Kapitola 1

Výroková logika

1.1 Výroky

1.1.1 Definice. 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 budeme vždy jako množinu atomických výroků (logických proměnných) A volit malá písmena anglické abecedy, případně je budeme indexovat. A tedy obsahuje $a, b, c, \ldots x, y, z$ nebo $x_1, x_2, \ldots$. Výrokové formule označujeme malými řeckými písmeny, např. $\alpha, \beta, \gamma, \ldots$ nebo $\varphi, \psi, \ldots$.

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 $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \Rightarrow \beta)$ a $(\alpha \Leftrightarrow \beta)$ odpovídá levý následník formuli α, pravý následník 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 Definice. *Pravdivostní ohodnocení*, též pouze *ohodnocení formulí*, je zobrazení $u\colon \mathcal{P}(A) \to \{0,1\}$, které splňuje následující 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 $a \in A$. □

Stručné zdůvodnění: Vezměme formuli α. Známe-li pravdivostní hodnotu logických proměnných, tj. listů syntaktického stromu formule α, pak jednoznačně určujeme pravdivostní hodnotu všech podformulí na základě 1.2.1, a to postupem ve stromě „směrem nahoru“.

Formálně správný důkaz je veden indukcí podle výšky syntaktického stromu formule α.

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 Poznámka. Uvědomte si, že pravdivostní ohodnocení je zobrazení u, které *každé* formuli α přiřazuje 0 (α není pravdivá v u) nebo 1 (α je pravdivá v u) takové, že u splňuje podmínky z definice 1.2.1. Věta 1.2.3 říká, že přestože všech formulí je spočetně mnoho, je různých pravdivostních ohodnocení je tolik, kolik existuje zobrazení množiny A do množiny $\{0,1\}$. V případě, že A je konečná množina, je takových různých zobrazení jen konečně mnoho; přesněji $2^{|A|}$, kde $|A|$ je počet prvků množiny A.

1.2.6 Definice tautologie, kontradikce a splnitelné formule. Formule se nazývá *tautologie*, jestliže je pravdivá ve všech ohodnocení; nazývá se *kontradikce*, jestliže je nepravdivá ve všech ohodnocení. Formule je *splnitelná*, jestliže existuje aspoň jedno ohodnocení, ve kterém je pravdivá. □

1.2.7 Příklady Jsou dány dvě formule α, β a logické proměnné a, b, pak

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.
4. Kontradikce je také každá negace tautologie. A naopak, negace libovolné kontradikce je tautologie.

1.3 Tautologická ekvivalence

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

Fakt, že dvě formule φ a ψ jsou tautologicky ekvivalentní, zapisujeme $\varphi \models\!\models \psi$.

1.3.2 Tvrzení. 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

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

□

1.3.3 Poznámka. Předchozí tvrzení můžeme využít i takto: Platí

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

(to zjistíme např. z pravdivostních tabulek formulí $\alpha \Rightarrow \beta$ a $\neg\alpha \vee \beta$). Na základě tvrzení 1.3.2 „dosazujeme“ ve formuli $((c \Rightarrow \neg(a \vee b)) \wedge \neg b$ za spojku $\Rightarrow$ v podformuli $c \Rightarrow \neg(a \vee b)$ a dostáváme

$$((c \Rightarrow \neg(a \vee b)) \wedge \neg b \models\!\models (\neg c \vee \neg(a \vee b)) \wedge \neg b.$$

(Zde $\alpha = c$ a $\beta = \neg(a \vee b)$.) Takovéto „dosazování“ předchozí tvrzení umožňuje.

1.3.4 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

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

□

1.3.5 Poznámka. Pro každé dvě formule α a β platí: $\alpha \models\!\models \beta$ právě tehdy, když $\alpha \Leftrightarrow \beta$ je tautologie.

1.3.6 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. To je proto, že formule $\neg\alpha \vee \alpha$ je tautologie a tudíž na pravdivosti či nepravdivosti formule β nezáleží.

1.3.7 Další spojky. Pravdivostní tabulka libovolné formule s jednou logickou proměnnou představuje zobrazení z množiny $\{0,1\}$ do množiny $\{0,1\}$. Existují čtyři zobrazení z množiny $\{0,1\}$ do množiny $\{0,1\}$:

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

Zobrazení f_1 je konstantní 0, a odpovídá libovolné formuli s jednou proměnnou, která je kontradikcí. Podobě zobrazení f_4 odpovídá formuli, která je tautologie. Zobrazení f_2 formuli x (zde x je logická proměnná), a zobrazení f_3 odpovídá formuli $\neg x$. Tedy nemáme důvod definovat „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 ze zobrazení $f_0, \ldots, f_{15}$ odpovídá nějaké formuli; např. f_0 odpovídá libovolné kontradikci, f_{15} libovolné tautologii, f_1 formuli $x \wedge y$, zobrazení f_{10} formuli $\neg y$, atd. Jako „nové“ spojky zavedeme spojky odpovídající zobrazením f_6, f_8 a f_{14}.

1.3.9 NAND. Definujeme novou logickou spojku $|$, nazývanou NAND (také *Shefferův operátor*), jako spojku, pro kterou $\alpha \mid \beta$ je nepravdivá právě tehdy, když α i β jsou obě pravdivé. □

1.3.10 Tvrzení. Platí

$$\alpha \mid \beta \models\!\models \neg(\alpha \wedge \beta).$$

□

Poznamenejme, že pravdivostní tabulka spojky NAND odpovídá zobrazení f_{14}.

1.3.11 NOR. Definujeme novou logickou spojku $\downarrow$, nazývanou *NOR* (také *Peirceova šipka*), jako spojku, pro kterou $\alpha \downarrow \beta$ je pravdivá právě tehdy, když α i β jsou obě nepravdivé. □

1.3.12 Tvrzení. Platí

$$\alpha \downarrow \beta \models\!\models \neg(\alpha \vee \beta).$$

□

Poznamenejme, že pravdivostní tabulka spojky NOR odpovídá zobrazení f_8.

1.3.13 XOR. Definujeme novou logickou spojku $\oplus$, nazývanou *XOR* (také *vylučovací nebo*), jako spojku, pro kterou $\alpha \oplus \beta$ je pravdivá právě tehdy, když buď α je nepravdivá a β je pravdivá, nebo α je pravdivá a β je nepravdivá. □

1.3.14 Tvrzení. Platí

$$\alpha \oplus \beta \models\!\!\models (\alpha \wedge \neg\beta) \vee (\neg\alpha \wedge \beta) \models\!\!\models \neg(\alpha \Leftrightarrow \beta).$$

□

Poznamenejme, že pravdivostní tabulka spojky XOR odpovídá zobrazení f_6.

1.3.15 Spojka F a T. Zavedeme ještě dvě spojky a to **F** a **T**; řetězce **F** a **T** budou také formulemi, přestože neobsahují žádnou logickou proměnnou. O těchto spojkách říkáme, že jsou *nulární*.

Platí: $\mathbf{F} \models\!\!\models \alpha$ pro každou kontradikci α. Tedy např. $\mathbf{F} \models\!\!\models (a \wedge \neg a)$. Dále $\mathbf{T} \models\!\!\models \neg\mathbf{F}$, tedy např. $\mathbf{T} \models\!\!\models (A \vee \neg a)$.

1.3.16 Poznámka. Protože jsme v předchozích odstavcích zavedli nové spojky NAND, NOR, XOR, **F** a **T**, měli bychom správně rozšířit definici formule; totiž tak, že

- **F** je formule, **T** je formule,
- jestliže α a β jsou formule, tak $(\alpha \mid \beta)$, $(\alpha \downarrow \beta)$ a $(\alpha \oplus \beta)$ jsou také formule.

Na druhou stranu víme, že každou formuli, která obsahuje spojky NAND, NOR, XOR, **F** nebo **T** jsme schopni nahradit tautologicky ekvivalentní formulí, která obsahuje jen původní spojky $\neg$, $\wedge$, $\vee$, $\Rightarrow$ a $\Leftrightarrow$.

1.3.17 Souvislost s logickými obvody. Každá výroková formule se dá realizovat jako logický obvod. Potřebujeme proto hradla pro všechny logické spojky (nejčastější jsou hradlo NOT – též zvaný invertor, hradlo AND a hradlo OR). Logický obvod má na každém vstupu 0 či 1 (přesněji logickou proměnnou) a každé kombinaci vstupních hodnot přiřazuje buď hodnotu 0 nebo hodnotu 1.

Tautologii odpovídá logický obvod, který na každou kombinaci vstupních hodnot dá výstup 1, kontradikci pak obvod, jehož výstup je vždy 0. Splnitelné formuli odpovídá logický obvod, který alespoň na jednu kombinaci vstupních hodnot dá výstup 1.

1.4 Úplné systémy logických spojek

1.4.1 Definice. Řekneme, že množina logických spojek Δ tvoří *úplný systém logických spojek* (též *funkčně úplný systém logických spojek*), jestliže pro každou formuli α existuje formule β s ní tautologicky ekvivalentní, která používá pouze spojky z množiny Δ. □

1.4.2 Tvrzení. Nechť Δ tvoří úplný systém logických spojek a nechť Π je množina spojek. Jestliže platí

1. pro každou binární spojku $\square \in \Delta$ existuje formule α obsahující pouze spojky z množiny Π a taková, že $\alpha \models\!\models x \square y$,
2. pro každou unární spojku $\diamond \in \Delta$ existuje formule β obsahující pouze spojky z množiny Π a taková, že $\beta \models\!\models \diamond x$,
3. pro každou nulární spojku $K \in \Delta$ existuje formule γ obsahující pouze spojky z množiny Π a taková, že $\gamma \models\!\models K$,

pak Π je také úplný systém logických spojek. □

1.4.3 Příklady úplných systémů logických spojek.

1. Množina $\{\neg, \wedge, \vee, \Rightarrow\}$ tvoří úplný systém logických spojek.

   To je proto, že další logické spojky $\Leftrightarrow$, $|$, $\downarrow$, $\oplus$, **F** a **T** můžeme „utvořit“ ze spojek z množiny $\{\neg, \wedge, \vee, \Rightarrow\}$. Přesněji

   $$\alpha \Leftrightarrow \beta \models\!\models (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha),$$

   zbytek plyne z 1.3.10, 1.3.12, 1.3.14 a 1.3.15.

2. Množina $\Delta = \{\neg, \vee\}$ tvoří úplný systém logických spojek.

   Víme, že $\{\neg, \wedge, \vee, \Rightarrow\}$ tvoří úplný systém logických spojek. Nyní si stačí uvědomit, že platí:

   $$(a \Rightarrow b) \models\!\models (\neg a \vee b) \quad \text{a} \quad (a \wedge b) \models\!\models \neg(\neg a \vee \neg b).$$

3. Množina $\Delta = \{\neg, \wedge\}$ tvoří úplný systém logických spojek.

   Protože množina $\Delta = \{\neg, \vee\}$ tvoří úplný systém logických spojek, stačí ověřit, že $\vee$ můžeme „vytvořit“ pomocí logických spojek $\neg, \wedge$. Přitom

   $$(a \vee b) \models\!\models \neg(\neg a \wedge \neg b).$$

4. Množina $\Delta = \{\neg, \Rightarrow\}$ tvoří úplný systém logických spojek.

   Obdobně jako výše si stačí uvědomit, že pomocí spojek $\neg$ a $\Rightarrow$ „vytvoříme“ spojku $\vee$ a/nebo $\wedge$.

   $$(a \vee b) \models\!\models (\neg a \Rightarrow b) \quad \text{nebo} \quad (a \wedge b) \models\!\models \neg(a \Rightarrow \neg b).$$

5. Množina $\{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$ (a tudíž ani žádná její podmnožina) netvoří úplný systém logických spojek.

   Stačí si uvědomit, že každá formule obsahující pouze spojky $\wedge$, $\vee$, $\Rightarrow$ a $\Leftrightarrow$ je pravdivá v pravdivostním ohodnocení u_0, v němž jsou pravdivé všechny logické proměnné. Ovšem formule $\neg a$ (a je logická proměnná) je v tomto ohodnocení u_0 nepravdivá.

6. Každá z množin $\{|\}$ (tj. NAND) a $\{\downarrow\}$ (tj. NOR) je úplný systém logických spojek.

   Uvažujme nejprve spojku NAND. Využijeme fakt, že $\{\neg, \wedge\}$ tvoří úplný systém logických spojek. Stačí proto „vytvořit“ spojky $\neg$ a $\wedge$ pomocí spojky $|$. A to je možné, protože

   $$\neg a \barwedge (a \mid a) \quad \text{a} \quad (a \wedge b) \barwedge \neg(a \mid b) \barwedge (a \mid b) \mid (a \mid b).$$

   Pro spojku $\downarrow$ postupujeme analogicky, pouze používáme $\{\neg, \vee\}$ jako úplný systém logických spojek místo množiny $\{\neg, \wedge\}$. Máme

   $$\neg a \barwedge (a \downarrow a) \quad \text{a} \quad (a \vee b) \barwedge \neg(a \downarrow b) \barwedge (a \downarrow b) \downarrow (a \downarrow b).$$

Všimněte si, že ani pro spojku NAND ani pro spojku NOR neplatí asociativní zákon, tj. neplatí $(\alpha \mid \beta) \mid \gamma \barwedge \alpha \mid (\beta \mid \gamma)$; obdobně pro $\downarrow$.

1.5 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 do logických proměnných. 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. V dalším ukážeme, že dokonce můžeme volit formuli ve speciálním tvaru, v tzv. *konjunktivním normálním tvaru* a/nebo *disjunktivním normálním tvaru*.

1.5.1 Booleova funkce.

Definice. *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 $f(x_1, x_2, \ldots, x_n)$). □

1.5.2 Formule v disjunktivním normálním tvaru.

Definice. *Literál* je logická proměnná nebo negace logické proměnné. *Minterm* je literál nebo konjunkce konečně mnoha literálů.

Řekneme, že formule je v *disjunktivním normálním tvaru*, zkráceně v *DNF*, jestliže je mintermem nebo disjunkcí konečně mnoha mintermů. □

1.5.3 Poznámka. Jestliže ve formuli, která je v DNF, obsahuje každý minterm všechny proměnné, říkáme, že se jedná o *úplnou DNF*.

Poznamenejme, že někteří autoři požadují, aby minterm neobsahoval současně proměnnou i její negaci. My to ale nevyžadujeme.

1.5.4 Konvence. Z tvrzení 1.3.4 víme, že pro spojky $\vee$ a $\wedge$ platí asociativní zákon; tj. jde-li nám jen o tautologickou ekvivalenci formulí, je jedno, jak „závorkujeme“ ve formuli, která je disjunkcí nebo konjunkcí n, $n > 1$, formulí. Budeme proto v dalším vynechávat vnitřní závorky tj. formuli, která je disjunkcí formulí α_i, pro $i = 1, 2, \ldots, n$, zapisujeme

$$\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n.$$

Obdobně pro konjunkci.

1.5.5 Věta. Ke každé Booleově funkci f existuje formule φ v DNF, která odpovídá Booleově funkci f. □

Zdůvodnění. Jestliže funkce f má všechny hodnoty rovny nule, je jí odpovídající formule kontradikce. Kontradikci můžeme reprezentovat např. formulí $x \wedge \neg x$ (v našem přístupu se jedná o minterm, tedy formuli v DNF).

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.

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

Ano, nejprve zkonstruujeme Booleovu funkci f odpovídající formuli α a k ní pak následně tautologicky ekvivalentní formuli β v DNF.

1.5.7 Formule v konjunktivním normálním tvaru. Definice. *Maxterm* je literál nebo disjunkce konečně mnoha literálů.

Řekneme, že formule je v *konjunktivním normálním tvaru*, zkráceně v *CNF*, jestliže je maxterm nebo konjunkcí konečně mnoha maxtermů. □

1.5.8 Poznámka. Poznamenejme, že maxtermu se také říká *klauzule* a to hlavně v rezoluční metodě. I my budeme dávat přednost termínu klauzule.

V některé literatuře se požaduje, aby maxterm neobsahoval současně logickou proměnnou i její negaci. My ale nic takového nevyžadujeme.

Jestliže každý maxterm/klauzule formule v CNF obsahuje všechny proměnné, říkáme, že se jedná o *úplnou CNF*.

