Matematické Základy Informatiky (FI: IB000)

Prof. RNDr. Petr Hliněný, Ph.D.

hlineny@fi.muni.cz

17. ledna 2023

Obsažný a dobře přístupný úvod do nezbytných formálních matematických základů moderní informatiky, doplněný řadou interaktivních cvičení v IS MU.

Výukový text pro předmět IB000 na FI MU od roku 2012, navazující z velké části na původní výukové texty Úvodu do Informatiky z dřívějších let.

Verze 3.03 (2023)

0.1 O tomto textu a jeho studiu

Vážení čtenáři,

dostává se vám do rukou výukový text Matematické Základy Informatiky, který je primárně určený pro studenty stejnojmenného předmětu na FI MU Brno. Seznamuje čtenáře s několika formálními matematickými oblastmi důležitými pro úspěšné studium moderní informatiky.

Náš text svým obsahem volně navazuje na původní slidy předmětu IB000 sepsané A. Kučerou do roku 2005 a rozšířené autorem v letech 2006–11 o volný text a komentáře. Od roku 2012 však byl obsah některých pasáží podstatně změněn či přímo vyměněn za nový – nejzřetelnější změnou je zahrnutí úvodu do teorie grafů. K další výrazné změně dochází v letech 2017–18, kdy je posílena látka matematické logiky a důkazů matematickou indukcí a zároveň omezeny části pojednávající o formalizaci algoritmů (ve prospěch předmětu IB002).

Učební text je psán strukturovaným matematickým přístupem, s velkým důrazem na přesnost a formalitu vyjadřování v nutných partiích. Na druhou stranu jsou strohá matematická vyjádření pokud možno doplněna obsáhlými neformálními komentáři, které mají studentům ulehčit pochopení látky. V žádném případě však čtenáři nemají zaměňovat neformální komentáře za matematické definice – v případě nejasností vždy platí to, co přesně říká formální definice.

Mimo textu samotného (jak jej zde vidíte) jsou z téhož zdoje vytvářeny i přednáškové slidy předmětu, které najdete například v IS MU. Slidy však pochopitelně obsahují jen část textu a jsou jinak formátovány.

Interaktivní osnova a odpovědníky

`http://is.muni.cz/el/1433/podzim2023/IB000/index.qwarp`

Nedílnou součástí celého výukového textu jsou interaktivní osnova předmětu IB000 v IS MU a z ní odkazované online odpovědníky sloužící k procvičování probrané látky na jednoduchých i složitějších příkladech. To je také důvod, proč tento text obsahuje jen poskrovnu příkladů k řešení k probírané látce – od čtenáře očekáváme, že si látku *bude procvičovat sám online na zmíněných odpovědnících*, obsahujících až tisíce příkladů desítek typů. Třebaže pro studenty předmětu IB000 jsou organizována také každotýdenní cvičení, ta nemohou zcela nahradit potřebnou rutinu získanou opakovaným samostatným procvičením látky. K praktickému pochopení přednesených znalostí i k jejich budoucím aplikacím je důkladné procvičení nezbytné.

Obsah

1 Základní formalismy matematiky

Úvod

Začněme nejprve několika obecnými poznámkami ke smyslu a směřování celého našeho předmětu. Jak sami poznáte, studium informatiky neznamená jen „naučit se nějaký programovací jazyk“, nýbrž zahrnuje celý soubor dalších relevantních předmětů, mezi nimiž najdeme i formální (matematicko–teoretické) základy moderní informatiky. A právě odborný nadhled nad celou informatikou včetně nezbytné formální teorie odliší od obyčejného počítačového dělníka („počítačové lopaty“) skutečného a mnohem lépe placeného IT experta.

Na tomto místě přichází náš odborný předmět Matematické Základy Informatiky, který má za úkol vás na studium formálních základů moderní informatiky připravit. K tomu je nezbytné začít poněkud zostra – zdůrazněním přesného matematického vyjadřování, výrokové logiky a principů logických úsudků a matematických důkazů. Nelekejte se však hned v první lekci a s chutí se dejte do dalšího studia, po překonání případného prvotního šoku to nebude až tak obtížné...

Cíle

Hlavním cílem této úvodní lekce je rozebrat formální strukturu a zápis matematických tvrzení (vět). Při tomto rozboru se naučíte chápat smysl matematických vyjádření (výroků) pohledem výrokové logiky a pracovat s touto formalizací. Také zjistíte, jak se v matematice správně argumentuje a jak by měl vypadat matematický důkaz.

1.1 Význam matematických tvrzení

Matematika coby vědní disciplína (tudíž i teoretická informatika jako její součást) se vyznačuje velmi přísnými formálními požadavky na korektnost vyjadřování a argumentace. Proto si již od prvních krůčků v tomto předmětu musíme ujasnit, jak mají matematická tvrzení (nazývaná prostě matematické *věty*) správně vypadat a jaký je jejich význam.

Matematické tvrzení je obvykle vysloveno ve tvaru

„Jestliže platí *předpoklady*, pak platí *závěr*“.

Tomuto tvaru se odborně říká *implikace*. Pro pochopení významu je klíčové vždy správně identifikovat, co jsou v dané větě ony zmíněné *předpoklady* a co je *závěrem*. A k dosažení této dovednosti nezbývá, než se cvičit a cvičit na příkladech...

`Komentář:` Příklady běžné formulace matematických vět:

* Je-li $x > 1$, pak platí $x^2 > x$.
* Konečná množina má konečně mnoho podmnožin.
* $\sin^2(\alpha) + \cos^2(\alpha) = 1$.
* Graf je rovinný, jestliže neobsahuje podrozdělení K_5 nebo $K_{3,3}$.

Co přesně nám uvedené matematické věty říkají? Všimněme si, že mnohdy jsou některé „samozřejmé“ předpoklady vět vynechány, například, že x je reálné číslo nebo že α ve třetím bodě je úhel, případně se tvrzení implicitně odvolávají na obecně známé definice, například, co je to graf K_5 ve čtvrtém bodě. Často nám k lepšímu pochopení toho, co je tvrzeno a je třeba dokázat, pomůže pouhé rozepsání definic pojmů, které se v dané větě vyskytují.

O pravdivosti implikace

Hned na začátek výkladu zdůrazňujeme, jak moc důležité je pochopit správný logický význam matematického tvrzení vysloveného zmíněnou formou implikace („jestliže ..., pak ...“). *Pravdivost* takového tvrzení je třeba chápat v následujícím významu:

Pro každou situaci, ve které jsou splněny všechny předpoklady, je platný i závěr tvrzení. (*)

Volně řečeno, z předchozího kritéria (*) vyplývá, že pokud předpoklady nejsou splněny nebo jsou sporné, tak celé tvrzení je platné bez ohledu na pravdivost závěru!

Příklad 1.1. *Je pravdivé následující matematické tvrzení?*

> **Věta**. *Mějme dvě kuličky, červenou a modrou. Jestliže červená kulička je těžší než modrá a zároveň je modrá kulička těžší než ta červená, tak jsou obě kuličky ve skutečnosti zelené.*
>
> *„To přece nemůže být pravda, jak může být jedna kulička těžší než druhá a naopak zároveň? Jak mohou být nakonec obě zelené? To je celé nějaká blbost...“*

Ano, výše uvedené jsou typické laické reakce na uvedenou větu. Přesto však tato věta pravdivá je! Stačí se vrátit o kousek výše ke kritériu – *Pro každou situaci, ve které jsou splněny všechny předpoklady, je platný i závěr tvrzení* – které je zjevně naplněno. Nenaleznete totiž situaci, ve které by byly splněny oba předpoklady zároveň, a tudíž ve všech takových neexistujících situacích si můžete říkat cokoliv, třeba že kuličky jsou zelené. □

Příklad 1.2. *Anna a Klára přišly na přednášku a usadily se do lavic. Proč je pravdivé toto matematické tvrzení?*

> **Věta**. *Jestliže Anna sedí v první řadě lavic a zároveň Anna sedí v poslední řadě lavic, tak Klára nesedí ve druhé řadě lavic.*

Opět je třeba se pečlivě zamyslet nad *významem předpokladů a závěru*. Avšak tentokrát není situace předpokladů tak triviálně sporná, jako byla v Příkladu 1.1. Kdy tedy mohou nastat oba předpoklady (o tom, kde sedí Anna) zároveň? Když první řada lavic je zároveň řadou poslední. Neboli posluchárna má jen (nejvýše) jednu řadu lavic a Klára tudíž v druhé řadě nemůže sedět. Pravdivost celé věty je tímto potvrzena. □

1.2 Úvod do matematického dokazování

S přísnými formálními požadavky na korektnost vyjádření a argumentace v matematice se pojí otázka, jak správně svá tvrzení zdůvodňovat. Krátce lze říci, že korektně postavená matematická argumentace musí vést od přijatých předpokladů v elementárních krocích směrem k požadovanému závěru (a nikdy ne naopak!).

Definice 1.3. ***Matematický důkaz*** .
Uvažme matematickou *větu* (neboli tvrzení) tvaru

„Jestliže platí *předpoklady*, pak platí *závěr*“.

Důkaz této věty je konečná posloupnost tvrzení, kde

- každé tvrzení je buď

  - ∗ *předpoklad*, nebo
  - ∗ obecně přijatá „pravda“ – *axiom*, nebo
  - ∗ plyne z předchozích a dříve dokázaných tvrzení podle nějakého „akceptovaného“ logického principu – *odvozovacího pravidla*;
- poslední tvrzení je *závěr*.

`Komentář:` O potřebné úrovni formality matematických důkazů a o běžných důkazových technikách se dozvíme dále v příštích lekcích. Pro úplný začátek si jen celou problematiku uvedeme názornými ukázkami. Čtěte si tyto ukázky pečlivě, i pokud jste se již s matematickým dokazováním setkali a (či nebo) vám následující příklady připadají zcela primitivní. Později, u složitějších důkazů, se právě k takovým jednoduchým ukázkám můžete vracet a lépe na nich pochopit, jak to správně a přesně v matematice „chodí“.

Příklad 1.4. *Uvažujme následující matematické tvrzení (které jistě už znáte).*

Věta. *Jestliže x je součtem dvou lichých čísel, pak x je sudé.*

`Poznámka` pro připomenutí:

- *Sudé* číslo je celé číslo dělitelné 2, tj. tvaru $2k$, kde k je celé číslo (ano, i 0 či -2 jsou sudá celá čísla a jsou dělitelná dvěma).
- *Liché* číslo je celé číslo nedělitelné 2, tj. tvaru $2k+1$, kde k je celé.

`Důkaz` postupuje v následujících *formálních krocích*:

	tvrzení	zdůvodnění
1)	$a = 2k+1$, k celé	předpoklad
2)	$b = 2l+1$, l celé	předpoklad
3)	$x = a+b = 2k+2l+1+1$	z 1,2) a komutativity sčítání (axiom)
4)	$x = 2(k+l)+2\cdot 1$	ze 3) a distributivnosti násobení (axiom)
5)	$x = 2(k+l+1)$	ze 4) a opět distributivnosti násobení
6)	$x = 2m$, m celé	z 5) a $m = k+l+1$ je celé číslo (axiom)

□

Příklad 1.5. *Dokažte následující tvrzení:*

Věta. *Jestliže x a y jsou racionální čísla pro která platí $x < y$, pak existuje racionální číslo z pro které platí $x < z < y$.*

`Důkaz` po krocích (s již trochu méně formálním zápisem) zní:

1) Nechť $z = \frac{x+y}{2} = x + \frac{y-x}{2} = y - \frac{y-x}{2}$.
2) Číslo z je racionální, neboť x a y jsou racionální.
3) Platí $z > x$, neboť $\frac{y-x}{2} > 0$.
4) Dále platí $z < y$, neboť opět $\frac{y-x}{2} > 0$.
5) Celkem $x < z < y$.

Jak čtenář asi sám vidí, tento strohý formální zápis (i když matematicky kompletní) si zaslouží alespoň krátký vysvětlující komentář. Takový komentář, ať už se objeví před nebo hned po formálních argumentech, sice není součástí důkazu samotného, velmi však napomáhá správnému pochopení a má své nezastupitelné místo. Nezapomínejte na to ve svých vlastních budoucích matematických důkazech.

Konkrétně v tomto příkladě je velmi vhodné poznamenat, že klíčový krok (1) popisuje námi vymyšlenou (prostě uhodnutou) algebraickou konstrukci, která vede k požadovanému číslu z. Zbylé kroky (2–5) pak jen snadno zdůvodňují, že nalezené z má všechny požadované vlastnosti. □

`Poznámka`. Alternativní matematické formulace Věty z Příkladu 1.5 mohou znít:

- Pro každé $x, y \in \mathbb{Q}$, kde $x < y$, existuje $z \in \mathbb{Q}$ takové, že $x < z < y$.
- Množina racionálních čísel je *hustá*.

1.3 Výroky

Důležitým *pevným mostem* mezi běžnou mluvou a přesným matematickým formalismem je pojem výroku. Jeho správné uchopení nám pomůže při chápání významu matematických výroků i v práci s nimi, což je klíčovou náplní tohoto oddílu.

Definice 1.6. ***Výrok*** v přirozené mluvě:
V běžné mluvě za *výrok* považujeme (každé) tvrzení, o kterém má smysl platně prohlásit, že je *buď* pravdivé, *nebo* nepravdivé.

`Komentář:` Ukážeme si několik příkladů – které z nich jsou výroky?

* Dnes už v Brně pršelo.
* Předmět FI: IB000 se vyučuje v prvním ročníku.
* Platí $2 + 3 = 6$.
* To je bez problémů. *(Co?)*
* Platí $x > 3$.
* Pro každé celé číslo x platí, že $x > 3$.

Všimněte si, že pravdivost výroku by mělo být možné rozhodnout bez skrytých souvislostí (kontextu), a proto čtvrtý a pátý příklad za výroky nepovažujeme. Poslední příklad již zase výrokem je, neboť obor hodnot x je jednoznačně vymezen tzv. kvantifikací.

Z více jednoduchých výroků vytváříme výroky složitější pomocí tzv. *logických spojek*.

`Komentář:` Následuje několik dalších příkladů.

* Množina $\{a, b\}$ má více než jeden prvek a není nekonečná.
* Jestliže Karel váží přes 90 kg, nejedu s ním výtahem.
* Jestliže má tato kráva 10 nohou, pak mají všechny domy modrou střechu.

Zastavme se na chvíli nad posledním výrokem. Co nám říká? Je pravdivý? Skutečně mají všechny domy modrou střechu a před námi stojí kráva s 10 nohama? (Ano, výrok je pravdivý, viz definice či pravdivostní tabulka implikace.)

Schopnost porozumět podobným větám je součást lidského způsobu uvažování a z tohoto hlediska nemá přímou souvislost s matematikou (je to „*přirozená logika*“). *Formální (matematická) logika* pak v podobném duchu definuje jazyk matematiky a přitom odstraňuje nejednoznačnosti přirozeného jazyka.

Ukázka argumentace s výroky

Matematické věty jsou často tzv. složené („platí ... a/nebo ..., tudíž ...“) a v tom případě je třeba umět správně vyvodit celkovou platnost z platnosti jednotlivých elementů. Tomuto se budeme věnovat přesněji v Oddíle 1.4, ale již nyní si pro ukázku rozebereme následující logickou hádanku, navazující na téma Příkladu 1.2.

Příklad 1.7. *Mějme trojici studentů X, Y, Z sedících v posluchárně na přednášce IB000 takových, že pro ně platí:*

a) X sedí v první řadě,

b) Y sedí v řadě hned za X,

c) Z sedí ve stejné řadě jako Y,

d) Z sedí ve druhé řadě.

Věta. *Jestliže zároveň platí (a),(b),(c),(d), pak student Z sedí ve druhé řadě.*

Tvrzení je samozřejmě triviální díky předpokladu (d), avšak úkolem je zjistit, které všechny minimální výběry z předpokladů (a),(b),(c),(d) ještě zajistí platnost uvedeného tvrzení.

Pro začátek osvětlíme (blíže viz Oddíl 7.3) význam slova *minimální*, kdy je myšlen výběr předpokladů takový, že odebráním libovolného z nich již platnost uvedeného tvrzení „Z sedí ve druhé řadě“ bude porušena.

Okamžitou odpovědí je samotný předpoklad (d), který je zřejmě minimální (bez předpokladů si může Z sednout i do první řady pro jakékoliv X, Y) a dostačující. Je to však úplná odpověď, nebo lze vybrat i jiné minimální předpoklady? Ano, ještě lze vybírat jinak – po vynechání (d) stále postačuje kombinace předpokladů (a),(b),(c) k vyslovení závěru „Z sedí ve druhé řadě“ (je to jasné?). Na druhou stranu, pokud kterýkoliv z předpokladů (a),(b),(c) vynecháme, rozebráním těchto tří možností najdeme platná rozesazení (tj. konkrétní protipříklady), při kterých zbylé dva předpoklady budou splněny, avšak Z v druhé řadě nebude:

- Například pokud vynecháme (b), může Y sedět v první řadě spolu s X a stejně tak Z. Formálně uvádíme *konkrétní protipříklad* studentů X, Y, Z, kteří sedí všichni v první řadě a tudíž splňují podmínky (a) a (c), ale nesplňují závěr, že Z sedí ve druhé řadě.
- Dále vezměme studenty X, Z v první řadě a Y ve druhé řadě. Tento protipříklad splňuje podmínky (a) a (b), ale zase nesplňuje závěr.
- Nakonec vezměme studenty X v druhé řadě a Y, Z ve třetí řadě. Tento protipříklad splňuje podmínky (b) a (c), ale zase nesplňuje závěr.

Takže výběr předpokladů (a),(b),(c) je minimální.

Pozor, ještě k úplnému vyřešení úlohy zbývá poslední důležitý krok – zdůvodnit, proč žádný jiný minimální výběr předpokladů úloze nevyhovuje. Zde je třeba rozebrat možnosti: Pokud náš hypotetický jiný výběr předpokladů obsahuje (d), pak nebude minimální, neboť lze odebrat cokoliv jiného, pokud (d) zůstane. Naopak výběr neobsahující (d) musí podle výše uvedeného obsahovat všechny tři předpoklady (a),(b),(c), neboť jinak není dostačující. A tím jsme hotovi – všechny minimální výběry předpokladů řešící úlohu jsou dva, výběr (a),(b),(c) a samotné (d). □

1.4 Formální výroková logika

Všimněte si, že podle Definice 1.6 každému výroku běžné mluvy prostě můžeme *přiřadit logickou hodnotu* 0 (*false*) nebo 1 (*true*) a dále se nestarat o jazykový význam... Proto jazykové výroky v matematice můžeme nahradit *výrokovými proměnnými*, které značíme velkými písmeny $A, B, C, \ldots$ a přiřadíme jim hodnotu 0 nebo 1.

`Definice`: *Výroková formule* (značíme $\varphi, \sigma, \psi, \ldots$) vzniká z výrokových proměnných pomocí *závorek* a logických spojek $\neg$ *negace* a $\Rightarrow$ *implikace*. Zároveň používáme v zápise následujících zkratek

* $\varphi \vee \psi$ (*disjunkce* / „nebo“) je jiný zápis formule $\neg\varphi \Rightarrow \psi$,
* $\varphi \wedge \psi$ (*konjunkce* / „a“) je jiný zápis formule $\neg(\neg\varphi \vee \neg\psi)$,
* $\varphi \Leftrightarrow \psi$ (*ekvivalence*) je jiný zápis formule $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$.

`Poznámka`: Uvedená definice je instancí tzv. *induktivní definice*, jíž se budeme blíže věnovat v Lekci 5, a proto ji zde zatím uvádíme jen v hodně zjednodušené podobě.

`Komentář:` Při zápise výrokových formulí je potřeba dávat pozor na správné závorkování, aby formule měla jednoznačný význam. Na intuitivní úrovni to ilustrujeme takto:

Správně A, $(A) \Rightarrow (B)$, $A \Rightarrow B$, $\neg A \Rightarrow B$, $A \vee B \vee \neg C$

a nesprávně $A \Rightarrow B \Rightarrow C$ – znamená toto $(A \Rightarrow B) \Rightarrow C$ nebo $A \Rightarrow (B \Rightarrow C)$?

`Komentář:` Výrokové logické spojky konjunkce a disjunkce mají přirozené jazykové vyjádření „a“ a „nebo“ (jen pozor, abyste při disjunkci pak nebrali „nebo“ jako výlučné, tedy bez uvažování platnosti obou možností zároveň). Pro logickou ekvivalenci jednoslovné a přirozené vyjádření v našem jazyce není, ale nejblíže krátkému a přesnému jazykovému vyjádření je obrat „*právě když*“, který mimo jiné používáme níže. Tento obrat však není jazykově pěkný a ve složitějších souvětích často vidíme víceslovná vyjádření ekvivalence mezi větami souvětí; příkladům se budeme věnovat v Oddíle 4.2.

Matematický význam logickým formulím pak dodává následující definice.

Definice 1.8. ***Sémantika*** (význam) výrokové logiky.
Nechť *valuace* (ohodnocení) je funkce $\nu : Prom \to \{0, 1\}$ na všech (dotčených) výrokových proměnných. Pro každou valuaci ν definujeme funkci $\mathcal{S}_\nu(\sigma)$, *vyhodnocení* formule σ, induktivně (tj. po krocích) takto:

* $\mathcal{S}_\nu(A) = \nu(A)$ pro každé $A \in Prom$.
* $\mathcal{S}_\nu(\neg\varphi) = \begin{cases} 1 & \text{jestliže } \mathcal{S}_\nu(\varphi) = 0; \\ 0 & \text{jinak.} \end{cases}$
* $\mathcal{S}_\nu(\varphi \Rightarrow \psi) = \begin{cases} 0 & \text{jestliže } \mathcal{S}_\nu(\varphi) = 1 \text{ a } \mathcal{S}_\nu(\psi) = 0; \\ 1 & \text{jinak.} \end{cases}$

`Poznámka`: Tento předpis (Def. 1.8) podává nejen *definici* funkce $\mathcal{S}_\nu$, ale také návod na to, jak ji pro daný argument *vypočítat*.

Pro odvozené logické spojky disjunkce, konjunkce a ekvivalence dostaneme následující zcela přirozený výsledek (což potvrzuje, že jsme definici sémantiky zvolili správně).

Tvrzení 1.9. *Přímým důsledkem Definice 1.8 a zkratek '$\vee$' za $\neg\varphi \Rightarrow \psi$, '$\wedge$' za $\neg(\neg\varphi \vee \neg\psi)$ a '$\Leftrightarrow$' za $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$, jsou tato fakta:*

* $\mathcal{S}_\nu(\varphi \vee \psi) = 1$ právě když $\big(\mathcal{S}_\nu(\varphi) = 1$ nebo $\mathcal{S}_\nu(\psi) = 1\big)$.
* $\mathcal{S}_\nu(\varphi \wedge \psi) = 1$ právě když $\big(\mathcal{S}_\nu(\varphi) = 1$ a současně $\mathcal{S}_\nu(\psi) = 1\big)$.
* $\mathcal{S}_\nu(\varphi \Leftrightarrow \psi) = 1$ právě když platí jedna z následujících podmínek
  - $\mathcal{S}_\nu(\varphi) = 1$ a současně $\mathcal{S}_\nu(\psi) = 1$,
  - $\mathcal{S}_\nu(\varphi) = 0$ a současně $\mathcal{S}_\nu(\psi) = 0$.

□

Pravdivostní tabulky

V praxi často vyhodnocení logické výrokové formule zapisujeme do tzv. *pravdivostní tabulky*. Tato tabulka typicky má sloupce pro jednotlivé proměnné, případné „meziformule“ (pomůcka pro snazší vyplnění) a výslednou formuli. Řádků je 2^p (počet valuací), kde p je počet použitých proměnných.

Pro naše účely postačí uvést pravdivostní tabulku instruktážním příkladem. Čtenáři nechť si vyplní podle Definice 1.8 vlastní pravdivostní tabulky dalších výrokových formulí, viz také příslušné cvičné odpovědníky v IS MU.

Příklad 1.10. *Jaká je pravdivostní tabulka pro formuli* $(A \Rightarrow B) \vee B \vee C$*?*

A	B	C	$A \Rightarrow B$	$(A \Rightarrow B) \vee B \vee C$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

□

Splnitelnost formulí a tautologie

`Definice`: Formule φ je *splnitelná*, pokud pro některou valuaci ν platí, že $\mathcal{S}_\nu(\varphi) = 1$. Formule je nesplnitelná (říká se také *kontradikce*), pokud není splnitelná. Formule φ je vždy pravdivá, neboli výroková *tautologie*, psáno $\models \varphi$, pokud pro každou valuaci ν platí, že $\mathcal{S}_\nu(\varphi) = 1$.

Řekneme, že dvě formule φ, ψ jsou *ekvivalentní*, právě když $\models \varphi \Leftrightarrow \psi$.

