# LGR

# 5. Syntaxe a sémantika výrokové a predikátové logiky. Základní pojmy teorie grafů.

---

\http://math.feld.cvut.cz/demlova/teaching/lgr/text_lgr_2017.pdf

https://math.fel.cvut.cz/en/people/demlova/lgr/predn_lgr.html

---

## Výroková logika

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

1. Každá logická proměnná (atomický výrok) a ∊ A je výroková formule.
2. Jsou-li $\alpha$, $\beta$ výrokové formule, pak $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \Rightarrow \beta)$, $(\alpha \Leftrightarrow \beta)$ 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 $P(A)$.
**Poznámka**: Spojka ¬ se nazývá *unární*, protože vytváří novou formuli z jedné formule. Ostatní zde zavedené spojky se nazývají *binární*, protože vytvářejí novou formuli ze dvou formulí.
V dalším textu logické proměnné označujeme malými písmeny např. a,b,c, … nebo x,y,z, … , výrokové formule označujeme malými řeckými písmeny např. α , β , γ , … nebo ϕ , ψ , … . Také většinou nebudeme ve formulích psát ty nejvíc vnější závorky - tj. píšeme a ∨ ( b ⇒ c ) místo ( a ∨ ( b ⇒ c ) ) .

#### 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. Jedná-li se o binární spojku, má vrchol dva následníky, jedná-li se o unární spojku, má vrchol pouze jednoho následníka. Přitom pro formule ( α ∧ β ) , ( α ∨ β ) , ( α ⇒ β ) odpovídá levý následník formuli α , pravý následník formuli β . Listy stromu jsou ohodnoceny logickými proměnnými.

#### Podformule

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

#### Pravdivostní ohodnocení

*Pravdivostní ohodnocení*, též pouze *ohodnocení formulí*, je zobrazení u:P( A ) → { 0,1 } , které splňuje pravidla

1. **Negace** ( ¬ α ) je pravdivá právě tehdy, když α je nepravdivá, tj. u( ¬ α ) =1 právě tehdy, když u( α ) =0 ;
2. **Konjunkce** ( α ∧ β ) je pravdivá právě tehdy, když α a β jsou obě pravdivé, tj. u( α ∧ β ) =1 právě tehdy, když u( α ) =u( β ) =1
3. **Disjunkce** ( α ∨ β ) je nepravdivá právě tehdy, když α a β jsou obě nepravdivé, tj. u( α ∨ β ) =0 právě tehdy, když u( α ) =u( β ) =0 ;
4. **Implikace** ( α ⇒ β ) je nepravdivá právě tehdy, když α je pravdivá a β nepravdivá, tj. u( α ⇒ β ) =0 právě tehdy, když u( α ) =1 a u( β ) =0 ;
5. **Ekvivalence** ( α ↔ β ) je pravdivá právě tehdy, když buď obě formule α a β jsou pravdivé nebo obě jsou nepravdivé, tj. u( α ↔ β ) =1 právě tehdy, když u( α ) =u( β ) .

#### Pravdivostní tabulky

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

| α | ¬ α |
|---|---|
| 0 | 1 |
| 1 | 0 |

| α | β | α ∧ β | α ∨ β | α ⇒ β | α ↔ β |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

**Věta**
Každé zobrazení $u_0$: A → { 0,1 } jednoznačně určuje ohodnocení u:P( A ) → { 0,1 } takové, že $u_0$ ( a ) =u( a ) pro všechna a ∈ A .

**Důsledek**
Dvě ohodnocení u,v:P( A ) → { 0,1 } jsou shodná právě tehdy, když pro všechny logické proměnné x ∈ A platí u( x ) = v( x ).

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

**Příklady**:

1. Formule α ∨ ¬ α , α ⇒ α , α ⇒ ( β ⇒ α ) jsou tautologie.
2. Formule a ∨ b , ( a ⇒ b ) ⇒ a jsou splnitelné, ale ne tautologie.
3. Formule α ∧ ¬ α je kontradikce. Kontradikce je také každá negace tautologie.

**Úplné systémy logických spojek**
Množiny logických spojek, pomocí, kterých lze popsat všechny možné formule.
například: {∨, ¬}, {∧, ¬}, {⇒, ¬}

**Tautologická ekvivalence formulí**
Řekneme, že formule ϕ a ψ jsou *tautologicky ekvivalentní* (také *sémanticky ekvivalentní*), jestliže pro každé ohodnocení u platí u( ϕ ) = u( ψ ) .
Tautologickou ekvivalenci značíme ϕ |=| ψ .

**Tvrzení**: Pro každé formule α , β , γ platí:

- α |=| α ,
- je-li α |=| β , pak i β |=| α ,
- je-li α |=| β a β |=| γ , pak i α |=| γ .

Jsou-li α , β , γ , δ formule splňující α |=| β a γ |=| δ , pak platí

- ¬ α |=| ¬ β ;
- ( α ∧ γ ) |=| ( β ∧ δ )
- ( α ∨ γ ) |=| ( β ∨ δ )
- ( α ⇒ γ ) |=| ( β ⇒ δ )
- ( α ↔ γ ) |=| ( β ↔ δ ) .

**Tvrzení**
Pro každé formule α , β , γ platí

- α ∧ α |=| α , α ∨ α |=| α (idempotence ∧ a ∨ );
- α ∧ β |=| β ∧ α , α ∨ β |=| β ∨ α (komutativita ∧ a ∨ );
- α ∧ ( β ∧ γ ) |=| ( α ∧ β ) ∧ γ , α ∨ ( β ∨ γ ) |=| ( α ∨ β ) ∨ γ (asociativita ∧ a ∨ );
- α ∧ ( β ∨ α ) |=| α , α ∨ ( β ∧ α ) |=| α (absorpce ∧ a ∨ );
- ¬ ¬ α |=| α ;
- ( α ⇒ β ) |=| ( ¬ α ∨ β ) .

**Poznámka**: Platí α |=| β právě tehdy, když α ↔ β je tautologie.
**Další spojky**
Každá formule s jednou (nebo žádnou) logickou proměnnou představuje zobrazení z množiny { 0,1 } do množiny { 0,1 } . Existují čtyři taková zobrazení:

| x | f1 | f2 | f3 | f4 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |

Funkce f1 je konstantní 0, jedná se o kontradikci a budeme ji značit **F**. Podobně funkce f4 je tautologie (konstantní 1), značíme je **T**. Funkce f2 je vlastně logická proměnná x a funkce f 3 je ¬ x . Tedy nemáme další unární spojky.
**NAND**
Logická spojka |, nazývaná NAND (také Sheffův operátor), je definována
x | y |=| ¬ ( x ∧ y ) .

**NOR**
Logická spojka ↓, nazývaná NOR (také Peiceova šipka), je definována

$x \downarrow y \mid=\mid \neg ( x \vee y )$ .

**XOR**
Lofická spojka $\oplus$ , nazývaná XOR (také vylučovací nebo), je definována
$x \oplus y \mid=\mid \neg ( x \leftrightarrow y )$ .