1.5.9 Věta. Ke každé Booleově funkci f existuje formule ψ v CNF, která odpovídá Booleově funkci f. □

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

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. To znamená

$$f'(x_1, x_2, \ldots, x_n) = 1 \text{ právě tehdy, když } ; f(x_1, x_2, \ldots, x_n) = 0.$$

K funkci f' najdeme formuli β v DNF podle 1.5.5. Hledanou formuli α v CNF odpovídající funkci f dostaneme z formule $\neg\beta$ dvojím použitím de Morganova pravidla. Uvědomte si, že indukcí podle n se lehce dokáže, že pro každé $n \geq 1$ platí

$$\neg(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \models\!\!\!\models (\neg\alpha_1 \vee \neg\alpha_2 \vee \ldots \vee \neg\alpha_n).$$

Poznamenejme, že takto vzniklá formule je úplná CNF.

1.5.10 Pozorování. V důkazu předchozí věty jsme také mohli postupovat přímo, podobně jako ve větě 1.5.5, jen s tím, že bychom popisovali řádky obsahující 0, v případě, že by x_i měla hodnotu 0, do formule bychom dali literál $\neg x_i$, jinak literál x_i a zaměnili disjunkci s konjunkcí a naopak.

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

1.5.12 Karnaughovy mapy. Pro zjednodušení formulí v disjunktivní a konjunktivní normální formě (viz 1.5.5 a 1.5.9) je možné použít tzv. Karnaughovy mapy. Vhodné je to hlavně pro tři nebo čtyři logické proměnné.

1.6 Booleovský kalkul.

Označme $x = u(a)$ a $y = u(b)$. Pak pro pravdivostní ohodnocení u platí:

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

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

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

Pro tyto operace platí řada rovností, tak, jak je známe z výrokové logiky. Uvědomte si, že se vlastně jedná o přeformulování tvrzení 1.3.4.

1.6.2 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. $x + 1 = 1,\ x + 0 = x$.

□

1.6.3 Booleovy funkce v DNF a CNF. Nyní můžeme 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 proměnné Booleovy funkce 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:

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

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

1.7 Sémantický důsledek

1.7.1 Množina formulí pravdivá v ohodnocení.

Definice. Řekneme, že množina formulí S je *pravdivá* v ohodnocení u, jestliže každá formule z S je pravdivá v u, tj. jestliže $u(\varphi) = 1$ pro všechna $\varphi \in S$. □

Uvědomte si, že množina formulí S je nepravdivá v ohodnocení u, jestliže existuje formule $\varphi \in S$, která je nepravdivá v ohodnocení u.

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

Poznamejme. že prázdná množina formulí je pravdivá v každém ohodnocení. Ano, každá množina je v daném ohodnocení u buď pravdivá nebo nepravdivá. Vezměme libovolné ohodnocení u; prázdná množina v něm nemůže být nepravdivá, to by musela existovat formule $\varphi \in \emptyset$, která by byla v u nepravdivá. Proto je $\emptyset$ v u pravdivá.

1.7.3 Splnitelná množina formulí.

Definice. 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*. □

1.7.4 Poznamenejme, že z předchozí poznámky vyplývá, že prázdná množina formulí je splnitelná, neboť je pravdivá nejen v jednom, ale dokonce ve všech pravdivostních ohodnocení.

1.7.5 Sémantický důsledek.

Definice. 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. □

1.7.6 Značení. 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 $\{\alpha\} \models \varphi$.

1.7.7 Poznámka. Definici sémantického důsledku můžeme přeformulovat následovně:

$$S \models \varphi \quad \text{právě tehdy, když} \quad u(S) \leq u(\varphi) \ \text{ pro každé ohodnocení } u.$$

Uvědomte si také, že φ *není sémantický důsledek* množiny formulí S, tj. neplatí $S \models \varphi$, právě tehdy, když existuje pravdivostní ohodnocení u takové, že množina S je v něm pravdivá a formule φ ne.

1.7.8 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. $\{(\alpha \wedge \beta) \Rightarrow \gamma, (\alpha \wedge \neg\beta) \Rightarrow \gamma\} \models (\alpha \Rightarrow \gamma)$.

1.7.9 Tvrzení.

1. Je-li S množina formulí a $\varphi \in S$, pak φ je konsekventem S. Jinými slovy $S \models \varphi$ pro každou $\varphi \in S$.

2. Tautologie je konsekventem každé množiny formulí S.

3. Formule φ je tautologie právě tehdy, když je konsekventem každé množiny S a to je právě tehdy, když je konsekventem prázdné množiny formulí.

4. Každá formule je konsekventem nesplnitelné množiny formulí.

5. Máme dvě množiny formulí M a N takové, že $M \subseteq N$. 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.

7. Jestliže $S \models \alpha$ i $S \models \neg\alpha$ pro nějakou formuli α, pak S je nesplnitelná množina formulí.

□

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

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

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

□

1.7.12 Věta. Pro množinu formulí S a formuli φ platí

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

□

Důkaz: Jedná se o ekvivalenci dvou tvrzení, musíme proto dokázat dvě implikace.

1. Předpokládejme, že platí $S \models \varphi$. Musíme ukázat, že neexistuje ohodnocení u, ve kterém by množina $S \cup \{\neg\varphi\}$ byla pravdivá. Uvažujme libovolné ohodnocení u. Pak buď platí $u(S) = 0$ (tj. S není pravdivá v u), nebo $u(S) = 1$ (tj. S je pravdivá v u).

   Když $u(S) = 0$, pak $u(S \cup \{\neg\varphi\}) = 0$, neboť přidáním formule se z nepravdivé množiny pravdivá stát nemůže.

   Když $u(S) = 1$, pak $u(\varphi) = 1$ a proto $u(\neg\varphi) = 0$ a množina $S \cup \{\neg\varphi\}$ je nepravdivá v u.

   Tedy neexistuje ohodnocení, ve kterém by množina $S \cup \{\neg\varphi\}$ byla pravdivá, tudíž je nesplnitelná.

2. Předpokládejme, že množina $S \cup \{\neg\varphi\}$ je nesplnitelná. Uvažujme ohodnocení u, ve kterém je S pravdivá. Pak $u(\neg\varphi) = 0$, jinak by $u(S \cup \{\neg\varphi\}) = 1$, a $S \cup \{\neg\varphi\})$ by byla splnitelná. Tedy $u(\varphi) = 1$, a proto $S \models \varphi$.

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

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

□

Důkaz: Opět je třeba dokázat dvě implikace.

1. Předpokládejme, že platí $S \cup \{\varphi\} \models \psi$. Uvažujme libovolné ohodnocení u, ve kterém je pravdivá množina S. V tomto ohodnocení je formule φ buď pravdivá nebo nepravdivá.

   Předpokládejme, že $u(\varphi) = 1$, pak $u(S \cup \{\varphi\}) = 1$ a proto $u(\psi) = 1$. To ale znamená, že formule $\varphi \Rightarrow \psi$ je pravdivá v u.

   Předpokládejme, že $u(\varphi) = 0$. Pak ale $u(\varphi \Rightarrow \psi) = 1$ nezávisle na pravdivosti či nepravdivosti formule ψ.

   V obou případech je formule $\varphi \Rightarrow \psi$ pravdivá. Proto platí $u(\varphi \Rightarrow \psi) = 1$, což dokazuje $S \models (\varphi \Rightarrow \psi)$.

2. Předpokládejme, že platí $S \models (\varphi \Rightarrow \psi)$. Uvažujme libovolné ohodnocení u, ve kterém je pravdivá množina $S \cup \{\varphi\}$. Pak platí $u(S) = 1$ a tudíž $u(\varphi \Rightarrow \psi) = 1$. Protože navíc $u(\varphi) = 1$, musí platit i $u(\psi) = 1$. Dokázali jsme $S \cup \{\varphi\} \models \psi$.

1.8 Rezoluční metoda ve výrokové logice

1.8.1 Klauzule – připomenutí. Je dána množina logických proměnných A. *Literál* je buď logická proměnná (tzv. *pozitivní literál*) nebo negace logické proměnné (tzv. *negativní literál*). *Komplementární literály* jsou literály x a $\neg x$, kde x je logická proměnná. *Klauzule* je literál nebo disjunkce konečně mnoha literálů.

Za klauzuli budeme považovat ještě formuli **F** zastupující kontradikci, kterou budeme nazývat *prázdná klauzule*.

Nepřesně se na prázdnou klauzuli **F** můžeme dívat jako na „disjunkci žádného literálu“.

1.8.2 Rezoluční metoda rozhoduje, zda daná množina klauzulí je splnitelná nebo je nesplnitelná. Tím je také „univerzální 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 klauzulí S_α taková, že S_α i α jsou pravdivé ve stejných ohodnocení.

1.8.3 Konvence. Pro jednoduchost zavedeme následující konvenci: Máme dánu klauzuli C a literál p, který se v C vyskytuje. Pak symbolem $C \setminus p$ označujeme klauzuli, která obsahuje všechny literály jako C kromě p. Jestliže $C = p$, pak $C \setminus p$ je prázdná klauzule **F**.

Tedy např. je-li $C = \neg x \vee y \vee \neg z$, pak

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

1.8.4 Resolventa dvou klauzulí.

Definice. Řekneme, že klauzule D je *resolventou klauzulí* C_1 a C_2, jestliže existuje logická proměnná x taková, že literál x se vyskytuje v klauzuli C_1, literál $\neg x$ se vyskytuje v klauzuli C_2 a D obsahuje všechny literály z klauzule $C_1 \setminus x$ a z klauzule $C_2 \setminus \neg x$ s tím, že každý lirerál obsahuje pouze jednou. Jestliže obě klauzule $C_1 \setminus x$, $C_2 \setminus \neg x$, neobsahují žádný literál, je resolventou prázdná klauzule **F**.

Říkáme též, že klauzule D je *resolventou* C_1 *a* C_2 *podle* x a značíme $D = res_x(C_1, C_2)$. □

Poznamenejme, že resolventu D můžeme zapsat též takto:

$$D = (C_1 \setminus x) \vee (C_2 \setminus \neg x)$$

s tím, že D obsahuje každý literál pouze jednou (i kdyby byl obsažen v $C_1 \setminus x$ i $C_2 \setminus \neg x$).

1.8.5 Poznámka. Mohou existovat dvě klauzule C_1 a C_2, které mají resolventy podle dvou různých logických proměnných. V takovém případě jsou obě resolventy tautologie.

1.8.6 Tvrzení. Máme dány dvě klauzule C_1, C_2 a označme D jejich resolventu. Pak D je sémantický důsledek množiny $\{C_1, C_2\}$. □

Důkaz: Předpokládejme, že množina $\{C_1, C_2\}$ je pravdivá v ohodnocení u. Uvažujme resolventu $D = (C_1 \setminus p) \vee (C_2 \setminus \neg p)$.

Mohou nastat dva případy: $u(p) = 0$ nebo $u(p) = 1$. Předpokládejme, že $u(p) = 0$. Protože $u(C_1) = 1$ a $C_1 = (C_1 \setminus p) \vee p$, dostáváme $u(C_1 \setminus p) = 1$. Proto $u(D) = 1$.

Předpokládejme, že $u(p) = 1$. Protože $u(C_2) = 1$ a $C_2 = (C_2 \setminus \neg p) \vee \neg p$, dostáváme $u(C_2 \setminus \neg p) = 1$. Proto $u(D) = 1$.

Ukázali jsme, že resolventa D je pravdivá v ohodnocení u.

1.8.7 Věta. Máme dánu množinu klauzulí S a označme D resolventu některých dvou klauzulí z množiny S. Pak množiny S a $S \cup \{D\}$ jsou pravdivé ve stejných ohodnocení. □

Jedná se o důsledek předchozího tvrzení. Jestliže v nějakém ohodnocení u je pravdivá množina $\{C_1, C_2, D\}$, tak je v něm pravdivá i množina $\{C_1, C_2\}$. Předpokládejme tedy, že množina $\{C_1, C_2\}$ je pravdivá v ohodnocení u. Protože D je sémantický důsledek množiny $\{C_1, C_2\}$, je D pravdivé v u. To ale znamená, že v u je pravdivá množina $\{C_1, C_2, D\}$.

1.8.8 Rezoluční princip. Je dána množina klauzulí S. Označme

$$R(S) = S \ \cup \{D \mid D \text{ je resolventa některých klauzulí z } S\}$$

Definujme

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

Je-li množina S konečná (obsahuje-li S jen konečně mnoho logických proměnných), pak pro výpočet $R^\star(S)$ nemusíme spočítat „nekonečně mnoho“ množin $R^n(S)$. Je to proto, že pro konečnou množinu logických proměnných existuje jen konečně mnoho klauzulí, které můžeme přidat. Musí tedy existovat n takové, že $R^n(S) = R^{n+1}(S)$. Pro toto n platí $R^n(S) = R^\star(S)$.

1.8.9 Věta (Rezoluční princip.) Konečná množina klauzulí S je splnitelná právě tehdy, když $R^\star(S)$ neobsahuje prázdnou klauzuli **F**. □

1.8.10 Základní postup. Předchozí věta dává návod, jak zjistit, zda daná konečná množina klauzulí je splnitelná nebo je nesplnitelná:

1. Ke každé formuli φ z množiny M najdeme tautologicky ekvivalentní formuli α_φ v CNF. Množinu M pak nahradíme množinou S všech klauzulí vyskytujících se v některé formuli α_φ. Klauzule, které jsou tautologiemi, vynecháme. Jestliže nám nezbude žádná klauzule, množina M se skládala z tautologií a je splnitelná (dokonce pravdivá v každém pravdivostním ohodnocení).
2. Vytvoříme množinu $R^\star(S)$.
3. Obsahuje-li $R^\star(S)$ prázdnou klauzuli, je množina S (a tedy i množina M) nesplnitelná, v opačném případě jsou S a M splnitelné.

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

1.8.11 Výhodnější postup – rezoluční algoritmus. Existuje ještě jeden postup, jak využít rezoluční metodu. Ten je jednodušší a nejen odpoví na otázku, zda konečná množina klauzulí S je splnitelná nebo nesplnitelná, ale umožní nám v případě splnitelnosti množiny sestrojit aspoň jedno pravdivostní ohodnocení, v němž je množina S pravdivá. Základní princip postupu spočívá v tom, že k dané množině klauzulí S s k logickými proměnnými zkonstruujeme novou množinu klauzulí S_1, která má o jednu logickou proměnnou méně a platí, že S je splnitelná právě tehdy, když S_1 je splnitelná. To děláme tak dlouho, dokud buď nedostaneme jako resolventu prázdnou klauzuli (výchozí množina klauzulí je nesplnitelná), nebo nedostaneme prázdnou množinu klauzulí (výchozí množina klauzulí je splnitelná).

Jeden krok postupu je tento: Máme konečnou neprázdnou množinu klauzulí S, kde žádná klauzule není tautologie. Vybereme jednu logickou proměnnou (označme ji x), která se v některé z klauzulí z S vyskytuje.

Najdeme množinu klauzulí S_1 s těmito vlastnostmi:

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

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

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

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

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

Označme N množinu všech resolvent klauzulí množiny S podle logické proměnné x (tj. resolvent vždy jedné klauzule z množiny M_x s jednou klauzulí z množiny $M_{\neg x}$), které nejsou tautologie.

Položíme $S_1 = M_0 \cup N$.

1.8.12 Tvrzení. Množina klauzulí S_1 zkonstruovaná výše je splnitelná právě tehdy, když je splnitelná množina S. □

Důkaz tohoto tvrzení uvedeme později, viz. 1.8.18.

1.8.13 Uvědomte si, že množina S_1 obsahuje o jednu logickou proměnnou méně než množina S: Klauzule z množiny M_0 logickou proměnnou x neobsahují a to samé platí klauzule z N — resolventy podle logické proměnné x. Navíc, množina S je splnitelná právě tehdy, když je splnitelná množina S_1.

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

1. Při vytváření resolvent dostaneme prázdnou klauzuli **F**. Tedy S je nesplnitelná.
2. Dostaneme prázdnou množinu klauzulí. V tomto případě je S splnitelná, protože prázdná množina formulí je splnitelná.

1.8.14 Je výhodné předchozí postup znázorňovat v tabulce. Na začátku práce utvoříme tabulku, která obsahuje pro každou klauzuli množiny S jeden sloupec. V prvním řádku vybereme jednu proměnnou, řekněme x, a řádek označíme x. Procházíme neoznačené sloupce tabulky. Jestliže klauzule odpovídající sloupci neobsahuje proměnnou x, sloupec přeskočíme. Jestliže klauzule sloupce obsahuje literál x (tj. klauzule patří do množiny M_x), napíšeme do sloupce 1. Jestliže klauzule odpovídající dosud neoznačenému sloupci obsahuje literál $\neg x$ (tj. patří do množiny $M_{\neg x}$), napíšeme do sloupce 0.