Tvrzení 1.11. *Následující formule jsou tautologiemi:*

* $\models A \vee \neg A$
* $\models \neg\neg A \Leftrightarrow A$
* $\models (A \wedge (A \Rightarrow B)) \Rightarrow B$
* $\models (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$
* $\models (\neg A \Rightarrow (B \wedge \neg B)) \Rightarrow A$

`Komentář:` Jak poznáme tautologie v pravdivostní tabulce? Snadno, v posledním sloupci jsou samé hodnoty 1. A jak ekvivalentní formule? Jejich poslední sloupce jsou shodné.

Navazující studium

Látka této úvodní lekce má velmi široký záběr. S potřebou formálního zápisu tvrzení a důkazů se setkáte nejen v tomto, ale i ve všech dalších matematických předmětech, které zároveň studujete či budete studovat, a principy budou stále stejné. Proto se je snažte pochopit a zažít už na zde uvedených jednoduchých příkladech a vše si procvičte. A pokud vám stále „jde hlava kolem“ z logického matematického vyjadřování (a to jsme teprve u pouhé výrokové logiky!), můžete se také zkusit podívat na hezkou úvodní (středoškolskou) knížku Antonína Sochora – Logika pro všechny ochotné myslet, Karolinum 2011.

2 Matematická logika a důkazy

Úvod

Po úvodním představení matematických formalismů již čtenář ví o výrokové logice a její užitečnosti, ale taký si jistě všiml velkých omezení samotné výrokové logiky, spočívajících především v nemožnosti využívat vnějších souvislostí (kontextu) ve výrocích a jejich vyhodnocení. Například, mluvíme-li v jednom výroku o studentovi s červeným svetrem a v jiném výroku o studentovi s brýlemi, výroková logika nám nijak neumožňuje nastolit, že se má u obou výroků jednat o stejného (nejmenovaného) studenta ze skupiny.

Zhruba řečeno, výrokové logice chybí možnost „parametrizace" výroků, což znamená zavedení omezeného a dobře definovaného kontextu pro každé tvrzení, ve kterém pak můžeme už jednotlivé výroky vyhodnocovat podle pravidel výrokové logiky. Takovou parametrizaci si lze v jednoduché ukázce představit jako proměnnou X, jež třeba nabývá hodnot ze skupiny našich studentů a umožňuje nám hovořit v různých výrocích o stejném nespecifikovaném studentovi X. S tímto přístupem následně operuje obecnější predikátová logika, která je hlavní náplní této lekce.

Z historického hlediska je ještě vhodné zmínit, že logika jako filozofická disciplína se intenzivně vyvíjí už od dob antiky, ale ke skutečnému rozmachu formální logiky coby součásti matematiky došlo až od začátku 20. století. (S přispěním třeba Russelova paradoxu a převratných Gödelových prací.)

Cíle

V této lekci pokračujeme ve formalizaci základů matematiky lehkým uvedením mocné predikátové logiky a souvisejícím vysvětlením role kvantifikátorů v logice. Také se blíže podíváme na problematiku nalézání či tvorby matematických důkazů obecně. Tím uzavřeme látku načatou v předchozí lekci.

2.1 Kvantifikace a predikátová logika

Dříve popsaná výroková logika je velmi omezená faktem, že každý výrok musí být (tzv. absolutně) vyhodnocen jako pravda nebo nepravda. Co když však chceme zpracovat tvrzení typu „den D v Brně pršelo"? Jeho pravdivostní hodnota přece závisí na tom, co dosadíme za den D, a tudíž jej nelze považovat za výrok výrokové logiky.

Řešení nám nabízí následující obecnější přístup predikátové matematické logiky.

- Predikátová logika pracuje s *predikáty*. Predikáty jsou „*parametrizované výroky*", které jsou buď pravdivé nebo nepravdivé pro každou konkrétní volbu parametrů. Výrokové proměnné lze chápat jako predikáty bez parametrů.
- *Predikátová logika* je tak *obecnější než logika výroková*; každá formule výrokové logiky je i formulí predikátové logiky, ale ne obráceně.

  `Komentář:` To, že predikátová logika zobecňuje výrokovou logiku, doslova znamená, že v predikátové logice můžete používat všechny výrokové proměnné a spojky ve stejném významu jako ve výrokové logice. K tomu navíc přibudou ony predikáty a takzvané kvantifikátory.

  Pro neformální přiblížení si uvedeme několik ukázek predikátů:

  * $x > 3$ (parametrem je zde $x \in \mathbb{R}$),
  * 'čísla x a y jsou nesoudělná' (parametry $x, y \in \mathbb{N}$),
  * obecně jsou predikáty psány $P(x, y)$, kde x, y jsou libovolné parametry.

Následně pak můžeme psát formule ve stylu (a více viz následující definice)

* $\big(P(x,y) \wedge Q(x,z)\big) \to R(y,z)$,
* $\big(P(x,y) \wedge Q(y,z)\big) \to \neg R(x,z)$.

Parametrům predikátů se také říká *proměnné*, ale je třeba dávat pozor, aby se nepletly s výrokovými proměnnými, jedná se o jiné objekty nabývající jiných hodnot.

Syntaxe a sémantika predikátové logiky

Čtenář by měl v tomto místě vzít na vědomí, že naše definice vztahující se k predikátové logice jsou velmi velmi zjednodušené a nenahrazují kurz matematické logiky! Přesto poskytnou alespoň základní povědomí o této problematice (a hlavně o použití kvantifikace v tvrzeních) pro potřeby matematické formalizace v našem kurzu.

`Definice`: Z predikátů (zahrnují i výrokové proměnné) lze vytvářet *predikátové formule* pomocí už známých výrokových spojek (viz Oddíl 1.4) a takzvaných *kvantifikátorů*. Přesněji, nechť φ je formule výrokové či predikátové logiky, pak také

* (*všeobecný kvantifikátor*) formule $\forall x\,.\,\varphi$ a
* (*existenční kvantifikátor*) formule $\exists x\,.\,\varphi$

jsou formulemi predikátové logiky.

`Komentář:` Pro ukázku si uvedeme například predikátovou formuli:

$$\forall x\,.\,\exists y\,.\,\big(x+2<y \;\wedge\; \forall z\,.\,(x+y+z>2023)\big)$$

Náš zápis predikátových formulí ve stylu $\forall x\,.\,\varphi$ není jediný možný, často se také setkáte se zápisy $\forall x\colon \varphi$ nebo $\forall x\big(\varphi\big)$. Všechny tyto tři způsoby můžete používat.

Definice 2.1. ***Zjednodušená sémantika*** (význam) predikátové logiky.
Uvažujme libovolnou množinu (nebo třídu) $\mathcal{U}$, zvanou *univerzum*, a předpokládejme, že každý predikátový parametr (proměnná) $x, y, \ldots$ má zvolenou hodnotu z $\mathcal{U}$. Vyhodnocení predikátové formule σ pak definujeme induktivně takto:

- Každý predikát $P(x,y,\ldots)$ se vyhodnotí (na 0 nebo 1) jako výrok v aktuálním kontextu, tj. vzhledem k aktuálně zvoleným hodnotám svých parametrů $x, y, \ldots$.
- Výrokové spojky jsou vyhodnoceny podle pravidel Definice 1.8.
- Formule tvaru $\forall x\,.\,\varphi$ se vyhodnotí jako pravdivá (1), právě když pro *každou* volbu parametru $x \in \mathcal{U}$ je pravdivá formule φ.
- Formule tvaru $\exists x\,.\,\varphi$ se vyhodnotí jako pravdivá (1), právě když *existuje* alespoň jedna volba parametru $x \in \mathcal{U}$, pro kterou je pravdivá formule φ.

`Komentář:` Pozorného čtenáře napadne, že Definice 2.1 nedává konzistentní návod na vyhodnocení takové predikátové formule, ve které se některá proměnná vyskytuje mimo kvantifikátor. Nejde však o chybu, nýbrž o přirozenou vlastnost – proměnné nemající svůj kvantifikátor (zvané *volné proměnné* formule) totiž musí být zafixovány zvenčí před samotným vyhodnocením. (Jde o obdobu neinicializovaných proměnných v počítačových programech.)

`Fakt`: Je-li *každá* proměnná – parametr predikátu – v dané formuli kvantifikovaná (tj. formule je *uzavřená*), pak je celá formule buď pravdivá nebo nepravdivá.

Příklad 2.2. *Ukažme si vyjádření následujících slovních výroků uzavřenými formulemi v predikátové logice:*

- Každé prvočíslo větší než 2 je liché;

$$\forall n \left[(n \in \mathbb{N} \wedge Pr(n) \wedge n{>}2) \;\Rightarrow\; Li(n)\right],$$

přičemž lze rozepsat $Li(n) \equiv \exists k \in \mathbb{N}\,(n = 2k+1)$ ($\equiv$ je „definiční rovnítko“ pro logické formule). Predikát $Pr(n)$ pak vyjadřuje fakt, že n je prvočíslem.

- Každé celé číslo $n > 1$, které není prvočíslem, je dělitelné nějakým celým číslem y kde $n \neq y$ a $y > 1$;

$$\forall n \left[\big(n \in \mathbb{Z} \wedge n > 1 \wedge \neg Pr(n)\big) \Rightarrow\; \exists y\big(y \in \mathbb{Z} \wedge y > 1 \wedge n \neq y \wedge y \,|\, n\big)\right].$$

□

`Poznámka`: V tomto předmětu počítáme 0 *za přirozené číslo.*

`Značení`: Při zápisu formulí predikátové logiky se často používají některé úpravy a zjednodušení (již jsme si řekli o možném psaní $\forall x\colon \varphi$), které nyní neformálně shrneme.

∗ Pokud píšeme více kvantifikátorů za sebou, vynecháváme zbytečně opakované symboly jako $\forall x \forall y \exists z\,.\,\varphi$, nebo dokonce $\forall x, y, z \exists s, t\,(\varphi)$.

∗ Jsou-li ve formuli φ některé parametry predikátů bez kvantifikace („neuzavřené“), například $x, y, \ldots$, pak mohou být v zápise zdůrazněny jako parametry samotné formule $\varphi(x, y, \ldots)$. Například $\varphi(x, y) \equiv \exists z(x < z < y)$.

∗ Další možná modifikace se týká univerza, ze kterého se vybírají hodnoty kvantifikovaných proměnných. Zatímco striktně bychom měli psát $\forall n\big(n \in \mathbb{Z} \Rightarrow n^2 \geq n\big)$, často vidíme jiný možný zápis $\forall n \in \mathbb{Z}\,(n^2 \geq n)$. Na naši úrovni zjednodušeného pojetí predikátové logiky mezi těmito dvěma zápisy není rozdíl.

Jak na pochopení kvantifikátorů

Používání kvantifikátorů si neformálně přiblížíme pár příklady. Na jednu stranu je (asi) význam použití jednotlivého všeobecného či existenčního kvantifikátoru zřejmý z výše uvedeného popisu, na druhou stranu už tak jednoduché není kombinování více kvantifikátorů v jednom tvrzení. Podívejme se třeba na následující ukázku:

Příklad 2.3. *Proč na pořadí kvantifikátorů velmi záleží:*

- Pro každého studenta A v posluchárně platí, že existuje student B v posluchárně takový, že A je kamarád B.
- Existuje student B v posluchárně, že pro každého studenta A v posluchárně platí, že A je kamarád B.

Vidíte ten propastný rozdíl ve významech uvedených dvou uzavřených predikátových formulí? Přitom jsme jen syntakticky zaměnili pořadí kvantifikace A a B. □

Druhý, již složitější příklad, nám ukazuje, že kvantifikátory potkáme (skrytě) i v situacích, kde na první syntaktický pohled žádná kvantifikace není.

Příklad 2.4. *Pepíček si přinesl ze školy domácí úkol z matematiky:*

Pro dané $a = 3, b = 4, c = 5$ vypočtěte hodnotu výrazu $(a + c - 5)(\bullet + 3) - b$.

Bohužel, jak vidíte, udělal do zadání kaňku $\bullet$ a teď neví, zda pod kaňkou bylo písmeno a, b nebo c. „To je ale smůla“, řekl si Pepíček, „můj příklad teď nemohu jednoznačně vypočítat. Přitom by stačilo změnit v zadání jen jednu z hodnot a, b nebo c a ten příklad by už měl jednoznačné řešení i s tou mou nešťastnou kaňkou!“.

Vysvětlete a zdůvodněte, co tímto Pepíček myslel.

V příkladě jsou dvě roviny, ve kterých musíme argumentovat. Za prvé, že pro uvedené zadání domácího úkolu existují dvě (alespoň) přiřazení symbolu z a, b, c kaňce taková, že výsledné hodnoty budou různé. To je snadno vidět, třeba pro b na místě kaňky vypočítáme hodnotu 17, kdežto pro c na místě kaňky vypočítáme 20. Takže Pepíček opravdu nemohl zadaný úkol jednoznačně vyřešit.

Ve druhé rovině je našim úkolem najít a ověřit, že existuje změna hodnoty jednoho z a, b, c taková, že pro každé přiřazení symbolu z a, b, c kaňce v domácím úkolu vyjde výsledná hodnota výrazu stejně. Potom jednoznačné řešení existuje i s kaňkou a chytrý Pepíček jej najde. Takových vhodných změn je dokonce více, například „$c = 5$“ změníme na „$c = 2$“ a hodnota výrazu vyjde $0 \cdot (\bullet + 3) - 4 = -4$ bez ohledu na to, co se skrývá pod kaňkou.

Vidíte, jak důležité bylo v našem řešení pochopit a explicitně vyjádřit skryté kvantifikátory Pepíčkových úvah? Zbytek pak už vychází docela snadno. □

2.2 Normální tvar logických formulí

Po formálních částech pojednání o logice se dostáváme k jednomu specifickému úskalí chápání přirozené logiky, totiž že přesný význam tvrzení se zanořenými negacemi (není pravda, že ...) je někdy skutečně obtížné pochopit. Co například říká následující?

„Není pravda, že nemohu neříct, že není pravda, že tě nemám nerad.“

Výrokové formule se proto pro lepší pochopení obvykle prezentují v tzv. normálním tvaru, ve kterém se negace zanořených podformulí nevyskytují, formálně:

`Definice`: Formule φ je v *normálním tvaru*, pokud se v ní operátor negace aplikuje pouze na výrokové proměnné (případně na predikáty v případě predikátové formule).

`Komentář:` Pro ilustraci, k formuli $\neg(A \Rightarrow B)$ je ekvivalentní normální tvar $A \wedge \neg B$, k formuli $\neg\big(C \wedge (\neg A \Rightarrow B)\big)$ je ekvivalentní $\neg C \vee (\neg A \wedge \neg B)$, k formuli $\neg\big((A \Rightarrow B) \Rightarrow C\big)$ je ekvivalentní $(A \Rightarrow B) \wedge \neg C$. A pokud důsledně aplikujeme přirozené pravidlo dvojí negace ($\models \neg\neg A \Leftrightarrow A$), tak výše napsané tvrzení „... nemám nerad“ si převedeme na lépe srozumitelný tvar:

„Nemusím říct, že tě mám nerad.“

Podobné převody složitých tvrzení tvaru „*Není pravda, že ...*“ na normální tvar je však velmi obtížné dělat z hlavy. Jak vůbec poznáme, že náš převod byl správný? Stačí si (třeba) vyplnit pravdivostní tabulku. Celkově výhodnější pak je se naučit čistě mechanický postup, kterým převedeme i velmi složitou formuli do normálního tvaru bez velké duševní námahy.

Negace formule v normálním tvaru

Použitím následujících neformálních „přepisovacích“ pravidel dokážeme převést každou formuli predikátové logiky na normální tvar:

- Je-li daná (již znegovaná) formule tvaru $\neg(\varphi \Rightarrow \psi)$, bude přepsána na tvar $(\varphi \wedge \neg\psi)$:

$$\neg(\varphi \Rightarrow \psi) \qquad \rightsquigarrow \qquad \varphi \wedge \neg\psi$$

- Podobně:

$$\begin{aligned} \neg(\varphi \vee \psi) \qquad &\rightsquigarrow \qquad \neg\varphi \wedge \neg\psi \\ \neg(\varphi \wedge \psi) \qquad &\rightsquigarrow \qquad \neg\varphi \vee \neg\psi \end{aligned}$$

- Pro kvantifikátory se pak použije následující intuitivní převod:

$$\begin{aligned} \neg(\exists x\,.\,\varphi) \qquad &\rightsquigarrow \qquad \forall x\,.\,\neg\varphi \\ \neg(\forall x\,.\,\varphi) \qquad &\rightsquigarrow \qquad \exists x\,.\,\neg\varphi \end{aligned}$$

Plný formální popis této metody převodu do normálního tvaru ponecháváme na budoucí Oddíl 5.4.

`Komentář:` Uvažme výrokovou formuli $\neg(A \Rightarrow \neg(B \vee \neg(C \Rightarrow \neg A)))$. Užitím uvedeného postupu ji upravíme takto:

$$\begin{aligned} \neg\big(A \Rightarrow \neg(B \vee \neg(C \Rightarrow \neg A))\big) \quad &= \quad A \wedge \neg\big(\neg(B \vee \neg(C \Rightarrow \neg A))\big) \quad = \\ A \wedge (B \vee \neg(C \Rightarrow \neg A)) \quad &= \quad A \wedge (B \vee (C \wedge \neg(\neg A))) \quad = \\ &A \wedge (B \vee (C \wedge A)) \end{aligned}$$

Formuli $A \wedge (B \vee (C \wedge A))$ lze dále zjednodušit na ekvivalentní formuli $A \wedge (B \vee C)$. To ale je již z našeho pohledu matematicky neformální (heuristický) postup.

Obdobně uvažme predikátovou formuli $\neg(\,\forall x \exists y\, \neg P(x,y)\,)$. Tu užitím našeho postupu upravíme následovně:

$$\begin{aligned} \neg\big(\,\forall x \exists y\, \neg P(x,y)\,\big) \quad &= \quad \exists x\, \neg\big(\,\exists y\, \neg P(x,y)\,\big) \quad = \\ \exists x \forall y\, \neg(\neg P(x,y)) \quad &= \quad \exists x \forall y\, P(x,y) \end{aligned}$$

2.3 Tvoření matematických důkazů

Závěrem lekce se vrátíme ke druhému pilíři správné formalizace matematiky – k přesným důkazům. Dokončíme nyní lehký úvod do matematického dokazování z Lekce 1.

V dřívějším textu jsme viděli několik různorodých ukázek matematických důkazů, od zcela formálního přístupu po velmi uvolněný text. Přirozenou otázkou teď je, jak *moc formální* mají správné matematické důkazy vlastně být?

- Záleží, komu je důkaz určen — *konzument* musí být schopen „snadno“ ověřit korektnost každého tvrzení v důkazu a plně pochopit, z čeho vyplývá.
- Je tedy hlavně na vás zvolit tu správnou úroveň formálnosti zápisu vět i důkazů podle situace.

- Avšak vůbec *neplatí*, že čím více formálních matematických symbolů v důkazu použijete místo běžného jazyka, tím by byl důkaz přesnější – často bývá výsledek právě opačný.

A jak na ten správný matematický důkaz máme přijít?

- No…, nalézání matematických důkazů je tvůrčí činnost, která není vůbec snadná a jako taková vyžaduje tvůrčí (přímo „*umělecké*“) matematické vlohy. I pokud takové vlohy (zatím) v sobě necítíte, snažte se jí alespoň trochu přiučit.

Dokazovat či vyvracet tvrzení?

Představme si, že našim úkolem je *rozhodnout platnost* matematického tvrzení. Neboli nám není shůry řečeno, že tvrzení platí, ale mohou nastat obě možnosti platí / neplatí. Jak pak matematicky správně zdůvodníme nalezenou odpověď?

- Záleží na odpovědi samotné… Pokud je to ANO (platí), prostě podáme důkaz tvrzení podle výše uvedených zvyklostí. Pokud je odpověď NE, tak naopak podáme důkaz *negace* daného tvrzení.

Poměrně častým případem je matematická věta T, která tvrdí nějaký závěr pro širokou oblast vstupních možností. Potom je postup následující:

- Pokud T platí, nezbývá než podat *vyčerpávající důkaz* platnosti pro všechny vstupy.
- Avšak pokud T je nepravdivá, stačí *uhodnout* vhodný *protipříklad* a jen pro něj dokázat, že při platnosti všech předpokladů závěr tvrzení platný není. Ke slovíčku „stačí“ ještě dodáváme, že pro správné matematické vyvrácení nepravdivého tvrzení T protipříklad uvést přímo musíte (viz pravidlo $\neg(\forall x\,.\,\varphi) \rightsquigarrow \exists x\,.\,\neg\varphi$).

  `Komentář:` Při dostatečně pokročilé úrovni znalostí matematiky si sice lze představit i jiné způsoby dokazování existence protipříkladu k našemu tvrzení T, než nalezení konkrétního protipříkladu, ale ty jsou zcela mimo oblast našeho předmětu i běžného studia na FI.

  `Komentář:` Také se již podruhé (po Příkladu 1.5) se dostáváme k fenoménu „*uhodnutí*“ některé součásti důkazu. Jak se k němu máme stavět (je to vůbec dovoleno udělat)? Ale ano, pokud váš důkaz bude matematicky správný, nikdo nebude namítat, že jste v něm něco jen šťastně uhodli. V takovém správném „*uhodnutí*“ věcí právě spočívá ta zmíněná tvůrčí stránka vytváření matematických důkazů.

  A nyní se znovu vraťte na začátek Oddílu 1.1 a srovnejte si výše uvedené se základními poznatky o významu matematických vět…

Příklad 2.5. *Rozhodněte platnost následujícího tvrzení: Pro všechna reálná x platí*

$$x^2 + 3x + 2 \geq 0\,.$$

`Důkaz`: Standardními algebraickými postupy si můžeme upravit vztah na $x^2+3x+2 = (x+1)\cdot(x+2) \geq 0$. Co nám tato úprava naznačuje? Například to, že k porušení daného tvrzení stačí volit x tak, aby jedna ze závorek byla kladná a druhá záporná. To nastane třeba pro $x = -\frac{3}{2}$.

Pro vyvrácení tvrzení nám tedy stačí začít volbou protipříkladu reálného čísla $x = -\frac{3}{2}$ (není nutno zdůvodňovat, jak jsme jej „uhodli“!) a následně dokázat úpravou

$$x^2 + 3x + 2 = (x+1)\cdot(x+2) = (-\frac{3}{2}+1)\cdot(-\frac{3}{2}+2) = (-\frac{1}{2})\cdot(+\frac{1}{2}) = -\frac{1}{4} < 0\,.$$

Dané tvrzení tudíž není platné. □

Konstruktivní a existenční důkazy

Z hlediska praktické využitelnosti je vhodné rozlišovat tyto dvě kategorie důkazů (třebaže z formálně–matematického pohledu mezi nimi kvalitativní rozdíl není).

- Důkaz z Příkladu 1.5 je *konstruktivní*. Dokázali jsme nejen, že číslo z existuje, ale podali jsme také návod (či algoritmus), jak ho pro dané x a y *sestrojit*.
- *Existenční* důkaz je takový, kde se prokáže existence nějakého objektu *bez toho*, aby byl podán použitelný návod na jeho konstrukci. Třebaže to může vypadat divně, nějak se snažit dokazovat existenci něčeho, co nedokážeme sestrojit, ale takové situace skutečně v matematice nastávají a existenční důkazy jsou v nich užitečné.

Příklad 2.6. *Čistě existenčního důkazu.*

Věta. *Existuje program, který vypíše na obrazovku čísla tažená ve 45. tahu sportky v roce 2023.*

`Důkaz`: Existuje pouze konečně mnoho možných výsledků losování 45. tahu sportky v roce 2023. Pro každý možný výsledek existuje program, který tento daný výsledek vypíše na obrazovku. Mezi těmito programy je tedy jistě takový, který vypíše právě ten výsledek, který bude ve 45. tahu sportky v roce 2023 skutečně vylosován. □

`Komentář:` To je ale *podvod*, že? A přece *není*... Formálně správně to je prostě tak a tečka.

Příklad 2.7. *Pravděpodobnostního existenčního důkazu.*

Věta. *Na dané množině bodů je zvoleno libovolně n^2 podmnožin, každá o n prvcích. Dokažte pro dostatečně velká n, že všechny body lze obarvit dvěma barvami tak, aby žádná množina nebyla jednobarevná.*

`Důkaz`: U každého bodu si „*hodíme mincí*“ a podle výsledku zvolíme barvu červeně nebo modře. (Nezávislé volby s pravděpodobností $\frac{1}{2}$.) Pravděpodobnost, že konkrétně zvolených n bodů vyjde jednobarevných, je jen $\frac{2}{2^n} = 2^{1-n}$.