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

**Disjunktivní normální tvar**
*Literál* je logická proměnná nebo negace logické proměnné. Řekneme, že formule je v *disjunktivním normálním tvaru*, zkráceně v *DNF*, jestliže je disjunkcí jedné nebo několika formulí, z nichž každá je literálem nebo konjunkcí literálů.
Poznamenejme, že literálu nebo konjunkci literálů se také říká *minterm*. Jestliže každý minterm obsahuje všechny proměnné, říkáme, že se jedná o *úplnou DNF*.
**Věta**: Ke každé Booleově funkci *f* existuje formule v *DNF* odpovídající *f*.
**Důsledek**: Ke každé formuli $\alpha$ existuje formule $\beta$ , která je v DNF a navíc $\alpha \mid=\mid \beta$ .

**Konjunktivní normální tvar**
Řekneme, že formule je v *konjunktivním normálním tvaru*, zkráceně v *CNF*, jestliže je konjunkcí jedné nebo několika formulí, z nichž každá je literálem nebo disjunkcí literálů.
Poznamenejme, že literálu nebo disjunkci literálů se také říká *maxterm* nebo *klausule*. Jestliže každá klausule obsahuje všechny proměnné, říkáme, že se jedná o *úplnou CNF*.
**Věta**: Ke každé Booleově funkci *f* existuje formule v *CNF* odpovídající *f*.
**Důsledek**: Ke každé formuli $\alpha$ existuje formule $\beta$ , která je v CNF a navíc $\alpha \mid=\mid \beta$ .

**Booleovský kalkul**
Víme, že pro pravdivostní ohodnocení formulí platí:
$u( a \vee b ) = \max \{ u( a ) ,u( b ) \} = \max \{ x,y \}$ ,
$u( a \wedge b ) = \min \{ u( a ) ,u( b ) \} = \min \{ x,y \}$ ,
$u( \neg a ) =1-u( a ) =1-x$.
kde $x=u( a )$ , $y=u( b )$ .

**Booleovské operace**
To motivuje zavedení booleovských operací (pro hodnoty 0,1):
součin $x \cdot y= \min \{ x,y \}$ ,
logický součet $x+y= \max \{ x,y \}$ ,
doplněk $x^{-} =1-x$.
Pro tyto operace platí řada rovností, tak, jak je známe z výrokové logiky:

**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. $x==x$ ;
7. $x+y^{-} = x^{-} \cdot y^{-}$ , $x \cdot y^{-} = x^{-} + y^{-}$ ;
8. $x+ x^{-} =1$ , $x \cdot x^{-} =0$ ;
9. $x \cdot 0=0$ , $x \cdot 1=x$ ;
10. $x+1=1$ , $x+0=x$ .

**Booleovy funkce v DNF a CNF**
Nyní můžeme pro Booleovu funkci psát pomocí výše uvedených operací, např.
$f( x,y,z ) = x^{-} y^{-} z^{-} + x^{-} y^{-} z+ x,,z^{-} + x^{-} yz+x y^{-} z$
a říkat, že jsme Booleovu funkci napsali v *disjunktivní normální formě*. Rovnost opravdu platí; dosadíme-li za logické proměnné jakékoli hodnoty, pak pravá strana rovnosti určuje hodnotu Booleovy funkce *f*. Obdobně jako jsme Booleovu funkci *f* napsali v disjunktivní normální formě, můžeme ji také napsat v *konjunktivní normální formě* a to takto:
$f( x,y,z ) =( x^{-} +y+z ) ( x^{-} + y^{-} +z ) ( x^{-} + y^{-} + z^{-} )$ .
**Věta**: Každou Booleovu funkci lze napsat v disjunktivní normální formě i v konjunktivní normální formě.

**Důsledek ve výrokové logice**
Řekneme, že formulí $\phi$ je *sémantickým důsledkem*, též *konsekventem* množiny formulí *S* právě tehdy, když je pravdivá ve všech ohodnoceních, kdy je pravdivá formule $S$. Tento fakt značíme $S \models \phi$.

**Příklady**
Pro každé tři formule $\alpha$, $\beta$, $\gamma$ platí:

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

Tvrzení

1. Je-li *S* množina formulí a $\phi \in S$ , pak $\phi$ je konsekventem *S*, tj. $S \models \phi$ pro každou $\phi \in S$ .
2. Tautologie je konsekventem každé množiny formulí *S*.
3. Formule $\phi$ je tautologie právě tehdy, když $\models \phi$ .
4. Každá formule je konsekventem nesplnitelné množiny formulí.

Poznámka
Uvědomme si, že $\alpha$ |=| $\beta$ právě tehdy, když platí současně $\alpha \models \beta$ a také $\beta \models \alpha$ .
Tvrzení
Pro každé dvě formule $\alpha$ a $\beta$ platí:
$\alpha \models \beta$ právě tehdy, když $\alpha \Rightarrow \beta$ je tautologie.

Věta
Pro množinu formulí S a formuli $\phi$ platí:
$S \models \phi$ právě tehdy, když $S \cup \{ \neg \phi \}$ je nesplnitelná.

Věta o dedukci
Pro množinu formulí S a formule $\phi$ a $\psi$ platí
$S \cup \{ \phi \} \models \psi$ právě tehdy, když $S \models ( \phi \Rightarrow \psi )$.

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

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

**Klausule**
Množinu všech logických proměnných označíme *A*. Připomeňme, že *literál* je buď logická proměnná (tzv. *positivní literál*) nebo negace logické proměnné (tzv. *negativní literál*). *Komplementární literály* jsou literály *p* a $\neg p$ . *Klausule* je literál nebo disjunkce konečně mnoha literálů (tedy i žádného). Zvláštní místo mezi klausulemi zaujímá *prázdná klausule*, tj. klausule, která neobsahuje žádný literál a tudíž se jedná o kontradikci. Proto ji budeme označovat *F*.
Pro jednoduchost zavedeme následující konvenci: Máme dánu klausuli *C* a literál *p*, který se v *C* vyskytuje. Pak symbolem $C \setminus p$ označujeme klausuli, která obsahuje všechny literály jako *C* kromě *p*. Tedy např. je-li $C= \neg x \vee y \vee \neg z$ , pak $C \setminus \neg z= \neg x \vee y$.

Rezoloventa
Řekneme, že klausule *D* je rezolventou klausulí $C_1$ a $C_2$ právě tehdy, když existuje literál *p* takový, že *p* se vyskytuje v klausuli $C_1$ , $\neg p$ se vyskytuje v klausuli $C_2$ a
$D=( C_1 \setminus p ) \vee ( C_2 \setminus \neg p )$ .
Také říkáme, že klausule *D* je rezolventou $C_1$ a $C_2$ podle literálu *p* a značíme $D=res_p ( C_1 , C_2 )$ .

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

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

**Rezoluční princip**
Označme
$R( S ) =S \cup \{ D | D$ je rezoloventa některých klausulí z $S \}$
$R^0 ( S ) =S$

$R^{i+1}(S) = R(R^i(S))$ pro $i \in N$
$R^\star(S) = \cup \{ R^i(S) | i \geq 0 \}$ .
Protože pro konečnou množinu logických proměnných existuje jen konečně mnoho klausulí, musí existovat *n* takové, že $R^n(S) = R^{n+1}(S)$ . Pro toto n platí $R^n(S) = R^\star(S)$ .