Vybereme libovolnou klauzuli C_1, která má v řádku proměnné x znak 1, a libovolnou klauzuli C_2, která má v řádku 0. Sloupec pro jejich resolventu podle x přidáme v případě, že se jedná o novou klauzuli, která není tautologie (tj. přidáváme sloupce pro nové klauzule z množiny N). Jestliže žádný sloupec 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 klauzuli, 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 množinu S_1). Opět vybereme proměnnou, která se v některé ze zbylých klauzulí vyskytuje. Postupujeme dále jako v kroku 1.

Celý postup tedy končí buď přidáním prázdné klauzule, 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 (příklad je uveden v **??**].

1.8.15 Příklad. Pomocí rezoluční metody rozhodněte, zda množina klauzulí

$$S = \{x \vee y \vee \neg z, \neg x, x \vee y \vee z, x \vee \neg y, z \ \vee t \vee v, \neg t \vee w\}$$

je splnitelná. Je-li splnitelná, najděte pravdivostní ohodnocení, ve kterém je S pravdivá.

Řešení. Postup si znázorníme v následující tabulce 1.1. Tabulka má jeden sloupec pro každou klauzuli množiny S.

	$x \vee y \vee \neg z$	$\neg x$	$x \vee y \vee z$	$x \vee \neg y$	$z \vee t \vee v$	$\neg t \vee w$					
y:	1		1	0			$x \vee \neg z$	$x \vee z$			
x:		0					1	1	$\neg z$	z	
z:					1				0	1	**F**

Tabulka 1.1: Tabulka pro rezoluční metodu

Nejprve odstraníme logickou proměnnou y: První řádek tabulky je označen y. Nyní v tomto řádku napíšeme do sloupce 1, jestliže daná klauzule obsahuje literál y, a napíšeme 0, jestliže odpovídající klauzule obsahuje literál $\neg y$. Jestliže daná klauzule neobsahuje proměnnou y, do sloupce nepíšeme nic. K tabulce přidáme za každou resolventu podle proměnné y, která není tautologie, další sloupec odpovídající této resolventě. Jsou to sloupce odpovídající klauzulím:

$$x \vee \neg z \ = res_y(x \vee y \vee \neg z, x \vee \neg y) \quad \text{a} \quad x \vee z \ = res_y(x \vee y \vee z, x \vee \neg y).$$

Množina S_1 se skládá ze všech klauzulí, jejichž sloupce nejsou označeny, tj. neobsahují ani 1, ani 0. Máme

$$S_1 = \{\neg x, z \ \vee t \vee v, \neg t \vee w, x \vee \neg z, x \vee z\}.$$

V dalším kroku vybereme další logickou proměnnou, např. x, a postupuje obdobně jako v kroku 1. Dostaneme množinu klauzulí S_2 (která již neobsahuje ani logickou proměnnou y, ani x):

$$S_2 = \{z \vee t \vee v, \neg t \vee w, \neg z, z\}.$$

Uvědomte si, že platí: množina S je splnitelná právě tehdy, když je splnitelná množina S_2.

Dále vybereme logickou proměnnou z. Protože devátý sloupec odpovídá klauzuli z a desátý klauzuli $\neg z$, je jejich resolventa prázdná klauzule **F**. Tím jsme ukázali, že množina S_2 je nesplnitelná a proto jsou nesplnitelné i množiny klauzulí S_1 a S. Tedy odpověď je: S je nesplnitelná.

1.8.16 Příklad. Pomocí rezoluční metody rozhodněte, zda je splnitelná množina klauzulí

$$S = \{a \vee \neg d,\ \neg b \vee \neg c,\ b \vee d,\ \neg b \vee \neg e,\ a \vee c \vee d,\ \neg a \vee \neg d\}$$

Jestliže je splnitelná, najděte pravdivostní ohodnocení, ve kterém je S pravdivá.

Řešení: Postupujeme obdobně jako v minulém příkladě. Dostaneme následující tabulku 1.2. (Uvědomte si, že tvar tabulky je určen pořadím výběru jednotlivých logických proměnných.)

	$a \vee \neg d$	$\neg b \vee \neg c$	$b \vee d$	$\neg b \vee \neg e$	$a \vee c \vee d$	$\neg a \vee \neg d$			
e:				0					
c:		0			1		$a \vee \neg b \vee d$		
b:			1				0	$a \vee d$	
d:	0					0		1	a
a:									1

Tabulka 1.2: Příklad tabulky pro splnitelnou množinu

Všimněte si, že v předposledním řádku jsme nepřidali sloupec pro jednu resolventu; je to proto, že $res_d(\neg a \vee \neg d, a \vee d)$ je tautologie $a \vee \neg a$.

Z tabulky je patrné, že množina S je splnitelná, protože nakonec jsme získali prázdnou množinu klauzulí — nezbyl žádný neoznačený sloupec — a prázdná množina je pravdivá ve všech ohodnoceních, tudíž je splnitelná.

Nyní z tabulky 1.2 odvodíme pravdivostní ohodnocení, ve kterém je množina S pravdivá. Postup je znázorněn v tabulce 1.3.

	$a \vee \neg d$	$\neg b \vee \neg c$	$b \vee d$	$\neg b \vee \neg e$	$a \vee c \vee d$	$\neg a \vee \neg d$			
e:				0					
c:		0			1		$a \vee \neg b \vee d$		
b:			1				0	$a \vee d$	
d:	0					0		1	a
a:									1
	$\uparrow_1$	$\uparrow_4$	$\uparrow_3$	$\uparrow_5$	$\uparrow_1$	$\uparrow_2$	$\uparrow_1$	$\uparrow_1$	$\uparrow_1$

Tabulka 1.3: Konstrukce pravdivostního ohodnocení

Poslední řádek tabulky byl ohodnocen proměnnou a. Protože v tomto řádku máme 1, prohlásíme literál a za pravdivý, tj. položíme $u(a) = 1$. Označíme si v tabulce **všechny** klauzule, které se touto volbou $u(a) = 1$ stanou pravdivé (označeny jsou šipkou s číslem 1).

Nyní přistoupíme k předposlednímu řádku tabulky. Sloupce, které nejsou označené šipkou 1, obsahují v tomto řádku už jen 0. Rozšíříme proto pravdivostní ohodnocení u tak, aby literál $\neg d$ by pravdivý, tj. položíme $u(d) = 0$. Označíme šipkou s indexem 2 ty klauzule, které se touto volbou stanou pravdivé (a nestaly se pravdivé již v předchozím kroku).

Obdobným způsobem postupujeme v tabulce nahoru až zajistíme, že všechny klauzule obsažené v tabulce budou pravdivé: dostaneme $u(b) = 1$, $u(c) = 0$ a $u(e) = 0$.

Je snadné se přesvědčit, že v takto definovaném pravdivostním ohodnocení je původní množina klauzulí S pravdivá.

1.8.17 Poznámka. V předchozím příkladě jsme definovali pravdivostní ohodnocení pro všechny logické proměnné, které se v množině S nacházely. To se nemusí stát vždy. V případě, že zajistíme pravdivost všech klauzulí nezávisle na některé logické proměnné, znamená to, že tuto proměnnou můžeme definovat libovolně.

Uvědomte si, že z jedné tabulky obecně nedostaneme **všechna** pravdivostní ohodnocení, ve kterých je původní množina pravdivá. Při jiném pořadí proměnných jsme mohli dostat jiná ohodnocení.

Též se může stát, že při zpětném postupu v některém řádku, řekněme logické proměnné t, nezbude ani 0, ani 1. V takovém případě musíme zkusit obě možnosti — definovat $u(t) = 1$ a když tato volba nevede k cíli, tak $u(t) = 0$. Jedna z možností povede k cíli.

1.8.18 Důkaz tvrzení 1.8.12. Připomeňme postup, jehož správnost budeme dokazovat.

Máme konečnou množinu klauzulí S, kde žádná klauzule není tautologií. Zvolíme jednu logickou proměnnou (označme ji x), která se v některé z klauzulí z S vyskytuje.

- Rozdělíme klauzule množiny S do tří skupin:

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

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

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

- Označíme N množinu všech resolvent klauzulí množiny S podle literálu x, které nejsou tautologiemi; tj. $N = \{D \mid D = res_x(C, C'), C \in M_x, C' \in M_{\neg x}$ a D není tautologie$\}$.

- Položíme $S_1 = M_0 \cup N$.

Dokážeme, že S_1 je splnitelná právě tehdy, když je splnitelná množina S.

Důkaz: Předpokládejme, že množina klauzulí S je splnitelná. To znamená, že existuje pravdivostní ohodnocení u takové, že $u(S) = 1$.

Pak $u(M_0) = 1$, protože $M_0 \subseteq S$. Navíc $u(N) = 1$, protože v N jsou jen resolventy klauzulí z množiny S a ty jsou (podle 1.8.7) pravdivé v u. Tedy $u(S_1) = 1$ a S_1 je splnitelná množina.

Předpokládejme, že množina S_1 je splnitelná. Existuje proto ohodnocení u takové, že $u(S_1) = 1$.

Nyní $u(M_0) = 1$, protože $M_0 \subseteq S_1$. Ukážeme, že můžeme zadefinovat hodnotu $u(x)$ tak, aby také obě množiny M_x a $M_{\neg x}$ byly pravdivé v u.

Může nastat jeden ze dvou případů

1. Všechny klauzule v M_x jsou pravdivé nezávisle na ohodnocení logické proměnné x; jinými slovy, v každé klauzuli z M_x je pravdivý aspoň jeden literál jiný než x.

   V tomto případě položme $u(x) := 0$, tj. $u(\neg x) = 1$ a všechny klauzule z $M_{\neg x}$ jsou automaticky pravdivé v u, protože obsahují literál $\neg x$. Tedy $u(S) = 1$.

2. Existuje klauzule $C \in M_x$, jejíž všechny literály různé od literálu x jsou nepravdivé; jinými slovy, $u(C \setminus x) = 0$.

   Uvažujme libovolnou klauzuli $C' \in M_{\neg x}$. Ukážeme, že $u(C' \setminus \neg x) = 1$, tj. ukážeme, že C' je pravdivá v u nezávisle na literálu $\neg x$.

   Protože $D = res_x(C, C')$ patří do množiny N, je $u(D) = 1$. Přitom $D = (C \setminus x) \vee (C' \setminus \neg x)$. Víme, že $u(C \setminus x) = 0$, musí proto být $u(C' \setminus \neg x) = 1$, a tudíž klauzule $C' \setminus \neg x$ je pravdivá v u nezávisle na hodnotě x. To nám umožňuje definovat $u(x) = 1$. Tím jsme dostali ohodnocení, ve kterém jsou pravdivé obě množiny M_x i $M_{\neg x}$; tedy je pravdivá i celá množina S.

1.9 Přirozená dedukce

V předchozím textu jsme se zabývali sémantickým důsledkem ve výrokové logice. Matematická logika studuje také tzv. logický důsledek, to je otázku, který výrok/formule logicky vyplývá z dané množiny výroků/formulí. Formalizace tohoto pojmu je zhruba následující: Formule α je logickým důsledkem množiny formulí S, jestliže je možno ji odvodit z S v některém „správném“ odvozovacím systému. Správný odvozovací systém pro nás bude ten, který je

- *bezesporný* — odvozováním z prázdné množiny předpokladů nevyplývá aspoň jedna formule,
- a *úplný* — vše, co je pravdivé v něm odvodíme, tj. odvodíme každý sémantický důsledek.

V následujícím textu přiblížíme odvozovací systém, který se nazývá přirozená dedukce. Uvědomte si, že se bude jednat jen o velmi stručný částečně „intuitivní“ úvod do problematiky

Zmíníme zde dva odvozovací systémy:

- **Hilbertův systém**: obsahuje tři typy axiomů — formulí, které se „mohou použít“ kdekoli, a jedno jediné odvozovací pravidlo, tzv. *Modus ponens*, kde z formulí α a $\alpha \Rightarrow \beta$ lze vyvodit formuli β.
- **Přirozená dedukce**, odvozovací systém, který nemá axiomy, ale využívá pomocných předpokladů uvnitř odvození.

1.9.1 Přirozená dedukce ve výrokové logice. Systém odvozovacích pravidel přirozené dedukce obsahuje pro každou logickou spojku $\neg$, $\vee$, $\wedge$ a $\Rightarrow$ pravidla. Pravidla, která spojku „zavádí“, tj. umožňují napsat formuli s touto spojkou, druhá spojku „odstraňují“. Prvním pravidlům říkáme *I-pravidla* („I“ pro introduction, tj. zavedení), druhým *E-pravidla* („E“ pro elimination, neboli odstranění).

1.9.2 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, která popíšeme dále;
- všechny pomocné předpoklady jsou již pasivní, též vysvětlíme dále.

□

1.9.3 Odvozovací pravidla. Jsou dány formule φ, ψ a α. Nejprve uvedeme pravidla, která nepotřebují tzv. boxy — „vložená odvození“:

I-pravidla pro $\wedge$: $\dfrac{\varphi,\ \ \psi}{\varphi \wedge \psi}$ $\dfrac{\varphi,\ \ \psi}{\psi \wedge \varphi}$ E-pravidla pro $\wedge$: $\dfrac{\varphi \wedge \psi}{\varphi}$ $\dfrac{\varphi \wedge \psi}{\psi}$

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

I-pravidla pro $\vee$: $\dfrac{\varphi}{\varphi \vee \psi}$ $\dfrac{\varphi}{\psi \vee \varphi}$ E-pravidlo pro $\neg$: $\dfrac{\neg\neg\varphi}{\varphi}$

Další pravidla vyžadují vložená odvození, též „boxy“. Boxy se používají tak, že box začíná tzv. pomocným předpokladem, pro každý box pouze jeden pomocný předpoklad, který uvnitř celého boxu můžeme „volně“ používat. Box je možné uzavřít pouze na základě některého pravidla, které boxy obsahuje. Tím, že toto pravidlo použijeme, stává se pomocný předpoklad „pasivní“ a box uzavíráme.

První z nich je

I-pravidlo pro $\Rightarrow$: $\dfrac{\boxed{\begin{array}{c}\varphi\\ \vdots\\ \psi\end{array}}}{\varphi \Rightarrow \psi}$

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

Další pravidla s pododvozeními jsou

E-pravidlo pro $\vee$:

$$\frac{\varphi \vee \psi, \quad \boxed{\begin{array}{c}\varphi\\ \vdots\\ \alpha\end{array}}, \quad \boxed{\begin{array}{c}\psi\\ \vdots\\ \alpha\end{array}}}{\alpha} \qquad \text{nebo} \qquad \frac{\varphi \vee \psi, \quad \varphi \Rightarrow \alpha, \quad \psi \Rightarrow \alpha}{\alpha}$$

I-pravidlo pro $\neg$:

$$\frac{\boxed{\begin{array}{c}\varphi\\ \vdots\\ \alpha \wedge \neg\alpha\end{array}}}{\neg\varphi}$$

1.9.4 Logický důsledek. Formule φ je *logický důsledek množiny formulí* S (říkáme jim *předpoklady*), též *logicky vyplývá z* S, jestliže existuje odvození z S takové, že $\varphi_n = \varphi$. Zapisujeme $S \vdash \varphi$. □

1.9.5 Theorém. Formule α se nazývá *theorém*, jestliže je logickým důsledkem prázdné množiny předpokladů. □

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

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

□

1.9.7 Poznámka. Ukázat, že formule α, která má odvození z předpokladů S, je sémantickým důsledkem množiny S, není obtížné. Je to proto, že závěr každého pravidla přirozené dedukce je sémantickým důsledkem předpokladů. Toto ukazuje, že přirozená dedukce je bezesporným odvozovacím systémem.

Druhou implikaci, totiž, že každý sémantický důsledek α množiny formulí S má též odvození z S, je daleko těžší dokázat. Důkaz je nepřímý; ukáže se, že není možné, aby odvození neměl. Tato implikace pak říká, že přirozená dedukce je úplný odvozovací systém.

1.9.8 Z věty o úplnosti také vyplývá následující tvrzení.
Tvrzení. Formule je theorém právě tehdy, když je to tautologie. □

Kapitola 2

Predikátová logika

Ne všechny logicky správné úsudky se dají zachytit/zdůvodnit výrokovou logikou. Např. z pravdivosti vět: *Následník sudého čísla je číslo liché. Číslo 3 je následník čísla 2. Číslo 2 je sudé.* vyplývá tvrzení *Číslo 3 je liché.*

Takovýto úsudek v predikátové logice zdůvodnit lze.