Pro n^2 podmnožin tak je pravděpodobnost, že některá z nich vyjde jednobarevná, shora ohraničená součtem

$$\underbrace{2^{1-n} + \cdots + 2^{1-n}}_{n^2} = \frac{n^2}{2^{n-1}} \to 0\,.$$

Jelikož je toto číslo (pravděpodobnost) pro $n \geq 7$ menší než 1, určitě bude existovat nějaké obarvení bez jednobarevných zvolených podmnožin. □

`Komentář:` Zkuste si sami v obou posledních příkladech přijít s konstruktivním důkazem, tj. takovým, který hledané řešení sestrojí. Pokud se vám to povede, budete skutečně dobří (a možná také bohatí)...

Navazující studium

Látka této lekce navazuje na velmi široký záběr Lekce 1 a dokončuje její nakousnuté poznatky. Přesto je celá matematická logika v našem textu uvedena jen velmi omezeně na nejlehčí akceptovatelné intuitivní úrovni a její další rozvoj je ponechán samostatným

kurzům. Zájemce o studium matematických hlubin logiky a třeba slavných Gödelových poznatků na FI MU můžeme později odkázat na předmět MA007 Matematická logika.

S poblematikou matematického dokazování se však budete dále setkávat v průběhu celé výuky tohoto předmětu i navazujících kurzů.

3 Množiny a množinové operace

Úvod

V přehledu matematických formalismů informatiky se v další lekci zaměříme na základní datové typy matematiky, tj. na množiny, relace a funkce. Netradiční sousloví „datové typy“ matematiky zde volíme záměrně, abychom zdůraznili jejich fundamentální roli pro výstavbu navazující matematické teorie v analogii k programování.

O množinách jste sice zajisté slyšeli už na základní škole, ale podstatou našeho předmětu je uvést povětšinou neformálně známé pojmy na patřičnou formální úroveň nutnou pro teoretické základy informatiky. V případě konečné teorie množin si zde vystačíme s tzv. „naivním“ pohledem, který dostatečně matematicky přesně vystihuje všechny jejich konečné aspekty (o komplikacích okolo nekonečných množin se krátce zmíníme až v Lekci 12).

Na rozdíl od množin se relacím v jejich abstraktní podobě mnoho pozornosti na nižších stupních výuky nevěnuje. Sice jste se již určitě setkali s pojmem funkce, ale to nejspíše jen ve spojení s aritmetickými a analytickými funkcemi („vzorečky“) typu $x+1$, x^2-y, či $\sin x$, $1+\cos x^2$, atd. Jak uvidíte v této lekci, pojmy relace i funkce jsou zcela abstraktní a nemusí se k nim vázat žádný analytický vzorec výpočtu či podobné předpisy.

Cíle

V této lekci si ukážeme první krok k seriózně vybudované matematické teorii množin – tzv. naivní teorii množin, která nám velmi dobře poslouží ve všech konečných případech. Na to navážeme zavedením základního množinového kalkulu a shrnutím běžných vlastností množin. Závěrem si řekneme o abstraktní definici relace a funkce na množinách, kteréžto pojmy budou rozvedeny v navazujících lekcích.

3.1 Pojem množiny

Položme si hned na úvod tu nejdůležitější otázku: **Co je vlastně *množina*?**
Na tuto otázku bohužel není zcela jednoduchá odpověď… Abychom se vůbec někam v našem úvodu dostali, spokojíme se zatím jen s přirozeným „naivním pohledem“.

`Definice` naivní teorie množin: „*Množina je soubor prvků a je svými prvky plně určena.*“

`Komentář:` Příklady zápisu množin výčtem prvků
$\emptyset$, $\{a,b\}$, $\{b,a\}$, $\{a,b,a\}$, $\{\{a,b\}\}$, $\{\emptyset,\{\emptyset\},\{\{\emptyset\}\}\}$, $\{x \mid x \text{ je liché přirozené číslo}\}$.

Co je ale pak *prvek*? Tady pozor, pojem *prvku* sám o sobě nemá matematický význam, svého významu totiž nabývá pouze ve spojení „*být prvkem množiny*“. Prvky množiny tak může být cokoliv, mimo jiné i další množiny.

`Komentář:` Relativitu významu vztahu „prvek–množina“ si můžeme přiblížit třeba na vztahu „*podřízený–nadřízený*“ z běžného pracovního života. Tam také nemá smysl jen říkat, že je někdo podřízeným, aniž řekneme také jeho nadřízeného. Přitom i vedoucí je někomu ještě podřízený a naopak i ten poslední podřízený pracovník může být pánem třeba svého psa. Podobně je tomu s množinou jako „*nadřízenou*“ svých prvků.

Ale přece jenom… v dobře definovaném kontextu lze (omezeně) mluvit o prvcích jako *samostatných entitách*. Formálně se například jedná o *prvky* pevně dané nosné množiny.

`Značení` množin a jejich prvků:

- $x \in M$ „x je *prvkem* množiny M“,

- $\emptyset$ je *prázdná* množina $\{\}$.

  `Komentář:` Některé vlastnosti vztahu „být prvkem“ jsou
  * $a \in \{a, b\}$, $a \notin \{\{a, b\}\}$, $\{a, b\} \in \{\{a, b\}\}$, $a \notin \emptyset$, $\emptyset \in \{\emptyset\}$, $\emptyset \notin \emptyset$,
  * *rovnost* množin dle svých prvků $\{a, b\} = \{b, a\} = \{a, b, a\}$, $\{a, b\} \neq \{\{a, b\}\}$.

`Značení`: Počet prvků (*mohutnost*) množiny A zapisujeme $|A|$.

`Komentář:` Například $|\emptyset| = 0$, $|\{\emptyset\}| = 1$, $|\{a, b, c\}| = 3$, $|\{\{a, b\}, c\}| = 2$.

Jednoduché srovnání množin

Vztah „*být prvkem množiny*“ nám přirozeně podává i způsob porovnávání množin mezi sebou. Jedná se o klíčovou část, čili hlavní nástroj teorie množin.

`Definice`: Množina A je *podmnožinou* množiny B, právě když každý prvek A je prvkem B. Píšeme $A \subseteq B$ nebo obráceně $B \supseteq A$. Říkáme také, že se jedná o *inkluzi*.

* Platí $\{a\} \subseteq \{a\} \subseteq \{a, b\} \not\subseteq \{\{a, b\}\}$, $\emptyset \subseteq \{\emptyset\}$,
* $A \subsetneq B$, právě když $A \subseteq B$ a $A \neq B$ (A je *vlastní* podmnožinou B).

Z naivní definice množiny pak přímo vyplývá následující:

`Definice`: Dvě množiny jsou si *rovny* $A = B$ právě tehdy, když $A \subseteq B$ a $B \subseteq A$.

* Podle naivní definice jsou totiž množiny A a B stejné, mají-li stejné prvky.
* Důkaz rovnosti množin $A = B$ má obvykle *dvě části*: Odděleně se dokáží inkluze $A \subseteq B$ a $B \subseteq A$.

Příklad 3.1. *Vyjádřete předchozí definice podmnožiny a rovnosti množin formálním jazykem predikátové logiky.*

Řešení: V tomto případě máme k dispozici jediný predikát ‘$\in$’ (být prvkem) a univerzum je neomezené. Překladem uvedených definic píšeme:

$$\text{‘}A \subseteq B\text{’} \equiv \forall x \,.\, x \in A \Rightarrow x \in B$$
$$\text{‘}A = B\text{’} \equiv \forall x \,.\, (x \in A \Rightarrow x \in B) \wedge (x \in B \Rightarrow x \in A)$$

□

Ukázky nekonečných množin

`Značení`: Běžné číselné množiny v matematice jsou následující

* $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ je množina přirozených čísel,
* $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, 3, \dots\}$ je množina celých čísel,
* $\mathbb{Z}^+ = \{1, 2, 3, \dots\}$ je množina celých kladných čísel,
* $\mathbb{Q}$ je množina racionálních čísel (zlomků).
* $\mathbb{R}$ je množina reálných čísel.

`Komentář:` Tyto uvedené číselné množiny jsou vesměs *nekonečné*, na rozdíl od konečných množin uvažovaných v předchozím „naivním“ pohledu. Pojem nekonečné množiny se přímo v matematice objevil až teprve v 19. století a brzy se s ním spojilo několik *paradoxů* ukazujících, že naivní pohled na teorii množin pro nekonečné množiny *nedostačuje*. My se k problematice nekonečných množin, Cantorově větě a Russelovu paradoxu vrátíme v závěru našeho předmětu v Lekci 12.

3.2 Množinové operace

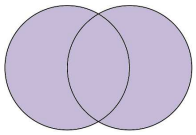
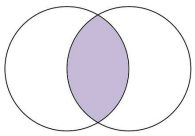
Začněme se základními operacemi množinového kalkulu, které zajisté na intuitivní úrovni již ovládáte ze školy. Pro formální úplnost podáme jejich přesné definice, na které se můžete odvolávat v důkazech.

Sjednocení a průnik

Definice 3.2. ***Sjednocení*** $\cup$ ***a průnik*** $\cap$ dvou množin A, B definujeme

$$\begin{aligned} A \cup B &= \{x \mid x \in A \text{ nebo } x \in B\}, \\ A \cap B &= \{x \mid x \in A \text{ a současně } x \in B\}. \end{aligned}$$

`Komentář:` Pro neformální pochopení věci je velmi užitečné si množinové operace a vztahy zobrazovat tzv. Vennovými diagramy, které zobrazují jednotlivé množiny a v nich „oblasti“ prvků s daným vztahem k zobrazeným množinám. V našem případě to je takto:

$A \cup B$  $A \cap B$ 

* Příklady $\{a,b,c\} \cup \{a,d\} = \{a,b,c,d\}$, $\{a,b,c\} \cap \{a,d\} = \{a\}$.
* Vždy platí „distributivita“ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ a $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
* a také „asociativita“ $A \cap (B \cap C) = (A \cap B) \cap C$ (stejně pro $\cup$) a „komutativita“ $A \cap B = B \cap A$ (stejně pro $\cup$).

  `Komentář:` Asociativita a komutativita operací sjednocení a průniku je na jednu stranu zcela přirozená, na druhou stranu však velmi důležitá například pro platnost následující definice.

`Definice`: Pro libovolný počet množin indexovaných pomocí I rozšířeně definujeme

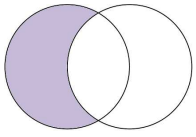
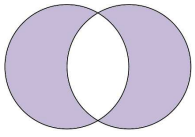
$$\begin{aligned} \bigcup_{i \in I} A_i &= \{x \mid x \in A_i \text{ pro } \textit{nějaké } i \in I\}, \\ \bigcap_{i \in I} A_i &= \{x \mid x \in A_i \text{ pro } \textit{každé } i \in I\}. \end{aligned}$$

`Komentář:` Nechť $A_i = \{2 \cdot i\}$ pro každé $i \in \mathbb{N}$. Pak $\bigcup_{i \in \mathbb{N}} A_i$ je množina všech sudých přirozených čísel. Nechť $B_i = \{x \mid x \in \mathbb{N}, x \geq i\}$ pro každé $i \in \mathbb{N}$. Pak $\bigcap_{i \in \mathbb{N}} B_i = \emptyset$.

Množinový rozdíl

Definice 3.3. ***Rozdíl*** $\setminus$ ***a symetrický rozdíl*** Δ dvou množin A, B definujeme

$$\begin{aligned} A \setminus B &= \{x \mid x \in A \text{ a současně } x \notin B\}, \\ A \,\Delta\, B &= \big(A \setminus B\big) \cup \big(B \setminus A\big). \end{aligned}$$

$A \setminus B$  $A \Delta B$ 

* Příklady $\{a,b,c\} \setminus \{a,b,d\} = \{c\}$, $\quad \{a,b,c\}\Delta\{a,b,d\} = \{c,d\}$.
* Vždy platí například $A \setminus (B \cap C) = (A \setminus B) \cup (A \setminus C)$ apod.

`Definice`: Pro libovolný počet množin indexovaných pomocí *konečné* I rozšířeně definujeme

$$\bigtriangleup_{i\in I} A_i = \{x \mid x \in A_i \text{ pro } \textit{lichý počet } i \in I\}\,.$$

`Komentář:` Povšimněte si, že operace rozdílu není asociativní ani komutativní (ostatně stejně je tomu u aritmetického rozdílu). Symetrický rozdíl už asociativní i komutativní je, není to však na první pohled vidět. Uměli byste toto dokázat?

`Definice`: Nechť $A \subseteq M$. *Doplňkem A vzhledem k M* je množina $\overline{A} = M \setminus A$.

`Komentář:` Jedná se o poněkud specifickou operaci, která *musí být vztažena* vzhledem k *nosné množině* M! Je-li $M = \{a,b,c\}$, pak $\overline{\{a,b\}} = \{c\}$. Je-li $M = \{a,b\}$, pak $\overline{\{a,b\}} = \emptyset$.

* Vždy pro $A \subseteq M$ platí $\overline{\overline{A}} = A$ („dvojí“ doplněk).
* Vždy pro $A, B \subseteq M$ platí $\overline{A \cup B} = \overline{A} \cap \overline{B}$ a $\overline{A \cap B} = \overline{A} \cup \overline{B}$. (Viz Věta 3.7.)

Potenční množina

Definice 3.4. ***Potenční množina*** množiny A,
neboli množina všech podmnožin, je definovaná vztahem

$$2^A = \{B \mid B \subseteq A\}\,.$$

* Platí například $2^{\{a,b\}} = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$,
* $2^\emptyset = \{\emptyset\}$, $\quad 2^{\{\emptyset,\{\emptyset\}\}} = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$,

Věta 3.5. *Počet prvků potenční množiny splňuje* $|2^A| = 2^{|A|}$.

`Důkaz`: Důkaz jen stručně naznačíme (pro přesnější zápis důkazu by se použila matematická indukce z Lekce 4). Jak vybereme jednu podmnožinu $B \subseteq A$? Například tak, že pro každý z $|A|$ prvků množiny A se nezávisle rozhodneme, zda jej zařadíme do B nebo ne. To jsou dvě možnosti pro výběr každého prvku $b \in A$ a podle principu nezávislých výběrů je celkový počet různých podmnožin množiny A roven

$$\left|2^A\right| = 2 \cdot 2 \cdot \ldots \cdot 2 = 2^{|A|}\,. \qquad \square$$

3.3 Kartézský součin

Zatímco prvky v množinách jsou zcela *neuspořádané*, jsou mnohé situace, kdy musíme pracovat se „*seřazenými*“ výčty prvků. V teorii množin lze takovéto seřazení definovat oklikou následovně:

`Definice`: *Uspořádaná dvojice* (a,b) je zadána množinou $\{\{a\},\{a,b\}\}$.

`Fakt`: Platí $(a,b) = (c,d)$ právě tehdy, když $a = c$ a současně $b = d$. Uměli byste toto sami dokázat přímo z podané definice?

* Co je podle definice (a, a)? Je to $(a, a) = \{\{a\}, \{a, a\}\} = \{\{a\}, \{a\}\} = \{\{a\}\}$.

S uspořádanými dvojicemi prvků je přímo svázán nadmíru důležitý pojem součinu dvou množin, který se zavádí takto:

Definice 3.6. ***Kartézský součin*** dvou množin A, B definujeme jako množinu všech uspořádaných dvojic ze složek z A a B

$$A \times B = \{(a, b) \mid a \in A, b \in B\}\,.$$

* Příklady $\{a, b\} \times \{a\} = \{(a, a), (b, a)\}$, $\{c, d\} \times \{a, b\} = \{(c, a), (c, b), (d, a), (d, b)\}$.
* Platí $\emptyset \times X = \emptyset = X \times \emptyset$ pro každou množinu X.
* Jednoduchá mnemotechnická pomůcka říká $|A \times B| = |A| \cdot |B|$. Každá dvojice z $A \times B$ totiž odpovídá nezávislému výběru prvku z A a prvku z B.

Uspořádání prvků a součiny množin nemusí být omezeny jen na dvojice, jejich snadného zobecnění na libovolné konečné počty členů lze dosáhnout následujícími definicemi.

`Definice`: Pro každé $k \in \mathbb{N}, k > 0$ definujeme *uspořádanou k-tici* $(a_1, \cdots, a_k)$ induktivně
- $(a_1) = a_1$,
- $(a_1, a_2, \cdots, a_i, a_{i+1}) = ((a_1, a_2, \cdots, a_i), a_{i+1})$.

Všimněte si, že tato definice používá tzv. rekurentní vztah, blíže viz Oddíl 5.1.

`Fakt`: Platí $(a_1, \cdots, a_k) = (b_1, \cdots, b_k)$, právě když $a_i = b_i$ pro každé $i \in \mathbb{N}$ kde $1 \leq i \leq k$.

`Definice` *kartézského součinu* více množin: Pro každé $k \in \mathbb{N}$ definujeme

$$A_1 \times A_2 \times \cdots \times A_k = \{(a_1, a_2, \cdots, a_k) \mid a_i \in A_i \text{ pro každé } 1 \leq i \leq k\}\,.$$

Navíc pokud jsou množiny A_i indexovány přes libovolnou konečnou množinu $I \ni i$, tak jejich kartézský součin krátce píšeme jako $\bigtimes_{i \in I} A_i$.

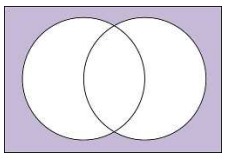
* Například $\mathbb{Z}^3 = \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} = \{(i, j, k) \mid i, j, k \in \mathbb{Z}\}$.
* Co je A^0? $\{\emptyset\}$, neboť jediná uspořádaná 0-tice je právě prázdná $\emptyset$.

`Poznámka`: Podle uvedené definice *není kartézský součin asociativní*, tj. obecně nemusí platit, že $A \times (B \times C) = (A \times B) \times C$. V matematické praxi je někdy výhodnější uvažovat upravenou definici, podle níž součin *asociativní je*. Pro účely této přednášky není podstatné, k jaké definici se přikloníme. Prezentované definice a věty „fungují“ pro obě varianty.

3.4 Porovnávání a určení množin

Pro obecnou ilustraci formálního množinového kalkulu si ukažme dva důkazy rovností mezi množinovými výrazy. Podobně lze (rozepsáním příslušných definic) rutinně dokazovat další množinové vztahy.

Věta 3.7. *Pro každé dvě množiny $A, B \subseteq M$ platí $\overline{A \cup B} = \overline{A} \cap \overline{B}$.*



`Důkaz` v obou směrech rovnosti (viz ilustrační obrázek).

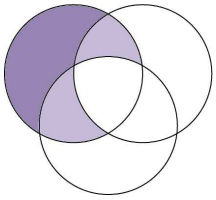
- $\overline{A \cup B} \subseteq \overline{A} \cap \overline{B}$: Pro $x \in M$ platí $x \in \overline{A \cup B}$, právě když $x \notin A \cup B$, neboli když zároveň $x \notin A$ a $x \notin B$. To znamená $x \in \overline{A}$ a zároveň $x \in \overline{B}$, z čehož vyplývá požadované $x \in \overline{A} \cap \overline{B}$.

- $\overline{A \cup B} \supseteq \overline{A} \cap \overline{B}$: Pro $x \in M$ platí $x \in \overline{A} \cap \overline{B}$, právě když $x \in \overline{A}$ a zároveň $x \in \overline{B}$, neboli když zároveň $x \notin A$ a $x \notin B$. To znamená $x \notin A \cup B$, z čehož vyplývá požadované $x \in \overline{A \cup B}$.

□

Věta 3.8. *Pro každé tři množiny A, B, C platí*

$$A \setminus (B \cap C) \;=\; (A \setminus B) \;\cup (A \setminus C)\,.$$



`Důkaz` (viz ilustrační obrázek).

- $A \setminus (B \cap C) \;\subseteq\; (A \setminus B) \;\cup (A \setminus C)$: Je-li $x \in A \setminus (B \cap C)$, pak $x \in A$ a zároveň $x \notin (B \cap C)$, neboli $x \notin B$ nebo $x \notin C$. Pro první možnost máme $x \in (A \setminus B)$, pro druhou $x \in (A \setminus C)$.

- Naopak $A \setminus (B \cap C) \;\supseteq\; (A \setminus B) \;\cup (A \setminus C)$: Je-li $x \in (A \setminus B) \;\cup (A \setminus C)$, pak $x \in (A \setminus B)$ nebo $x \in (A \setminus C)$. Pro první možnost máme $x \in A$ a zároveň $x \notin B$, z čehož plyne $x \in A$ a zároveň $x \notin (B \cap C)$, a tudíž $x \in A \setminus (B \cap C)$. Druhá možnost je analogická.

□

Charakteristický vektor (pod)množiny

V případech, kdy všechny uvažované množiny jsou podmnožinami nějaké konečné *nosné množiny X*, což není neobvyklé v programátorských aplikacích, s výhodou využijeme následující reprezentaci množin.

`Definice`: Mějme nosnou množinu $X = \{x_1, x_2, \ldots, x_n\}$. Pro $A \subseteq X$ definujeme *charakteristický vektor* χ_A jako

$$\chi_A = (c_1, c_2, \ldots, c_n), \text{ kde } c_i = 1 \text{ pro } x_i \in A \text{ a } c_i = 0 \text{ jinak.}$$

Užitečnost této reprezentace je ilustrována také následnými fakty.

- Platí $A = B$ právě když $\chi_A = \chi_B$.

- Množinové operace jsou realizovány „bitovými funkcemi“
  sjednocení $\sim$ OR, průnik $\sim$ AND, symetrický rozdíl $\sim$ XOR.

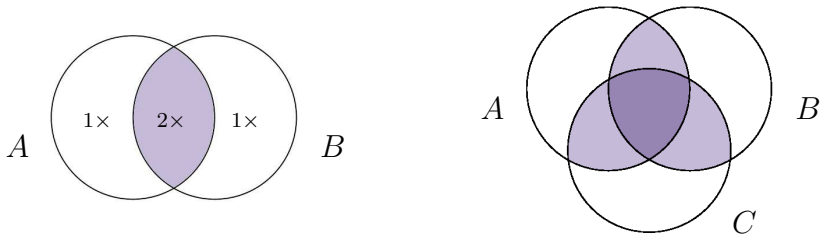
Princip inkluze a exkluze

Tento důležitý a zajímavý kombinatorický princip je někdy také nazýván „princip *zapojení a vypojení*“. Používá se ke zjišťování počtů prvků množin s neprázdnými průniky.

Věta 3.9. *Počet prvků ve sjednocení dvou či tří množin spočítáme:*

$$|A \cup B| = |A| + |B| - |A \cap B|$$

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|$$



`Důkaz`: Uvedeme si podrobný důkaz prvního vztahu $|A \cup B| = |A| + |B| - |A \cap B|$, přičemž druhá část je analogická (viz obrázková ilustrace). Nechť $x \in A \cup B$. Pokud $x \notin B$, tak je $x \in A$ a prvek x je započítán na pravé straně $|A| + |B| - |A \cap B|$ právě jednou v $|A|$. Obdobně je to v případě $x \notin A$. Nakonec pro $x \in A \cap B$ je ve vztahu $|A| + |B| - |A \cap B|$ tento prvek x započítán také $1 + 1 - 1 = 1$ krát. □

`Komentář:` Všimněte si, že Větu 3.9 lze stejně tak využít k výpočtu počtu prvků v průniku množin...

Příklad 3.10. *Z* 1000 *televizí jich při první kontrole na výrobní lince má* 5 *vadnou obrazovku,* 10 *je poškrábaných a* 12 *má jinou vadu. Přitom* 3 *televize mají současně všechny tři vady a* 4 *jiné jsou poškrábané a mají jinou vadu. Kolik televizí je celkem vadných?*

Řešení: Dosazením $|A| = 5$, $|B| = 10$, $|C| = 12$, $|A \cap B \cap C| = 3$, (zde pozor) $|A \cap B| = 3 + 0$, $|A \cap C| = 3 + 0$, $|B \cap C| = 3 + 4$ do Věty 3.9 zjistíme výsledek 17. □

`Poznámka`. Jen stručně, bez důkazu a bližšího vysvětlení, si uvedeme *obecnou formu principu inkluze a exkluze*:

$$\left|\bigcup_{j=1}^{n} A_j\right| = \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} (-1)^{|I|-1} \cdot \left|\bigcap_{i \in I} A_i\right|$$

(Jeho znalost nebude v předmětu vyžadována.)

3.5 Relace a funkce

Vedle množin jsou dalším důležitým základním „*datovým typem*“ matematiky relace, které nyní zavedeme a kterým vzhledem k jejich mnohotvárnému použití v informatice věnujeme významnou pozornost i v příštích lekcích.

Definice 3.11. ***Relace*** mezi množinami $A_1, A_2, \cdots, A_k$, pro $k \in \mathbb{N}$, je *libovolná* podmnožina kartézského součinu

$$R \subseteq A_1 \times A_2 \times \cdots \times A_k \,.$$

Pokud $A_1 = A_2 = \cdots = A_k = A$, hovoříme o *$k$-ární relaci R na A*. Speciálně tak mluvíme třeba o *binární* ($k = 2$), *ternární* ($k = 3$) nebo *unární* ($k = 1$) relaci.