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

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

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

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

Výhodnější postup
Existuje ještě jeden postup, který usnadní práci s použitím rezoluční metody. Ten nejenom že nám odpoví na otázku, zda konečná množina klausulí *S* je splnitelná nebo nesplnitelná, ale dokonce nám umožní v případě splnitelnosti sestrojit aspoň jedno pravdivostní ohodnocení, v němž je množina *S* pravdivá.
Máme konečnou množinu klausulí *S*, kde žádná klausule není tautologií. Zvolíme jednu logickou proměnnou (označme ji *x*), která se v některé z klausulí z *S* vyskytuje. Najdeme množinu klausulí $S_1$ s těmito vlastnostmi:

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

Množinu $S_1$ vytvoříme takto: Rozdělíme klausule množiny *S* do tří skupin: $M_0$ se skládá ze všech klausulí množiny *S*, které neobsahují logickou proměnnou *x*.
$M_x$ se skládá ze všech klausulí množiny *S*, které obsahují positivní literál *x*.
$M_{\neg x}$ se skládá ze všech klausulí množiny *S*, které obsahují negativní literál $\neg x$.
Označme *N* množinu všech rezolvent klausulí množiny *S* podle literálu *x*; tj. rezolvent vždy jedné klausule z množiny $M_x$ s jednou klausulí z množiny $M_{\neg x}$ . Všechny tautologie vyřadíme.
Položíme $S_1 = M_0 \cup N$ .

**Tvrzení**: Množina klausulí $S_1$ zkonstruovaná výše je splnitelná právě tehdy, když je splnitelná množina *S*.
Dostali jsme tedy množinu klausulí $S_1$ , která již neobsahuje logickou proměnnou *x* a je splnitelná právě tehdy, když je splnitelná množina *S*. Navíc, množina $S_1$ má o jednu logickou proměnnou méně než množina *S*.
Nyní opakujeme postup pro množinu $S_1$ . Postup skončí jedním ze dvou možných způsobů:

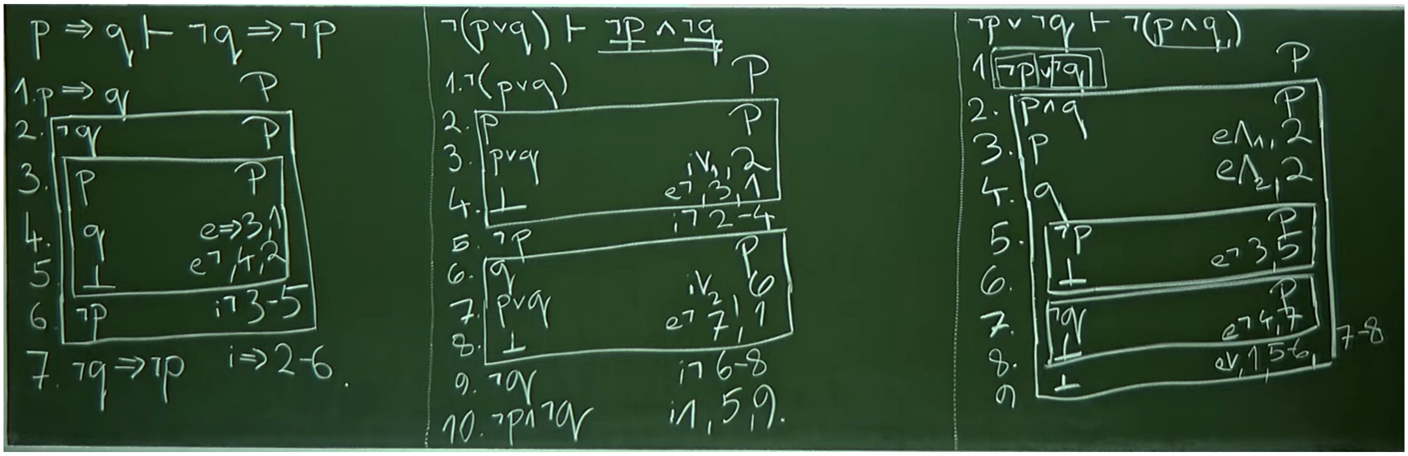
1. Při vytváření rezolvent dostaneme prázdnou kalusuli *F*. Tedy *S* je nesplnitelná.
2. Dostaneme prázdnou množinu klausulí. V tomto případě je množina *S* splnitelná.

**Přirozená dedukce**
Formální důkazové systémy - zabývají se odvoditelností formulí, patří do syntaxe, nezabývají se pravdivostí, odvozují logickou ekvivalenci
Pracuje se základní sadou spojek kromě ekvivalence
i-pravidlo - spojku zavádí (introduction)

| Logická spojka | Pravidlo pro zavedení | Pravidlo pro vyloučení |
|---|---|---|
| $\neg$ | $\varphi \to \{\psi, \neg\psi\} \to \neg\varphi$ *princip nepřímého důkazu* | $\varphi \vee \neg\varphi \to \mathrm{T}$; $\neg\neg\varphi \to \varphi$ *princip vyloučení třetího a princip dvojí negace* |
| $\wedge$ | $\{\varphi, \psi\} \to \{\varphi \wedge \psi, \psi \wedge \varphi\}$ *definice konjunkce* | $\varphi \wedge \psi \to \{\varphi, \psi\}$ *definice konjunkce* |
| $\vee$ | $\varphi \to \{\varphi \vee \psi, \psi \vee \varphi\}$ *definice disjunkce* | $\{\varphi \vee \psi, \varphi \to \alpha\ \alpha, \psi \to \alpha\} \to \alpha$ *princip důkazu rozborem případů* |
| $\Rightarrow$ | $\{\varphi \to \psi\} \to \varphi \Rightarrow \psi$ *definice implikace* | $\{\varphi, \varphi \Rightarrow \psi\} \to \psi$ *pravidlo modus ponens* |
| **Kvalifikátor** | **Pravidlo pro zavedení** | **Pravidlo pro vyloučení** |
| $\forall$ | $\varphi(x) \to \forall x\, \varphi(x)$ | $\varphi(x)\ \forall x\, \varphi(x) \to \varphi(x)$ |
| $\exists$ | $\varphi(a) \to \exists x\, \varphi(x)$ | $\{\exists x\, \varphi(x), \varphi(y) \to \psi\} \to \psi$ |

e-pravidlo - spojku odstraňuje (elimination)

**Tab. 4.4 – Odvozovací pravidla pro přirozenou dedukci**




## Predikátová logika

**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*.

Jazyk predikátové logiky L
*Jazyk predikátové logiky* se skládá z

1. *logických symbolů*, tj.:
   1. spočetné množiny individuálních proměnných: Var={ x,y, … , x1 , x2 , … }
   2. výrokových logických spojek: ¬ , ∧ , ∨ , ⇒ , ↔
   3. obecného kvantifikátoru ∀ a existenčního kvantifikátoru ∃
2. *speciálních symbolů*, tj.:
   1. množiny Pred predikátových symbolů (nesmí být prázdná)
   2. množiny Kons konstantních symbolů (může být prázdná)
   3. množiny Func funkčních symbolů (může být prázdná)
3. pomocných symbolů, jako jsou závorky “ ( ,[, ) ,] ” a čárka “,”.