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: $\mathbf{T}$, $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \mathbf{F}$
   c) obecného kvantifikátoru $\forall$ a existenčního kvantifikátoru $\exists$
2. *speciálních symbolů*, tj. po dvou disjunktních množin Pred, Kons a Func, kde
   a) Pred je množina predikátových symbolů (není prázdná)
   b) Konsje množina konstantních symbolů (může být prázdná)
   c) Func je množina funkčních symbolů
3. pomocných symbolů, jako jsou závorky „(, [,) ,]“ a čárka „,“

Pro každý predikátový i funkční symbol máme dáno přirozené číslo n, které udává kolika objektů se daný predikát týká, nebo kolik proměnných funkční symbol má. 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 na to v textu upozorníme.

Poznamenejme, 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.

Definice. 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.

Definice. *Atomická formule* je řetězec $P(t_1, t_2, \ldots, t_n)$, kde $P \in \mathsf{Pred}$ kladné arity n a $t_1, t_2, \ldots, t_n$ je n-tice termů, nebo je řetězec P, kde $P \in \mathsf{Pred}$ je predikátový symbol arity 0. □

Jinými slovy, atomická formule je buď predikátový symbol arity 0, nebo predikátový symbol kladné arity aplikovaný na tolik termů, kolik je jeho arita.

2.1.6 Formule.

Definice. Množina *formulí* je definována těmito pravidly:

1. Každá atomická formule je formule.
2. **F** je formule a 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 definujeme „ty nejjednodušší“ formule (atomické formule) a potom pomocí logických spojek a kvantifikátorů 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 formule 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* (někdy nazývaný *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 proměnnou, konstantou, predikátovým symbolem arity 0 nebo **F**.

2.1.10 Podformule.

Definice. *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.

Definice. 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, otevřená formule.

Definice. 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

- se jedná 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.

Definice. 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. Takovým formulím se tadé říká *formule s čistými proměnnými*.

2.2 Sémantika predikátové logiky

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

2.2.1 Interpretace jazyka predikátové logiky.

Definice. *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 $n > 0$ přiřazuje podmnožinu $[\![P]\!]$ množiny U^n, tj. n-ární relaci na množině U, každému predikátovému symbolu P arity 0 přiřazuje buď 0 nebo 1.
  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, update kontextu proměnných.

Definice. 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.

Definice. Je dána interpretace $\langle U, [\![-]\!]\rangle$ a kontext proměnných ρ. Pak termy interpretujeme následujícím způsobem.

1. Je-li term konstantní 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, hodnotu termu $f(t_1, \ldots, t_n)$ získáme tak, že funkci $[\![f]\!]$ provedeme 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. Je dána atomická formule $\varphi = P(t_1, \ldots, t_n)$, kde P je predikátový symbol arity $n > 0$ 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 a kontextu proměnných pravdivá právě tehdy, když n-tice hodnot termů $([\![t_1]\!]_\rho, \ldots, [\![t_n]\!]_\rho)$ má vlastnost $[\![P]\!]$.) Je-li P predikát arity 0, pak $\varphi = P$ je pravdivá právě tehdy, když $[\![(]\!]P) \neq \emptyset$.

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

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

□

2.2.5 Pravdivostní hodnota sentence.

Definice. Sentence φ je *pravdivá v interpretaci* $\langle U, [\![-]\!]\rangle$, jestliže je pravdivá v každém kontextu proměnných ρ. □

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

2.2.6 Model sentence.

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

2.2.7 Tautologie, kontradikce, splnitelná sentence.

Definice.

- Sentence φ se nazývá *tautologie*, jestliže je pravdivá v každé interpretaci.
- Sentence se nazývá *kontradikce*, jestliže je nepravdivá v každé interpretaci.
- Sentence se nazývá *splnitelná*, jestliže je pravdivá v aspoň jedné interpretaci.

□

Předchozí definice jsme také mohli definovat pomocí pojmu „model“. Tautologie je sentence, pro kterou je každá interpretace jejím modelem; sentence je splnitelná, má-li model; sentence je kontradikce, nemá-li model.

2.2.8 Příklady. Následující sentence jsou tautologie. (P je unární predikátový symbol, Q je binární predikátový symbol a a je konstantní symbol.)

1. $(\forall x\, P(x)) \Rightarrow P(a)$;
2. $P(a) \Rightarrow (\exists x\, P(x))$;
3. $\neg(\forall x\, P(x)) \Leftrightarrow (\exists x\, \neg P(x))$;
4. $\neg(\exists x\, P(x)) \Leftrightarrow (\forall x\, \neg P(x))$;
5. $(\forall x\, \forall y\, Q(x,y)) \Leftrightarrow (\forall y\, \forall x\, Q(x,y))$;
6. $(\exists x\, \exists y\, Q(x,y)) \Leftrightarrow (\exists y\, \exists x\, Q(x,y))$.

2.2.9 Následující sentence jsou splnitelné formule:

1. $\forall x\, \exists y\, Q(x,y)$,
2. $\forall x\, \forall y\, (x + y = y + x)$,

kde Q a $=$ jsou binární predikátové symboly, $+$ je binární funkční symbol. (Upozorňujeme, že místo zápisu $=(t_1, t_2)$ a $+(x,y)$ používáme čitelnější zápis $t_1 = t_2$ a $x + y$.)

Uvědomte si, že i když funkční symbol značíme $+$, jeho interpretace může být libovolná binární operace na množině U; tedy ne nutně komutativní. Interpretací, ve které je třetí sentence pravdivá, je každá neprázdná množina spolu se symetrickou binární relací.

2.2.10 Zvláštní příklady kontradikcí neuvádíme. Kontradikce jsou přesně ty formule, jejichž negace jsou tautologie. Tak např. formule $(\forall x\, P(x) \wedge \neg(\forall x\, P(x))$ je kontradikce. Je dobré si uvědomit, že jde o „dosazení“ formule $\forall x\, P(x)$ do výrokové kontradikce $p \wedge \neg p$.

2.2.11 Splnitelné množiny sentencí.

Definice. Množina sentencí M je *splnitelná*, jestliže 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, tj. nemá model. □

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

2.3 Tautologická ekvivalence

Obdobně jako ve výrokové logice zavádíme i v predikátové logice pojem *tautologicky ekvivalence*. V predikátové logice ale pouze pro sentence, tedy formule bez volných výskytů proměnných.

2.3.1 Tautologická ekvivalence sentencí.

Definice. Řekneme, že dvě sentence φ a ψ jsou *tautologicky ekvivalentní*, jestliže mají stejné modely, tj. jsou pravdivé ve stejných interpretacích. Tento fakt značíme $\varphi \models\!\models \psi$. □

Někdy se též 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, že následující sentence jsou tautologicky ekvivalentní (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.

Definice. Je dána množina sentencí S a sentence φ. Řekneme, že φ je *sémantickým důsledkem*, též *konsekventem*, množiny S, jestliže 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 sentence φ. Jedná se tedy o obdobný pojem jako ve výrokové logice, „pouze“ místo o pravdivostním ohodnocení mluvíme o interpretaci. (Je dobré si ale uvědomit, že mezi pojmy pravdivostní ohodnocení a interpretace je podstatný rozdíl — např. to, že pro interpretace neexistuje nic obdobného pravdivostním tabulkám.)

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

   Zdůvodnění: Jestliže všechny objekty z daného universa mají vlastnost odpovídající predikátovému symbolu P, pak existuje alespoň 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 universum U interpretace nikdy není prázdná množina.)

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

   Zdůvodnění: Není těžké najít interpretaci $\langle U, [\![-]\!]\rangle$, ve které je pravdivá sentence $\exists x\, P(x)$ a sentence $\forall x\, P(x)$ pravdivá není. 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. V této interpretaci je sentence $\exists x\, P(x)$ pravdivá (např. číslo 2 je sudé), ale $\forall x\, P(x)$ pravdivá není (např. číslo 3 není sudé).

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

   Zdůvodnění: Jestliže v nějaké interpretaci existuje objekt, který má vlastnost odpovídající predikátovému symbolu P, pak z pravdivosti implikace $\forall x\, (P(x) \Rightarrow Q(x))$ tento objekt musí mít vlastnost odpovídající predikátovému Q. Pravdivost sentence $\exists x\, P(x)$ zaručuje, že stejný objekt, který má vlastnost odpovídající P, má i vlastnost odpovídající Q.

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

   Zdůvodnění: Sentence $(\forall x\, P(x)) \vee (\forall x\, Q(x))$ a sentence $(\forall x\, P(x)) \vee (\forall y\, Q(y))$ jsou stejné (liší se pouze legálním přejmenováním vázané proměnné x na y), budeme tedy pracovat s ní.

   Sentence $(\forall x\, P(x)) \vee (\forall y\, Q(y))$ je pravdivá v každé interpretaci, ve které všechny objekty mají vlastnost odpovídající P nebo všechny objekty mají vlastnost odpovídající 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. $(\forall x\, (P(x) \vee Q(x)) \not\models (\forall x\, (P(x)) \vee (\forall x\, Q(x))$.

Zdůvodnění: 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; proto sentence $(\forall x\,P(x)) \vee (\forall x\,Q(x))$ je nepravdivá. Uvedeme konkrétní interpretaci, ve které je sentence $\forall x\,(P(x) \vee Q(x))$ pravdivá, ale sentence $(\forall x\,P(x)) \vee (\forall x\,Q(x))$ nepravdivá.

Uvažujme interpretaci, kde U je množina přirozených čísel, vlastnost odpovídající P je vlastnost být sudým číslem, vlastnost odpovídající Q je vlastnost 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. $(\forall x\,P(x)) \vee (\forall y\,Q(y)) \models \forall x\,\forall y\,(P(x) \vee Q(y))$.

   Zdůvodnění: Nejprve označme

   $$\alpha = (\forall x\,P(x)) \vee (\forall y\,Q(y)) \quad \text{a} \quad \beta = \forall x\,\forall y\,(P(x) \vee Q(y)).$$

   Zdůvodnění tohoto tvrzení je trochu obtížnější. Půjdeme na to tak, že ukážeme, že není-li sentence β pravdivá, není pravdivá ani sentence α. Tím bude tvrzení z příkladu 6) 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 vlastnost $[\![P]\!]$ a d nemá vlastnost $[\![Q]\!]$. 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. $\exists x\,\forall y\,R(x, y) \models \forall y\,\exists x\,R(x, y)$.

   Zdůvodnění: 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, tj. $(c, d) \in [\![R]\!]$. 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. $\forall x\,\exists y\,R(x, y) \not\models \exists y\,\forall x\,R(x, y)$.

   Zdůvodnění: 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 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 sentence $\forall x\,\exists y\,R(x, y)$ 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; její pravdivost by znamenala, že bychom museli mít největší přirozené číslo (a to by ještě muselo být větší než ono samo) a takové číslo neexistuje.

2.4.4 Obdobně jako pro výrokovou logiku, dostáváme řadu jednoduchých pozorování.

Tvrzení. Pro množiny sentencí M, N a sentenci φ platí:

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

5. Je-li $N \subseteq M$ a platí $N \models \varphi$, pak platí i $M \models \varphi$.
6. Je-li φ konsekventem množiny sentencí $\{\alpha_1, \ldots, \alpha_k\}$ a každá sentence α_i je konsekventem množiny sentencí M, pak φ je konsekventem M.
7. Jestliže $M \models \alpha$ i $M \models \neg\alpha$ pro nějakou sentenci α, pak M je nesplnitelná množina sentencí.

□

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

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

□

Ano, oboje znamená, že sentence φ a ψ mají stejné modely.

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

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

□

Ano, oboje znamená, že neexistuje interpretace, která by byla modelem φ a nebyla modelem ψ.

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

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

□

Zdůvodnění je stejné jako ve výrokové logice (viz z **??**), pouze místo o pravdivostním ohodnocení mluvíme o interpretacích.

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. Postupujeme obdobně jako ve výrokové logice (je dobré sledovat to, co je společné, i to, co je rozdílné, s rezoluční metodou výrokové logiky).

Nejprve zavedeme literály a klauzule v predikátové logice.

2.5.1 Literál.

Definice. *Literál* je atomická formule (tzv. *pozitivní literál*), nebo negace atomické formule (tzv. *negativní literál*). *Komplementární literály* jsou dva literály, z nichž jeden pozitivní, jeden je negativní a ten negativní je negací pozitivního. □

2.5.2 Klauzule

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

Za klauzuli budeme považovat ještě formuli **F** zastupující kontradikci, kterou budeme nazývat *prázdná klauzule*. □

Ve výrokové logice jsme pro každou formuli α našli k ní tautologicky ekvivalentní množinu klauzulí S_α a to tak, že α i S_α byly pravdivé ve stejných pravdivostních ohodnoceních. Takto jednoduchá situace v predikátové logice není. Ukážeme si nejprve, jak k dané sentenci φ najít množinu klauzulí S_φ a to tak, že φ je splnitelná právě tehdy, když je splnitelná množina S_φ.

2.5.3 Převedení sentence na . Pro každou sentenci φ existuje množina klauzulí S_φ taková, že sentence φ je splnitelná právě tehdy, když je splnitelná množina S_φ. (Poznamenejme, že se jedná o obdobu CNF formule pro danou výrokovou formuli.)

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

1. Přejmenujeme proměnné sentence 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

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

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

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

   Přitom dáváme přednost prvním dvěma rovnostem. Teprve v případě, že ani první, ani druhou rovnost nelze aplikovat, používáme třetí rovnost. Uvědomte si, že třetí rovnost je opravdu tautologická ekvivalence pouze proto, že formule α neobsahuje proměnnou x (viz krok 1).

5. V případě, že formule ψ obsahuje existenční kvantifikátor, provedeme skolemizaci, která je vysvětlena a popsána v dalších odstavcích.

6. Použijeme tautologickou ekvivalenci

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

   k distribuci obecného kvantifikátoru. Dostali jsme sentenci ψ, která je konjunkcí klauzulí. Sentenci ψ nahradíme množinou jejích klauzulí — a to je hledaná množina S_φ.

2.5.5 Poznámka. Místo abychom skolemizovali až v kroku 5, mohli jsme skolemizaci použít již v kroku 4. V takovém případě bychom nemuseli přesunovat existenční kvantifikátor ve stromě formule „nahoru“ nad logickou spojku $\vee$. **Nesmíme** ale skolemizovat před krokem 3, tj. před přesunutím negace před atomické formule.

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

Skolemizací nahradíme sentenci ψ sentencí ψ' takovou, že sentence ψ je splnitelná právě tehdy, když je splnitelná sentence ψ' (obecně ale může existovat interpretace, ve které je ψ pravdivá, ale ψ' nikoli). Dříve než ukážeme obecný postup, uvedeme čtyři příklady.

2.5.7 Příklad 1. Najděme klauzuli 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'. Navíc, má-li ψ model, můžeme interpretovat konstantu a tak, abychom dostali model ψ'.

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

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

Nejprve skolemizujeme první existenční kvantifikátor, tj. $\exists x$. Tím dostaneme sentenci je $\exists y\, Q(a, y)$, kde a je nějaký nový konstantní symbol. Nyní skolemizujeme druhý existenční kvantifikátor $\forall y$ a dostáváme klauzuli $\psi' = Q(a, b)$. Opět musí platit, že b je nový konstantní symbol, který se nikde jinde neobjevuje.

Opět není obtížné nahlédnout, že sentence ψ je tautologickým důsledkem sentence ψ'. Navíc, má-li ψ model, můžeme interpretovat konstanty a, b tak, abychom dostali model ψ'.

2.5.9 Příklad 3. Najděme klauzuli ψ', která je splnitelná právě tehdy, když je splnitelná sentence $\psi = \forall x\, \exists y\, Q(x, y)$.

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

Funkčnímu symbolu f se říká *skolemizační funkční symbol*.

2.5.10 Příklad 4. Najděme klauzuli ψ', která je splnitelná právě tehdy, když je splnitelná sentence $\psi = \forall x\, \exists y\, \forall z\, \exists u\, R(x, y, z, u)$.

Skolemizovat začneme kvantifikátory od kořene syntaktického stromu ψ; tedy nejprve odstraníme $\exists y$. Protože před tímto kvantifikátorem je jeden obecný kvantifikátor $\forall x$, nahradíme $\exists y$ novým unárním funkčním symbolem $f(x)$. Dostáváme tedy sentenci $\forall x\, \forall z\, \exists v\, R(x, f(x), x, v$. Nyní při skolemizaci $\exists v$ od tohoto kvantifikátoru do kořene syntaktického stromu máme dva universální kvantifikátory; ano, $\forall x$ a $\forall z$. Zavedeme tedy nový binární funkční symbol $g(x, z)$. Výsledná sentence ψ' je rovna $\psi' = \forall x\, \forall z\, R(x, f(x), z, g(x, z))$, kde f je nový unární funkční symbol a g je nový binární funkční symbol. Nyní opět platí, že sentence ψ je splnitelná právě tehdy, když je splnitelná formule ψ'.