`Komentář:` Příklady relací.

* $\{(1, a), (2, a), (2, b)\}$ je relace mezi $\{1, 2, 3\}$ a $\{a, b\}$.
* $\{(i, 2 \cdot i) \mid i \in \mathbb{N}\}$ je *binární* relace na $\mathbb{N}$.
* $\{(i, j, i + j) \mid i, j \in \mathbb{N}\}$ je *ternární* relace na $\mathbb{N}$.
* $\{3 \cdot i \mid i \in \mathbb{N}\}$ je *unární* relace na $\mathbb{N}$.
* Jaký význam vlastně mají unární a nulární relace na A?

Uvědomme si, jak obecně je relace definována – její definice umožňuje podchytit skutečně libovolné „vztahy“ mezi prvky téže i různých množin. V praxi se relace velmi široce využívají třeba v relačních databázích...

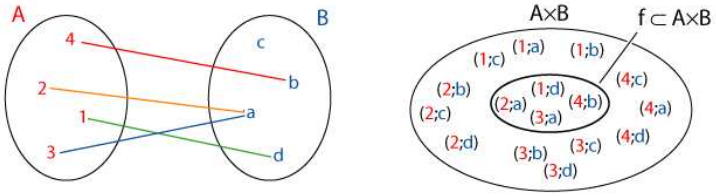
Funkce mezi množinami

První školní setkání s instancemi relací povětšinou proběhne u pojmu funkce. Ovšem tradiční školní pojetí *funkce* ji obvykle identifikuje s nějakým výpočetním (analytickým) předpisem či vzorcem. Pojem funkce je však daleko obecnější a zcela abstraktní, což si v tomto oddíle bohatě ilustrujeme a poslouží nám to i pro lepší pochopení obecnějšího pojmu relace.

Definice 3.12. ***(Totální) funkce*** z množiny A do množiny B je relace f mezi A a B taková, že pro každé $x \in A$ existuje právě jedno $y \in B$ takové, že $(x, y) \in f$. Množina A se nazývá *definiční obor* funkce f. Funkcím se také říká *zobrazení*.

`Poznámka`: Množinu B lze nazvat oborem hodnot, ale častější terminologie je, že *oborem hodnot* funkce f s definičním oborem A je podmnožina množiny B sestávající z těch y, pro něž existuje $x \in A$ takové, že $(x, y) \in f$. Pro náš výklad není podstatné, k čemu se přikloníme.

`Komentář:` Neformálně řečeno, ve funkci f je každé „vstupní“ hodnotě x přiřazena *jednoznačně* „výstupní“ hodnota y. Naopak v obecné relaci počty „přiřazených“ hodnot neomezujeme. Přiřazené „výstupní“ hodnoty tvoří obor hodnot (příp. jeho podmnožinu). Například v ukázce na obrázku to jsou $\{a, b, d\} \subseteq B$.



Pokud je funkce dána výčtem všech svých dvojic, lze definiční obor z tohoto výčtu jednoznačně odvodit (je to množina těch prvních složek dvojic), avšak obecně je definiční obor nedílnou součástí definice konkrétní funkce, neboť v obvyklém případě implicitních definic funkcí (třeba vztahem) jej nelze vždy jednoznačně odvodit.

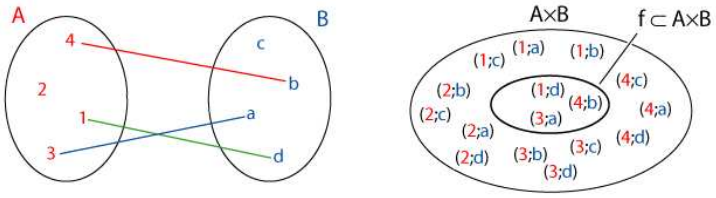
`Značení`: Pro funkce místo $(x, y) \in f$ píšeme obvykle $f(x) = y$. Zápis $f : A \to B$ říká, že f je funkce s definičním oborem A a oborem hodnot, který je podmnožinou B.

`Komentář:` Příklady funkcí jsou třeba následující.

* Definujeme funkci $f : \mathbb{N} \to \mathbb{N}$ předpisem $f(x) = x + 8$. Pak $f = \{(x, x + 8) \mid x \in \mathbb{N}\}$.
* Definujeme funkci $plus : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ předpisem $plus(i, j) = i + j$.
  Pak $plus = \{(i, j, i + j) \mid i, j \in \mathbb{N}\}$.

`Definice`: Pokud naši Definici 3.12 upravíme tak, že požadujeme pro každé $x \in A$ nejvýše jedno $y \in B$ takové, že $(x, y) \in f$, obdržíme definici *parciální funkce* z A do B.

`Komentář:`



V parciální funkci f nemusí být pro některé „vstupní“ hodnoty x funkční hodnota definována (viz například $f(2)$ v uvedeném obrázku).

Pro *nedefinovanou* hodnotu používáme znak $\perp$.

`Komentář:` Následuje několik příkladů parciálních funkcí.

* Definujeme parciální funkci $f : \mathbb{Z} \to \mathbb{N}$ předpisem

$$f(x) = \begin{cases} 3 + x & \text{jestliže } x \geq 0, \\ \perp & \text{jinak.} \end{cases}$$

  Tj. $f = \{(x, 3 + x) \mid x \in \mathbb{N}\}$.
* Také funkce $f : \mathbb{R} \to \mathbb{R}$ daná běžným analytickým předpisem $f(x) = \log x$ je jen parciální – není definována pro $x \leq 0$.
* Co je relace, přiřazující lidem v ČR jejich (česká) rodná čísla?

Navazující studium

I když jste se s "množinami" setkávali už od základní školy, do formálně matematické teorie množin asi nahlédnete na VŠ poprvé. I když některé čtenáře množství zde uvedených symbolů a pojmů možná nejprve vyleká, jedná se vesměs o velmi intuitivní věci, které není problém pochopit byť z ilustračních obrázků a příkladů. Naučte se s konečnými množinami a relacemi dobře pracovat alespoň na té intuitivní úrovni, jakou v této lekci vidíte. Ostaně, jak již bylo psáno, množiny jsou základním stavebním kamenem celé matematiky.

V dalších lekcích budeme důležité pojmy relací a funkcí dále rozvíjet. Mimo naší tzv. naivní teorie množin si lehký náhled na obecně nekonečné množiny a jejich zdánlivé paradoxy i informatické aplikace (jako třeba problém zastavení) ukážeme nakonec v Lekci 12...

4 Techniky matematických důkazů

Úvod

Náš hlubší úvod do matematických formalismů pro informatiku pokračujeme základním přehledem technik matematických důkazů. Pro vysvětlení, z Lekce 1 sice víme, že matematický důkaz má jednotnou formu Definice 1.3, ale nyní se budeme věnovat takříkajíc šablonám, podle nichž můžeme (snadněji) korektní důkaz sestavit.

Třebaže matematické dokazování a příslušné techniky mohou někomu připadat neprůhledné (a zpočátku možná zbytečné), jejich pochopení a zvládnutí není samoúčelné, neboť nám pomáhá si mnoho uvědomit o studovaných problémech samotných. Konečně, jak si můžeme být jisti svými poznatky, když bychom pro ně nebyli schopni poskytnout důkazy? Během studia tohoto předmětu poznáte (a ti méně šťastní až s překvapením u zkoušky), že podstata toho, k čemu naším výkladem směřujeme, se dá neformálně shrnout slovy „naučit se přesně vyjadřovat a být si svými tvrzeními naprosto jisti“ a analogicky později „naučit se navrhovat správné algoritmy a být si i svými programy naprosto jisti“.

Cíle

Cílem této lekce je popsat základní techniky matematických důkazů, z nichž největší důraz klademe na matematickou indukci. Každý student by se měl alespoň naučit formálně psané důkazy číst a chápat, nejlépe i sám umět jednoduché důkazy tvořit.

4.1 Přehled základních důkazových technik

V matematice je často používaných několik následujících způsobů – technik, jak k danému tvrzení sestavit korektní formální důkaz. (Uvědomme si, že jedno tvrzení mívá mnoho různých, stejně korektních důkazů; ty se však mohou výrazně lišit svou složitostí a čitelností. Uváděné techniky nám pomohou najít důkaz co nejjednodušší.) Tyto techniky si v bodech shrneme zde:

- *Přímé odvození.* To je způsob, o kterém jsme se dosud bavili.

  `Komentář:` Postupujeme přímo od předpokladů k závěru, což vypadá nejpřirozeněji, ale sami poznáte, že taková „přímá“ cesta bývá obtížně k nalezení.

- *Kontrapozice* (také *obměnou* – „obrácením“, či *nepřímý důkaz*). Místo věty

  > „Jestliže platí *předpoklady*, pak platí *závěr*.“

  budeme dokazovat ekvivalentní větu

  > „Jestliže neplatí *závěr*, pak neplatí alespoň jeden z *předpokladů*.“

  `Komentář:` Na rozdíl od přímého odvození se obvykle lépe hledá cesta „pozpátku“, určitě tento fenomén znáte například z hledání cest v dětských bludištích.

- *Důkaz sporem.* Místo věty

  > „Jestliže platí *předpoklady*, pak platí *závěr*.“

  budeme dokazovat větu

  > „Jestliže platí *předpoklady* a platí *opak závěru*, pak platí. . . “
  >
  > – nějaké *zjevně nepravdivé tvrzení*, nebo případně
  > – závěr (tj. opak jeho opaku) či opak jednoho z předpokladů.

`Komentář:` Třebaže tato šablona vypadá trochu složitě, stará dobrá matematická moudrost praví: „Nevíte-li, jak nějakou větu dokázat, zkuste důkaz sporem…“

- *Matematická indukce.* Pokročilá technika, kterou popíšeme později…

Tyto techniky asi nejlépe ilustrujeme následujícími příklady důkazů.

Příklad důkazu kontrapozicí

`Definice`: *Prvočíslo* je celé číslo $p > 1$, které nemá jiné dělitele než 1 a p.

Příklad 4.1. *Na důkaz kontrapozicí (obměnou).*

Věta. *Jestliže p je prvočíslo větší než 2, pak p je liché.*

`Důkaz` *obměněného tvrzení*: Místo uvedeného znění věty budeme dokazovat, že

je-li p sudé, pak p není větší než 2 nebo p není prvočíslo.

Připomínáme, že podle definice je p sudé, právě když lze psát $p = 2 \cdot k$, kde k je celé. Jsou jen dvě snadno řešitelné možnosti:

- $k \leq 1$. Pak $p = 2 \cdot k$ není větší než 2.
- $k > 1$. Pak $p = 2 \cdot k$ není prvočíslo podle definice. □

`Poznámka`: Důkazy kontrapozicí pracují s *negací* (opakem) *předpokladů* a *závěru*. Je-li např. závěrem komplikované tvrzení tvaru

„z toho, že z A a B plyne C, vyplývá, že z A nebo C plyne A a B“,

není snadné pouhou intuicí zjistit, co je vlastně jeho negací. Proto je vhodné si nejprve opak závěru přepsat do takzvaného normálního tvaru, který byl vysvětlen v Oddílu 2.2.

Příklady důkazu sporem

Příklad 4.2. *Jiný, kratší přístup k Důkazu 4.1.*

Věta. *Jestliže p je prvočíslo větší než 2, pak p je liché.*

`Důkaz` *sporem*: Nechť tedy p je prvočíslo větší než 2, které je sudé. Pak $p = 2 \cdot k$ pro nějaké $k > 1$, tedy p není prvočíslo, spor (s předpokladem, že p je prvočíslo). □

Důkaz sporem je natolik specifický a důležitý v matematice, že si zaslouží širší vysvětlení. Co je vlastně jeho podstatou? Je to (zcela přirozený) předpoklad, že v *konzistentní teorii* nelze zároveň odvodit tvrzení i jeho negaci. Jestliže tedy ve schématu

„Jestliže platí *předpoklady* a platí *opak závěru*, pak platí závěr nebo opak jednoho z *předpokladů*, nebo platí jiné *zjevně nepravdivé tvrzení*.“

odvodíme k některému předpokladu jeho opak, nebo případně jiné tvrzení, které odporuje všeobecně přijatým faktům (přesněji axiomům, například $0 = 1$), pak něco musí být „*špatně*“. Co však v našem tvrzení může (nezapomeňte předpoklad konzistence) být chybné? Původní předpoklady byly dány, takže zbývá jedině náš dodatečný předpoklad, že platí *opak závěru*. Tudíž opak závěru nemůže nikdy platit a dvojí negací odvodíme, že platí původní závěr.

Příklad 4.3. *Opět sporem.*

Věta. *Číslo $\sqrt{2}$ není racionální.*

`Důkaz` *sporem*: Nechť tedy $\sqrt{2}$ je racionální, tj. nechť existují nesoudělná celá kladná čísla r, s taková, že $\sqrt{2} = r/s$.

– Pak $2 = r^2/s^2$, tedy $r^2 = 2 \cdot s^2$, proto r^2 je dělitelné dvěma. Z toho plyne, že i r je dělitelné dvěma (proč? opět na to můžete jít sporem...).

– Jelikož r je dělitelné dvěma, je r^2 dělitelné dokonce čtyřmi, tedy $r^2 = 4 \cdot m$ pro nějaké m. Pak ale také $4 \cdot m = 2 \cdot s^2$, tedy $2 \cdot m = s^2$ a proto s^2 je dělitelné dvěma.

– Z toho plyne, že s je také dělitelné dvěma. Celkem dostáváme, že r i s jsou dělitelné dvěma, jsou tedy soudělná a to je spor (se základní vlastností racionálního čísla). □

4.2 Věty typu „právě tehdy (když)“

Uvažujme nyní (v matematice poměrně hojné) *věty* tvaru

„Nechť platí *předpoklady P*. Pak *tvrzení A* platí *právě tehdy*, platí-li *tvrzení B*.“

Příklady jiných jazykových formulací téže věty jsou:

* Nechť platí předpoklady P. Pak tvrzení A platí *tehdy a jen tehdy*, když platí tvrzení B.
* Za předpokladů P je tvrzení B *nutnou a postačující* podmínkou pro platnost tvrzení A.
* Za předpokladů P je tvrzení A nutnou a postačující podmínkou pro platnost tvrzení B.

`Fakt`: *Plný důkaz* vět tohoto tvaru má vždy *dvě části(!)*. Je třeba dokázat:

* Jestliže platí předpoklady P a tvrzení A, pak platí tvrzení B.
* Jestliže platí předpoklady P a tvrzení B, pak platí tvrzení A.

Následující příklad je záměrně poněkud krkolomný, aby ukázal situaci, ve které se oba směry dokazované ekvivalence výrazně liší.

Příklad 4.4. *Na důkaz typu „právě tehdy (když)“.*

Věta. *Pro každá dvě celá čísla a, b platí, že $a < b$ právě tehdy, když $2^a < 2^b$.*

`Důkaz`: Nezapomínáme, že našim úkolem je dokázat oba směry tvrzení (implikace zleva doprava a zprava doleva). Začneme s prvním: Předpoklad $a < b$ je definičně ekvivalentní tomu, že $b = a + k$ pro nějaké přirozené $k > 0$. Potom $2^b = 2^{a+k} = 2^a \cdot 2^k = \ell \cdot 2^a$ pro nějaké přirozené $\ell = 2^k \geq 2$, a tudíž $2^b - 2^a = (\ell - 1) \cdot 2^a \geq 1 \cdot 2^a > 0$. Tím je první část důkazu hotova.

V opačném směru postupujeme z předpokladu $2^a < 2^b$. Vydělením obou stran nerovnosti kladným číslem 2^a získáme $2^b/2^a = 2^{b-a} > 1$. Dále postupujeme sporem. Nechť tedy $a \geq b$, neboli $a - b = m \geq 0$. Potom máme $2^{a-b} = \underbrace{2 \cdot \ldots 2}_{m\times} \geq 1$. Avšak celkově $1 = 2^0 = 2^{a-b} \cdot 2^{b-a} > 1 \cdot 1 = 1$ je sporné. Proto zbývá jen požadovaný závěr $a < b$. □

`Komentář:` Je možno se někdy oddělenému zpracování obou směrů takového tvrzení vyhnout? Ano, stačí dělat pouze tzv. ekvivalentní úpravy, ale je to velmi nebezpečné. Co například tvrzení „$a = b$ právě když $ax = bx$“? To přece dokážeme vynásobením obou stran rovnosti stejným číslem x...

Je to sice téměř pravda, ale(!) musíme si všimnout, že násobení obou stran rovnosti je ekvivalentní úpravou jen pro $x \neq 0$; jinak nazpět dělíme nulou (což na světě dokáže snad pouze Chuck Norris). Zapamatujte si, že nejlepší cestou, jak se takovým chybám a problémům vyhnout, je skutečně poctivě dokazovat každý směr ekvivalence zvlášť.

4.3 Matematická indukce

Pokud se souhrnně podíváme na důkazové techniky v matematice, všimneme si, že matematickou indukci lze považovat za „dvorní“ důkazovou technikou diskrétní matematiky. To proto, že umožňuje pohodlně dokazovat i složitá tvrzení po jednotlivých (malých a diskrétních) krocích od jednoduchých počátků.

Uvažme tedy větu ve tvaru:

„Pro každé přirozené (celé) $n \geq k_0$ platí $T(n)$.“

Zde k_0 je nějaké pevné přirozené číslo a $T(n)$ je tvrzení parametrizované číslem n. Příkladem je třeba tvrzení:

Pro každé $n \geq 0$ platí, že n přímek dělí rovinu nejvýše na $\frac{1}{2}n(n+1)+1$ oblastí.

Definice 4.5. ***Princip matematické indukce*** říká, že k důkazu věty

„Pro každé přirozené (celé) $n \geq k_0$ platí $T(n)$.“

stačí ověřit platnost těchto dvou tvrzení:

- $T(k_0)$ (tzv. *báze* neboli základ indukce)
- *Pro každé $k \geq k_0$; jestliže platí $T(k)$,* *(indukční předpoklad)*
  pak platí také $T(k+1)$. *(indukční krok)*

Formálně řečeno, matematická indukce je axiomem aritmetiky přirozených čísel.

Opět jako v předešlém si tuto techniku ilustrujeme množstvím názorných příkladů.

Příklady důkazů indukcí

Příklad 4.6. *Velmi jednoduchá a přímočará indukce.*

Věta. *Pro každé přirozené. $n \geq 1$ je stejná pravděpodobnost, že při současném hodu n kostkami bude výsledný součet sudý, jako, že bude lichý.*

`Důkaz`: *Základ indukce* je zde zřejmý: Na jedné kostce (poctivé!) jsou tři lichá a tři sudá čísla, takže obě skupiny padají se stejnou pravděpodobností.

Indukční krok pro $k \geq 1$: Nechť p_k^s pravděpodobnost, že při hodu k kostkami bude výsledný součet sudý, a p_k^l je pravděpodobnost lichého. Podle indukčního předpokladu je $p_k^s = p_k^l = \frac{1}{2}$. Hoďme navíc $(k+1)$-ní kostkou. Podle toho, zda na ní padne liché nebo sudé číslo, je pravděpodobnost celkového sudého součtu rovna

$$\frac{3}{6}p_k^l + \frac{3}{6}p_k^s = \frac{1}{2}$$

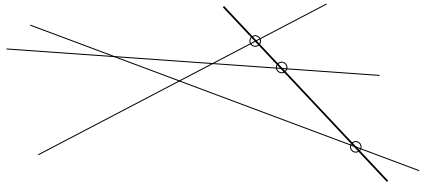
a stejně pro pravděpodobnost celkového lichého součtu. □

Příklad 4.7. *Ukázka skutečné důkazové „síly“ principu matematické indukce.*

Věta. *Pro každé $n \geq 0$ platí, že n přímek dělí rovinu nejvýše na*

$$\frac{1}{2}n(n+1)+1$$

oblastí.



`Důkaz`: Pro bázi indukce postačí, že $n = 0$ přímek dělí rovinu na jednu oblast. (Všimněte si také, že 1 přímka dělí rovinu na dvě oblasti, jen pro lepší pochopení důkazu.)

Mějme nyní rovinu rozdělenou $n = k$ přímkami na nejvýše $\frac{1}{2}k(k+1)+1$ oblastí. Další, $(k+1)$-ní přímka je rozdělena průsečíky s předchozími přímkami na nejvýše $k+1$ úseků a každý z nich oddělí novou oblast roviny. Celkem tedy bude rovina rozdělena našimi přímkami na nejvýše tento počet oblastí:

$$\frac{1}{2}k(k+1)+1 \ + \ (k+1) \ = \ \frac{1}{2}k(k+1)+\frac{1}{2}\cdot 2(k+1) \ + \ 1 \ = \ \frac{1}{2}(k+1)(k+2) \ + \ 1$$

A toto je přesně naše tvrzení pro $n = k+1$, takže jsme hotovi.

□

Příklad 4.8. *Další indukční důkaz rozepsaný v podrobných krocích.*

Věta. *Pro každé $n \geq 0$ platí $\sum_{j=0}^{n} j = \frac{n(n+1)}{2}$.*

`Důkaz` indukcí vzhledem k n.

- Báze: Zde musíme dokázat platnost tvrzení pro $n := 0$, což je v tomto případě rovnost $\sum_{j=0}^{0} j = \frac{0(0+1)}{2}$. Tato rovnost (zjevně) platí.
- Indukční krok: Musíme dokázat, pro každé $k \geq 0$, že z platnosti tvrzení pro $n := k$ vyplývá platnost pro $n := k+1$, což konkrétně znamená:

$$\textit{Jestliže platí } \textstyle\sum_{j=0}^{k} j = \frac{k(k+1)}{2}\textit{, pak platí } \sum_{j=0}^{k+1} j = \frac{(k+1)(k+1+1)}{2}.$$

Předpokládejme tedy, že $\sum_{j=0}^{k} j = \frac{k(k+1)}{2}$ a pokusme se dokázat, že pak také

$$\sum_{j=0}^{k+1} j = \frac{(k+1)(k+1+1)}{2} = \frac{(k+1)(k+2)}{2}.$$

To už plyne přímočarou úpravou:

$$\sum_{j=0}^{k+1} j \ = \ \sum_{j=0}^{k} j \ + (k+1) \ = \ \frac{k(k+1)}{2} + (k+1) \ = \ \frac{k(k+1)+2(k+1)}{2} \ = \ \frac{(k+1)(k+2)}{2}$$

Podle principu matematické indukce je celý důkaz hotov. □

4.4 Komentáře k matematické indukci

Pro správné a úspěšné použití indukce v dokazování je vhodné si zapamatovat několik cenných rad:

* Základní trik všech důkazů matematickou indukcí je vhodná *reformulace* tvrzení $T(n+1)$ tak, aby se „odvolávalo“ na tvrzení $T(n)$.
  - Dobře se vždy podívejte, v čem se liší tvrzení $T(n+1)$ od tvrzení $T(n)$. Tento „rozdíl“ budete muset v důkaze zdůvodnit.
* Především dokud se matematickou indukci teprve učíte, důsledně používejte následující pomůcku. Zaveďte si další proměnnou k a formulujte svá tvrzení a úpravy stylem „platí-li naše tvrzení $T(n)$ pro $n := k$, pak bude platit i pro $n := k+1$“. Elegantně se tak vyhnete nejasnostem a chybám vznikajícím popletením n a $n+1$.
* Pozor, občas je potřeba „*zesílit*“ tvrzení $T(n)$, aby indukční krok správně „fungoval“ (a jsou případy, kdy tento trik velmi pomáhá).
  - Velkým problémem bohužel je, že není možno podat návod, jak vhodné zesílení nalézt (ani kdy jej vůbec hledat). Jedná se vlastně o pokusy a „hádání z křišťálové koule“. Viz Příklad 4.9.
* Často se chybuje v důkazu indukčního kroku, neboť ten bývá většinou výrazně obtížnější než báze, ale o to *zrádnější* jsou chyby v samotné zdánlivě snadné bázi!
  - Dejte si dobrý pozor, od které hodnoty $n \geq k_0$ je indukční krok univerzálně platný a jestli případně báze nezahrnuje více než jednu hodnotu…

Zesílení tvrzení v indukčním kroku

Potřebu nahrazení, v některých případech, daného tvrzení něčím „silnějším“ pro hladký průběh matematické indukce si ukážeme v Příkladu 4.9. Jaké zesílení však máme na mysli? V prvé řadě to musí být nějaké tvrzení $T'(n)$, ze kterého původní tvrzení $T(n)$ jednoduše vyplývá. Jak si tím ale v indukčním kroku pomůžeme? Je na jedné straně pravdou, že nově musíme dokazovat silnější závěr $T'(n+1)$, ale zároveň mám k dispozici silnější indukční předpoklad $T'(n)$ a to právě může být klíčovou pomocí.