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

**Poznámka**
Predikátové symboly budeme většinou značit velkými písmeny, tj. např. P, Q, R, … , P1 , P 2 , … ; konstantní symboly malými písmeny ze začátku abecedy, tj. a, b, c, … , a1 , a2 , … , a funkční symboly většinou f, g, h, … , f1 , f2 , … . Formule predikátové logiky budeme označovat malými řeckými písmeny (obdobně, jako jsme to dělali pro výrokové formule). Kdykoli se od těchto konvencí odchýlíme, tak v textu na to upozorníme.
Poznamejme, že přestože často budeme mluvit o *n*-árních predikátových symbolech a *n*-árních funkčních symbolech, v běžné praxi se setkáme jak s predikáty, tak funkcemi arity nejvýše tři. Nejběžnější jsou predikáty a funkční symboly arity 1, těm říkáme též *unární*, nebo arity 2, těm říkáme též *binární*.
Predikátové symboly arity 0 představují nestrukturované výroky (netýkají se žádného objektu). Tímto způsobem se v predikátové logice dá popsat i výrok: „Prší”.

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

1. Každá proměnná a každý konstantní symbol je term.
2. Jestliže *f* je funkční symbol arity *n* a t1, t2, … , tn jsou termy, pak f( t1 , t2 , … , tn ) je také term.
3. Nic, co nevzniklo konečným použitím pravidel 1 a 2, není term.

**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”.

**Atomické formule**
*Atomická formule* je predikátový symbol *P* aplikovaný na tolik termů, kolik je jeho arita. Jinými slovy, pro každý predikátový symbol P ∈ Pred arity *n* a pro každou *n*-tici termů t1, t2, … , tn je P( t1 , t2 , … , tn ) atomická formule.

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

1. Každá atomická formule je formule.
2. Jsou-li ϕ a ψ dvě formule, pak ( ¬ ϕ ) , ( ϕ ∧ ψ ) , ( ϕ ∨ ψ ) , ( ϕ ⇒ ψ ) , ( ϕ ↔ ψ ) jsou opět formule.
3. Je-li ϕ formule a x proměnná, pak ( ∀ x ϕ ) a ( ∃ x ϕ ) jsou opět formule.
4. Nic, co nevzniklo pomocí konečně mnoha použití bodů 1 až 3, není formule.

**Konvence**

1. Úplně vnější závorky nepíšeme. Píšeme tedy např. ( ∃ xP( x ) ) ∨ R( a,b ) místo ( ( ∃ xP( x ) ) ∨ R( a,b ) ) .
2. Spojka „negace” má vždy přednost před výrokovými logickými spojkami a proto píšeme např. ∀ x( ¬ P( x ) ⇒ Q( x ) ) místo ∀ x( ( ¬ P( x ) ) ⇒ Q( x ) ) .

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

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

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

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

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

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

**Rovnost formulí**
Dvě formule považujeme za *stejné*, jestliže se liší pouze legálním přejmenováním vázaných proměnných.
Každou formuli ϕ lze napsat tak, že každá proměnná má ve formuli buď jen volné výskyty nebo jen vázané výskyty.

**Sémantika predikátové logiky**
Nyní se budeme zabývat sémantikou formulí, tj. jejich významem a pravdivostí.

**Intepretace jazyka predikátové logiky**
*Interpretace* predikátové logiky s predikátovými symboly Pred , konstantními symboly Kons a funkčními symboly Func je dvojice ⟨ U,[ - ] ⟩ , kde

- *U* je neprázdná množina nazývaná *universum*;
- ⟨ [ - ] ⟩ je přiřazení, které
    1. každému predikátovému symbolu P ∈ Pred arity *n* > 0 přiřazuje podmnožinu $U^n$ , tj. *n*-ární relaci na množině *U*. Když n = 0, tak přiřazuje P nulu nebo jedničku.
    2. každému konstantnímu symbolu a ∈ Kons přiřazuje prvek z *U*, značíme jej [[*a*]],
    3. každému funkčnímu symbolu f ∈ Func arity *n* přiřazuje zobrazení množiny $U^n$ do *U*, značíme je [[*f*]],

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

**Kontext proměnných (konkrétní přiřazení proměnných)**
Je dána interpretace $\langle U, [-] \rangle$. Kontext proměnných je zobrazení $\rho$, které každé proměnné $x \in Var$ přiřadí prvek $\rho(x) \in U$. Je-li $\rho$ kontext proměnných, $x \in Var$ a $d \in U$, pak
$p[x:=d]$
označuje kontext proměnných, který má stejné hodnoty jako $\rho$, a liší se pouze v proměnné $x$, kde má hodnotu $d$. Kontextu proměnných $p[x:=d]$ též říkáme *update* kontextu $\rho$ o hodnotu $d$ v $x$.

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

1. Je-li term konstantní symbol $a \in Kons$, pak jeho hodnota je prvek $[a]\rho = [a]$. Je-li term proměnná $x$, pak jeho hodnota je $[x]\rho = \rho(x)$.
2. Je-li $f(t1, t2, \ldots, tn)$ term. pak jeho hodnota je $[f(t1, t2, \ldots, tn)]\rho = [f]([t1]\rho, \ldots, [tn]\rho)$.
3. Jinými slovy, hodnota termu $f(t1, t2, \ldots, tn)$ je funkční hodnota funkce $[f]$ provedené na $n$-tici prvků $[t1]\rho, \ldots, [tn]\rho$ z $U$.

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

1. Nechť $\phi$ je atomická formule. Tj. $\phi = P(t1, t2, \ldots, tn)$, kde $P$ je predikátový symbol arity $n$ a $t1, t2, \ldots, tn$ jsou termy. Pak $\phi$ je pravdivá v interpretaci $\langle U,[-] \rangle$ a kontextu $\rho$ právě tehdy, když
2. $([t1]\rho, \ldots, [tn]\rho) \in [P]$.
3. Jinými slovy: $\phi$ je v naší interpretaci pravdivá právě tehdy, když $n$-tice hodnot termů $([t1]\rho, \ldots, [tn]\rho)$ má vlastnost $[P]$.
4. Jsou-li $\phi$ a $\psi$ formule, jejichž pravdivost v interpretaci $\langle U,[-] \rangle$ a kontextu $\rho$ již známe, pak
   - $\neg\phi$ je pravdivá právě tehdy, když $\phi$ není pravdivá.
   - $\phi \wedge \psi$ je pravdivá právě tehdy, když $\phi$ a $\psi$ jsou pravdivé.
   - $\phi \vee \psi$ je nepravdivá právě tehdy, když $\phi$ i $\psi$ jsou nepravdivé.
   - $\phi \Rightarrow \psi$ je nepravdivá právě tehdy, když $\phi$ je pravdivá a $\psi$ je nepravdivá.
   - $\phi \leftrightarrow \psi$ je pravdivá právě tehdy, když buď obě formule $\phi$ a $\psi$ jsou pravdivé, nebo obě formule $\phi$ a $\psi$ jsou nepravdivé.
5. Je-li $\phi$ formule a $x$ proměnná, pak
   - $\forall x \phi(x)$ je pravdivá právě tehdy, když fromule $\phi$ je pravdivá *v každém* kontextu $p[x:=d]$, kde $d$ je prvek $U$.
   - $\exists x \phi(x)$ je pravdivá právě tehdy, když fromule $\phi$ je pravdivá *v aspoň jednom* kontextu $p[x:=d]$, kde $d$ je prvek $U$.