2.5.11 Obecný postup (bod 5 z postupu 2.5.4).

5. Obsahuje-li formule existenční kvantifikátor, nahradíme každou uzavřenou podformuli tvaru $\forall x_1 \ldots \forall x_n\, \exists y\, \alpha(x_1, \ldots, x_n, y)$ formulí $\forall x_1 \ldots \forall x_n\, \alpha(x_1, \ldots, x_n, f(x_1, \ldots, x_n))$, 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 6 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.

2.5.12 Resolventy klauzulí. Ve výrokové logice jsme resolventy vytvářeli tak, že jsme si vždy vzali dvě klauzule, které obsahovaly dvojici komplementárních literálů, a výsledná resolventa byla disjunkcí všech ostatních literálů z obou klauzulí. Situace v predikátové logice je složitější. Dříve než zadefinujeme pojem resolventy dvou klauzulí v predikátové logice, uvedeme postup vytváření resolvent na příkladech.

2.5.13 Příklad 1. Najděme resolventu klauzulí

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

kde P a R jsou unární predikátové symboly a Q je binární predikátový symbol, x, y jsou proměnné.

Klauzule 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. Obdobně jako ve výrokové logice resolventou klauzulí K_1 a K_2 je klauzule $K = \forall x\, \forall y\, (P(x) \vee R(y))$. (Uvědomte si, že jsme v klauzulích K_1 a K_2 vynechali dvojici komplementárních literálů a klauzule K se skládá ze zbylých literálů.)

2.5.14 Příklad 2. Najděme resolventu klauzulí

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

kde P, Q jsou unární predikátové symboly a a, b jsou konstanty.

Klauzule 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 klauzule

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

a jejich resolventa je klauzule $P(a) \vee R(b)$. (Uvědomte si, že platí $K_1 \models K_1'$.)

2.5.15 Příklad 3. Najděme resolventu klauzulí

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

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 klauzule, dostaneme klauzule K_1' a K_2', jejichž resolventa bude i resolventou klauzulí 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 resolventa je $K = \forall v\, (\neg P(g(v),a) \vee R(v,a))$. (Uvědomte si, že platí $K_1 \models K_1'$ a $K_2 \models K_2'$.)

2.5.16 Poznámka. Ne vždy resolventa existuje. Resolventa klauzulí

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

neexistuje (zde P je unární predikátový symbol, Q a R jsou binární predikátové symboly, f je unární funkční symbol a a je konstanta). Případy, kdy existuje či neexistuje vhodná substituce, řeší unifikační algoritmus. V případě, že substituce existuje, unifikační algoritmus najde nejobecnější substituci (tj. provedeme pouze nutné substituce a žádné jiné).

2.5.17 Unifikační algoritmus.

Vstup: Dva pozitivní 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.

2.5.18 Definice resolventy. Máme dvě klauzule K_1 a K_2. Předpokládejme, že K_1 obsahuje literál L_1 a K_2 obsahuje literál L_2 pro něž existuje substituce θ taková, že $\theta(L_1)$ a $\theta(L_2)$ tvoří dvojici komplementárních literálů.

Pak resolventa klauzulí K_1 a K_2 je klauzule K, která je určena všemi literály obsaženými v $\theta(K_1) \setminus \theta(L_1)$ a $\theta(K_2) \setminus \theta(L_2)$; (tj. literály doplněné o obecné kvantifikátory všech proměnných, které se nacházejí v $\theta(K_1) \setminus \theta(L_1)$ a $\theta(K_2) \setminus \theta(L_2)$). □

Neformálně řečeno, resolventu K dostaneme tak, že v „těle“ klauzule necháme všechny literály z v $\theta(K_1) \setminus \theta(L_1)$ a v $\theta(K_2) \setminus \theta(L_2)$ a na začátek doplníme obecné kvantifikátory se všemi proměnnými, které v klauzuli zbyly.

2.5.19 Věta. Jsou dány dvě klauzule K_1 a K_2. Jestliže existuje jejich resolventa K, pak platí: kdykoli jsou pravdivé klauzule K_1 a K_2 v některé interpretaci, pak v této interpretaci je pravdivá i jejich resolventa K. □

Jinými slovy, množiny klauzulí $\{K_1, K_2\}$ a $\{K_1, K_2, K\}$ mají stejné modely.

2.5.20 Rezoluční princip. Je obdobný jako rezoluční princip ve výrokové logice:
Je dána množina klauzulí S. Označme

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

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

Věta. Množina klauzulí S je splnitelná právě tehdy, když $R^\star(S)$ neobsahuje prázdnou klauzuli **F**. □

2.5.21 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 klauzální tvar; množinu klauzulí, kterou dostaneme, označme M.

Jestliže při vytváření množiny $R^\star(M)$ získáme prázdnou klauzuli, pak množina $S \cup \{\neg\alpha\}$ je nesplnitelná a tvrzení $S \models \alpha$ je pravdivé; jestliže jsme vytvořili celou množinu $R^\star(M)$ a prázdná klauzule do ní nepatří, pak $S \cup \{\neg\alpha\}$ je splnitelná a tvrzení $S \models \alpha$ není pravdivé.

2.5.22 Příklad. Je dána množina tří 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 $\varphi = R(g(b))$, (zde 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 $S \models \varphi$.

Řešení. Nejprve vytvoříme množinu $S \cup \{\neg\varphi\}$, ta je rovna

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

Nyní převedeme sentence na klauzální tvar. Přitom přejmenujeme proměnné tak, aby v celé množině každý kvantifikátor vázal jinou proměnnou. (Poslední tři sentence už jsou klauzule.) Dostaneme množinu klauzulí

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

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

Nejprve utvoříme resolventu klauzulí K_1 a K_3. Abychom resolventu mohli utvořit, za proměnnou x dosadíme term $f(b)$. Dostaneme klauzuli

$$K_1' = \forall y\, (\neg P(f(b)) \vee \neg Q(f(b), y) \vee R(y)).$$

Klauzule 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 resolventa je klauzule

$$K_5 = \forall y\, (\neg Q(f(b), y) \vee R(y)).$$

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

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

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

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

2.5.23 Příklad. Je dána sentence

$$\varphi = (\exists x\, (M(x) \wedge \neg P(x)) \wedge \forall y\, (M(y) \Rightarrow S(y))) \Rightarrow \exists z\, (S(z) \wedge \neg P(z)).$$

Rezoluční metodou rozhodněte, zda se jedná o tautologii.

2.5.24 Řešení. φ je tautologie právě tehdy, když je sentence $\neg\varphi$ nesplnitelná. K sentenci $\neg\varphi$ najdeme množinu klauzulí S, která je splnitelná právě tehdy, když je splnitelná sentence $\neg\varphi$.

Sentenci $\neg\varphi$ převedeme na klauzální tvar a dostaneme množinu S, kde

$$S = \{M(a), \neg P(a), \forall y\,(\neg M(y) \vee S(y)), \forall z\,(\neg S(z) \vee P(z))\}.$$

Resolventa klauzulí $K_1 = M(a)$ a $K_3 = \forall y\,(\neg M(y) \vee S(y))$ je klauzule $S(a)$. Dále resolventou klauzulí $K_2 = \neg P(a)$ a $K_4 = \forall z\,(\neg S(z) \vee P(z)$ je klauzule $\neg S(a)$. Nyní resolventa klauzulí $S(a)$ a $\neg S(a)$ je prázdná klauzule.

Protože jsme dostali prázdnou klauzuli ($\mathbf{F} \in R^2(S)$), je množina S nesplnitelná a φ je tautologie. □

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

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

Řešení. Abychom ověřili, že $\varphi \models\!\models \psi$, musíme ověřit, zda platí

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

Ad 1. Utvoříme množinu $\{\varphi, \neg\psi\}$. Této množině odpovídá množina sentencí (přejmenovali jsme proměnné v sentenci $\neg\psi$)

$$\{\exists x\,\forall y\,Q(x,y), \exists t\,\forall z\,\neg Q(z,t)\}.$$

Skolemizujeme a dostaneme množinu klauzulí

$$S = \{\forall y\,Q(a,y), \forall z\,\neg Q(z,b)\}.$$

Substituci vytvoříme takto: za proměnnou y dosadíme konstantu b a za proměnnou z dosadíme konstantu a. Tím dostaneme dvojici klauzulí $K_1 = Q(a,b)$ a $K_2 = \neg Q(a,b)$; jejich resolventa je prázdná klauzule. Proto $\varphi \models \psi$ je pravdivé.

Ad 2. Obdobně utvoříme množinu $\{\psi, \neg\varphi\}$. Této množině odpovídá množina sentencí

$$\{\forall y\,\exists x\,Q(x,y), \forall t\,\exists z\,\neg Q(t,z)\}.$$

Skolemizujeme a dostaneme množinu klauzulí

$$S = \{\forall y\,Q(f(y),y), \forall t\,\neg Q(t,g(t))\}.$$

Nyní sice můžeme za proměnnou t dosadit term $f(y)$, ale dostáváme dvě klauzule $K_1 = \forall y\,Q(f(y),y)$ a $K_2 = \forall y\,\neg Q(f(y),g(f(y)))$, jejichž resolventa neexistuje (za y nemůžeme dosadit term $f(g(y))$, protože term $f(g(y))$ již proměnnou y obsahuje). Proto $\psi \models \varphi$ není pravdivé a pravdivé proto není ani $\varphi \models\!\models \psi$. □

2.5.27 Teorie. *Axiomatická teorie* $\mathcal{T}$ v predikátové logice je dána volbou jazyka predikátové logiky a množinou sentencí T. O prvcích množiny T mluvíme jako o *axiomech* teorie $\mathcal{T}$. *Model teorie* $\mathcal{T}$ je každá interpretace, ve které jsou všechny axiomy z T pravdivé.

Příklad: Jazyk predikátové logiky se skládá z jednoho binárního predikátového symbolu R a axiomy jsou následující tři sentence:

$$\forall x\, R(x,x);\ \forall x\, \forall y (R(x,y) \Rightarrow R(y,x));$$

$$\forall x\, \forall y\, \forall z\, ((R(x,y) \wedge R(y,z)) \Rightarrow R(x,z)).$$

Pak modelem této teorie je každá neprázdná množina U, na které je predikát R interpretován jako binární relace, která je reflexivní, symetrická a tranzitivní; tudíž jedná se o množinu a na ní danou relaci ekvivalence R.

2.5.28 Teorie s rovností je teorie, kde kromě množiny predikátových symbolů Pred máme ještě binární predikátový symbol $=$, který má vlastnosti rovnost a vždy se interpretuje jako rovnost objektů „světa“, tj. universa U. (Poznamenejme, že v případě teorie s rovností už může být množina predikátových symbolů prázdná.)

2.5.29 Použití rezoluční metody v programovacím jazyce Prolog. Převzato z přednášky prof. Štěpánkové (PROLOG, PROgramování v LOGice.)

Program v Prologu je teorie v predikátové logice. Přesněji: *Logický program* je množina klauzulí s právě jedním pozitivním literálem. (Klauzulím, které obsahují právě jeden pozitivní literál, se také říká *Hornovy* klauzule.)

Logický program tvoří

- *fakta* — elementární pravdivá tvrzení vyjádřené jako jeden atom.
- *Pravidla – podmíněná tvrzení* „jestliže jsou splněny současně všechny podmínky tvořící tělo pravidla, pak platí i hlava p“ (hlava je vždy popsána jediným atomem).

Program se spouští pomocí dotazu.

2.5.30 Poznámka. V Prologu se pravidla většinou píší ve formě implikace, ale „naopak“, tj. nejprve se napíše hlava pravidla p, pak symbol $:-$ (který má význam implikace zprava do leva) a pak teprve předpoklady implikace.

2.5.31 Příklad. Jazyk predikátové logiky se skládá z

- $\mathsf{Pred} = \{\texttt{rodic}, \texttt{zena}, \texttt{muz}, \texttt{matka}\}$, kde `rodic` a `matka` jsou binární predikátové symboly a `zena` a `muz` jsou unární predikátové symboly.
- $\mathsf{Kons} = \{\texttt{boleslav}, \texttt{drahomira}, \texttt{ludmila}, \texttt{spytihnev}, \texttt{vaclav}, \texttt{vratislav}\}$.

Program:

- Fakta:

$$\texttt{rodic}(\texttt{ludmila}, \texttt{spytihnev}),\quad \texttt{rodic}(\texttt{ludmila}, \texttt{vratislav}),$$
$$\texttt{rodic}(\texttt{drahomira}, \texttt{vaclav}),\quad \texttt{rodic}(\texttt{drahomira}, \texttt{boleslav}),$$
$$\texttt{rodic}(\texttt{vratislav}, \texttt{vaclav}),\quad \texttt{rodic}(\texttt{vratislav}, \texttt{boleslav}).$$
$$\texttt{zena}(\texttt{ludmila}),\ \texttt{zena}(\texttt{drahomira}),\ \texttt{muz}(\texttt{vaclav}),\ \texttt{muz}(\texttt{boleslav}).$$

- Pravidla:

$$\texttt{matka}(x,y)) :- \forall x\, \forall y\, ((\texttt{zena}(x) \wedge \texttt{rodic}(x,y))$$

  Toto pravidlo je ekvivalentní klauzuli

$$\forall x\, \forall y\, (\neg\texttt{zena}(x) \vee \neg\texttt{rodic}(x,y) \vee \texttt{matka}(x,y)).$$

Dotaz může být $\exists z\, \mathtt{matka}(z, \mathtt{vaclav})$?

K programu přidáme negaci dotazu, tedy fakt

$$\neg\exists z\, \mathtt{matka}(z, \mathtt{vaclav}) \models\!\models \forall z\, \neg\mathtt{matka}(z, \mathtt{vaclav})$$

a použijeme rezoluční metodu ke zjištění, zda program spolu s negací dotazu tvoří nesplnitelnou množinu (pak odpověď je ano), nebo splnitelnou množinu (pak odpověď je ne).

Není těžké zjistit, že

- Resolventa klauzulí

  $$\forall z\, \neg\mathtt{matka}(z, \mathtt{vaclav}) \text{ a } \forall x\, \forall y\, (\neg\mathtt{zena}(x) \vee \neg\mathtt{rodic}(x, y) \vee \mathtt{matka}(x, y)))$$

  (druhá klauzule je pravidlo programu) je klauzule

  $$\forall x\, (\neg\mathtt{zena}(x) \vee \neg\mathtt{rodic}(x, \mathtt{vaclav})).$$

- Rezolventa klauzulí

  $$\forall x\, (\neg\mathtt{zena}(x) \vee \neg\mathtt{rodic}(x, \mathtt{vaclav})) \text{ a } \mathtt{zena}(\mathtt{drahomira})$$

  (druhá klauzule je fakt programu) je klauzule

  $$\neg\mathtt{rodic}(\mathtt{drahomira}, \mathtt{vaclav}).$$

- Rezolventa klauzulí

  $$\neg\mathtt{rodic}(\mathtt{drahomira}, \mathtt{vaclav}) \text{ a } \mathtt{rodic}(\mathtt{drahomira}, \mathtt{vaclav})$$

  (druhá klauzule je fakt programu) je prázdná klauzule **F**.

Zjistili jsme, že program spolu s negaci dotazu tvoří nesplnitelnou množinu klauzulí, a proto odpověď Prologu je **ano**.

2.6 Přirozená dedukce v predikátové logice

Upozorňujeme čtenáře, že zde uvádíme jen velmi „letmé“ přiblížení přirozené dedukce. Jedná se o rozšíření přirozené dedukce z výrokové logiky o odvozovací pravidla pro kvantifikátory $\forall$ a $\exists$.

Pravidla pro kvantifikátory jsou v podstatě následující:

E-pravidlo pro $\forall$: $\dfrac{\forall x\, \varphi(x)}{\varphi(t)}$ I-pravidlo pro $\forall$: $\dfrac{\varphi(x)}{\forall x\, \varphi(x)}$

I-pravidlo pro $\exists$: $\dfrac{\varphi(t)}{\exists x\, \varphi(x)}$ E-pravidlo pro $\exists$: $\dfrac{\exists x\, \varphi(x) \quad \boxed{\begin{array}{c}\varphi(y)\\ \psi\end{array}}}{\psi}$

Aby tato čtyři odvozovací pravidla byla opravdu „správně“, tj. aby závěry pravidel byly sémantickým důsledkem předpokladů pravidla, musí být ještě splněno:

- V E-pravidle pro $\forall$ a I-pravidle pro $\exists$ musí být term t „volný pro x“, tj. musí obsahovat pouze proměnné, které jsou volné ve formuli $\phi[x := t]$, kde jsme za proměnnou x dosadili term t.
- V I-pravidle pro $\forall$ a E-pravidle pro $\exists$ nesmí být proměnná x volná v žádném aktivním předpokladu.