Příklad 4.9. *Kdy je vhodné (a v zásadě také nutné) indukční krok zesílit…*

Věta. *Pro každé $n \geq 1$ platí*

$$s(n) = \frac{1}{1\cdot 2} + \frac{1}{2\cdot 3} + \frac{1}{3\cdot 4} + \cdots + \frac{1}{n(n+1)} \; < 1\,.$$

`Důkaz`: *Báze indukce* je zřejmá, neboť $\frac{1}{1\cdot 2} < 1$.
Co však *indukční krok*? Předpoklad $s(n) < 1$ je sám o sobě „*příliš slabý*“ na to, aby bylo možno tvrdit $s(n+1) = s(n) + \frac{1}{(n+1)(n+2)} < 1$.

Neznamená to ještě, že by tvrzení nebylo platné, jen je potřeba náš indukční předpoklad *zesílit*. Budeme dokazovat

Tvrzení. *Pro každé přirozené $n \geq 1$ platí $s(n) \leq 1 - \frac{1}{n+1} < 1$.*

To platí pro $n = 1$ (nezapomeňte ověřit!) a dále už úpravou jen dokončíme zesílený indukční krok:

$$s(n+1) \;=\; s(n) + \frac{1}{(n+1)(n+2)} \;\leq\; 1 - \frac{1}{n+1} + \frac{1}{(n+1)(n+2)} =$$

$$= \; 1 + \frac{-(n+2)+1}{(n+1)(n+2)} \;=\; 1 - \frac{1}{n+2} \qquad \square$$

Rozšíření báze a předpokladu

Mimo zesilování tvrzení indukčního kroku jsme někdy okolnostmi nuceni i k *rozšiřování* samotné báze indukce a s ní indukčního předpokladu na více než jednu hodnotu parametru n.

- Můžeme například předpokládat platnost (parametrizovaných) tvrzení $T(n)$ *i* $T(n+1)$ *zároveň*, a pak odvozovat platnost $T(n+2)$. Toto lze samozřejmě zobecnit na jakýkoliv počet předpokládaných parametrů.
- Můžeme dokonce předpokládat platnost tvrzení $T(j)$ *pro všechna* $j = k_0, k_0+1, \ldots, n$ najednou a dokazovat $T(n+1)$. Toto typicky využijeme v případech, kdy indukční krok „rozdělí“ problém $T(n+1)$ na dvě menší části a z nich pak odvodí platnost $T(n+1)$.

`Fakt`: Obě prezentovaná „rozšíření“ jsou v konečném důsledku jen speciálními instancemi základní matematické indukce; použité rozšířené možnosti pouze zjednodušují formální zápis důkazu.

Příklad 4.10. *Když je nutno rozšířit bázi a indukční předpoklad...*

Věta. *Nechť funkce f pro každé $n \geq 0$ splňuje vztah*

$$f(n+2) = 2f(n+1) - f(n).$$

Pokud platí $f(0) = 1$ a zároveň $f(1) = 2$, tak platí $f(n) = n + 1$ pro všechna přirozená $n \geq 0$.

`Důkaz`: Už samotný pohled na daný vztah $f(n+2) = 2f(n+1) - f(n)$ naznačuje, že bychom měli *rozšířit indukční předpoklad* (a krok) zhruba takto:

Pro každé přirozené $k \geq 0$; jestliže platí $f(n) = n + 1$ pro $n := k$ a zároveň pro $n := k + 1$ (neboli $f(k) = k + 1$ a $f(k + 1) = k + 2$), pak $f(n) = n + 1$ platí také pro $n := k + 2$ (neboli $f(k + 2) = k + 3$).

Báze indukce – pozor, zde už musíme ověřit dvě hodnoty pro $n := 0$ a $n := 1$:

$$f(0) = 0 + 1 = 1, \quad f(1) = 1 + 1 = 2$$

Náš *indukční krok* tak nyní může využít celého rozšířeného předpokladu, znalosti hodnot $f(k)$ i $f(k + 1)$, pro ověření požadovaného vztahu pro $n := k + 2$

$$f(n) = f(k+2) = 2f(k+1) - f(k) = 2 \cdot (k+1+1) - (k+1) = k+3 = n+1\,. \qquad \square$$

`Komentář:` Jak by tento důkaz měl být formulován v tradiční indukci? („Substitucí“ nového tvrzení.)

Závěrem malý „problém“

Příklad 4.11. *Aneb jak snadno lze v matematické indukci udělat chybu.*

Věta. *(„nevěta“)*
V každém stádu o $n \geq 1$ koních mají všichni koně stejnou barvu.

`Důkaz` indukcí vzhledem k n.
Báze: Ve stádu o jednom koni mají všichni koně stejnou barvu.
Indukční krok: Nechť $S = \{K_1, \ldots, K_{n+1}\}$ je stádo o $n+1$ koních. Dokážeme, že všichni koně mají stejnou barvu. Uvažme dvě menší stáda:

- $S' = \{K_1, \underline{K_2}, \ldots, K_n\}$
- $S'' = \{\underline{K_2}, \ldots, K_n, K_{n+1}\}$

Podle indukčního předpokladu mají všichni koně ve stádu S' stejnou barvu B'. Podobně všichni koně ve stádu S'' mají podle indukčního předpokladu stejnou barvu B''. Dokážeme, že $B' = B''$, tedy že všichni koně ve stádu S mají stejnou barvu. To ale plyne z toho, že koně $K_2, \ldots, K_n$ patří jak do stáda S', tak i do stáda S''. □

`Komentář:` *Ale to už je podvod! Vidíte, kde?*

Navazující studium

Jak jsme již řekli, matematické důkazy a jejich chápání jsou samozřejmě nezbytné ke studiu vysokoškolských matematických předmětů. Bez schopnosti přesného vyjadřování a chápání definic a vět se ale informatik neobejde, ani pokud se zaměřuje čistě aplikovaným směrem; příkladem je třeba správné pochopení různých norem a specifikací.

Na druhou stranu umění “tvořit” nové matematické důkazy je dosti obtížné a nedá se jemu jen tak snadno naučit – vyžaduje to mnoho tréninku a pokročilých matematických zkušeností. Jelikož je schopnost formálního matematického dokazování nezbytná (převážně jen) v teoretických informatických disciplínách, není tato část kritická v celém rozsahu našeho předmětu (a u zkoušek se objeví s menším důrazem), ale to neznamená, že byste se jí mohli zcela vyhýbat. Obzvláště techniku matematické indukce by měl každý informatik aspoň trochu ovládat, neboť s jejím použitím se zajisté ještě mnohokrát setkáte v budoucím studiu. Například znovu v Lekcích 5 a 11 tohoto textu.

5 Rekurze, Strukturální indukce

Úvod

Mimo „přímočarých“ definic množin a funkcí, jako výčtem prvků či explicitním zadáním, se obzvláště v informatice setkáváme s definicemi nepřímými, které se odvolávají na sebe sama. Ostatně jde o problematiku velmi podobnou rekurentním programům, které jste už sami mohli potkat. Odborně se taková situace nazývá rekurzí či induktivní definicí. My si celou problematiku probereme z matematického pohledu od jednoduchých rekurentních definic posloupností po induktivní definice množin a funkcí a dokazování nad nimi.

Cíle

Náplní našeho výkladu je vysvětlit nelehkou oblast induktivních (tj. na sebe se odvolávajících) definic množin a funkcí, se kterými se hojně pracuje ve všech oblastech informatiky. Spojíme si tak látku Lekcí 4 a 3 do nového formalismu strukturální matematické indukce.

5.1 Posloupnosti a rekurentní vztahy

Pokud chceme obecně v matematice vyjádřit uspořádaný soubor objektů, používáme pojem *posloupnost*. Důležité je si uvědomit, že na rozdíl od množin je opakování prvků v posloupnosti povoleno(!). Uspořádané k-tice (viz Oddíl 3.2) z daného oboru hodnot H jsou tak nazývány *konečnými posloupnostmi* délky k (nad H). Pojem posloupnosti zobecňuje toto pojetí na „nekonečnou délku“ takto:

Definice 5.1. ***Posloupnost*** (nekonečná) je zobrazením z přirozených čísel $\mathbb{N}$ do oboru hodnot H, neboli

$$p : \mathbb{N} \to H.$$

Místo „funkčního“ zápisu n-tého členu posloupnosti jako $p(n)$ častěji používáme „indexovou“ formu jako p_n, ve které se celá posloupnost zapíše $(p_n)_{n\in\mathbb{N}}$.

`Komentář:` Oborem hodnot H posloupnosti obvykle bývá nějaká číselná množina, ale může to být i jakákoliv jiná množina.

`Poznámka`: Třebaže to není zcela formálně přesné, běžně se setkáme s posloupnostmi indexovanými od nuly nebo od jedničky, jak se to v aplikacích hodí (bez ohledu na to, zda nulu považujeme za přirozené číslo nebo ne). I my se budeme touto „uvolněnou“ konvencí řídit a vždy si určíme počáteční index podle potřeby.

Pojem poslopupnosti si nejlépe ilustrujeme několika příklady:

- $p_0 = 0,\ p_1 = 2, \ldots, p_i = 2i, \ldots$ je posloupnost sudých nezáporných čísel,
- $(3,\ 3.1,\ 3.14,\ 3.141, \ldots)$ je posloupnost postupných dekadických rozvojů čísla π,
- třeba (*jablko, hruška, švestka, hruška, hruška,...*) je posloupnost nad druhy ovoce coby oborem hodnot.
- $(1,\ -1,\ 1,\ -1, \ldots)$ je posloupnost určená vztahem $p_i = (-1)^i$, $i \geq 0$,
- avšak pokud bychom chtěli stejnou posloupnost $(1,\ -1,\ 1,\ -1, \ldots)$ určit indexy od jedné, tj. zadat ji jako q_i, $i \geq 1$, tak definiční vzorec bude vypadat (proč?) $q_i = (-1)^{i-1}$.

`Definice`: Posloupnost $(p_n)_{n\in\mathbb{N}}$ je
* *rostoucí* pokud $p_{n+1} > p_n$ a *nerostoucí* pokud $p_{n+1} \leq p_n$,
* *klesající* pokud $p_{n+1} < p_n$ a *neklesající* pokud $p_{n+1} \geq p_n$

platí pro všechna n.

Rekurentní zadání posloupnosti

Když je naše posloupnost definována jako zobrazení z přirozených čísel, proč se jí věnujeme zvlášť, mimo běžné funkce? Asi hlavním důvodem pro tuto zvláštní péči je možnost definovat posloupnosti pomocí odkazů na sebe sama – tzv. *rekurentně* (což třeba pro reálné funkce možno není).

`Definice`: Říkáme, že posloupnost $(p_n)_{n\in\mathbb{N}}$ je zadána *rekurentně*, pokud je dán její počáteční člen p_0 (či několik počátečních členů) a dále máme předpis, jak určit hodnotu členu p_{n+1} z hodnot p_i pro nějaká $i \leq n$.

`Komentář:` Ukázky rekurentně zadaných posloupností:
- Zadáme-li posloupnost p_n počátečním členem $p_0 = 1$ a rekurentním vztahem $p_{n+1} = 2p_n$ pro $n \geq 0$, pak platí $p_n = 2^n$ pro všechna n.
- Funkce „faktoriál“ (na přirozených číslech) je dána počáteční hodnotou $0! = 1$ a rekurentním vztahem $(n+1)! = (n+1)\cdot n!$ pro $n \geq 0$.
- Známá Fibonacciho posloupnost je zadaná počátečními členy $f_1 = f_2 = 1$ a vztahem $f_{n+2} = f_{n+1} + f_n$ pro $n \geq 1$.

Všimněte si v poslední ukázce, že není potřeba doslova dodržovat formu rekurentního vztahu „p_{n+1} z hodnot p_i“, ale vždy je třeba hodnotu následujícího členu posloupnosti odvozovat z předchozích (tj. už určených) členů, v tom případě „f_{n+2} z hodnot f_i, $i = n, n+1$“.

Příklad 5.2. *Posloupnost $f : \mathbb{N} \to \mathbb{Z}$ je zadaná rekurentní definicí*

$$f(0) = 3 \quad a \quad f(n+1) = 2\cdot f(n) + 1$$

pro všechna přirozená n. Určete hodnotu $f(n)$ explicitním vzorcem v závislosti na n.

Řešení: V první fázi řešení takového příkladu musíme nějak „uhodnout“ hledaný vzorec pro $f(n)$. Jak? Zkusíme vypočítat několik prvních hodnot a uvidíme…

$$\begin{aligned} f(1) &= 2\cdot f(0) + 1 = 2\cdot 3 + 1 = 7 \\ f(2) &= 2\cdot f(1) + 1 = 2\cdot 7 + 1 = 15 \\ f(3) &= 2\cdot f(2) + 1 = 2\cdot 15 + 1 = 31 \\ f(4) &= 2\cdot f(3) + 1 = 2\cdot 31 + 1 = 63 \end{aligned}$$

Nepřipomínají nám tato čísla něco? Co třeba výrazy $8-1$, $16-1$, $32-1$, $64-1\ldots$? Bystrému čtenáři se již asi podařilo uhodnout, že půjde o mocniny dvou snížené o 1. Přesněji, $f(n) = 2^{n+2} - 1$ (proč je exponent $n+2$? inu proto, aby správně vyšlo $f(0)$).

Ve druhé fázi nesmíme ale zapomenout správnost našeho „věštění“ dokázat, nejlépe matematickou indukcí podle n.

- Báze pro $n := 0$ říká $f(0) = 2^{0+2} - 1 = 4 - 1 = 3 = f(0)$, což platí.

- Indukční krok využije indukční předpoklad platnosti vztahu $f(n) = 2^{n+2}-1$ pro $n := i$ (tj. $f(i) = 2^{i+2}-1$), kde i je libovolné přirozené číslo, a pokračuje úpravou ze zadaného rekurentního vzorce

$$f(i+1) = 2 \cdot f(i) + 1 = 2 \cdot (2^{i+2} - 1) + 1 = 2 \cdot 2^{i+2} - 2 + 1 = 2^{(i+1)+2} - 1,$$

  což po dosazení pro $n := i + 1$ znamená opět požadovaný vztah $f(n) = 2^{n+2} - 1$.

Podle principu matematické indukce je nyní dokázáno, že pro zadanou rekurentní posloupnost f platí $f(n) = 2^{n+2} - 1$ pro všechna přirozená n. □

Příklad 5.3. *Posloupnost $(s_n)_{n\in\mathbb{N}}$ je zadaná rekurentní definicí*

$$s_0 = 0 \quad \text{a} \quad s_n = s_{n-1} + n^2$$

pro všechna $n \geq 1$. Dokažte, že $s_n = \frac{1}{6}n(n+1)(2n+1)$ pro všechna přirozená n.

Řešení: Opět postupujeme matematickou indukcí podle n.

- Báze $n := 0$: $s_0 = \frac{1}{6} \cdot 0 \cdot (0+1)(2 \cdot 0 + 1) = 0$, což platí.
- Indukční krok: za využití $s_{i-1} = \frac{1}{6}(i-1)(i-1+1)(2i-2+1) = \frac{1}{6}(i-1)i(2i-1)$, což je indukční předpoklad pro $n := i - 1$ a libovolné $i \geq 1$, upravujeme daný rekurentní vzorec jako

$$s_i = s_{i-1} + i^2 = \frac{1}{6}(i-1)i(2i-1) + i^2 = \frac{1}{6}i\,(2i^2 - 3i + 1) + i^2$$

$$= \frac{1}{6}i\,(2i^2 - 3i + 1 + 6i) = \frac{1}{6}i(i+1)(2i+1),$$

  což dokazuje požadovaný vztah $s_n = \frac{1}{6}n(n+1)(2n+1)$ také pro $n := i$ a podle principu matematické indukce jsme hotovi. □

`Poznámka`: Výsledek Příkladu 5.3 je ukázkou tzv. sumačního vzorce pro řadu

$$1^2 + 2^2 + \cdots + n^2 = \frac{1}{6}n(n+1)(2n+1).$$

Jinou podobnou ukázkou je už dříve zmíněný základní vztah $1 + 2 + \cdots + n = \frac{1}{2}n(n+1)$.

A podobně můžeme odvodit například

$$1 + 3 + \cdots + (2n-3) + (2n-1) = 1 + (2n-1) \; + \; 3 + (2n-3) \; + \cdots = 2n \cdot \frac{n}{2} = n^2,$$

nebo už bez zdůvodnění $1^3 + 2^3 + \cdots + n^3 = (1 + 2 + \cdots + n)^2 = \frac{1}{4}n^2(n+1)^2$. Jakmile vztah tohoto typu „uhodneme“, není těžké jej vždy dokázat indukcí. Jak ale dojdeme k tomu uhodnutí takového vztahu? Pomoci si můžeme jednoduchým trikem – součet prvních řekněme n polynomiálních členů lze vyjádřit *nějakým* polynomem v n stupně o jedna vyššího. Napišme si tedy třeba $1^2 + 2^2 + \cdots + n^2 = An^3 + Bn^2 + Cn + D$ a postupným dosazením $n = 0, 1, 2, 3$ získejme soustavu čtyř lineárních rovnic o čtyřech neznámých, které nám pomohou určit správné hodnoty A, B, C, D. Vyzkoušejte si sami.

Závěrem zopakujeme, že odvozování a práce s jednoduchými sumačními vzorci by měla náležet do výbavy každého schopného informatika. Princip matematické indukce je pro tuto práci klíčový.

5.2 Induktivní definice množin a funkcí

Rekurzivně lze určit nejen hodnoty členů posloupnosti, ale mnohem šířeji lze určit i prvky množiny a definovat obecnou funkci na nich. Koncept induktivních definic množin a funkcí není zcela lehký, ale uvidíme, že v informatice je velmi přirozený a důležitý.

Definice 5.4. ***Induktivní definice*** množiny.
Jedná se obecně o popis (nějaké) množiny M v následujícím tvaru:

- Je dáno několik pevných (*bázických*) prvků $a_1, a_2, \ldots, a_k \in M$.
- Je dán soubor *induktivních pravidel* typu

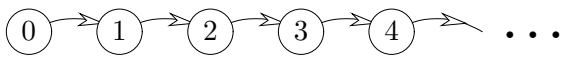
  Jsou-li (libovolné prvky) $x_1, \ldots, x_\ell \in M$, pak také $y \in M$.

  V tomto případě je y typicky funkcí $y = f_i(x_1, \ldots, x_\ell)$.

Pak naše *induktivně definovaná množina* M je určena jako nejmenší (inkluzí) množina vyhovující všem těmto pravidlům.

`Komentář:` Několik ukázek...

- Pro nejbližší příklad induktivní definice se obrátíme na množinu všech přirozených čísel, která je formálně zavedena následovně.
  - $0 \in \mathbb{N}$
  - Je-li $i \in \mathbb{N}$, pak také $succ(i) \in \mathbb{N}$ (kde následník $succ(i)$ odpovídá hodnotě $i + 1$).

- Pro každé $y \in \mathbb{N}$ můžeme definovat jinou množinu $M_y \subseteq \mathbb{N}$ induktivně takto:
  - $y \in M_y$
  - Jestliže $x \in M_y$ a $x + 1$ je liché, pak $x + 2 \in M_y$.

  Pak například $M_3 = \{3\}$, nebo $M_4 = \{4 + 2i \mid i \in \mathbb{N}\}$.
- Dalším příkladem induktivní definice je už známé zavedení *výrokových formulí* z Oddílu 1.4. Uměli byste teď přesně říci, co tam byly bázické prvky a jaká byla induktivní pravidla? A jaká byla v definici formulí role závorek? K tomu se blíže vyjádříme na závěr této lekce.

Jednoznačnost induktivních definic

`Definice`: Řekneme, že daná induktivní definice množiny M je *jednoznačná*, právě když každý prvek M lze odvodit z bázických prvků pomocí induktivních pravidel právě *jedním způsobem*.

`Komentář:` Definujme například množinu $M \subseteq \mathbb{N}$ induktivně takto:
– $2, 3 \in M$
– Jestliže $x, y \in M$ a $x \leq y$, pak také $x^2 + y^2$ a $x \cdot y$ jsou prvky M.
Proč tato induktivní definice není jednoznačná? Například číslo $8 \in M$ lze odvodit způsobem $8 = 2 \cdot (2 \cdot 2)$, ale zároveň zcela jinak $8 = 2^2 + 2^2$.

V čem tedy spočívá důležitost jednoznačných induktivních definic množin? Je to především v dalším možném využití induktivní definice množiny jako „základny“ pro odvozené vyšší definice, viz následující Definice 5.5 a třeba doplňková Věta 5.11. Stručně a

neformálně řečeno, hlavní role jednoznačnosti induktivní definice je v možnosti pak přiřadit prvkům této induktivní množiny nějaký „jednoznačný význam“.

Induktivně definovaná množina povětšinou nemá valný význam sama o sobě, avšak poskytuje *definiční obor* pro následnou induktivně definovanou funkci:

Definice 5.5. ***Induktivní definice funkce*** z induktivní množiny.
Nechť množina M je dána jednoznačnou induktivní definicí. Pak říkáme, že funkce $\mathcal{F} : M \to X$ je definována *induktivně* (vzhledem k induktivní definici M), pokud je řečeno:

- Pro každý z bázických prvků $a_1, a_2, \ldots, a_k \in M$ je určeno $\mathcal{F}(a_i) = c_i$, kde c_i je konstanta.
- Pro každé induktivní pravidlo typu

  “Jsou-li (libovolné prvky) $x_1, \ldots, x_\ell \in M$, pak také $f(x_1, \ldots, x_\ell) \in M$”

  je definováno

  $$\mathcal{F}\big(f(x_1, \ldots, x_\ell)\big) \text{ na základě hodnot } \mathcal{F}(x_1), \ldots, \mathcal{F}(x_\ell).$$

`Komentář:` Ilustrujme si induktivní definici funkce dětskou hrou na „tichou poštu“. Definičním oborem je řada sedících hráčů, kde ten první je bázickým prvkem a každý následující (mimo posledního) odvozuje hráče sedícího hned za ním jako další prvek hry. Hodnotou bázického prvku je první (vymyšlené) posílané slovo. Induktivní pravidlo pak následujícímu hráči přiřazuje slovo, které je odvozeno („zkomolením“) ze slova předchozího hráče. Výsledkem hry pak je hodnota–slovo posledního hráče.

Pro další příklad se podívejme třeba do manuálových stránek unixového příkazu `test EXPRESSION`:

```
EXPRESSION is true or false and sets exit status.  It is one of:
 ( EXPRESSION )               EXPRESSION is true
 ! EXPRESSION                 EXPRESSION is false
 EXPRESSION1 -a EXPRESSION2   both EXPRESSION1 and EXPRESSION2 are true
 EXPRESSION1 -o EXPRESSION2   either EXPRESSION1 or EXPRESSION2 is true
 [-n] STRING                  the length of STRING is nonzero
 STRING1 = STRING2            the strings are equal
   ......
```

Vidíte, jak tato ukázka krásně koresponduje s Definicí 5.5? No, ne úplně, poněkud problematická je otázka jednoznačnosti této definice – jednoznačnost není vynucena (jen umožněna) syntaktickými pravidly, jinak je pak dána nepsanými konvencemi implementace příkazu. To je pochopitelně z matematického hlediska špatně, ale přesto jde o pěknou ukázku z praktického života informatika.

5.3 Použití strukturální indukce

Induktivní definici množiny a funkce z ní si znovu podrobněji ilustrujeme primitivní ukázkou. Účelem předloženého příkladu je názorně vysvětlit, kterak lze přirozeně zkombinovat induktivní definice s pokročilou formou matematické indukce v dokazování, s tzv. *strukturální indukcí*.

Příklad 5.6. *Jednoduché aritmetické výrazy a jejich význam.*
Nechť je dána abeceda $\Sigma = \{0,1,2,3,4,5,6,7,8,9,\odot,\oplus,(,)\}$. Definujme množinu *jednoduchých výrazů* $SExp \subseteq \Sigma^*$ induktivně takto:

* Dekadický zápis každého přirozeného čísla $\mathbf{n}$ je prvkem *SExp* (bázické prvky).
* Jestliže $x, y \in SExp$, pak také $(x) \odot (y)$ a $(x) \oplus (y)$ jsou prvky *SExp*.
* Jak snadno nahlédneme, díky nucenému závorkování je tato induktivní definice jednoduchých výrazů *SExp jednoznačná*.

Tímto jsme jednoduchým aritmetickým výrazům přiřadili jejich „formu“, tedy *syntaxi*.

Pro přiřazení „významu“, tj. *sémantiky* aritmetického výrazu, následně definujme funkci vyhodnocení $Val : SExp \to \mathbb{N}$ induktivně takto:

* Pro bázické prvky: $Val(\mathbf{n}) = n$, kde $\mathbf{n}$ je dekadický zápis přirozeného čísla n.
* První induktivní pravidlo: $Val((x) \oplus (y)) = Val(x) + Val(y)$.
* Druhé induktivní pravidlo: $Val((x) \odot (y)) = Val(x) \cdot Val(y)$.