**Pravdivostní hodnota sentence**
Sentence $\phi$ je *pravdivá v interpretaci* $\langle U,[-] \rangle$ právě tehdy, když je pravdivá v každém kontextu proměnných $\rho$.
Poznamenejme, že pro sentence v předchozí definici jsme mohli požadovat pravdivost v alespoň jednom kontextu.

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

**Tautologie, kontradikce, splnitelná sentence**
Sentence $\phi$ se nazývá *tautologie*, jestliže je pravdivá v každé interpretaci. Sentence se nazývá *kontradikce*, jestliže je nepravdivá v každé interpretaci. Nazývá se *splnitelná*, jestliže je pravdivá v aspoň jedné interpretaci.

Také jsme mohli formulovat předchozí definice 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.
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. $P(a) \Rightarrow (\exists x P(x))$;

Následující sentence jsou splnitelné formule:

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

kde $Q$ = jsou binární predikátové symboly, + je binární funkční symbol.
Zvláštní příklady kontradikcí neuvádíme. Kontradikce jsou přesně ty formule, jejichž negace je 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 $p \wedge \neg p$.

**Splnitelné množiny sentencí**
Množina sentencí $M$ je *splnitelná* právě tehdy, když existuje interpretace $\langle U,[-] \rangle$, v níž jsou všechny sentence z $M$ pravdivé. Takové interpretaci pak říkáme *model* množiny sentencí $M$.
Množina sentencí $M$ je *nesplnitelná*, jestliže ke každé interpretaci $\langle U,[-] \rangle$ existuje formule z $M$, která je v $\langle U,[-] \rangle$ nepravdivá.
Z poslední definice vyplývá, že prázdná množina sentencí je splnitelná.

**Tautologická ekvivalence**
Tautologická ekvivalence sentencí
Řekneme, že dvě sentence $\phi$ a $\psi$ jsou *tautologicky ekvivalentní* právě tehdy, když mají stejné modely, tj. jsou pravdivé ve stejných interpretacích. Jinými slovy, mají stejnou pravdivostní hodnotu ve všech interpretacích.
Někdy se říká, že sentence jsou *sémanticky* ekvivalentní místo, že jsou tautologicky ekvivalentní.

**Poznámka**: Dá se jednoduše dokázat, že tautologická ekvivalence je relace ekvivalence na množině všech sentencí daného jazyka L a že má podobné vlastnosti jako tautologická ekvivalence formulí výrokové logiky.

**Tvrzení**
Nechť $\phi$ a $\psi$ jsou sentence. Pak platí:
$\phi$ |=| $\psi$ právě tehdy, když $\phi \leftrightarrow \psi$ je tautologie.
Tautologické ekvivalence: ( P a Q jsou unární predikátové symboly.)

1. $\neg(\forall x P(x))$ |=| $(\exists x \neg P(x))$ ,
2. $\neg(\exists x P(x))$ |=| $(\forall x \neg P(x))$ .
3. $(\forall x P(x)) \wedge (\forall x Q(x))$ |=| $\forall x(P(x) \wedge Q(x))$ ;

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

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

**Konvence**
Jestliže množina sentencí *S* je jednoprvková, tj. $S=\{\psi\}$ , pak píšeme $\psi \vDash \phi$ místo $\{\psi\} \vDash \phi$ . Je-li množina *S* prázdná, píšeme $\vDash \phi$ místo $\emptyset \vDash \phi$ .
Obdobně jako pro výrokovou logiku, dostáváme řadu jednoduchých pozorování. Pro množiny sentencí *M*, *N* a sentenci $\phi$ platí:

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

**Tvrzení**
Nechť $\phi$ a $\psi$ jsou sentence. Pak platí:
$\phi$ |=| $\psi$ právě tehdy, když $\phi \vDash \psi$ a $\psi \vDash \phi$ .

**Tvrzení**
Nechť $\phi$ a $\psi$ jsou sentence. Pak platí:
$\phi \vDash \psi$ právě tehdy, když $\phi \Rightarrow \psi$ je tautologie.

**Věta**
Pro každou množinu sentencí *S* a každou sentenci $\phi$ platí:
$S \vDash \phi$ právě tehdy, když $S \cup \{\neg \phi\}$ je nesplnitelná množina.

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

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

**Klausule**
*Klausule* je sentence taková, že všechny kvantifikátory jsou obecné ($\forall$) a stojí na začátku sentence (na jejich pořadí nezáleží) a za nimi následují literál nebo disjunkce literálů.
Ve výrokové logice jsme pro každou formuli $\alpha$ našli k ní tautologicky ekvivalentní množinu klausulí $S\alpha$ a to tak, že $\alpha$ i $S\alpha$ byly pravdivé ve stejných pravdivostních ohodnocení. Takto jednoduchá situace v predikátové logice není. Ukážeme si, jak k dané sentenci $\phi$ najít množinu klausulí $S\phi$ a to tak, že $\phi$ je splnitelná právě tehdy, když množina $S\phi$ je splnitelná.

**Rezoloventy klausulí**
Ve výrokové logice jsme rezolventy vytvářeli tak, že jsme si vždy vzali dvě klausule, které obsahovaly dvojici komplementárních literálů, a výsledná

rezolventa byla disjunkcí všech ostatních literálů z obou klausulí. Situace v predikátové logice je složitější. Postup, jak vytváříme rezolventy v predikátové logice, si ukážeme na příkladech.

**Poznámka**: Ne vždy rezolventa existuje.
**Příklad**
Najděme rezoloventu klausulí K1 = ∀x ∀y (P(x) ∨ ¬Q (x, y)) a K2 = ∀x ∀y(Q(x, y) ∨ R(y)) , kde P a R jsou unární predikátové symboly a Q je binární predikátový symbol, x, y jsou proměnné.
Klausule K1 a K2 obsahují dvojici komplementárních literálů, totiž ¬ Q( x,y ) je literál K1 a Q(x, y) je literál K2 . Rezoloventou klausulí K1 a K2 je tedy K= ∀x ∀y(P(x) ∨ R(y) ) .

**Unifikační algoritmus**
Vstup: Dva positivní literály L1 , L2 , 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 E1 := L1 , E2 := L2 , θ := ∅
2. Jsou-li E1 , E2 prázdné řetězce, stop. Množina θ určuje hledanou substituci. V opačném případě položíme E1 := E1 θ , E2 := E2 θ (tj. na E1 , E2 provedeme substituci θ ).
3. Označíme *X* první symbol řetězce E1 , *Y* první symbol řetězce E2 .
4. Je-li X=Y , odstraníme *X* a *Y* z počátku E1 a E2 . 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.
6. Je-li *Y* proměnná (a *X* nikoli), přehodíme E1 , E2 a *X*, *Y* .
7. Není-li ani *X* ani *Y* proměnná, stop. Výstup neexistuje.
8. Je-li *Y* proměnná nebo konstanta, položíme θ := θ ∪ { X/Y } . Odstraníme *X* a *Y* ze začátků řetězců E1 a E2 (spolu s čárkami, je-li třeba) a jdeme na krok 2.
9. 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.
10. V opačném případě položíme θ := θ ∪ { X/Z } , odstraníme *X* a *Z* ze začátků E1 a E2 (odstraníme čárky, je-li třeba) a jdeme na krok 2.