I v predikátové logice definujeme odvození a logický důsledek stejně jako v 1.9.2 a 1.9.4. A co je důležité, i v predikátové logice, podobně jako ve výrokové logice, platí věta o úplnosti 1.9.6.

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

- každá sentence φ_i je buď předpoklad (tj. $\varphi_i \in S$), nebo je pomocný předpoklad, nebo vznikla z předcházejících sentencí pomocí některého odvozovacího pravidla;
- všechny pomocné předpoklady jsou již pasivní.

2.6.2 Logický důsledek. Sentence 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 $S \vdash \varphi$.

2.6.3 Věta o úplnosti. Pro každou množinu sentencí S a sentenci φ platí

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

Kapitola 3

Grafy

3.1 Orientované a neorientované grafy

3.1.1 Orientovaný graf. Definice. *Orientovaný graf* G je trojice (V, E, ε), kde V je neprázdná 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*. □

Další pojmy spojené s orientovanými grafy. 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 Neorientovaný graf. Definice. *Neorientovaný graf* je trojice $G = (V, E, \varepsilon)$, kde V je neprázdná konečná množina *vrcholů* (též zvaných *uzlů*), E je konečná množina jmen *hran* a ε je přiřazení, které každé hraně $e \in E$ přiřazuje množinu $\{u, v\}$ (kde $u, v \in V$ jsou vrcholy) a nazývá se *vztah incidence*. □

Další pojmy spojené s neorientovanými grafy. 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. Definice. Jestliže v orientovaném nebo neorientovaném grafu existují dvě různé hrany e_1, e_2, pro které platí, že $\varepsilon(e_1) = \varepsilon(e_2)$, říkáme, že hrany e_1, e_2 jsou *paralelní*. □

Uvědomte si, že pro orientované grafy to znamená, že počáteční vrcholy i koncové vrcholy hran e_1, e_2 jsou stejné, zatímco pro neorientované grafy to pouze znamená, že krajní vrcholy hran e_1, e_2 jsou stejné.

3.1.4 Prostý graf. Definice. Graf (orientovaný nebo neorientovaný) se nazývá *prostý graf*, nemá-li paralelní hrany. □

3.1.5 Poznámka. V některé literatuře se termínem *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.

3.1.6 Stupně vrcholů. Definice. 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 (které do vrcholu v vstupují), 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 (které z vrcholu v vystupují), tj.

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

Stupeň vrcholu v, značíme jej $d(v)$, je roven počtu hran, které jsou incidentní s vrcholem v, tj.

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

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. □

Všimněte si, že v definici stupně vrcholu je smyčka započítána dvakrát i v případě orientovaných grafů.

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

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

kde $|E|$ značí počet hran grafu G. □

3.1.8 Důsledek. Každý graf má sudý počet vrcholů lichého stupně. □

3.1.9 Zadávání grafu. Orientovaný i neorientovaný graf můžeme zadat seznamem jeho vrcholů, a pro každý vrchol seznamem hran, které v něm začínají a/nebo v něm končí. Graf též můžeme zadat maticí sousednosti nebo maticí incidence.

3.1.10 Matice sousednosti. Je dán graf $G = (V, E, \varepsilon)$, kde jsme očíslovali vrcholy, tj. $V = \{v_1, v_2, \ldots, v_n\}$. Čtvercová matice $\mathbf{M} = (m(i,j))$ řádu n se nazývá *matice sousednosti* grafu G, splňuje-li:

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

□

Poznamenejme, že pro neorientovaný graf je matice sousednosti symetrická.

3.1.11 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, splňuje-li:

- 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 orientovaného grafu má v každém sloupci jednu 1 a jednu -1, pro neorientované grafy má v každém sloupci dvě 1.

3.1.12 Porovnávání grafů. Definice. 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 pro orientované grafy

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

a pro neorientované grafy

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

□

3.1.13 Sled v orientovaném grafu. Definice. 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. □

Poznámka: Neorientovaný sled se od orientovaného sledu liší tím, že můžeme „jít proti směru“ hrany.

3.1.14 Sled v neorientovaném grafu. Definice. Je dán neorientovaný graf. Pak *neorientovaný sled* je 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 $i = 1, 2, \ldots, k-1$. □

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

3.1.16 Uzavřené sledy. Máme dán orientovaný nebo neorientovaný sled $v_1, e_1, \ldots, e_{k-1}, v_k$. 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.

Definice. Orientovaný (neorientovaný) sled se nazývá *uzavřený*, jestliže $k > 1$ a $v_1 = v_k$. V opačném případě mluvíme o *otevřeném* sledu. □

Tedy triviální sled nepovažujeme za uzavřený.

3.1.17 Tah a cesta. Definice. 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*. □

Poznámka. 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.

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

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

3.1.19 Souvislý graf. Definice. Řekneme, že (orientovaný nebo neorientovaný) graf je *souvislý*, jestliže pro každé dva vrcholy u, v grafu existuje neorientovaná cesta z u do v. □

Poznámka. 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 (stačí cestu „číst pozpátku“).

3.2 Stromy

3.2.1 Definice. Orientovaný nebo neorientovaný graf se nazývá *strom*, je-li souvislý a neobsahuje-li kružnici. □

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

Kdyby totiž v nějakém grafu měl každý vrchol stupeň alespoň 2, pak by v grafu existovala kružnice.

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

Zdůvodnění: Větu je možné dokázat indukcí podle počtu vrcholů n. Tvrzení zřejmě platí pro stromy o 1 nebo 2 vrcholech.

Předpokládejme, že tvrzení věty platí pro všechny stromy s n vrcholy. Vezměme libovolný strom G s $n + 1$ vrcholy. Označme G' strom, který dostaneme tak, že z G odstraníme vrchol stupně 1. Graf G' je opět strom a má n vrcholů, tedy obsahuje přesně $n - 1$ hran. Proto G má $n - 1 + 1 = n$ hran.

3.2.4 Důsledek. V každém stromu s alespoň dvěma vrcholy existují (alespoň) dva vrcholy stupně 1. □

Zdůvodnění: Má-li souvislý graf n vrcholů ($n > 1$) a $n - 1$ hran, nemůže mít jen jeden vrchol stupně 1.

Také není těžké nahlédnout, že vezmeme-li v grafu, který nemá kružnice, cestu o největším počtu hran, pak oba koncové vrcholy této cesty musí mít stupeň 1.

3.2.5 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.

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

3.2.6 Tvrzení. 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 souvislý.

□

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

3.2.7 Podgrafy. Definice. 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', taková, že trojice $G' = (V', E', \varepsilon')$ je také graf.

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

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

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.

Faktor je podgraf, kde jsme ponechali všechny vrcholy. Podgraf indukovaný množinou vrcholů A je podgraf s množinou vrcholů A obsahující všechny hrany grafu G, které obsahovat může.

3.2.8 Komponenty souvislosti. Definice. 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 množině A libovolný vrchol, podgraf indukovaný touto větší množinou už souvislý nebude.

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

3.3 Minimální kostra

3.3.1 Kostra grafu. Definice. Je dán graf G. Faktor grafu G, který je stromem, se nazývá *kostra* grafu G. □

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

3.3.3 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).

Definice. *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). □

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

3.3.5 Kruskalův algoritmus. Jedná se o modifikaci postupu 3.3.8:

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).

□

Uvědomte si, že v tomto případě vybíráme a přidáváme vždy hranu, která je nejlevnější pro **obě** množiny S a S'.

3.3.6 Primův algoritmus. Jedná se o modifikaci postupu 3.3.8:

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.

□

Uvědomte si, že přestože v Primově algoritmu „neudržujeme“ systém komponent grafu (V, L), komponenty známe. Jsou to: komponenta S obsahující na začátku vybraný vrchol v, ostatní komponenty jsou jednoprvkové. Hrana, kterou přidáváme, je vždy nejlevnější hrana, která vede ven z komponenty S.

3.3.7 Příklad. Je dán neorientovaný graf $G = (V, E)$ s množinou vrcholů $V = \{1, \ldots, 7\}$ kde následující matice udává ceny hran (to znamená, že na místě (i, j) máme buď $c(\{i, j\})$, když $\{i, j\} \in E$, nebo "$-$", jestliže $\{i, j\} \notin E$). Najdeme minimální kostru grafu (G, c) nejprve Kruskalovým algoritmem

$$\begin{pmatrix} - & 6 & 9 & - & - & - & 9 \\ 6 & - & 2 & 1 & 3 & - & - \\ 9 & 2 & - & 1 & - & - & 15 \\ - & 1 & 1 & - & 10 & 13 & 3 \\ - & 3 & - & 10 & - & 10 & 1 \\ - & - & - & 13 & 10 & - & 15 \\ 9 & - & 15 & 3 & 1 & 15 & - \end{pmatrix}$$

Řešení. Nejprve uspořádáme hrany podle ceny neklesajícím způsobem (v závorce je vždy cena odpovídající hrany)

$$e_1 = \{2,4\}(1), e_2 = \{3,4\}(1), e_3 = \{5,7\}(1), e_4 = \{2,3\}(2), e_5 = \{2,5\}(3), e_6 = \{4,7\}(3),$$

$$e_7 = \{1,2\}(6), e_8 = \{1,3\}(9), e_9 = \{1,7\}(9), e_{10} = \{4,5\}(10), e_{11} = \{5,6\}(10), e_{12} = \{4,6\}(13),$$

$$e_{13} = \{3,7\}(15), e_{14} = \{6,7\}(15).$$

Položíme $T = \emptyset$, $\mathcal{S} = \{\{i\} \mid i \in V\}$.

Nyní procházíme hrany v tomto pořadí a hranu e_i přidáme do L právě tehdy, když neuzavře kružnici. Přitom upravíme komponenty souvislosti grafu (V, T).

1. e_1 neuzavře kružnici, proto $T := \{\{2,4\}\}$, komponenty souvislost v $\mathcal{S}$ jsou $\{1\}$, $\{2,4\}$, $\{3\}$, $\{5\}$, $\{6\}$ a $\{7\}$.
2. e_2 neuzavře kružnici, proto $T := \{\{2,4\},\{3,4\}\}$, komponenty souvislost v $\mathcal{S}$ jsou $\{1\}$, $\{2,3,4\}$, $\{5\}$, $\{6\}$ a $\{7\}$.
3. e_3 neuzavře kružnici, proto $T := \{\{2,4\},\{3,4\},\{5,7\}\}$, komponenty souvislost v $\mathcal{S}$ jsou $\{1\}$, $\{2,3,4\}$, $\{5,7\}$ a $\{6\}$.
4. e_4 uzavře kružnici tvořenou e_1, e_2 a e_4, proto T i $\mathcal{S}$ jsou stejné jako v 3.
5. e_5 neuzavře kružnici, proto $T := \{\{2,4\},\{3,4\},\{5,7\},\{2,5\}\}$, komponenty souvislost v $\mathcal{S}$ jsou $\{1\}$, $\{2,3,4,5,7\}$ a $\{6\}$.
6. e_6 uzavře kružnici tvořenou e_1, e_5, e_3 a e_6, proto T i $\mathcal{S}$ jsou stejné jako v 5.
7. e_7 neuzavře kružnici, proto $T := \{\{2,4\},\{3,4\},\{5,7\},\{2,5\},\{1,2\}\}$, komponenty souvislost v $\mathcal{S}$ jsou $\{1,2,3,4,5,7\}$ a $\{6\}$.
8. e_8 uzavře kružnici tvořenou e_1, e_2, e_7 a e_8, proto T i $\mathcal{S}$ jsou stejné jako v 7.
9. e_9 uzavře kružnici tvořenou e_3, e_5, e_7 a e_9, proto T i $\mathcal{S}$ jsou stejné jako v 7.
10. e_{10} uzavře kružnici tvořenou e_1, e_5 a e_{10}, proto T i $\mathcal{S}$ jsou stejné jako v 7..
11. e_{11} neuzavře kružnici, proto $T := \{\{2,4\},\{3,4\},\{5,7\},\{2,5\},\{1,2\},\{5,6\}\}$, v $\mathcal{S}$ je jediná komponenta souvislosti a to $\{1,2,3,4,5,6,7\}$.

Protože T obsahuje 6 hran (a $\mathcal{S}$ je jednoprvková, algoritmus končí a

$$T = \{\{2,4\},\{3,4\},\{5,7\},\{2,5\},\{1,2\},\{5,6\}\}$$

je množina hran minimální kostry grafu G. Cena konstry L je

$$c(T) = 1+1+1+3+6+10 = 22.$$

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

1. Na začátku položíme $L = \emptyset$. $\mathcal{S}$ je množina 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. Výsledkem je množina hran L.

3.3.9 Tvrzení. Obecný postup 3.3.8 skončí po konečně mnoha krocích a výsledkem je některá minimální kostra. □

3.4 Kořenové stromy

3.4.1 Kořen. Definice. Je dán orientovaný graf $G = (V, E, \varepsilon)$. 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.

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

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

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

3.4.3 Kořenový strom. Definice. 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.

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

Zdůvodnění. Kdyby nějaký strom měl dva kořeny, řekněme r_1 a r_2, pak by existovala orientovaná cesta z r_1 do r_2, a také orientovaná cesta z r_2 do r_1. Spojením těchto dvou cest bychom dostali uzavřený orientovaný sled, a ten vždy obsahuje cyklus, tedy i kružnici a to strom obsahovat nemůže.

3.4.5 Následník, předchůdce a list. Definice. 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*. □

3.4.6 Hladiny a výška kořenového stromu. Definice. Je dán kořenový strom $G = (V, E)$ s kořenem r. Řekneme, že vrchol v *leží v hladině* k, jestliže orientovaná cesta z r do v má přesně k hran.

Výška kořenového stromu je největší k takové, že k-tá hladina je neprázdná. □

Víme, že pro každý vrchol v v kořenovém stromě existuje právě jedna orientovaná cesta z kořene r do vrcholu v. Proto jsou hladiny kořenového stromu korektně definované.

Výšku kořenového stromu jsme také mohli definovat jako počet hran v nejdelší orientované cestě (ta musí vést z kořene do některého z listů).

3.4.7 Podstrom určený vrcholem. Definice. Je dán kořenový strom G. *Podstrom určený vrcholem* v je podgraf G indukovaný množinou všech vrcholů, které jsou orientovaně dostupné z vrcholu v. □

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

3.4.8 Binární kořenové stromy. Definice. 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.

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

Halda je stromová datová struktura s touto vlastností: Je-li vrchol v orientovaně dostupný z vrcholu x, pak číselná hodnota ve vrcholu v je větší nebo rovna číselné hodnotě ve vrcholu x. Navíc se jedná o úplný binární strom; tj. každý vrchol s výjimkou listů a vrcholů v předposlední hladině má vždy dva následníky, a jestliže v předposlední hladině má vrchol pouze jeden následník, pak je to levý následník a všechny vrcholy „vpravo“ od něho jsou již listy.

Z definice haldy je zřejmé, že nejmenší hodnotu má kořen haldy, proto je nalezení minima velmi jednoduché. Musí se však vyřešit dvě úlohy — a to odstranění kořene a vložení prvku do haldy. Obě tyto operace je možné provést v čase úměrném $\log_2 n$, kde n je počet prvků, které má halda.

Velkou výhodou haldy je to, že ji v počítači nemusíme držet jako stromovou strukturu, ale můžeme se v haldě „pohybovat“ pomocí násobení dvěma a celočíselného dělení dvěma.

3.5 Acyklické grafy

3.5.1 Definice. Orientovaný graf se nazývá *acyklický*, jestliže neobsahuje žádný cyklus. □

3.5.2 Topologické očíslování vrcholů. Definice. 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.

3.5.3 Topologické očíslování hran. Definice. 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'.

3.5.4 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.

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

Myšlenka důkazu. Kdyby každý vrchol nějakého grafu měl vstupní stupeň alespoň 2, pak bychom (podobnou úvahou jako v důkazu faktu, že každý strom o $n > 1$ vrcholech má aspoň jeden vrchol stupně 1) dostali existenci cyklu.

3.5.6 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.