Co je pak „správným významem“ (*hodnotou*) uvedených jednoduchých aritmetických výrazů? (Příklad 5.7) □

Příklad 5.7. *Důkaz správnosti přiřazeného „významu“* $Val : SExp \to \mathbb{N}$.

> **Věta**. *Pro každý výraz $\sigma \in SExp$ je hodnota $Val(\sigma)$ z Příkladu 5.6 číselně rovna výsledku vyhodnocení výrazu σ podle běžných zvyklostí aritmetiky.*

Jelikož pojednáváme o induktivně definované funkci *Val*, je přirozené pro důkaz jejích vlastností aplikovat *matematickou indukci*. Avšak na rozdíl od dříve probíraných příkladů zde nevidíme žádný celočíselný „parametr n“, a proto si jej budeme muset nejprve definovat podle *struktury* výrazu σ (mluvíme proto o *strukturální indukci*).

Přesněji, naši indukci povedeme podle „délky ℓ odvození výrazu σ“ definované jako *počet aplikací induktivních pravidel* potřebných k odvození $\sigma \in SExp$.

`Důkaz`: V *bázi indukce* ověříme vyhodnocení bázických prvků, kteréžto jsou zde dekadické zápisy přirozených čísel. Platí $Val(\mathbf{n}) = n$, což skutečně odpovídá zvyklostem aritmetiky.

V *indukčním kroku* se podíváme na vyhodnocení $Val((x) \oplus (y)) = Val(x) + Val(y)$. Podle běžných zvyklostí aritmetiky by hodnota $Val((x) \oplus (y))$ měla být rovna *součtu* vyhodnocení výrazu x, což je podle indukčního předpokladu rovno $Val(x)$ (x má zřejmě kratší délku odvození), a vyhodnocení výrazu y, což je podle indukčního předpokladu rovno $Val(y)$. Takže skutečně $Val((x) \oplus (y)) = Val(x) + Val(y)$.

Druhé pravidlo $Val((x) \odot (y))$ se dořeší analogicky. □

5.4 Nazpět k matematické logice

Vybaveni aparátem induktivních definic, můžeme nyní podat matematicky přesnější definici formulí výrokové logiky z Oddílu 1.4 a jejich sémantiky.

`Definice`: Nechť $\mathcal{P} = \{A, B, C, \dots\}$ je (nekonečná spočetná) množina výrokových proměnných. Množina $\mathcal{F}$ výrokových formulí je dána těmito pravidly induktivní definice:

* Bázickými prvky $\mathcal{F}$ jsou výrokové proměnné $\mathcal{P}$, tj. $X \in \mathcal{F}$ pro každé $X \in \mathcal{P}$.
* (*negace*) Je-li $\varphi \in \mathcal{F}$, pak také $\neg(\varphi)$ je prvkem $\mathcal{F}$.
* (*implikace*) Je-li $\varphi, \psi \in \mathcal{F}$, pak také $(\varphi) \Rightarrow (\psi)$ je prvkem $\mathcal{F}$.

Zároveň platí, že závorky okolo φ lze vynechat pouze pokud $\varphi \in \mathcal{P}$ nebo φ bylo vytvořeno induktivním pravidlem negace. To stejné platí pro vynechávání závorek okolo ψ.

`Poznámka`: Již nad rámec předchozí definice patří dříve uvedené syntaktické zkratky

* $\varphi \vee \psi$ (*disjunkce* /„nebo“) je jiný zápis formule $\neg\varphi \Rightarrow \psi$,
* $\varphi \wedge \psi$ (*konjunkce* /„a“) je jiný zápis formule $\neg(\neg\varphi \vee \neg\psi)$,
* $\varphi \Leftrightarrow \psi$ (*ekvivalence*) je jiný zápis formule $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$.

Na základě *jednoznačné* induktivní definice množiny $\mathcal{F}$ výrokových formulí nyní můžeme podat přesnou induktivní definici funkce vyhodnocení (logické hodnoty formulí), kterou jsme doposud znali z Definice 1.8.

Nechť *valuace* (ohodnocení) je funkce $\nu : \mathcal{P} \to \{0, 1\}$.

Definice 5.8. ***Sémantika*** (význam) výrokové logiky.
Pro každou valuaci ν definujeme funkci

$$\mathcal{S}_\nu : \mathcal{F} \to \{0, 1\},$$

zvanou *vyhodnocení* formule σ vzhledem k ν, induktivně takto:

* $\mathcal{S}_\nu(X) := \nu(X)$ pro každé $X \in \mathcal{P}$.
* $\mathcal{S}_\nu(\neg\varphi) := \begin{cases} 1 & \text{jestliže } \mathcal{S}_\nu(\varphi) = 0; \\ 0 & \text{jinak.} \end{cases}$
* $\mathcal{S}_\nu(\varphi \Rightarrow \psi) := \begin{cases} 0 & \text{jestliže } \mathcal{S}_\nu(\varphi) = 1 \text{ a } \mathcal{S}_\nu(\psi) = 0; \\ 1 & \text{jinak.} \end{cases}$

Převod formulí do normálního tvaru

V dalším se vrátíme k logickým formulím v normálním tvaru (viz Oddíl 2.2), tedy k formulím, ve kterých se operátor negace vztahuje pouze k výrokovým proměnným. Význam normálního tvaru pro lidské chápání významu formulí jsme si již zdůvodnili a nyní si formálně dokážeme, že vždy můžeme normálního tvaru dosáhnout.

Tvrzení 5.9. *Každou výrokovou formuli lze převést do normálního tvaru, pokud k $\Rightarrow$ povolíme i užívání odvozených spojek $\wedge$ a $\vee$.*

Důkaz Tvrzení 5.9 pak přímo vyplývá z jednoduchého mechanického postupu převodu formulí do normálního tvaru popsaného v následující Metodě 5.10.

Metoda 5.10. ***Převod formule φ do normálního tvaru $\mathcal{F}(\varphi)$.***
Pro převod použijeme funktory $\mathcal{F}$ a $\mathcal{G}$ s neformálním významem $\mathcal{F}(X)$ jako „je pravda, že X“ a $\mathcal{G}(X)$ jako „není pravda, že X“. Tyto funktory definujeme induktivními předpisy následovně:

$$\begin{array}{lcllcl}
\mathcal{F}(A) & = & A & \mathcal{G}(A) & = & \neg A \\
\mathcal{F}(\neg\varphi) & = & \mathcal{G}(\varphi) & \mathcal{G}(\neg\varphi) & = & \mathcal{F}(\varphi) \\
\mathcal{F}(\varphi \Rightarrow \psi) & = & \mathcal{F}(\varphi) \Rightarrow \mathcal{F}(\psi) & \mathcal{G}(\varphi \Rightarrow \psi) & = & \mathcal{F}(\varphi) \wedge \mathcal{G}(\psi) \\
\mathcal{F}(\varphi \wedge \psi) & = & \mathcal{F}(\varphi) \wedge \mathcal{F}(\psi) & \mathcal{G}(\varphi \wedge \psi) & = & \mathcal{G}(\varphi) \vee \mathcal{G}(\psi) \\
\mathcal{F}(\varphi \vee \psi) & = & \mathcal{F}(\varphi) \vee \mathcal{F}(\psi) & \mathcal{G}(\varphi \vee \psi) & = & \mathcal{G}(\varphi) \wedge \mathcal{G}(\psi) \\
\mathcal{F}(\varphi \Leftrightarrow \psi) & = & \mathcal{F}(\varphi) \Leftrightarrow \mathcal{F}(\psi) & \mathcal{G}(\varphi \Leftrightarrow \psi) & = & \mathcal{F}(\varphi) \Leftrightarrow \mathcal{G}(\psi)
\end{array}$$

Pro predikátové formule toto rozšíříme ještě o pravidla:

$$\begin{array}{lcllcl}
\mathcal{F}(\forall x\,.\,\varphi) & = & \forall x\,.\,\mathcal{F}(\varphi) & \mathcal{G}(\forall x\,.\,\varphi) & = & \exists x\,.\,\mathcal{G}(\varphi) \\
\mathcal{F}(\exists x\,.\,\varphi) & = & \exists x\,.\,\mathcal{F}(\varphi) & \mathcal{G}(\exists x\,.\,\varphi) & = & \forall x\,.\,\mathcal{G}(\varphi)
\end{array}$$

`Komentář:` Uvažme formuli $\neg(A \Rightarrow \neg(B \vee \neg(C \Rightarrow \neg A)))$. Užitím uvedeného postupu Metody 5.10 získáme:

$$\begin{array}{lcll} \mathcal{F}(\neg(A \Rightarrow \neg(B \vee \neg(C \Rightarrow \neg A)))) & = & \mathcal{G}(A \Rightarrow \neg(B \vee \neg(C \Rightarrow \neg A))) & = \\ \mathcal{F}(A) \wedge \mathcal{G}(\neg(B \vee \neg(C \Rightarrow \neg A))) & = & A \wedge \mathcal{F}(B \vee \neg(C \Rightarrow \neg A)) & = \\ A \wedge (\mathcal{F}(B) \vee \mathcal{F}(\neg(C \Rightarrow \neg A))) & = & A \wedge (B \vee \mathcal{G}(C \Rightarrow \neg A)) & = \\ A \wedge (B \vee (\mathcal{F}(C) \wedge \mathcal{G}(\neg A))) & = & A \wedge (B \vee (C \wedge \mathcal{F}(A))) & = \\ A \wedge (B \vee (C \wedge A)) & & & \end{array}$$

Formuli $A \wedge (B \vee (C \wedge A))$ lze dále zjednodušit na ekvivalentní formuli $A \wedge (B \vee C)$. To ale je již z našeho pohledu matematicky neformální (heuristický) postup.

Důkaz pro normální tvar formule

Na závěr následuje další ilustrativní ukázka důkazu strukturální indukcí.

Věta 5.11. *Pro libovolnou výrokovou formuli φ platí (viz Metoda 5.10), že*

a) $\mathcal{F}(\varphi)$ je formule v normálním tvaru ekvivalentní k φ

b) a $\mathcal{G}(\varphi)$ je formule v normálním tvaru ekvivalentní negaci $\neg\varphi$.

`Důkaz` povedeme *indukcí ke struktuře formule*, neboli indukci povedeme podle „délky“ ℓ – počtu aplikací induktivních pravidel při sestavování formule φ.

- Báze indukce ($\ell = 0$): Pro všechny atomy, tj. výrokové proměnné, zřejmě platí, že $\mathcal{F}(A) = A$ je ekvivalentní A a $\mathcal{G}(A) = \neg A$ je ekvivalentní $\neg A$.
- V indukčním kroku předpokládejme, že a) i b) platí pro všechny formule φ délky nejvýše ℓ. Vezmeme si formuli ψ délky $\ell + 1$, která je utvořená jedním z následujících způsobů:
  - $\psi \equiv \neg\varphi$. Podle výše uvedeného induktivního předpisu je $\mathcal{F}(\psi) = \mathcal{F}(\neg\varphi) = \mathcal{G}(\varphi)$. Podle indukčního předpokladu pak je $\mathcal{G}(\varphi)$ formule v normálním tvaru ekvivalentní $\neg\varphi = \psi$. Obdobně pro funktor $\mathcal{G}$ vyjádříme $\mathcal{G}(\psi) = \mathcal{G}(\neg\varphi) = \mathcal{F}(\varphi)$. Podle indukčního předpokladu pak je $\mathcal{F}(\varphi)$ formule v normálním tvaru ekvivalentní φ a to je dále ekvivalentní $\neg\neg\varphi = \neg\psi$ podle Tvrzení 1.11.
  - $\psi \equiv (\varphi_1 \Rightarrow \varphi_2)$. Podle výše uvedeného induktivního předpisu je $\mathcal{F}(\psi) = \mathcal{F}(\varphi_1 \Rightarrow \varphi_2) = \mathcal{F}(\varphi_1) \Rightarrow \mathcal{F}(\varphi_2)$. Podle indukčního předpokladu jsou $\mathcal{F}(\varphi_1)$ i $\mathcal{F}(\varphi_2)$ formule v normálním tvaru ekvivalentní φ_1 a φ_2. Potom i $\mathcal{F}(\varphi_1) \Rightarrow \mathcal{F}(\varphi_2)$ je v normálním tvaru dle definice a podle sémantiky $\Rightarrow$ je ta ekvivalentní formuli $(\varphi_1 \Rightarrow \varphi_2) = \psi$.
    Obdobně rozepíšeme $\mathcal{G}(\psi) = \mathcal{G}(\varphi_1 \Rightarrow \varphi_2) = \mathcal{F}(\varphi_1) \wedge \mathcal{G}(\varphi_2)$. Jelikož $\wedge$ je pro nás jen zkratka, výraz dále rozepíšeme $\mathcal{G}(\psi) = \neg(\mathcal{F}(\varphi_1) \Rightarrow \neg\mathcal{G}(\varphi_2))$. Podle indukčního předpokladu (a dvojí negace) jsou $\mathcal{F}(\varphi_1)$ a $\neg\mathcal{G}(\varphi_2)$ po řadě ekvivalentní formulím φ_1 a φ_2. Tudíž nakonec odvodíme, že $\mathcal{G}(\psi)$ je ekvivalentní negaci formule $\varphi_1 \Rightarrow \varphi_2$, což jsme zde měli dokázat.
  - $\psi \equiv (\varphi_1 \vee \varphi_2)$. Zde si musíme opět uvědomit, že spojka $\vee$ je pro nás jen zkratka, a přepsat $\psi \equiv (\neg\varphi_1 \Rightarrow \varphi_2)$. Potom podle předchozích dokázaných případů víme, že $\mathcal{F}(\psi) = \mathcal{F}(\neg\varphi_1 \Rightarrow \varphi_2) = \mathcal{F}(\neg\varphi_1) \Rightarrow \mathcal{F}(\varphi_2)$ je ekvivalentní formuli $(\neg\varphi_1 \Rightarrow \varphi_2) = \psi$, což bylo třeba dokázat. Stejně tak $\mathcal{G}(\psi) = \mathcal{G}(\neg\varphi_1 \Rightarrow \varphi_2) = \mathcal{F}(\neg\varphi_1) \wedge \mathcal{G}(\varphi_2)$ je podle předchozích případů důkazu ekvivalentní $(\neg\varphi_1 \wedge \neg\varphi_2) = \neg\psi$.

∗ $\psi \equiv (\varphi_1 \wedge \varphi_2)$ a $\psi \equiv (\varphi_1 \Leftrightarrow \varphi_2)$ už dokončíme analogicky. □

Rozšiřující studium

Poslední částí látky o důkazech a množinách byla problematika induktivních definic, které ač ve formálním podání mohou nejprve vypadat těžko pochopitelné, jsou ve skutečnosti zcela přirozenou popisnou metodou v mnoha aplikačních sférách informatiky a jejich alespoň intuitivní ovládání pro vás bude v dalším studiu informatiky nezbytné. Schválně, zkuste se podívat do kterékoliv vaší oblíbené oblasti informatiky, kde všude induktivní definice (tj. ty odvolávající se rekurzivně samy na sebe pro „menší případy"), najdete. Ať už přímo či nepřímo. V Lekci uvedený příklad příkazu '`test EXPRESSION`*' je jen vybraným zrníčkem písku v hromadě dalších použití.*

6 Relace a jejich vlastnosti

Úvod

Jak jsme si říkali v Lekci 3, relace a funkce patří mezi základní „datové typy“ matematiky. Zároveň na pojem (zcela obecné) relace velmi brzy narazí každý informatik při studiu dat a databází, které na něm staví. Není to však jen oblast relačních databází, ale i jiná místa informatiky, kde se abstraktní relace skrývají či přímo explicitně objevují. Nejčastěji se tak setkáte s binárními relacemi nad množinami, na něž se v textu hlouběji zaměříme.

Cíle

Rozšíříme látku závěru Lekce 3 o prezentace relací, zvláště v případě binárních relací. Ve vztahu k binárním relacím je zavedeno množství pojmů, které jsou používány v různých oblastech matematiky i informatiky. Ty zahrnují základní vlastnosti relací a jejich uzávěry. V látce pak plynule pokračujeme Lekcí 7.

6.1 Reprezentace konečných relací

Oblastí, kde informatici nejčastěji potkají obecné relace, je bezesporu ukládání dat. To proto, že shromažďovaná data, stejně jako relace, především sledují vztahy mezi objekty. Na druhou stranu je *relační databáze* zcela obecnou ukázkou reprezentace jakékoliv relace, kterou si ilustrujeme příkladem.

Příklad 6.1. *Tabulka relační databáze prezentuje obecnou relaci.*

Definujme následující množiny („elementární typy“)

* ZNAK $= \{a, \cdots, z, A, \cdots, Z, mezera\}$,
* CISLICE $= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Dále definujeme tyto množiny („odvozené typy“)

* JMENO $=$ ZNAK^{15}, PRIJMENI $=$ ZNAK^{20}, VEK $=$ CISLICE^3,
* ZAMESTNANEC „$\in$“ JMENO $\times$ PRIJMENI $\times$ VEK.

Relaci „typu“ *ZAMESTNANEC* pak lze reprezentovat tabulkou:

JMENO	*PRIJMENI*	*VEK*
Jan	Novák	42
Petr	Vichr	28
Pavel	Zíma	26
Stanislav	Novotný	52

□

Reprezentace binárních relací na množině

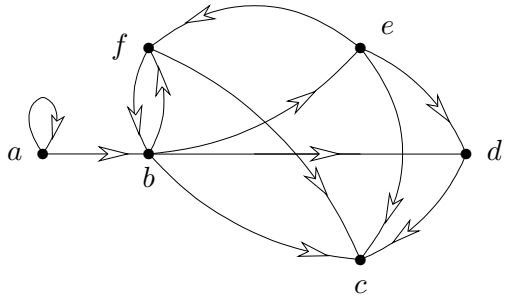
Jistě čtenáři uznají, že zadání relace výčtem jejích složek není pro člověka (na rozdíl od počítače) tím nejpříjemnějším způsobem. Je tedy přirozené se ptát, jak co nejnázorněji takovou relaci, alespoň v její nejčastější binární podobě, ukázat.

`Značení`: Binární relaci $R \subseteq M \times M$ lze jednoznačně znázornit jejím *grafem*:

- Prvky M znázorníme jako body v rovině (tj. na papíře).

- Prvek $(a, b) \in R$ znázorníme jako *orientovanou hranu* („šipku“) z a do b. Je-li $a = b$, pak je touto hranou „smyčka“ na a.

  `Komentář:` Například mějme $M = \{a, b, c, d, e, f\}$ a $R = \{(a, a), (a, b), (b, c), (b, d), (b, e), (b, f), (d, c), (e, c), (f, c), (e, d), (e, f), (f, b)\}$, pak:

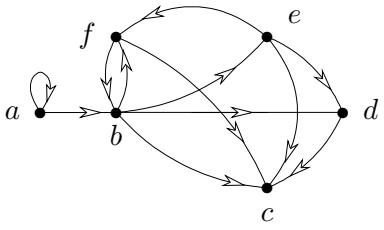


  Pozor, nejedná se o „grafy funkcí“ známé třeba z matematické analýzy.

V případě, že M je nekonečná nebo „velká“, může být reprezentace R jejím grafem nepraktická (záleží také na míře „pravidelnosti“ R).

`Značení`: Binární relaci $R \subseteq M \times M$ lze jednoznačně zapsat také pomocí *matice* relace – matice $\boldsymbol{A}$ typu $M \times M$ s hodnotami z $\{0, 1\}$, kde $a_{i,j} = 1$ právě když $(i, j) \in R$.

`Komentář:`



$$\rightarrow \quad \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} = \boldsymbol{A}$$

6.2 Vlastnosti binárních relací

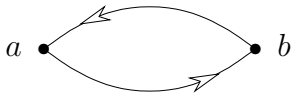
Pro začátek dalšího matematického výkladu relací si výčtem a doprovodnými schematickými obrázky uvedeme pět základních vlastností binárních relací na stejné množině, které nás typicky v matematické teorii zajímají. Je třeba si uvědomit, že se budeme zabývat velmi speciálním případem, neboť uvedené vlastnosti dávají dobrý smysl pouze pro uspořádané dvojice na téže množině, ale na druhou stranu se jedná o užitečný speciální případ.

Definice 6.2. Nechť $R \subseteq M \times M$. Binární relace R je

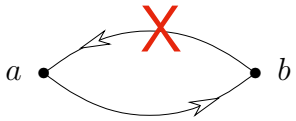
- *reflexivní*, právě když pro každé $a \in M$ platí $(a, a) \in R$;

- *ireflexivní*, právě když pro každé $a \in M$ platí $(a, a) \notin R$;

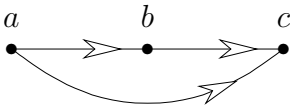
- *symetrická*, právě když pro každé $a, b \in M$ platí, že jestliže $(a, b) \in R$, pak také $(b, a) \in R$;



- *antisymetrická*, právě když pro každé $a, b \in M$ platí, že jestliže $(a, b), (b, a) \in R$, pak $a = b$;



- *tranzitivní*, právě když pro každé $a, b, c \in M$ platí, že jestliže $(a, b), (b, c) \in R$, pak také $(a, c) \in R$.



`Komentář:` Pozor, může být relace *symetrická i antisymetrická zároveň*?

Ano!

Vezměte si relaci $R = \{(x, x) \mid x \in M\}$, která obě jmenované vlastnosti splňuje. Neboli není pravda, že relace je antisymetrická, právě když není symetrická. Proto pokud jste třeba dotázáni, zda relace je symetrická, nestačí se v odpovědi odvolávat na fakt, že je antisymetrická (či naopak), neboť tyto vlastnosti se nevylučují.

Ukázkové binární relace

Příklad 6.3. *Jak poznat vlastnosti relací z matice:*

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Které z vlastností z Definice 6.2 mají binární relace reprezentované těmito maticemi?

Řešení: Ony vlastnosti si probereme (trochu neformálně) jednu po druhé.

∗ Podle definice je relace reflexivní, právě když každý prvek na hlavní diagonále je 1. Tudíž vlevo relace reflexivní je a vpravo není.

∗ Obdobně je relace ireflexivní, právě když je její hlavní diagonála vyplněna nulami. Tudíž ani jedna z našich relací ireflexivní není.

∗ Podle definice je relace symetrická, právě když je matice „stejná“ v zrcadlovém obraze podle hlavní diagonály. Tudíž vlevo symetrickou relaci nemáme a vpravo máme.

∗ Obdobně je relace antisymetrická, právě když po „přeložení“ matice podle hlavní diagonály nebudou nikde mimo tuto diagonálu dvě jedničky „na sobě“. Tudíž vlevo relace antisymetrická je a vpravo není.

* Zbývá posoudit tranzitivitu, což není úplně jednoduché… Pokusíme se velmi neformálně popsat názorný postup vycházející z definice tranzitivity.

  Relace je tranzitivní, právě když *vždy* vyjde následující test. Najdeme v libovolném řádku 1, od ní ve svislém směru (nahoru nebo dolů) dojdeme na hlavní diagonálu (její prvek nás nezajímá) a z toho místa nakonec ve vodorovném směru dojdeme na jinou 1. Poté na průsečíku počátečního řádku a koncového sloupce musí také být 1. □

Příklad 6.4. *Několik příkladů relací definovaných v přirozeném jazyce.*

Nechť M je množina všech studentů 1. ročníku FI. Uvažme postupně relace $R \subseteq M \times M$ definované takto

* $(x, y) \in R$ právě když x a y mají stejné rodné číslo;
* $(x, y) \in R$ právě když x má stejnou výšku jako y (dejme tomu na celé mm);
* $(x, y) \in R$ právě když výška x a y se neliší více jak o 2 mm;
* $(x, y) \in R$ právě když x má alespoň takovou výšku jako y;
* $(x, y) \in R$ právě když x má jinou výšku než y (dejme tomu na celé mm);
* $(x, y) \in R$ právě když x je zamilován(a) do y.

Zamyslete se, které z definovaných vlastností tyto jednotlivé relace mají… □

`Komentář:` Co dělat v situaci, že naše relace některou z poptávaných vlastností nemá, ale nám by se ta vlastnost hodila? V některých případech lze chybějící vlastnost relaci „doplnit“ pomocí takzvaného *uzávěru*, čemuž se budeme věnovat v dalším oddíle.

- Například reflexivní uzávěr jednoduše přidá do relace všechny dvojice (x, x) nad nosnou množinou relace.
- Nebo symetrický uzávěr zahrne do relace všechny obrácené dvojice k existujícím dvojicím, čili všechny chybějící (x, y) pokud $(y, x) \in R$.