**Rezoluční princip**
Je obdobný jako rezoluční princip ve výrokové logice:
Je dána množina klausulí *S*. Označme
R( S ) =S ∪ { K|K je nejobecnější rezoloventa některých klausulí z S }
R 0 ( S ) =S
R i+1 ( S ) =R( R i ( S ) ) pro i ∈ N
R ★ ( S ) = ∪ { R i ( S ) |i ≥ 0 } .
Množina klausulí *S* je splnitelná právě tehdy, když R ★ ( S ) neobsahuje prázdnou klausuli *F*.
Jestliže je množina *S* konečná, existuje přirozené číslo n 0 takové, že R n 0 ( S ) = R n 0 1 . Pak R ★ ( S ) = R n 0 ( S ) .

**Skolemizace**

- Odstranění existenčních kvantifikátorů s zachováním ekvisplnitelnosti
- $\exists x P(x) \approx P(c)$, kde c je nový konstantní symbol
- $\forall x \exists y R(x, y) \approx \forall x R(x, f(x))$, kde f() je nový funkční symbol arity 1.

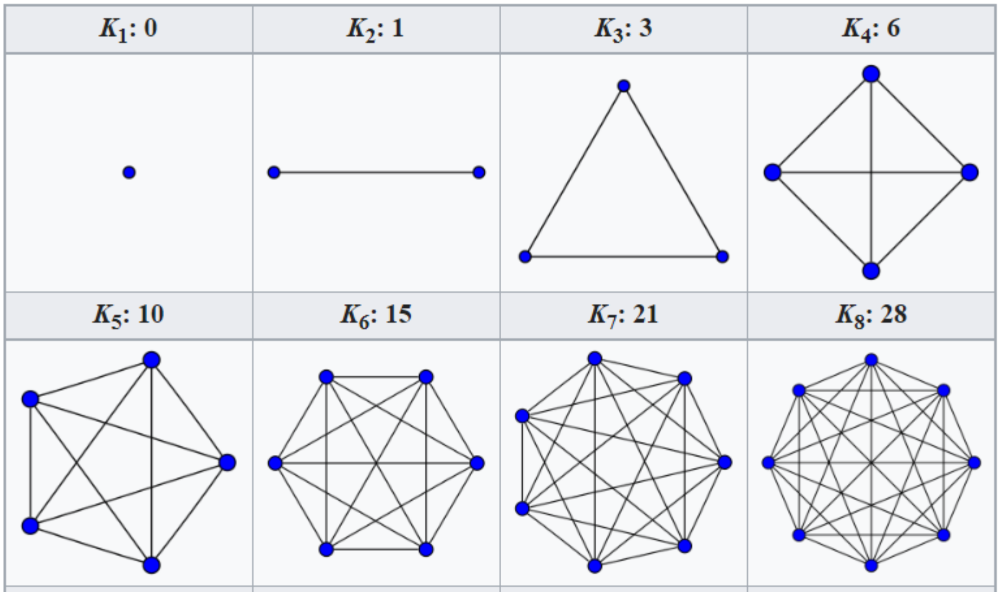
# Grafy

## Definice orientovaného grafu

Orientovaný graf je trojice G=(V,E, ε ) , kde V je konečná množina vrcholů (též zvaných uzlů), E je konečná množina jmen hran (též nazývaných orientovaných hran) a ε je přiřazení, které každé hraně e ∈ E přiřazuje uspořádanou dvojici vrcholů a nazývá se vztah incidence.
Jestliže ε(e)=(u,v) pro u,v ∈ V , říkáme, že vrchol u je počáteční vrchol hrany e a vrchol v je koncový vrchol hrany e. O vrcholech u,v říkáme, že jsou krajní vrcholy hrany e, též že jsou incidentní s hranou e. Jestliže počáteční vrchol a koncový vrchol jsou stejné, říkáme, že hrana e je orientovaná smyčka.

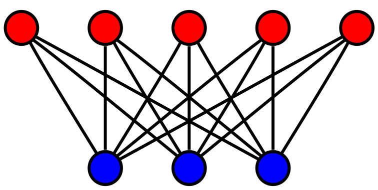
## Speciální případy grafů

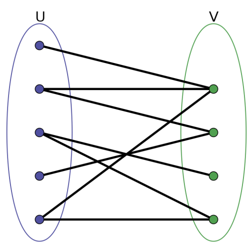
Graf o $n$ vrcholech, ve kterém $E = \binom{v}{2}$ (v! / 2! * (v - 2)!) je množina všech dvouprvkových podmnožin množiny $V$ se nazývá **úplný graf,** značí se ****$K_n$.




Vlastnosti: $|E| = n \cdot \frac{n-1}{2}$, je regulární a celý graf je svojí největší klikou.

**Bipartitní graf** je graf, jehož množina vrcholů se dá rozdělit na dvě disjunktní podmnožiny $V_1$, $V_2$ tak, že každá hrana grafu má jeden koncový vrchol ve $V_1$ a druhý ve $V_2$.

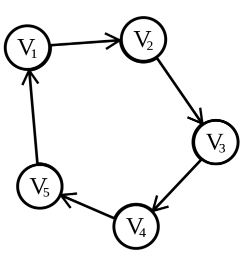


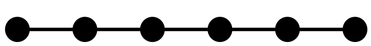



Vlastnosti: Graf je bipartitní právě tehdy, když neobsahuje kružnici liché délky.

Bipartitní graf, který obsahuje všechny možné hrany, se nazývá **úplný bipartitní graf**. Značí se $K_{m,n}$, kde $|V_1| = m$, $|V_2| = n$ jsou velikosti partit. Na obrázku s červenými a modrými vrcholy je např. $K_{5,3}$.
Vlastnosti: $|E| = m \cdot n$. Je regulární, právě když $m = n$.

**Cesta** délky $n$ je graf isomorfní grafu $V = \{0, 1, \ldots, n\}$ a s množinou hran $E = \{\{0,1\}, \{1,2\}, \ldots, \{n-1, n\}\}$. Značíme ji $P_n$.