□

Nástin zdůvodnění. Není těžké ukázat, že má-li graf topologické očíslování vrcholů, má i topologické očíslování hran – stačí hrany vypisovat takto: Nejprve vypíšeme hrany začínající v prvním vrcholu (topologického očíslování), pak v druhém vrcholu, atd. (Uvědomte si, že z posledního vrcholu topologického očíslování vrcholů už žádná hrana nevede.)

Obdobně se ukáže i opačná implikace; totiž, že graf, který má topologické očíslování hran, má i topologické očíslování vrcholů. Vypisujeme počáteční vrcholy hran v topologickém očíslování hran a to vždy první výskyt vrcholu. Na konci nám zůstanou některé vrcholy (aspoň jeden) a ty pak vypíšeme na konec v libovolném pořadí.

Také je zřejmé, že cyklus není možné topologicky uspořádat. Tudíž, má-li graf cyklus, nemá topologické očíslování (ani vrcholů, ani hran). Fakt, že acyklický graf má topologické očíslování vrcholů ukážeme následujícím algoritmem.

3.5.7 Postup na nalezení topologického očíslování vrcholů. Je dán prostý orientovaný graf $G = (V, E)$. Úkolem je je najít některé topologické očíslování vrcholů G.

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.

3.5.8 Věta. Postup 3.5.7 skončí po konečně mnoha krocích a po skončení je posloupnost $v_1, \ldots, v_n$ topologické očíslování vrcholů grafu G. □

3.5.9 Poznámka. Kdybychom chtěli získat topologické očíslování hran, stačilo by vypisovat hrany do posloupnosti v pořadí, jak je zpracováváme v kroku 3b).

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

1. Pro každou hranu e s počátečním vrcholem $PV(e) \in K$ platí $KV(e) \notin K$.
2. Pro každý vrchol v, který neleží v K, existuje hrana e s $PV(e) = v$ a $KV(e) \in K$.

□

Podmínka 1 vlastně říká, že neexistuje hrana, která by vedla z množiny K do sebe. Podmínka 2 se dá formulovat: z každého vrcholu, který leží mimo K, se můžeme dostat po hraně zpět do K.

3.5.11 Tvrzení. V každém acyklickém grafu existuje jádro a je určeno jednoznačně. □

Důkaz. Jádro je možné sestrojit z topologického očíslování vrcholů a to následujícím způsobem: Předpokládejme, že

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

je topologické očíslování vrcholů grafu G. Do jádra K vložíme poslední vrchol $v := v_n$ a z posloupnosti vyškrtáme všechny vrcholy, ze kterých vede hrana do zařazeného vrcholu v. Jestliže ještě nebyly vyškrtány všechny vrcholy, vezmeme nevyškrtnutý vrchol s největším indexem v_i zařadíme ho do jádra K a opět vyškrtneme ze zbylé posloupnosti všechny vrcholy, z nichž vede hrana do v_i. Postup opakujeme, dokud není posloupnost prázdná.

Není těžké nahlédnout, že postup je správný; stačí si uvědomit, že vrchol z něhož nevychází žádná hrana, musí ležet v jádře.

Předpokládejme, že v některém acyklickém grafu by existovala dvě různá jádra K_1 a K_2. Protože jsou různá, je množina $(K_1 \setminus K_2) \cup (K_2 \setminus K_1)$ neprázdná.

Uvažujme topologické očíslování vrcholů acyklického grafu G

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

Z množiny $(K_1 \setminus K_2) \cup (K_2 \setminus K_1)$ vybereme vrchol s největším pořadovým číslem v_r. Předpokládejme bez újmy na obecnosti, že $v_r \in K_1$ a $v_r \notin K_2$. Protože $v_r \notin K_2$, existuje hrana e s $PV(e) = v_r$ a $KV(e) = v_s$. Musí platit $s > r$ a $v_s \notin K_1$ (v opačném případě by K_1 nebylo jádro grafu). A to je spor s volbou vrcholu v_r.

3.6 Silná souvislost

3.6.1 Silně souvislé grafy. Definice. Řekneme, že orientovaný graf G je *silně souvislý*, jestliže pro každou dvojici vrcholů u, v existuje orientovaná cesta z vrcholu u do vrcholu v a orientovaná cesta z vrcholu v do vrcholu u. □

3.6.2 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.

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

Zdůvodnění. Předpokládejme, že graf G je silně souvislý a vyberme jeho libovolnou hranu e s $PV(e) = x$, $KV(e) = y$. Protože je graf silně souvislý, existuje orientovaná cesta z vrcholu y do vrcholu x. Přidáme-li k této cestě hranu e, dostaneme cyklus, který e obsahuje.

Předpokládejme, že graf G je souvislý a každá jeho hrana je obsažena v některém cyklu. Vyberme libovolně dva vrcholu u, v v grafu. Protože je graf souvislý, existuje neorientovaná cesta z vrcholu u do vrcholu v. Každou hranu v této cestě, kterou jde cesta „proti směru hrany“, nahraďme částí cyklu, který tuto hranu obsahuje. Tím dostaneme orientovaný sled z u do v, a ten jistě obsahuje orientovanou cestu. Ukázali jsme, že G je silně souvislý.

3.6.4 Silně souvislé komponenty. Definice. Je dán orientovaný graf G. Množina vrcholů B se nazývá *silně souvislá komponenta*, též *komponenta silné souvislosti*, jestliže je maximální podmnožina vrcholů taková, podgraf indukovaný B je silně souvislý. □

Poznamenejme, že termín „maximální“ zde znamená „nedá se přidat vrchol při zachování silné souvislosti“. Jinými slovy, pro každý vrchol $v \notin B$ graf indukovaný množinou $B \cup \{v\}$ není silně souvislý.

3.6.5 Poznámka. Každý vrchol orientovaného grafu leží přesně v jedné silně souvislé komponentě. O hranách už takové tvrzení neplatí – mohou existovat hrany, které vedou mezi silně souvislými komponentami.

3.6.6 Hledání silně souvislých komponent — Kosaraju, Sharir algoritmus.

Vstup: Orientovaný graf G.

Výstup: Silně souvislé komponenty grafu G.

1. Prohledáme graf do hloubky a vypíšeme vrcholy v pořadí, ve kterém jsme vrcholy opouštěli.
2. V grafu G obrátíme hrany, dostaneme graf G'.
3. Prohledáme graf G' do hloubky a to v pořadí opačném pořadí v kroku 1.
4. Vrcholy stromů druhého prohledávání jsou pak vrcholy jednotlivých silně souvislých komponent.

3.6.7 Tarjanův algoritmus pro nalezení silně souvislých komponent. Existuje algoritmus, který nalezne silně souvislé komponenty a je rozšířením algoritmu pro prohledávání do hloubky DFS. Autorem algoritmu je Robert Tarjan.

Uvedeme myšlenku rozšíření DFS pro nalezení silně souvislých komponent. Kromě pořadových čísel budeme vrcholům přiřazovat tzv. zpětná čísla. Vrcholy budeme dělit do tří skupin:

- Ještě nenavštívené vrcholy, to budou vrcholy, které ještě nemají pořadové číslo ani zpětné číslo.
- Vrcholy, které již byly navštíveny, ale ještě nejsou zařazeny do žádné komponenty silné souvislosti.
- Vrcholy, které již byly zařazeny do některé komponenty silné souvislosti.

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 používáme zásobník ZAS z prohledávání do hloubky.

1. *[Inicializace.]* Všechny vrcholy jsou nenavštívené a zásobník ZAS je prázdný.
2. *[Volba vrcholu].* Vybereme libovolný dosud nenavštívený vrchol x, označíme ho v, přiřadíme v jeho pořadové číslo $P(v)$, $Z(v) := P(v)$ a vložíme v na vrchol zásobníku ZAS.

   Jestliže nenavštívený vrchol neexistuje, výpočet končí.

3. *[Volba hrany e].* Vezmeme některou dosud nepoužitou hranu e začínající ve vrcholu v a označíme w koncový vrchol této hrany e.

   Pokud hrana neexistuje, pokračujeme krokem 7.

4. *[Koncový vrchol e nebyl navštíven.]* Je-li vrchol w ještě nenavštívený, přiřadíme mu pořadové číslo $P(w)$, položíme $Z(w) := P(w)$, zařadíme w na vrchol zásobníku ZAS a položíme $v := w$. Pokračujeme krokem 3.

5. *[Koncový vrchol e byl navštíven, ale není zařazen do některé komponenty.]* Je-li vrchol w ještě nezařazený do některé komponenty silné souvislosti, zkontrolujeme, zda $Z(v) > P(w)$. Jestliže ano, položíme $Z(v) := P(w)$ a pokračujeme krokem 3.

6. *[Koncový vrchol e je zařazen do některé komponenty.]* Je-li vrchol w již zařazen do některé komponenty silné souvislosti, pokračujeme krokem 3.

7. *[Možná nová komponenta.]* Jestliže $Z(v) = P(v)$, utvoříme novou komponentu silné souvislosti. Komponenta bude obsahovat dosud nezařazené vrcholy w, pro které platí $P(w) \geq P(v)$. Vrcholy komponenty odstraníme ze zásobníku, pokračujeme krokem 8.

   Je-li $Z(v) < P(v)$, pokračujeme krokem 8.

8. *[Test, zda je ZAS prázdný.]* Je-li zásobník ZAS prázdný, pokračujeme krokem 2.

   Není-li prázdný, označíme x vrchol zásobníku ZAS

   Jestliže nebyla uzavřena komponenta a $Z(x) > Z(v)$, položíme $Z(x) := Z(v)$.

   Dále položíme $v := x$ a pokračujeme krokem 3.

3.6.8 Poznámka. Oba algoritmy pracují v lineárním čase, to je v čase úměrném počtu hran a počtu vrcholů. Tarjanův algoritmus použije pouze jedno prohledávání do hloubky, algoritmus autorů Kosaraju a Sharir dvě prohledávání do hloubky. Poznamenejme, že Tarjanův algoritmus je součástí velmi rychlého (také lineárního) algoritmu pro zjištění, zda daný graf lze nakreslit do roviny tak, aby se jednotlivé hrany nekřížily (to je, zda je rovinný).

3.6.9 Kondenzace grafu. Definice. 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. □

3.6.10 Poznámka. Kondenzace grafu už je vždy acyklický graf. Kdyby totiž nebyl, pak byly špatně spočítané silně souvislé komponenty.

3.7 Eulerovy grafy

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

3.7.2 Eulerovské tahy. Definice. Tah v grafu se nazývá *eulerovský*, jestliže prochází každou hranou. □

Jinými slovy, tah se nazývá eulerovský, jestliže obsahuje každou hranu přesně jedenkrát. Eulerovské tahy se dělí na uzavřené a otevřené, orientované a neorientované.

3.7.3 Eulerův graf. Definice. Orientovaný (neorientovaný) graf G se nazývá *eulerovský graf*, jestliže v něm existuje uzavřený orientovaný (neorientovaný) eulerovský tah. □

3.7.4 Aplikace. Eulerovské tahy mají řadu aplikací.

- **Kreslení s co nejmenším počtem tahů.** Je dán neorientovaný souvislý graf. Úkolem je najít co nejmenší počet hranově disjunktních tahů tak, aby všechny hrany grafu byly obsaženy v některém z nich (to znamená, že každá hrana bude obsažena v přesně jednom tahu).

  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?

  Vytvoříme neorientovaný graf takto: vrcholy jsou křižovatky a hrany ulice mezi nimi, kterými musí pošťák projít. Hrany jsou ohodnoceny počtem kilometrů, které daná ulice má.

  Jestliže v grafu existuje eulerovský tah, pak je to nejkratší možné řešení — pošťák projde každou ulicí přesně jednou. Jestliže eulerovský tah neexistuje, musí pošťák projít některou ulicí dvakrát. Tedy hledáme „zdvojení“ některých hran tak, aby vzniklý graf byl eulerovský a aby součet přidaných hran byl nejmenší možný.

- **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 této posloupnosti nebyly stejné. Úloha se dá řešit nalezením uzavřeného orientovaného eulerovského tahu ve speciálním orientovaném grafu.

3.7.5 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 právě tehdy, když každý vrchol má sudý stupeň. □

3.7.6 Postup 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 C. Jestliže C obsahuje všechny hrany, je to hledaný uzavřený eulerovský tah.

Neobsahuje-li C všechny hrany, pak na C existuje vrchol w takový, že v něm začíná ještě nepoužitá hrana. (To vyplývá ze souvislosti grafu.) Tah C ve vrcholu w rozpojíme a vložíme do něj náhodně zkonstruovaný uzavřený tah z dosud nepoužitých hran, který začíná (a končí) ve vrcholu w. Dostaneme nový tah C'.

Jestliže C' obsahuje všechny hrany, je to hledaný eulerovský tah. V opačném případě opět najdeme na C' vrchol, ve kterém začíná dosud nepoužitá hrana a postup opakujeme dokud nedostaneme tah obsahující všechny hrany.

3.7.7 Algoritmus pro hledání uzavřeného eulerovského tahu.

Vstup: Souvislý orientovaný graf $G = (V, E)$, kde pro každý vrchol platí $d^+(v) = d^-(v)$.

Výstup: Orientovaný uzavřený eulerovský tah F v grafu G.

Pomocné proměnné: Seznam F, který je na začátku prázdný a na konci obsahuje hledaný tah; množina ještě nepoužitých hran H.

1. *[Inicializace]*
   $F := \emptyset$; $H := E$

2. *[Počátek tahu]*
   zvolíme libovolný vrchol v; vložíme v do F

3. *[Test ukončení]*
   `if` $H = \emptyset$ `then stop`
   `else` $v :=$ poslední vrchol tahu F

4. *[Prodloužení tahu]*
   `if` existuje $(v, w) \in H$, `then do`
   `begin`
   přidáme hranu (v, w) a vrchol w do tahu F
   $H := H \setminus \{(v, w)\}$;
   `end`
   `go to 3`

5. *[Tah se nedá prodloužit]*
   `else` neexistuje hrana $(v, w) \in H$, `then do`
   `begin`
   najdeme vrchol y v F takový, že existuje $(y, z) \in H$;
   rozpojíme F ve vrcholu y, tj. y je poslední vrchol F
   `end`
   `go to 3`

3.7.8 De Bruijnova posloupnost podruhé. 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 této posloupnosti nebyly stejné.

Utvoříme orientovaný graf $G = (V, E)$, kde vrcholy jsou všechny různé $k-1$-tice 0 a 1; z vrcholu $a_1, a_2, \ldots, a_{k-1}$ vedou přesně dvě orientované hrany a to do vrcholu $a_2, a_3, \ldots, a_{k-1}, 0$ a do vrcholu $a_2, a_3, \ldots, a_{k-1}, 1$. První hrana je ohodnocena 0, druhá hrana 1.

Je zřejmé, že graf G je souvislý a má 2^k hran. Pro každý vrchol v platí $d^+(v) = 2 = d^-(v)$. Proto v něm existuje eulerovský uzavřený tah. Vytvoříme-li posloupnost z ohodnocení jednotlivých hran v eulerovském tahu, dostaneme hledanou cyklickou posloupnost.

3.7.9 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. □

Důkaz je podobný jako důkaz věty 3.7.5. Pouze při konstrukci otevřeného eulerovského tahu musíme začínat ve vrcholu u_2, tj. z vrcholu, ze kterého vychází o jednu hranu víc než do něho vchází.

3.7.10 Tvrzení. Je dán souvislý neorientovaný graf G s $2k$, $k > 0$, vrcholy lichého stupně. Pak existuje k hranově disjunktních otevřených tahů takových, že každá hrana grafu G leží v právě jednom z těchto tahů. □

3.8 Hamiltonovské grafy

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

3.8.1 Hamiltonovské cesty, kružnice, cykly. Definice. 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. □

3.8.2 Hamiltonovské grafy. Definice. Orientovaný (neorientovaný) graf G se nazývá *hamiltonovský*, jestliže obsahuje hamiltonovský cyklus (hamiltonovskou kružnici). □

3.8.3 Úlohy spojené s hamiltonovskými cestami dělíme na existenční, vyhodnocovací a optimalizační. V existenční úloze jde o to zjistit, zda v daném grafu existuje hamiltonovská cesta, kružnice nebo cyklus. Ve vyhodnocovací úloze jde o to najít (aspoň) jednu hamiltonovskou cestu, kružnici nebo cyklus v případě, že existují. 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 cesty, kružnice nebo cyklu.