6.3 Uzávěry relací

V předchozím jsme se krátce zmínili o postupu, jak danou relaci můžeme „obohatit“ o zvolenou vlastnost. (To se v praxi může hodit například tehdy, když naše data o relaci jsou neúplná a potřebná vlastnost je tak poškozena.) Takové obohacení se pochopitelně může týkat pouze některých vlastností – těch, které lze jednoznačně zajistit prostým přidáním dvojic do existující relace. Podchytit uzavíratelné vlastnosti má za úkol následující definice.

`Definice`: Nechť V je libovolná vlastnost binárních relací. Řekneme, že V je *uzavíratelná*, pokud splňuje následující podmínky:

* Pro každou množinu M a každou relaci $R \subseteq M \times M$ existuje alespoň jedna relace $S \subseteq M \times M$, která má vlastnost V a pro kterou platí $R \subseteq S$.
* Nechť I je množina a nechť $R_i \subseteq M \times M$ je relace mající vlastnost V pro každé $i \in I$. Pak relace $\bigcap_{i \in I} R_i$ má vlastnost V.

`Fakt`: Libovolná kombinace vlastností *reflexivita*, *symetrie*, *tranzitivita* je uzavíratelná vlastnost. To plyne přímo z toho, že úplná relace $M \times M$ na naší množině M má všechny

uvedené vlastnosti zároveň. Ireflexivita a antisymetrie naopak nejsou uzavíratelné vlastnosti.

Věta 6.5. *Nechť V je uzavíratelná vlastnost binárních relací. Buď M množina a R libovolná binární relace na M. Pak pro množinu všech relací $S \supseteq R$ na M majících vlastnost V existuje infimum R^V (vzhledem k množinové inkluzi), které samo má vlastnost V. Platí*

$$R^V = \bigcap_{S \supseteq R,\ S \text{ má } V \text{ na } M} S\,.$$

`Definice`: Tuto minimální relaci R^V s vlastností V nazýváme *V-uzávěr* relace R.

Srovnáme-li Větu 6.5 s Definicí 5.4, pohledem sběhlého matematika uvidíme, že V-uzávěr R^V relace je vlastně daný induktivní definicí, ve které bázickými prvky jsou dvojice původní relace R a induktivní pravidla odpovídají vlastnosti V. Tento konstruktivní pohled je podchycen případ od případu v následujícím tvrzení.

Tvrzení 6.6. Nechť R je binární relace na M. Pak platí následující poznatky.

* *Reflexivní uzávěr* R je přesně relace $R \cup \{(x,x) \mid x \in M\}$.
* *Symetrický uzávěr* R je přesně relace $\overset{\leftrightarrow}{R} = \{(x,y) \mid (x,y) \in R \text{ nebo } (y,x) \in R\}$.
* *Tranzitivní uzávěr* R je přesně relace $R^+ = \bigcup_{i=1}^{\infty} \mathcal{T}^i(R)$, kde $\mathcal{T}$ je zobrazení, které pro každou binární relaci S vrátí relaci

  $$\mathcal{T}(S) := S \cup \{(x,z) \mid \text{ existuje } y \text{ takové, že } (x,y),(y,z) \in S\}$$

  a $\mathcal{T}^i(S) = \underbrace{\mathcal{T}(\mathcal{T}\ldots(\mathcal{T}}_{i}(S))\ldots)$ je i-krát iterovaná aplikace zobrazení $\mathcal{T}$.
* Reflexivní a tranzitivní uzávěr R je přesně relace $R^* = Q^+$, kde Q je reflexivní uzávěr R.
* Reflexivní, symetrický a tranzitivní uzávěr R (tj. nejmenší ekvivalence obsahující R) je přesně relace $(\overset{\leftrightarrow}{Q})^+$, kde Q je reflexivní uzávěr R.

`Poznámka`: Na pořadí aplikování uzávěrů vlastností záleží! Zhruba řečeno, tranzitivní uzávěr obvykle aplikujeme coby poslední.

Závěrem se ještě pozastavíme nad případem tranzitivního uzávěru $R^+ = \bigcup_{i=1}^{\infty} \mathcal{T}^i(R)$, jehož definice se nezdá až tak „konstruktivní“ – jak bychom počítali nekonečné sjednocení? To skutečně nepotřebujeme, neboť platí následující:

Tvrzení 6.7. *Nechť R je relace na konečné množině. Pak existuje přirozené m takové, že tranzitivní uzávěr relace R lze zapsat $R^+ = \bigcup_{i=1}^{m} \mathcal{T}^i(R)$.*

`Důkaz`: Pro něj si uvědomme zásadní fakt – pokud $\mathcal{T}^{i+1}(R) = \mathcal{T}^i(R)$, tak už platí $\mathcal{T}^{i+k}(R) = \mathcal{T}^i(R)$ pro všechna přirozená k. Neboli je potřeba sjednotit jen tolik prvních členů výrazu $\bigcup_{i=1}^{\infty} \mathcal{T}^i(R)$, dokud se hodnota $\mathcal{T}^i(R)$ „zvětšuje“, což může nastat jen konečně krát nad konečnou množinou. □

6.4 Tranzitivní relace

Mezi základními vlastnostmi relací popsanými Definicí 6.2 je jedna – *tranzitivita*, jejíž uchopení a zvládnutí je výrazně těžší než u ostatních vlastností. Proč? Definice reflexivity, symetrie či antisymetrie nám ihned řeknou, které dvojice v relaci chybí nebo přebývají. Avšak vlastnost tranzitivity je fundamentálně *induktivní* – všimněte si předchozího tranzitivního uzávěru, kde po každém přidání dvojice do relace je nutno znovu podmínku tranzitivity kontrolovat, což „indukuje“ nové dvojice k přidávání.

K hlubšímu pochopení tranzitivity pomůže (snad) následující poznatek.

Tvrzení 6.8. *Mějme relaci R na konečné množině M a nechť R^+ je tranzitivním uzávěrem R. Pak pro každé $x, y \in M$ platí $(x, y) \in R^+$ právě tehdy, když existují $z_0, z_1, \ldots, z_k \in M$ takové, že $z_0 = x, z_k = y$ a $(z_{i-1}, z_i) \in R$ pro $i = 1, \ldots, k$.*

`Komentář:` Tvrzení 6.8 o tranzitivním uzávěru R^+ je potřeba (neformálně) číst takto: Do tranzitivního uzávěru patří všechny ty dvojice (x, y), že v původní relaci R se lze „dostat po šipkách“ z x do y. Pro velmi jednoduchou ilustraci:

V rozšířeném příkladě buď $R \subseteq \mathbb{N} \times \mathbb{N}$ definovaná takto: $R = \{(i, i+1) \mid i \in \mathbb{N}\}$. Pak tranzitivní uzávěr R^+ je běžné ostré lineární uspořádání $<$ přirozených čísel.

`Důkaz`: Nezapomeňte, že tvrzení vyslovené jako ekvivalence se musí dokazovat ve dvou směrech implikace. Nejprve předpokládejme, že existují $z_0, z_1, \ldots, z_k \in M$ takové, že $z_0 = x, z_k = y$ a $(z_{i-1}, z_i) \in R$ pro $i = 1, \ldots, k$. Indukcí podle $i \geq 1$ snadno dokážeme, že $(z_0, z_i) \in R$: zde platí $(z_0, z_{i-1}) \in R$ z indukčního předpokladu a $(z_{i-1}, z_i) \in R$ z předpokladu našeho tvrzení. Potom $(z_0, z_i) \in R$ plyne z definice tranzitivity.

Naopak předpokládejme $(x, y) \in R^+$. Tento směr důkazu není lehký a pomůžeme si tímto trikem: Nechť $D \subseteq M$ je množina těch prvků M, do kterých se lze „dostat po šipkách“ z x (induktivní definice množiny). Potom $x \in D$ a pokud také $y \in D$, jsme s důkazem hotovi. Pro spor tedy předpokládáme, že $y \notin D$.

Z toho, jak jsme množinu D zavedli, plyne, že pro každé $z \in D$ a $t \in M \setminus D$ platí $(z, t) \notin R$ (neformálně, z D nevede ven žádná šipka). Podle definice tranzitivního uzávěru pak ale nevede z D ven žádná šipka ani v uzávěru R^+ a jelikož $y \in M \setminus D$, dostáváme $(x, y) \notin R^+$, což je spor. Důkaz opačného směru je hotov. □

S tranzitivními relacemi (mezi objekty) se nejčastěji setkáváme, pokud tyto objekty porovnáváme vztahy „je stejný jako“ nebo „je větší/lepší než“. Takové porovnání přímo vede k následujícím pojmům.

Definice 6.9. Daná binární relace R je

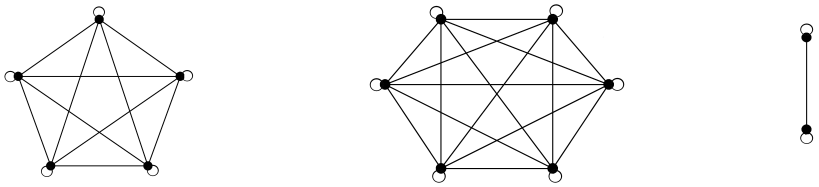
* relace *ekvivalence*, právě když je R reflexivní, symetrická a tranzitivní;
* *částečné uspořádání*, právě když je R reflexivní, antisymetrická a tranzitivní (často říkáme jen *uspořádání*).

Příklad 6.10. *Jaké vlastnosti mají následující relace?*

- Buď $R \subseteq \mathbb{N} \times \mathbb{N}$ definovaná takto $(x, y) \in R$ právě když x dělí y. (*Částečné uspořádání*, ale ne každá dvě čísla jsou porovnatelná.)
- Buď $R \subseteq \mathbb{N} \times \mathbb{N}$ definovaná takto $(x, y) \in R$ právě když x a y mají stejný zbytek po dělení číslem 5. (*Ekvivalence.*)
- Nechť $F = \{f \mid f : \mathbb{N} \to \mathbb{N}\}$ je množina funkcí. Buď $R \subseteq F \times F$ definovaná takto $(f, g) \in R$ právě když $f(x) < g(x)$ pro všechna x. (*Ireflexivní*, *antisymetrická* a *tranzitivní*, ale ne reflexivní – striktně řečeno není uspořádáním.) □

Příklad 6.11. *Jak vypadá graf relace ekvivalence?*

Poměrně příznačně, jak nám ukazuje následující obrázek (všimněte si absence šipek, která je dána symetrií relace).



Neformálně řečeno; ekvivalence je relace $R \subseteq M \times M$, taková, že $(x, y) \in R$ právě když x a y jsou v nějakém smyslu „*stejné*“ (*leží na stejné hromádce*). Tyto *hromádky* pak nazýváme komponentami či *třídami ekvivalence* dané relace. Více se o vztahu ekvivalence k jejím třídám dozvíme v příští lekci. □

Zvídavým čtenářům ještě přidáme jeden obtížnější příklad, který se už trochu vztahuje k příští látce...

Příklad 6.12. *Studenti předmětu IB000 sedí v lavicích v obdélníkové posluchárně (však to dobře znáte). Ne všechny sedačky jsou nutně obsazené. Definujeme relaci R tak, že dvojice studentů (a, b) náleží do R, právě když b sedí ve stejné řadě vlevo od a nebo b sedí ve stejném sloupci před a.*

a) Dokažte, že tranzitivní uzávěr R je antisymetrická relace.

b) Kolik nejméně a nejvíce tříd ekvivalence může definovat reflexivní, symetrický a tranzitivní uzávěr relace R?

Část (a) dokážeme sporem; nechť tedy platí $(a, b), (b, a) \in R^+$ kde $a \neq b$. Proč by toto nemělo nastat? Pro důkaz si pomůžeme jednoduchým trikem: pro studenta x definujeme hodnotu $s(x) := i + j$, kde i je řada a j sloupec, ve kterých x sedí, číslováno zleva a zepředu. Pak podle popisu relace R zřejmě platí $s(x) > s(y)$ pokud $(x, y) \in R$. Podle definice tranzitivního uzávěru pak $s(a) > s(b)$ jakmile $(a, b) \in R^+$ a $a \neq b$, avšak také $s(a) < s(b)$ jelikož $(b, a) \in R^+$. To je zřejmě spor.

Nyní přejdeme k části (b). Označme S reflexivní, symetrický a tranzitivní uzávěr relace R. Nejméně tříd, jednu, získáme při plném obsazení posluchárny, ať už má jakékoliv rozměry. Vezměme kterékoliv dva různé studenty a, b. Podle definice reflexivního a symetrického uzávěru existuje student c sedící ve stejné řadě jako a a ve stejném sloupci jako b (může to být i jeden z a, b), přičemž platí $(a, c), (c, b) \in S$. Podle definice tranzitivního uzávěru pak $(a, b) \in S$, a proto a, b leží ve stejné jedné třídě podle Věty 7.7.

Naopak nejvíce tříd získáme, pokud žádní dva studenti nesedí ve stejné řadě nebo stejném sloupci (třeba sedí jen na jedné úhlopříčce). Počet tříd pak je přesně ℓ, kde ℓ je

rovno minimu z počtu řad a sloupců posluchárny. Předpokládejme pro spor, že by mohlo při nějakém obsazení posluchárny existovat více než ℓ tříd ekvivalence uzávěru S. Zvolme z každé třídy jednoho studenta jako reprezentanta. Podle Dirichletova principu někteří dva a, b z vybraných studentů sedí ve stejné řadě či sloupci (neboť studentů je více, než minimum z počtu řad a sloupců). Podle definice naší relace R a symetrického uzávěru pak platí $(a, b) \in S$. Avšak a, b pochází z různých tříd ekvivalence a to je ve sporu s Větou 7.7. □

Navazující studium

Mějte na paměti, že na pojmech množin, relací a funkcí jsou vystavěny prakticky všechny skutečné datové struktury používané v dnešní informatice. Explicitně toto můžete vidět na relačních databázích, ale i na mnoha jiných implicitních výskytech. Mnohem více si nejprve o relacích a jejich skládání a poté speciálně o funkcích vysvětlíme v dalších dvou lekcích. Dalším důležitým matematickým datovým typem odvozeným z binárních relací jsou pak grafy probírané od Lekce 9, na nichž v různé míře staví většina základních algoritmů, které se budete učit.

7 Ekvivalence, Uspořádané množiny

Úvod

V této lekci pokračujeme rozebíráním relací na množinách jako nástrojů vyjadřujících vztahy mezi objekty. Zaměříme se přitom pouze na relace binární, které nějakým způsobem srovnávají objekty podle jejich vlastností (obvykle ve smyslu „stejný jako“ nebo „lepší/větší než“). Takto vágně opsané binární relace mívají jasné společné znaky, které se pak objevují ve zde uvedených formálních definicích relace ekvivalence a uspořádání.

Co se týče ekvivalencí, lze si je neformálně představit jako rozdělení prvků nosné množiny na „hromádky“, přičemž prvky každé jednotlivé hromádky jsou si navzájem v jistém smyslu stejné. Naopak uspořádání nám již podle svého názvu udává srovnání prvků nosné množiny, neboli které prvky si „stojí lépe“ než jiné.

Cíle

Definujeme relace ekvivalence a uspořádání a mnohé další pojmy k nim se vztahující, například rozklady a Hasseovské diagramy. Studující by měli porozumět matematické problematice rozkladů množin a uspořádaných množin a naučit se je správně používat.

7.1 Relace ekvivalence a rozklady

Nyní se hlouběji podívejme na první specifický typ *binární* relace zmíněný v Lekci 6: Podle Definice 6.2 je relace $R \subseteq M \times M$ *ekvivalence* právě když R je reflexivní, symetrická a tranzitivní. Tyto *tři vlastnosti* tedy musí být splněny a ověřeny k důkazu toho, že daná relace R je ekvivalencí.

`Značení`. V případě relace ekvivalence se poměrně často lze setkat s označením jako $\sim$ či $\approx$ místo R. Místo $(x, y) \in R$ se pak píše $x \approx y$.

Příklad 7.1. *Nechť M je množina všech studentů 1. ročníku FI. Uvažme postupně relace $R \subseteq M \times M$ definované následovně a zkoumejme, zda se jedná o ekvivalence:*

- $(x, y) \in R$ právě když x má stejnou výšku jako y;
- $(x, y) \in R$ právě když x má stejnou barvu vlasů jako y;
- $(x, y) \in R$ právě když x, y mají stejnou výšku a stejnou barvu vlasů;
- $(x, y) \in R$ právě když x, y mají stejnou výšku *nebo* stejnou barvu vlasů.

U kterého body se nejedná o relaci ekvivalence a proč? Je to poslední případ, kdy není splněna tranzitivita. □

Uvedený příklad ukazuje na následující univerzální poznatek, který může platit (a taky platí) pouze pro průnik a nikoliv pro sjednocení.

Tvrzení 7.2. *Nechť R, S jsou dvě relace ekvivalence na stejné množině M. Pak jejich průnik $R \cap S$ je opět relací ekvivalence.*

`Důkaz` (náznak): Jelikož reflexivita a symetrie jsou zřejmé, stačí snadno ověřit platnost tranzitivity na $R \cap S$. Nechť $(a, b), (b, c) \in R \cap S$, pak podle tranzitivity každé jedné z R, S plyne $(a, c) \in R$ a zároveň $(a, c) \in S$. □

Příklad 7.3. *Nechť $R \subseteq \mathbb{N} \times \mathbb{N}$ je binární relace definovaná takto: $(x, y) \in R$ právě když $|x - y|$ je dělitelné třemi. V jakém smyslu jsou zde x a y „stejné“?*

Platí $(x, y) \in R$ právě když x, y dávají stejný zbytek po dělení třemi. Je to opět relace ekvivalence. □

Příklad 7.4. *Buď R binární relace mezi všemi studenty na přednášce FI:IB000 definovaná takto: $(x, y) \in R$ právě když x i y sedí v první lavici.*

Už na první pohled jde o relaci symetrickou a tranzitivní. Proč se v tomto případě *nejedná* o relaci ekvivalence? Protože není reflexivní pro studenty sedící v dalších lavicích. (Takže si dávejte dobrý pozor na správné pochopení definic.) □

Rozklady a jejich vztah k ekvivalencím

Náplní následující části výkladu je ukázat jiný přirozený pohled na ekvivalence. Tento nový pohled nám matematicky formalizuje již nastíněnou představu ekvivalence jako rozdělení prvků nosné množiny M na „*hromádky*“, přičemž prvky každé jednotlivé hromádky jsou si navzájem v jistém smyslu „stejné“.

Definice 7.5. ***Rozklad množiny.*** Nechť M je množina.
Rozklad (na) M je množina podmnožin $\mathcal{N} \subseteq 2^M$ splňující následující tři podmínky:

- $\emptyset \notin \mathcal{N}$ (tj. každý prvek $\mathcal{N}$ je neprázdná podmnožina M);
- pokud $A, B \in \mathcal{N}$, pak buď $A = B$ nebo $A \cap B = \emptyset$;
- $\bigcup_{A \in \mathcal{N}} A = M$.

Prvkům $\mathcal{N}$ se také říká *třídy rozkladu*.

`Komentář:`

* Buď $M = \{a, b, c, d\}$. Pak $\mathcal{N} = \{\{a\}, \{b, c\}, \{d\}\}$ je rozklad na M.
* Nechť $A_0 = \{k \in \mathbb{N} \mid k \bmod 3 = 0\}$, $A_1 = \{k \in \mathbb{N} \mid k \bmod 3 = 1\}$, $A_2 = \{k \in \mathbb{N} \mid k \bmod 3 = 2\}$. Pak $\mathcal{N} = \{A_0, A_1, A_2\}$ je rozklad všech přirozených čísel $\mathbb{N}$ podle zbytkových tříd.

Každý rozklad $\mathcal{N}$ na M jednoznačně určuje jistou ekvivalenci $R_\mathcal{N}$ na M:

Věta 7.6. *Nechť M je množina a $\mathcal{N}$ rozklad na M. Nechť $R_\mathcal{N} \subseteq M \times M$ je relace na M definovaná takto*

$$(x, y) \in R_\mathcal{N} \text{ právě když existuje } A \in \mathcal{N} \text{ taková, že } x, y \in A.$$

Pak $R_\mathcal{N}$ je ekvivalence na M.

`Důkaz`: Dokážeme, že $R_\mathcal{N}$ je *reflexivní, symetrická a tranzitivní* (*Definice 6.2*).

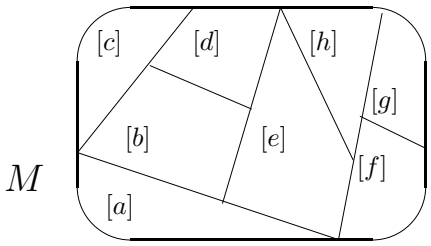
- Reflexivita: Buď $x \in M$ libovolné. Jelikož $\mathcal{N}$ je rozklad na M, musí existovat $A \in \mathcal{N}$ takové, že $x \in A$ (jinak spor se třetí podmínkou z *Definice 7.5*). Proto $(x, x) \in R_\mathcal{N}$, tedy $R_\mathcal{N}$ je reflexivní.
- Symetrie: Nechť $(x, y) \in R_\mathcal{N}$. Podle definice $R_\mathcal{N}$ pak existuje $A \in \mathcal{N}$ taková, že $x, y \in A$. To ale znamená, že také $(y, x) \in R_\mathcal{N}$ podle definice $R_\mathcal{N}$, tedy $R_\mathcal{N}$ je symetrická.
- Tranzitivita: Nechť $(x, y), (y, z) \in R_\mathcal{N}$. Podle definice $R_\mathcal{N}$ existují $A, B \in \mathcal{N}$ takové, že $x, y \in A$ a $y, z \in B$. Jelikož $A \cap B \neq \emptyset$, podle druhé podmínky z *Definice 7.5* platí $A = B$. Tedy $x, z \in A = B$, proto $(x, z) \in R_\mathcal{N}$ podle definice $R_\mathcal{N}$. □

Každá ekvivalence R na M naopak jednoznačně určuje jistý rozklad M/R na M:

Věta 7.7. *Nechť M je množina a R ekvivalence na M. Pro každé $x \in M$ definujeme množinu*

$$[x] = \{y \in M \mid (x, y) \in R\}\,.$$

Pak $\{[x] \mid x \in M\}$ je rozklad na M, který značíme M/R a čteme „rozklad M podle R“.



Důkaz: Dokážeme, že M/R splňuje podmínky *Definice 7.5.*

- Pro každé $[x] \in M/R$ platí $[x] \neq \emptyset$, neboť $x \in [x]$.
- Nechť $[x], [y] \in M/R$. Ukážeme, že pokud $[x] \cap [y] \neq \emptyset$, pak $[x] = [y]$.

  Jestliže $[x] \cap [y] \neq \emptyset$, existuje $z \in M$ takové, že $z \in [x]$ a $z \in [y]$. Podle definice $[x]$ a $[y]$ to znamená, že $(x, z), (y, z) \in R$. Jelikož R je symetrická a $(y, z) \in R$, platí $(z, y) \in R$. Jelikož $(x, z), (z, y) \in R$ a R je tranzitivní, platí $(x, y) \in R$. Proto také $(y, x) \in R$ (opět ze symetrie R). Nyní dokážeme, že $[y] = [x]$:

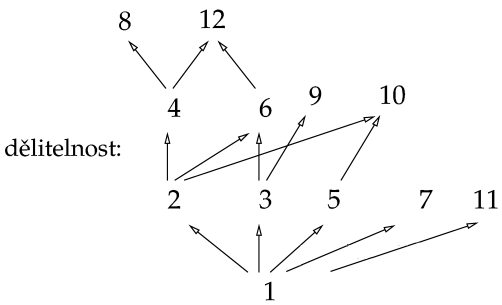
  * „$[x] \subseteq [y]$“: Nechť $v \in [x]$. Pak $(x, v) \in R$ podle definice $[x]$. Dále $(y, x) \in R$ (viz výše), tedy $(y, v) \in R$ neboť R je tranzitivní. To podle definice $[y]$ znamená, že $v \in [y]$.
  * „$[y] \subseteq [x]$“: Nechť $v \in [y]$. Pak $(y, v) \in R$ podle definice $[y]$. Dále $(x, y) \in R$ (viz výše), tedy $(x, v) \in R$ neboť R je tranzitivní. To podle definice $[x]$ znamená, že $v \in [x]$.
- Platí $\bigcup_{[x] \in M/R} [x] = M$, neboť $x \in [x]$ pro každé $x \in M$. □

7.2 Uspořádané množiny

Přirozený vágní pojem *porovnání/uspořádání* objektů má také svou přesnou matematickou definici: Podle Definice 6.2 je relace $R \subseteq M \times M$ *(částečné) uspořádání* právě když R je reflexivní, antisymetrická a tranzitivní. Tyto *tři vlastnosti* tedy musí být splněny a ověřeny k důkazu toho, že daná relace R je uspořádáním.

Komentář: Neformálně řečeno: uspořádání je taková relace $R \subseteq M \times M$, kde $(x, y) \in R$ právě když x je v nějakém smyslu „*menší nebo rovno*“ než y. Mohou ovšem existovat taková $x, y \in M$, kde neplatí $(x, y) \in R$ ani $(y, x) \in R$. (Pak říkáme, že x a y jsou *nesrovnatelné*.)