**Kružnice** délky $n \geq 3$ je graf isomorfní grafu s množinou vrcholů $V = \{1, \ldots n\}$ a s množinou hran $E = \{\{1,2\}, \{2,3\}, \ldots \{n-1, n\}\}$. Značíme ji $C_n$.

| **Regulární graf** | **Všechny vrcholy stejný stupeň** |
|---|---|
| Podgraf | $G' = (V', E')$, kde $V' \subseteq V$, $E' \subseteq E$ |
| Faktor grafu | Podgraf $G' = (V', E')$, kde $V' = V$ |
| Podgraf indukovaný množinou $V'$ | Podgraf $G' = (V', E')$, kde $E'$ obsahuje všechny hrany, které mají oba krajní vrcholy v množině $V'$ |
| Sled | Sled délky $k$ je posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, \ldots, v_{k-1}, e_k, v_k$ taková, že hrana $e_i$ je incidentní s vrcholy $v_{i-1}$ a $v_i$. |
| Uzavřený sled | Sled, kde $v_0 = v_k$ |

| Regulární graf | Všechny vrcholy stejný stupeň |
|---|---|
| Tah (uzavřený) | Sled (uzavřený), ve kterém se neopakují hrany. |
| Eulerovský tah (uzavřený) | Tah (uzavřený), který obsahuje všechny hrany a všechny vrcholy grafu. |
| Cesta (uzavřená) | Tah (uzavřený), ve kterém se neopakují vrcholy (s výjimkou $v_0 = v_k$). |
| Cyklus | Uzavřená orientovaná cesta, ve které se neopakují vrcholy. |
| Kružnice | Uzavřená cesta, která má aspoň jednu hranu |
| Souvislý graf | Mezi každými jeho vrcholy existuje cesta. |
| Strom | Souvislý graf, který neobsahuje kružnici. |
| Kořenový strom | Orientovaný graf, který je strom a má kořen. |
| Les | Graf, který neobsahuje kružnice. |
| Prostý graf | Graf bez paralelních hran. |
| Obyčejný graf | Prostý graf bez smyček. |
| Eulerovský graf | Graf, kde existuje eulerovský tah. |
| Silně souvislý graf | Orientovaný graf, kde mezi každými dvěma vrcholy existuje orientovaná cesta. |
| Doplněk grafu | $G^{\text{dop}} = (V, \binom{V}{2} - E)$, tedy s hranami, které nejsou v původním grafu. |
| Vzdálenost vrcholů | Délka nejkratší cesty mezi dvěma vrcholy. |
| Komponenta souvislosti | Maximální souvislý podgraf. |
| Skóre grafu | Sestupně setříděná $n$-tice stupňů vrcholů. |
| Kostra grafu | Faktor, který je stromem. |
| Komponenta silné souvislosti | Maximální podgraf, který je souvislý. |
| Kořen | Vrchol u orientovaného grafu, z něhož vede orientovaná cesta do všech ostatních vrcholů. |
| Jádro | Jádro orientovaného grafu $G = (V, E)$ je množina $J \subseteq V$ jeho vrcholů taková, že mezi libovolnými dvěma nevede hrana a z každého vrcholu mimo $J$ vede aspoň jedna hrana do $J$. |
| Obarvení vrcholů | Zobrazení $b : V \to B$ takové, že žádné dva vrcholy spojené hranou nemají stejnou barvu. |
| $k$-barevný graf | Graf, který se dá obarvit $k$ barvami. |
| Barevnost grafu | Nejmenší počet barev, které jsou potřeba k obarvení jeho vrcholů. |
| Klika | Maximální podmnožina vrcholů s vlastností, že každé dva různé vrcholy jsou spojeny hranou. |
| Klikovost grafu | Počet vrcholů v nejpočetnější klice. |
| Nezávislá množina v grafu | Množina vrcholů, jejíž žádné dva vrcholy nejsou spojeny hranou. |
| Nezávislost grafu | Počet vrcholů v maximální nezávislé množině. |

# Definice neorientovaného grafu

Neorientovaný graf je trojice G=(V,E,ε), kde V je konečná množina vrcholů (též zvaných uzlů), E je konečná množina jmen hran a ε je přiřazení, které každé hraně e ∈ E přiřazuje množinu {u,v} pro vrcholy u,v ∈ V a nazývá se vztah incidence. Jestliže ε(e)={u,v} pro u,v є 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.

# Stromy

Orientovaný nebo neorientovaný graf se nazývá strom, je-li souvislý a neobsahuje-li kružnici
V každém stromu s alespoň dvěma vrcholy existuje vrchol stupně 1.
Každý strom o n vrcholech má n-1 hran.

**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.

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)

# Souvislost

**Souvislé grafy**
Řekneme, že graf je souvislý, jestliže pro každé dva vrcholy u, v grafu existuje neorientovaná cesta z u do v. Poznamenejme, že vždy existuje cesta z vrcholu u do sebe, totiž triviální cesta. Také platí, že neorientovaná cesta z vrcholu u do vrcholu v je také neorientovanou cestou z v do u.

**Komponenty souvislosti**
Máme dán 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.

# Silná souvislost

**Silně souvislé grafy**
Řekneme, že orientovaný graf G je silně souvislý, jestliže pro každé dva vrcholy u, v existuje orientovaná cesta z vrcholu u do vrcholu v a orientovaná cesta z vrcholu v do vrcholu u.
Poznámka V definici silně souvislého grafu jsme mohli požadovat pouze existenci orientované cesty z vrcholu u do vrcholu v. Je to proto, že existenci takové cesty vyžadujeme pro všechny dvojice vrcholů, tedy i pro dvojici v, u. Dále si uvědomte, že vždy existuje orientovaná cesta z vrcholu do sebe – je to triviální cesta.
Souvislý graf je silně souvislý právě tehdy, když každá hrana leží v nějakém cyklu.

# Minimální kostra

**Kostra grafu**
Je dán souvislý graf G. Faktor grafu G, který je stromem, se nazývá kostra grafu G. Připomeňme, že faktor grafu G je podgraf grafu G, který má stejnou množinu vrcholů jako G.

**Minimální kostra**
Je dán souvislý graf G spolu s ohodnocením hran c, tj. pro každou hranu e ∈ E( G ) je dáno číslo c(e) (číslo c(e) nazýváme cenou hrany e).
Minimální kostra grafu G=(V,E) je taková kostra grafu K=(V,L), že ∑e ∈ Lc( e ) je nejmenší (mezi všemi kostrami grafu G).

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

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

1. Na začátku máme L=0. Označíme S množinu všech komponent souvislosti grafu K=(V,L); tj. na začátku je s= v ;v ∈ V .
2. Dokud není graf K=(V,L) souvislý (tj. dokud S se neskládá z jediné množiny), vybereme hranu e podle následujících pravidel:
   1. E spojuje dvě různé komponenty souvislosti S,S‘ grafu K (tj. dvě množiny z S)
   2. A pro S nebo S‘ je nejlevnější hranou která vede z komponenty ven
3. Hranu e přidáme do množiny L a množiny S a S‘ nahradíme jejich sjednocením.
4. Postup ukončíme, jestliže jsme přidali n-1 hran (tj. jestliže se S skládá z jediné množiny).

**Kruskalův algoritmus**
Jedná se o modifikaci obecného postupu pro hledání minimální kostry:

1. Setřídíme hrany podle ceny do neklesající posloupnosti, tj. c( e1 ) ≤ c( e2 ) ≤ … .. ≤ c( em ) . Položíme L=0, S= v ;v ∈ V .
2. Probíráme hrany v daném pořadí. Hranu ei přidáme do L, jestliže má oba krajní vrcholy v různých množinách S,S‘ ∈ S . V S množiny S a S‘ nahradíme jejich sjednocením. V opačném případě hranu přeskočíme.
3. Algoritmus končí, jestliže jsme přidali n-1 hran (tj. S se skládá z jediné množiny).