Na rozdíl od hledání eulerovských tahů, je hledání hamiltonovských cest, hamiltonovských kružnic a hamiltonovských cyklů velmi obtížná úloha. Přesněji, není znám algoritmus, který by pro obecný graf zjistil, zda v daném grafu existuje hamiltonovská cesta, kružnice nebo cyklus, a přitom provedl počet kroků, který závisí na počtu vrcholů a hran daného grafu jen polynomiálně. Přesto, nebo právě proto, jsou úlohy tohoto typu v praxi rozšířené.

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

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

3.8.5 Věta. Je dán neorientovaný prostý graf $G = (V, E)$ s $n \geq 3$ vrcholy. Označme

$$d_1 \leq d_2 \leq d_3 \leq \ldots \leq d_n$$

posloupnost stupňů grafu G (tj. v posloupnosti jsou stupně všech vrcholů grafu G a jsou uspořádány neklesajícím způsobem). Jestliže platí následující podmínka:

$$\text{kdykoli existuje } k < \tfrac{n}{2} \text{ takové, že } d_k \leq k, \text{ pak platí } d_{n-k} \geq n - k,$$

pak graf G je hamiltonovský. □

3.8.6 Předchozí věta se nazývá Chvátalova věta. Jedná se o nejlepší možnou postačující podmínku pro existenci hamiltonovské kružnice založenou pouze na vlastnostech stupňů vrcholů. Platí totiž: Jestliže posloupnost

$$d_1 \leq d_2 \leq d_3 \leq \ldots \leq d_n$$

nesplňuje podmínku věty, pak existuje prostý neorientovaný graf G, který není hamiltonovský a jeho posloupnost stupňů majorizuje posloupnost $d_1, \ldots, d_n$. (To znamená, že je pro každé i alespoň tak velká jako d_i.)

Důkaz Chvátalovy věty není jednoduchý a je nad rámec této přednášky.

3.8.7 Aplikace hamiltonovských cest. Uvedeme tři aplikace 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 zař9zení na kterém se provádějí 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.

3.8.8 Protože doposud není znám žádný rychlý algoritmus, který by našel hamiltonovské cesty, kružnice či cykly, často se jejich hledání provádí metodou větví a mezí. Jak metoda větví a mezí pracuje, si ukážeme na úloze hledání nejlevnější hamiltonovské cesty v ohodnoceném grafu.

3.8.9 Hledání nejkratší hamiltonovské cesty. Vyjdeme z následujících pozorování: Každá hamiltonovská cesta v grafu G je

1. kostrou grafu G,
2. ve které má každý vrchol stupeň nejvýše 2.

Tohoto pozorování využijeme takto:

V daném grafu G najdeme minimální kostru K grafu G. Je-li tato kostra cesta, dostali jsme nejkratší hamiltonovskou cestu v grafu G.

Není-li tato kostra cesta, tj. obsahuje-li vrchol v stupně aspoň 3, zakážeme vždy jednu hranu incidentní s v. Tím rozvětvíme danou úlohu na k podúloh, kde k je stupeň vrcholu v. Přitom každá z podúloh má méně hran než původní úloha. Nyní každou z podúloh řešíme podobně.

Navíc, cena kterékoli již získané hamiltonovské cesty je horní odhad délky nejkratší hamiltonovské cesty. Jestliže se v některé podúloze stane, že délka minimální kostry je delší nebo stejná jako délka dosud známé hamiltonovské cesty, nemusíme se danou podúlohou již zabývat – nemůžeme dostat cestu s menší délkou.

Obdobně, jestliže v některé podúloze již minimální kostra neexistuje (zakázali jsme tolik hran, ze graf přestal být souvislý), nemusíme se touto podúlohou zabývat.

3.8.10 Poznámka. Pro zjištění horního odhadu nejkratší hamiltonovské cesty nemusíme čekat na to, až jako minimální kostru dostaneme některou hamiltonovskou cestu. Často nejprve vytvoříme některou hamiltonovskou cestu (třeba „hladovým postupem“) a ta nám pak slouží jako „prvotní“ odhad.

Poznamenejme, že popsaná metoda může pracovat dlouho. Proto v praxi přidáváme časové omezení; překročí-li doba práce toto omezení, bereme jako „nejlepší“ řešení to nejlepší z dosud známých řešení.

Popsaná metoda je zvláštním případem tzv. *metody větví a mezí*, kterou popíšeme přesněji v následujícím odstavci.

3.8.11 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 „složité“ podmínky B. Jako $\mathcal{U}'$ označíme optimalizační úlohu, která je dána pouze podmínkami A a se stejnou účelovou funkcí c.

Nejprve řešíme úlohu $\mathcal{U}'$ (tj. zapomeneme na podmínky B — říkáme, že podmínky B relaxujeme) a najdeme optimální řešení *opt* úlohy $\mathcal{U}'$. 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.

Je-li možné vybrat z několika podúloh, vybíráme vždy podúlohu s nejlepším odhadem. Výpočet ukončíme tehdy, když pro všechny podúlohy jsme buď našli optimální řešení, nebo jsme je umrtvili.

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

- A – hamiltonovská cesta je kostra,
- B – každý vrchol v hamiltonovské cestě má stupeň nejvýše 2,
- pro každou množinu hran C je hodnota účelové funkce rovna součtu cen jednotlivých hran v C,
- cena minimální kostry slouží jako odhad optimálního řešení dané podúlohy.

3.8.13 Poznámka. Pro zmenšení počtu větvení při výpočtu nejkratší hamiltonovské cesty můžeme použít ještě jiný způsob větvení úlohy než je uveden výše.

Označme v vrchol, který má v minimální kostře stupeň $d(v) = k \geq 3$ 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 přípustných řešení úloh $\mathcal{U}_1, \mathcal{U}_2$ a $\mathcal{U}_3$ jsou po dvou disjunktní.

3.9 Nezávislost, kliky a barevnost grafu

3.9.1 Nezávislé množiny. Definice. 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.

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

Jinými slovy, nezávislá množina 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.

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

3.9.4 Poznámky.

1) Nejpočetnější nezávislá množina je jistě také maximální, ale ne každá maximální nezávislá množina je současně nejpočetnější.
2) Jádro orientovaného grafu G definované v 3.5.10 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 3.5.10.

3.9.5 Úplný neorientovaný graf. Připomeňme pojem úplného grafu; jedná se o graf, který má nejvíce hran a je prostý a bez smyček.

Definice. Neorientovaný graf G se nazývá *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.

3.9.6 Klika v grafu. Definice. 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.

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

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

□

3.9.8 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}. □

3.9.9 Obarvení grafu. Definice. 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. □

3.9.10 k-barevný graf. Definice. Graf G se nazývá *k-barevný*, jestliže má aspoň k vrcholů a dá se obarvit k barvami (tj. jestliže existuje obarvení grafu G takové, že množina barev B má k prvků). □

3.9.11 Barevnost grafu. Definice. 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 $\chi(G)$. □

3.9.12 Pozorování. Každá množina vrcholů, která je obarvena stejnou barvou, tvoří nezávislou množinu grafu.

Graf je jednobarevný právě tehdy, když nemá žádnou hranu.

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

Zdůvodnění. Jestliže je graf G obsahuje kružnici liché délky, pak není možné jej obarvit dvěma barvami (k obarvení kružnice liché délky je třeba 3 barev).

Druhou implikaci dokazuje následující postup, jak zjistit, že daný graf je dvoubarevný.

Metoda zjištění, zda graf je dvoubarevný. Zjistit, zda je daný graf dvoubarevný, se dá jednoduchou modifikací prohledávání do šířky. Jestliže graf není souvislý, obarvíme každou komponentu souvislosti zvlášť. Budeme se proto nyní zabývat jen souvislými grafy.

Souvislý graf G prohledáme do šířky. Vrcholům, které ležely v sudých hladinách, přiřadíme barvu 1; vrcholům, které ležely v lichých hladinách, přiřadíme barvu 2.

Dá se dokázat, že neobsahuje-li graf kružnici liché délky, jedná se o korektní obarvení grafu a graf je tedy dvoubarevný. To je proto, že vede-li hrana mezi dvěma vrcholy v hladinách stejné parity, obsahuje graf kružnici liché délky a není proto dvoubarevný.

3.9.14 Bipartitní grafy. Definice. Graf G se nazývá *bipartitní*, jestliže jeho množinu vrcholů můžeme rozdělit na dvě neprázdné disjunktní množiny X a Y tak, že každá hrana má jeden krajní vrchol v množině X a druhý krajní vrchol v množině Y. □

3.9.15 Tvrzení. Graf G je bipartitní právě tehdy, když je dvoubarevný. □

3.9.16 Poznámka. Zjistit, zda daný graf je tříbarevný, je těžký problém (nezná se pro jeho řešení rychlý algoritmus).

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

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

□

Myšlenka důkazu. Jestliže graf má barevnost k, pak je k-barevný, to znamená, že je možné jeho vrcholy rozdělit do k disjunktních množin, kde každá množina je nezávislá. Máme tedy nezávislé množiny vrcholů $N_1, \ldots, N_k$. Kdyby mezi některými z těchto množin nevedla žádná hrana, byl by G i $k-1$ barevný, což není. Tedy mezi každými dvěma množinami N_i a N_j $(i \neq j)$ vede hrana. Proto $m \geq \frac{k(k-1)}{2}$, úpravou dostáváme odhad z věty. □

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

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

□

Předchozí tvrzení dokážeme tím, že uvedeme postup, jak graf G obarvit $\Delta(G) + 1$ barvami.

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

1. Seřadíme vrcholy grafu 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.

3.9.20 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).

3.9.21 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.

Zdůvodnění. Vytvoříme obarvení barvami $1, 2, \ldots, \alpha(G) + 1$ barvami. Tím dokážeme, že barevnost je nejvýše $\alpha(G) + 1$.

Vyberme nezávislou množinu N o $\alpha(G)$ vrcholech. Tuto množinu obarvíme jednou barvou. Zbývá $n - \alpha(G)$ neobarvených vrcholů. Každý z těchto vrcholů obarvíme jednou (jinou) barvou. Tím jsme dostali požadované obarvení. □

3.10 Rovinné grafy

3.10.1 Nakreslení grafu. Je dán neorientovaný graf G s množinou vrcholů V a množinou hran E. *Nakreslení grafu* se skládá z přiřazení, které vrcholům grafu přiřazuje různé body roviny a hranám přiřazuje křivky a to tak, že 1. různým hranám jsou přiřazeny různé křivky, a 2. jestliže hrana e vede mezi vrcholy u, v, pak křivka odpovídající e vede mezi body odpovídajícími vrcholům u, v. Formálně:
Definice. Nakreslení grafu $G = (V, E, \varepsilon)$ je dvojice prostých zobrazení f, g, takových, že f přiřazuje vrcholům $v \in V$ body eukleidovské roviny, g přiřazuje hranám prosté křivky a to tak, že platí

$$\text{je-li } \varepsilon(e) = \{x, y\}, \text{ pak } g(e) \text{ má krajní body } f(x) \text{ a } f(y).$$

□

3.10.2 Rovinné nakreslení grafu. Definice. Nakreslení grafu G se nazývá *rovinné*, jestliže se křivky odpovídající různým hranám nekříží (tj. jestliže dvě křivky, které odpovídají dvěma různým hranám, mají společné nejvýše krajní body). □

3.10.3 Rovinný (planární) graf. Definice. Graf G se nazývá *rovinný*, jestliže má rovinné nakreslení. □

Jinými slovy, graf je rovinný, jestliže je možné ho nakreslit v rovině tak, aby se jeho hrany „nekřížily“.

Poznamenejme, že i rovinný graf se dá (často) nakreslit tak, aby se křivky odpovídající jeho hranám křížily. K tomu, aby graf byl rovinný, stačí, abychom našli jedno jeho nakreslení, které je rovinné.

3.10.4 Definice. Rovinný graf spolu se svým rovinným nakreslením se nazývá *topologický rovinný graf*. □

3.10.5 Nakreslení grafu na kulové ploše. Stejně jako jsme v 3.10.1 a 3.10.2 definovali nakreslení grafu v rovině, můžeme definovat nakreslení grafu (rovinné nakreslení grafu) na kulové ploše. Nedostaneme tím ovšem nic nového – grafy, které je možno nakreslit aniž by se jejich hrany křížily na kulové ploše, jsou přesně rovinné grafy z definice 3.10.3.

3.10.6 Stěna grafu. Definice. Je dáno rovinné nakreslení grafu G. *Stěna topologického rovinného grafu* G je minimální část roviny, která je ohraničena křivkami odpovídajícími hranám grafu. Jedna ze stěn je vždy neomezená, ostatní jsou omezené. Dvě stěny jsou *sousední*, jestliže společná část jejich hranice je tvořena jednou nebo více křivkami odpovídajícími hranám grafu.

Řekneme, že *stěna je incidentní s hranou*, jestliže křivka odpovídající hraně je částí hranice stěny nebo celá ve stěně leží.

Stupeň stěny je počet hran s nimiž je stěna incidentní s tím, že každou hranu, která leží celá v jedné stěně, počítáme dvakrát. □

3.10.7 Tvrzení. Sečteme-li stupně všech stěn rovinného grafu, dostaneme dvojnásobek počtu hran. □

3.10.8 Věta (Eulerova formule). Pro každý souvislý rovinný graf, který má n vrcholů, m hran a s stěn, platí

$$n + s = m + 2.$$

□

Zdůvodnění. Každý souvislý graf má kostru. Kostra má o jednu hranu méně než je počet jejích vrcholů, a navíc má jen jednu stěnu – neomezenou. Tedy pro kostru tvrzení platí.

Přidáním libovolné hrany k již souvislému grafu zvětšíme počet stěn o 1; ano, jednu stěnu rozdělíme na dvě. Tvrzení proto platí pro každý souvislý graf. □

3.10.9 Věta. Je dán prostý souvislý rovinný graf bez smyček s $n \geq 3$ vrcholy a m hranami, pak platí

$$m \leq 3n - 6.$$

□

Zdůvodnění. Jestliže G je rovinný, souvislý, prostý a bez smyček, má každá stěna stupeň alespoň 3. Přitom součet všech stupňů stěn je roven dvakrát počtu hran. Máme proto $2m \geq 3s$, tedy $s \leq \frac{2}{3}\, m$. Dosazením do Eulerovy formule dostáváme

$$m + 2 = n + s \leq n + \frac{2}{3}m, \quad \text{tj. } 3m + 6 \leq 3n + 2m,$$

Poslední nerovnost dává $m \leq 3n - 6$.

3.10.10 Věta. Je dán prostý souvislý rovinný graf bez smyček a trojúhelníků s $n \geq 3$ vrcholy a m hranami, pak platí

$$m \leq 2n - 4.$$

□

Zdůvodnění. Toto tvrzení se dokazuje obdobně jako předchozí; pouze v případě, že graf G neobsahuje trojúhelník, má každá stěna stupeň alespoň 4. Z tohoto vztahu dostáváme $2m \geq 4s$, tj. $s \leq \frac{1}{2}\, m$ a

$$m + 2 \leq n + \frac{1}{2}m, \quad \text{tj. } m \leq 2n - 4.$$

□

3.10.11 Věta. V každém prostém rovinném grafu G bez smyček existuje vrchol v, který má stupeň nejvýše 5. □

Zdůvodnění. Ukážeme, že každý prostý rovinný graf má v každé komponentě souvislosti aspoň jeden vrchol stupně nejvýše 5. To je zřejmé pro komponenty s nejvýše 6 vrcholy. Pro komponenty s aspoň 7 vrcholy tvrzení dokážeme sporem.

Předpokládejme, že by existovala komponenta souvislosti prostého rovinného grafu bez smyček, kde každý vrchol by měl stupeň alespoň 6. Pak by platilo

$$2m \geq 6n, \text{ tj. } m \geq 3n > 3n - 6,$$

což je ve sporu s větou 3.10.9. □

3.10.12 Tvrzení. Úplný neorientovaný graf K_5 na 5 vrcholech není rovinný.

Zdůvodnění. Stačí si uvědomit, že K_5 má $n = 5$ vrcholů a $m = 10$ hran, přitom je prostý a bez smyček. Nesplňuje tedy 3.10.9. □

3.10.13 Tvrzení. Úplný bipartitní graf se stranami o třech vrcholech $K_{3,3}$ není rovinný. □

Zdůvodnění. Stačí si uvědomit, že $K_{3,3}$ má $n = 6$ vrcholů a $m = 9$ hran, přitom je prostý a bez smyček a bez trojúhelníků. Nesplňuje tedy 3.10.10.

3.10.14 Věta o čtyřech barvách. Každý rovinný graf bez smyček lze obarvit čtyřmi barvami. □

Věta o čtyřech barvách byla dlouho nedokázaná. Nakonec ve druhé polovině minulého století byla dokázána s pomocí počítačů.