**Primův algoritmus**
Jedná se o modifikaci obecného postupu pro hledání minimální kostry:

1. Vybereme libovolný vrchol v. Položíme L=0, 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.

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

1. Pro každou hranu e s počátečním vrcholem PV(e) ϵ K platí KV(e) (NENÍ)ϵ K. (Neexistuje hrana, která by vedla z množiny K do sebe)

2. Pro každý vrchol v, který neleží v K, existuje hrana e s PV(e)=v a KV(e) є K. (z každého vrcholu, který leží mimo K, se můžeme dostat po hraně zpět do K)

// Věci níže je asi dobré vědět, není v požadavkách

# Eulerovy grafy

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

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

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

Aplikace eulerovských tahů

- Kreslení s co nejmenším počtem tahů
- Úloha čínského pošťáka
- De Bruijnova posloupnost

V silně souvisém 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.

**Postup na hledání uzavřeného orientovaného eulerovského tahu**
Vybereme libovolný vrchol v grafu. Protože graf je souvislý, v každém vrcholu začíná i končí alespoň jedna hrana. Z vrcholu v vytváříme náhodně orientovaný tah; tj. procházíme hrany tak, abychom žádnou hranou neprošli dvakrát. Takto pokračujeme, dokud je to možné, tj. dokud se nevrátíme do výchozího vrcholu v a ve vrcholu v již nezačíná žádná dosud nepoužitá hrana. Tím jsme dostali uzavřený tah. Jestliže tento tah obsahuje všechny hrany, je to hledaný uzavřený eulerovský tah. Neobsahuje-li takto zkonstruovaný tah všechny hrany, pak na tahu existuje vrchol w takový, že v něm začíná nepoužitý hrana. (To vyplývá ze souvislosti grafu.) Získaný tah ve vrcholu w rozpojíme a náhodně konstruujeme uzavřený tah (z dosud nepoužitých hran) začínající a končící ve vrcholu w. Tento postup opakujeme, dokud nedostaneme tah obsahující všechny hrany.

**Tvrzení**
V souvislém orientovaném grafu existuje otevřený orientovaný eulerovský tah právě tehdy, když existují vrcholy u1, u2 takové, že d-( u1 ) =d+( u1 ) +1,d-( u2 ) =d+( u2 ) -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.
Tvrzení Je dán souvislý neorientovaný graf G s 2k vrcholy lichého stupně. Pak existuje k hranově disjunktních otevřených tahů takových, že každá hrana grafu G leží v právě jednom z těchto tahů. Ke grafu G přidáme k hran a to tak, že každá nově přidaná hrana spojuje vždy dva vrcholy lichého stupně. Tím dostaneme eulerovský graf G'' (ano, každý vrchol má již sudý stupeň). V grafu G‘‘ najdeme eulerovský uzavřený tah. Jestliže z něj odstraníme všechny přidané vrcholy, rozpadne se na k hranově disjunktních tahů. Tyto tahy splňují podmínky tvrzení.

# Hamiltonovské grafy

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

**Hamiltonovské cesty, kružnice, cykly**
Je dán graf G. Otevřená cesta se nazývá hamiltonovská cesta, obsahuje-li všechny vrcholy (a tudíž všechny vrcholy přesně jedenkrát). Obdobně hamiltonovská kružnice je kružnice, která obsahuje každý vrchol grafu; hamiltonovský cyklus je cyklus, který obsahuje každý vrchol v grafu.
Úlohy dělíme na existenční a optimalizační. V existenční úloze jde o to, zjistit zda v daném grafu existuje hamiltonovská cesta, kružnice nebo cyklus. V optimalizačních úlohách máme hrany grafu navíc ohodnoceny délkami a požaduje se nalezení hamiltonovské cesty, kružnice nebo cyklu s co nejmenším součtem délek jednotlivých hran tvořících cestu, kružnici nebo cyklus.
Na rozdíl od hledání eulerovských tahů, je hledání hamiltonovských cest, kružnic nebo cyklů velmi obtížná úloha. Přesněji, zjištění, zda v daném grafu existuje hamiltonovská cesta, kružnice nebo cyklus je tzv. NP-úplná úloha. Přesto, nebo právě proto, jsou úlohy tohoto typu v praxi rozšířené.

Aplikace

- Problém obchodného cestujícího
- Dopravní úlohy
- Plánování procesů

Jednoduché nutné podmínky pro existenci hamiltonovské cesty, kružnice nebo cyklu

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

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

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

//See also: Metoda větví a mezí

# Nezávislé množiny

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

**Maximální nezávislá množina**
Je dán graf G. Nezávislá množina N se nazývá maximální nezávislá množina, jestliže jakákoli její nadmnožina už není nezávislá. Jinými slovy, N je maximální nezávislá množina, jestliže pro každý vrchol v, který neleží v N, existuje vrchol w є N takový, že v G existuje hrana mezi v a w.

**Nezávislost grafu**
Je dán neorientovaný nebo orientovaný graf G. Počet vrcholů v nejpočetnější nezávislé množině grafu G se nazývá nezávislost grafu G a značíme jej α(G).
Nejpočetnější nezávislá množina je jistě také maximální, ale ne každá maximální nezávislá množina je současně nejpočetnější.

**Poznámka**
Jádro orientovaného grafu G 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 splňovat obě podmínky (viz výše).
Úplný neorientovaný graf G nazýváme úplným grafem, jestliže je prostý, nemá smyčky a každé dva různé vrcholy jsou spojené hranou. Úplný neorientovaný graf G s n vrcholy má (n(n-1))/2 hran.

# Obarvení grafu

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

Množina vrcholů obarvená stejnou barvou tvoří nezávislou množinu grafu. Graf je jednobarevný právě tehdy, když nemá žádnou hranu.
Graf G je dvoubarevný právě tehdy, když neobsahuje kružnici liché délky.

**Dvoubarevné grafy**
Zjistit, zda je daný graf dvoubarevný, se dá jednoduchou modifikací prohledávání do šířky: Provedeme prohledání grafu do šířky. Vrcholům, které ležely v sudých hladinách, přiřadíme barvu 1; vrcholům, které ležely v lichých hladinách, přiřadíme barvu 2.
Jestliže graf neobsahoval kružnici liché délky, jedná se o obarvení grafu a graf je tedy dvoubarevný. Vede-li hrana mezi dvěma vrcholy v hladinách stejné parity, obsahuje graf kružnici liché délky a není proto dvoubarevný.
Poznámka Zjistit, zda daný graf je tříbarevný, je těžký problém (obecně NP-úplný problém). Pro každý graf G, který mý m hran platí
χ ( G ) ≤ 1 2 + 2m+ 1 4
Tvrzení označíme ∆ největší stupeň vrcholu grafu G. Pak χ(G)<=∆+1 Sekvenční barvení
Následující postup obarví graf ∆+1 barvami. Označíme množinu barev B={1,…., ∆+1}.

1. Seřadíme vrcholy do posloupnosti (libovolně). Např. v1,v2,….,vn
2. Probíráme vrcholy v tomto pořadí a vrcholu vi přiřadíme vždy tu nejmenší barvu, kterou nemá žádný jeho soused vrcholu.

Tento algoritmus 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 n/2 barvami (kde n je počet vrcholů grafu)
Tvrzení Pro každý neorientovaný graf G bez smyček platí: α( G ) +χ( G ) ≤ n+1 . Kde n je počet vrcholů grafu G.

//See also: Biparitní grafy, klika v grafu, doplňkový graf