Jak *názorně* zobrazit (částečné) uspořádání? Příklad zjednodušeného zakreslení (jsou vynechány šipky vyplývající z reflexivity a tranzitivity, viz Oddíl 7.3) je zde:



Všimněte si, že je někdy zvykem „větší“ prvky kreslit nad ty „menší“.

`Značení`. Pro větší přehlednost se relace uspořádání často značí symboly jako $\sqsubseteq$ či $\preceq$ místo prostého R. I my tak často budeme psát třeba $x \preceq y$ místo $(x, y) \in R$.

`Poznámka`: Zajisté jste se již neformálně setkali s „neostrým“ uspořádáním čísel $\leq$ a „ostrým“ uspořádáním $<$. Všimněte si dobře, že námi definované uspořádání je vždy neostré. Pokud byste naopak chtěli definovat *ostré* uspořádání, mělo by vlastnosti *ireflexivní*, antisymetrické a tranzitivní. (Příliš se však tato varianta nepoužívá.) Nadále budeme pracovat pouze s neostrým uspořádáním, ale smyčky vyplývající z reflexivity a případně i šipky z tranzitivity budeme pro větší přehlednost v obrázcích vynechávat.

Uspořádaná množina

Matematicky důležitějším pojmem než samotná relace uspořádání je uspořádaná množina, neboli soubor prvků s pevně zvoleným uspořádáním na nich.

Definice 7.8. ***Uspořádaná množina*** je dvojice $(M, \preceq)$,
kde M je množina a $\preceq$ je (částečné) uspořádání na M.

`Definice`: Uspořádání $\preceq$ na M je *lineární* (nebo také *úplné*), pokud každé dva prvky M jsou v $\preceq$ srovnatelné.

Příklady

Příklad 7.9. *Nechť M je množina všech studentů 1. ročníku FI. Uvažme postupně relace $R \subseteq M \times M$ definované následovně (jedná se úplně vždy o uspořádání?):*

* $(x, y) \in R$ právě když x má alespoň takovou výšku jako y;
* $(x, y) \in R$ právě když y má alespoň takovou výšku jako x.

Všimněte si dobře možného problému – pokud by dva studenti měli přesně stejnou výšku, nebyla by splněna podmínka antisymetrie. Prakticky však můžeme uvažovat, že toto nenastane (matematicky bychom řekli, že pravděpodobnost výskytu dvou studentů přesně stejné výšky bez omezení přesnosti měření je nula) a bude se jednat o uspořádání. □

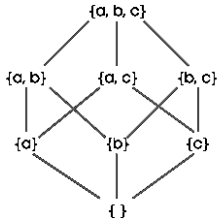
Příklad 7.10. *Nechť M je množina všech českých studentů 1. ročníku FI. Uvažme relaci $R \subseteq M \times M$ definovanou tak, že $(x, y) \in R$ právě když x a y mají stejné rodné číslo. Je možné, aby R byla uspořádáním na M?*

Ano, za zákonného předpokladu jedinečnosti rodného čísla. Dobře si promyslete proč (které dvojice jsou vůbec porovnatelné?). □

Příklad 7.11. *Další ukázky uspořádaných množin následují zde:*

* $(\mathbb{N}, \leq)$ je lineárně uspořádaná množina, kde $\leq$ má „obvyklý“ význam.
* $(\mathbb{N}, |\,)$, kde $a|b$ je relace dělitelnosti „a dělí b“ na přirozených číslech, je uspořádaná množina. Toto uspořádání není lineární.

∗ Buď M množina. Pak $(2^M, \subseteq)$ je uspořádaná množina (říkáme *inkluzí*). Které dvojice jsou v uspořádání inkluzí *nesrovnatelné*?



□

`Komentář:` Zamyslete se také, jak vypadá relace dělitelnosti na celých (tj. i záporných) číslech. Proč se už nejedná o uspořádání? Blíže viz konec Oddílu 7.3.

Multikriteriální uspořádání

Přirozenou ukázkou praktického problému, ve kterém potkáme problematiku netriviálně uspořádaných objektů, je výběr toho *nejlepšího* podle více nezávislých kritérií. Co třeba znamená požadavek na „*nejlepší počítač*“? I pokud se soustředíme jen na užitné / výkonové charakteristiky počítače, můžeme porovnávat nejen podle rychlosti procesoru, ale také třeba podle výkonu grafiky, velikosti paměti a disku, nebo i počtu nakouslých jablek na víku. A jen málokdy se setkáme s dostupným počítačem, který by jasně dominoval ve všech kritériích najednou.

Takže jak lze objekty podle více kritérií seřadit?

- Pokud jsou všechna *kritéria numerická*, zkoumané objekty lze řadit podle jejich váženého průměru (či součtu). Avšak numerické hodnoty jsou ne vždy dostupné a především „průměrování“ kritérií často dává velmi nepřirozené výsledky.
- Objekty můžeme uspořádat podle průniku seřazení jednotlivých kritérií (říká se *po složkách*). Výhodou je, že pokud objekt (počítač) P je lepší než Q, tak P v žádném kritériu nemůže zaostávat za Q. Avšak výsledkem bude typicky existence několika „nejlepších“ objektů, které nebudeme umět vzájemně porovnat. (Podívejte se na pojem maximálního prvku v Definici 7.14.)
- Nakonec si můžeme pomoci *slovníkovým uspořádáním*: Kritéria seřadíme od nejdůležitějšího a první odlišná hodnota kritéria rozhoduje. Například nás v prvé řadě zajímá rychlost procesoru, poté velikost paměti a až nakonec výkon grafiky či velikost disku... (Máme přece počítač na práci a ne na hraní her či sledování filmů.)

Předchozí vágní popisy uspořádání nyní formalizujeme matematicky. Formálně mějme několik kritérií označených indexovou množinou I (například uvažme čtyři kritéria vypsaná výše s $I = \{1, 2, 3, 4\}$). Pro $i \in I$ nechť hodnoty nabývané i-tým kritériem tvoří uspořádanou množinu $(A_i, \leq_i)$. Ve výsledku nás pak zajímá „složené“ uspořádání na kartézském součinu $\bigtimes_{i\in I} A_i$, které definujeme následovně.

Definice 7.12. ***Uspořádání podle více kritérií.***
Pro $I \subseteq \mathbb{N}$ mějme systém uspořádaných množin $(A_i, \leq_i)$ kde $i \in I$. Na množině $M := \bigtimes_{i\in I} A_i$ definujeme binární relace $\sqsubseteq$ a $\preceq$ takto;

pro $\ell, m \in M$ kde $\ell = (\ell_i : i \in I)$ a $m = (m_i : i \in I)$ platí

∗ (*po složkách*)

$$\ell \sqsubseteq m \quad \text{právě když} \quad \ell_i \leq_i m_i \text{ pro všechna } i \in I,$$

∗ (*lexikograficky*)

$$\ell \preceq m \quad \text{právě když} \quad \begin{array}{l} \ell = m\,, \quad \text{nebo} \\ \text{existuje } j \in I \text{ tak, že } \ \ell_j \lneq_j m_j \text{ a} \\ \text{pro všechna } i \in I,\ i < j \text{ platí } \ \ell_i = m_i. \end{array}$$

`Komentář:` Lexikografické uspořádání z Definice 7.12 je staré známé uspořádání slov ve slovníku podle abecedy – kde I indexuje jednotlivá písmena slov zleva. Všimněte si, že pro definici lexikografického uspořádání je potřebné, aby také indexová množina I byla lineárně uspořádaná. Porovnejte si jej s detaily naší definice pro lepší porozumnění věci.

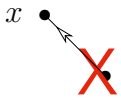
Tvrzení 7.13. *Nechť $\sqsubseteq$ a $\preceq$ jsou binární relace na množině M z Definice 7.12. Pak se v obou případech jedná o relace uspořádání. Navíc, pokud všechny $(A_i, \leq_i)$ jsou lineárně uspořádané množiny, tak lexikografické uspořádání $(M, \preceq)$ nad nimi je taktéž lineární.*

7.3 Pojmy uspořádaných množin

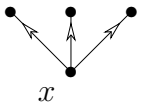
K tématu uspořádaných množin se vztahuje množství formálních pojmů, které potkáte v různých oblastech matematiky i informatiky.

Definice 7.14. Nechť $(M, \preceq)$ je uspořádaná množina.

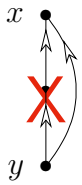
- $x \in M$ je *minimální* právě když pro každé $y \in M$ platí, že z $y \preceq x$ vyplývá $x \preceq y$, čili podle antisymetrie pro každé $y \preceq x$ platí $y = x$.



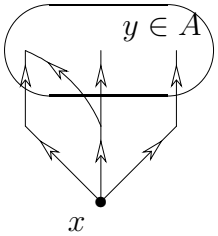
- $x \in M$ je *maximální* právě když pro každé $y \in M$ platí, že z $x \preceq y$ vyplývá $y \preceq x$, čili podle antisymetrie pro každé $y \succeq x$ platí $y = x$. (Tj. x je maximální, právě když neexistuje žádný prvek ostře větší než x, a x je minimální, právě když neexistuje žádný prvek ostře menší než x.)
- $x \in M$ je *nejmenší* právě když pro každé $y \in M$ platí, že $x \preceq y$.



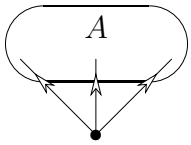
- $x \in M$ je *největší* právě když pro každé $y \in M$ platí, že $y \preceq x$.
- $x \in M$ *pokrývá* $y \in M$ právě když $x \neq y$, $y \preceq x$ a neexistuje žádné $z \in M$ takové, že $x \neq z \neq y$ a $y \preceq z \preceq x$.



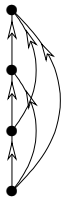
- $x \in M$ je *dolní závora* (mez) množiny $A \subseteq M$ právě když $x \preceq y$ pro každé $y \in A$.



- $x \in M$ je *horní závora* (mez) množiny $A \subseteq M$ právě když $y \preceq x$ pro každé $y \in A$.
- $x \in M$ je *infimum* množiny $A \subseteq M$ právě když x je největší dolní závora (mez) množiny A.



- $x \in M$ je *supremum* množiny $A \subseteq M$, právě když x je nejmenší horní závora (mez) množiny A.
- $A \subseteq M$ je *řetězec* v uspořádání $\preceq$ právě když $(A, \preceq)$ je *lineárně* uspořádaná množina.



`Komentář:` Tuto dlouhou definici se sluší poněkud neformálně okomentovat. Za prvé, s pojmy nejmenšího a největšího prvku jste se už intuitivně setkali mnohokrát, ale (matematicky slabší) pojmy minimálního a maximálního působí někdy problémy. Zapamatujte si proto dobře, že minimálních prvků může mít množina několik, jsou to prostě všechny ty "vespod", ale nejmenší prvek existuje nejvýše jeden a je to pouze ten *unikátní* minimální prvek množiny. Stejně pro maximální...

Další poznámka se vztahuje k infimu a supremu množiny. Jak jsme napsali (a asi totéž znáte z matematické analýzy), množina nemusí mít nejmenší ani největší prvek, ale v mnoha případech je lze "nahradit" po řadě infimem a supremem, které hrají v jistých ohledech podobnou roli. Avšak ani supremum a infimum nemusí vždy existovat.

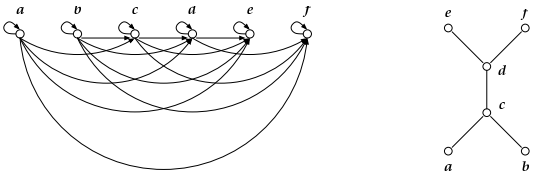
Nakonec si *dejte pozor*, že některé uvedené definice mají „netriviální chování" na *nekonečných* množinách. Proto je budeme obvykle uvažovat jen nad konečnými množinami.

Příklad 7.15. *Proč má každá uspořádaná množina nejvýše jeden největší prvek?*

Tvrzení dokážeme sporem: Nechť m i n jsou největší prvky uspořádané množiny $(M, \preceq)$. Pak podle Definice 7.14 platí $n \preceq m$ i $m \preceq n$ zároveň. Ovšem jelikož uspořádání musí být antisymetrické, pak platí $m = n$ a největší prvek je jen jeden. □

Hasseovské diagramy

Motivací zavedení tzv. Hasseovských diagramů uspořádaných množin jsou přehlednější „obrázky" než u grafů relací. Například si srovnejte následující ukázky:



Zjednodušení a konvence přinesené následující definicí jsme ostatně měli možnost vidět už na některých z předchozích ilustračních obrázků uspořádání.

`Definice:` *Hasseovský diagram* konečné uspořádané množiny $(M, \preceq)$ je její jednoznačné grafické znázornění definované takto:

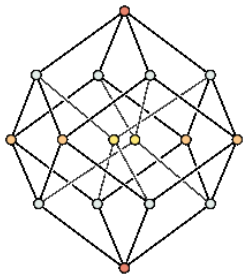
* Prvky nosné množiny M jsou zakresleny body v rovině („puntíky“).
* Mezi každou dvojicí prvků $x \neq y$ vede hrana (tj. „čára“, ale bez šipek), právě když prvek x pokrývá prvek y. V takovém případě x musí být zakreslen *výše než* y. Pojem pokrývání prvku najdete v Definici 7.14.

`Komentář:` Všimněte si, jak přímo z definic vyplývá, že původní relace uspořádání $\preceq$ je reflexivním a tranzitivním uzávěrem relace pokrývání z Hasseovského diagramu. To znamená, že samotné hrany tohoto diagramu spolu s podmínkou, které prvky hran jsou kresleny výše, už poskytují plnou informaci o naší uspořádané množině $(M, \preceq)$.

Metoda 7.16. ***Zakreslení Hasseovského diagramu***
Pro danou konečnou uspořádanou množinu $(M, \preceq)$ jej jednoduše získáme takto:

* Do první horizontální vrstvy zakreslíme body odpovídající minimálním prvkům $(M, \preceq)$. (Tj. které *nepokrývají* nic.)
* Máme-li již zakreslenu vrstvu i, pak do vrstvy $i+1$ (která je „nad“ vrstvou i) zakreslíme všechny nezakreslené prvky, které *pokrývají pouze* prvky vrstev $\leq i$. Pokud prvek x vrstvy $i+1$ pokrývá prvek y vrstvy $\leq i$, spojíme x a y neorientovanou hranou (tj. „čárou bez šipky“).

Příklad 7.17. *Relaci inkluze na čtyřprvkové množině* $\{a, b, c, d\}$ *zakreslíme Hasseovským diagramem takto:*



□

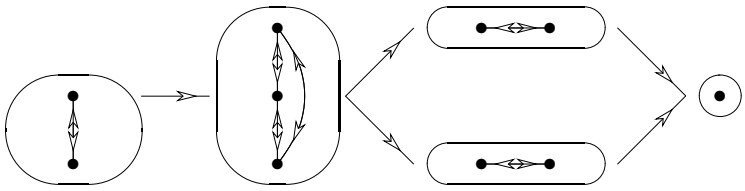
`Komentář:` Znovu jinými méně formálními slovy opakujeme, že v Hasseovském diagramu „*vynecháváme*“ ty hrany relace $\preceq$, které vyplývají z *reflexivity či tranzitivity*. To celý obrázek výrazně zpřehlední, a přitom nedochází ke ztrátě informace. Lze vynechat i šipky na hranách, neboť dle definice všechny míří „vzhůru“. Také pojem „*vrstvy*“ v Metodě 7.16 je jen velmi neformální, důležité je, že větší (pokrývající) prvky jsou *nad menšími* (pokrývanými).

7.4 Relace předuspořádání

Mimo uspořádání chápaných *striktně antisymetricky* se v praxi často setkáme se „polouspořádáními“, ve kterých některé očividně různé prvky stojí na stejné úrovni naší preference. Co třeba dostaneme, když studenty na zkoušce „*uspořádáme*“ podle výsledné známky A–F? Bude to vždy uspořádání ve smyslu Definice 6.2? Samozřejmě ne, obvykle více než jeden student bude mít třeba známku E, což zjevně porušuje antisymetrii. Přesto cítíme, že studenty podle známky tak nějak uspořádané máme. Z toho důvodu je užitečné zavést i následující obecnější pojem:

`Definice`: Relace $R \subseteq M \times M$ je *předuspořádání* (také *kvaziuspořádání*, nebo *polouspořádání*) právě když R je *reflexivní* a *tranzitivní*.

`Komentář:` Rozdíl mezi uspořádáním a předuspořádáním je (jen neformálně řečeno!) v tom, že u předuspořádání srovnáváme prvky podle kritéria, které není pro prvek jedinečné. V předuspořádání takto mohou vznikat „*balíky*“ (třídy) se stejnou hodnotou kritéria, což schematicky ilustrujeme na následujícím obrázku.



Důležitým poznatkem je, že předuspořádání se „až tak moc“ neliší od dřívějšího uspořádání – přirozeně totiž odvodíme následující:

Věta 7.18. *Je-li $\sqsubseteq$ předuspořádání na M, můžeme definovat relaci $\sim$ na M předpisem*

$$x \sim y \quad \text{právě když} \quad x \sqsubseteq y \text{ a } y \sqsubseteq x.$$

Pak $\sim$ je ekvivalence na M, která se nazývá jádro předuspořádání $\sqsubseteq$.

Na rozkladu $M/\sim$ pak lze zavést relaci $\preceq$ definovanou takto

$$[x] \preceq [y] \quad \text{právě když } x \sqsubseteq y.$$

Pak $(M/\sim, \preceq)$ je uspořádaná množina indukovaná $\sqsubseteq$.

`Komentář:` Pro ukázku si vezměme relaci dělitelnosti na $\mathbb{Z}$. Pak třeba $-2 \sim 2$. Jádrem zde jsou dvojice čísel stejné absolutní hodnoty.

`Důkaz` (náznak): Tranzitivita a reflexivita relace $\sim$ vyplývá z tranzitivity a reflexivity relace $\sqsubseteq$. Symetrie $\sim$ pak je přímým důsledkem její definice. Tudíž $\sim$ skutečně je relací ekvivalence a $M/\sim$ je platný rozklad nosné množiny.

Tranzitivita a reflexivita relace $\preceq$ se opět dědí z relace $\sqsubseteq$. Její antisymetrie vyplývá následující úvahou: Pokud $[x] \preceq [y]$ a $[y] \preceq [x]$, pak podle naší definice $x \sqsubseteq y$ a $y \sqsubseteq x$, neboli $x \sim y$ a $[x] = [y]$ podle definice tříd rozkladu. Pozor, nejdůležitější částí této větve důkazu je však ještě zdůvodnění, že naše podaná definice vztahu $[x] \preceq [y]$ je korektní, což znamená, že její platnost nezávisí na konkrétní volbě reprezentantů x z $[x]$ a y z $[y]$.

Poslední zmíněné tvrzení dokážeme sporem: Nechť $x, x' \in [x]$ a $y, y' \in [y]$ jsou (možná různí) reprezentanti dvou zkoumaných tříd rozkladu $M/\sim$, pro které však platí $x \sqsubseteq y$ a $x' \sqsupseteq y'$. Podle definice třídy rozkladu je $x \sim y$ a $x' \sim y'$ a z tranzitivity $x \sqsubseteq y \sqsubseteq y' \sqsubseteq x' \sqsubseteq x$ a tudíž $[x] = [y]$ a na volbě reprezentanta skutečně nezáleží. □

Rozšiřující studium

S relacemi ekvivalence i jimi implicitně definovanými rozklady množin se lze setkat všude tam, kde nějaké objekty "rozdělujeme do přihrádek" podle nějakých sdílených znaků nebo kritérií. Principy takových rozkladů vám zajisté byly intuitivně známy ještě dříve, než jste vůbec slyšeli o matematickém pojmu ekvivalence, v této lekci jsme si je jen zavedli

na pevných formálních základech. Toto formální pojetí relací ekvivalence a rozkladů budete porůznu potkávat i v dalších informatických předmětech, například u teorie automatů apod.

Druhá část lekce pojednává o látce, kterou opět určitě na intuitivní úrovni znáte (však schopnost si věci „uspořádat“ je jednou ze základních lidských dovedností). Přesto správné matematické uchopení podstaty uspořádaných množin vyžaduje se prokousat několika nesnadnými formálními definicemi, které byly v této lekci vysvětleny a okomentovány. Všimněte si, že hlavní těžkosti konceptu uspořádání se vztahují k částečným, tedy nelineárním uspořádáním, kdy běžnému lidskému uvažování není zcela intuitivně jasná práce s nesrovnatelnými dvojicemi. S trochou příslušného matematického formalismu se správné nakládání s konečnými částečnými uspořádáními stane snadnou rutinou.

8 Skládání relací a funkcí

Úvod

Na závěr učiva o relacích se vracíme k základním definicím abstraktní relace a funkce z Oddílu 3.5. Náplní výkladu bude to, jak relace obracet a jak je mezi sebou skládat, což si mimo obecných relací vysvětlíme i na speciálním případu funkcí (konrétně na tzv. permutacích). Mimochodem, kde jste se již přirozeně se skládáním funkcí setkali – jak například spočítáte na kalkulačce výsledek složitějšího vzorce? Obecné skládání relací je základním nástrojem práce v relačních databázích. A ke skládání funkcí se pak váže několik dalších základních matematických pojmů, jako jsou injektivní, surjektivní a bijektivní funkce.

Cíle

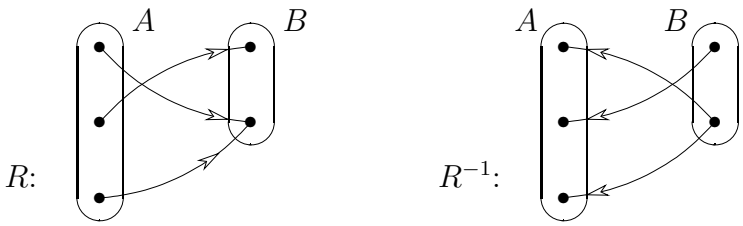
V této lekci definujeme inverzi a skládání více relací a ukážeme na zásadní význam těchto operací pro relační databáze. Na to navážeme definicí základních vlastností funkcí a diskusí souvisejících operací inverze a skládání funkcí. V rámci této diskuse si podrobně probereme problematiku cyklů permutací a skládání permutací.

8.1 Inverze a skládání relací

Ve výkladu se nyní vrátíme k obecnému pojetí relací mezi dvěma (třeba různými) množinami. K využitelné práci s relacemi v aplikacích se dostaneme, pokud budeme umět relace správně „převracet“ a „skládat“. To nám umožní následující definice.

`Definice`: Nechť $R \subseteq A \times B$ je *binární* relace mezi A a B. *Inverzní relace* k relaci R se značí R^{-1} a je definována takto:

$$R^{-1} = \{(b, a) \mid (a, b) \in R\}$$

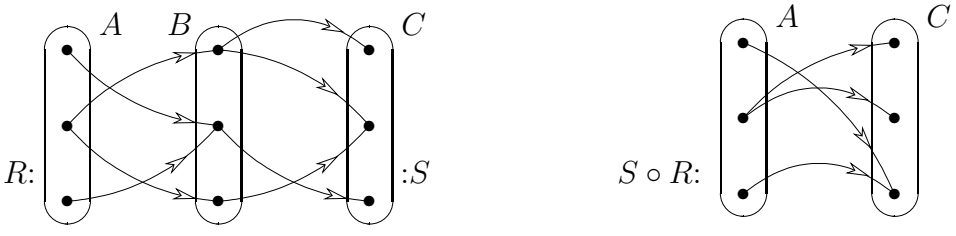


R^{-1} je tedy relace *mezi* B *a* A.

Následuje klíčová definice, pro jejíž bližší ilustraci odkazujeme také na Příklad 8.3.

Definice 8.1. ***Složení (kompozice)*** relací R a S.
Nechť $R \subseteq A \times B$ a $S \subseteq B \times C$ jsou binární relace. *Složení* relací R a S (v tomto pořadí!) je relace $S \circ R \subseteq A \times C$ definovaná takto:

$$S \circ R = \{(a, c) \mid \text{existuje } b \in B \text{ takové, že } (a, b) \in R, (b, c) \in S\}$$



Složení relací čteme „*R složeno s S*“ nebo (pozor na pořadí!) „*S po R*“.

`Komentář:` Několik matematických příkladů skládání relací následuje zde.

* Je-li
  - $A = \{a, b\}, \quad B = \{1, 2\}, \quad C = \{X, Y\},$
  - $R = \{(a, 1), (b, 1), (b, 2)\}, \quad S = \{(1, X)\},$

  pak složením vznikne relace
  - $S \circ R = \{(a, X), (b, X)\}.$
* Složením funkcí $h(x) = x^2$ a $f(x) = x + 1$ na $\mathbb{R}$ vznikne funkce
$$(f \circ h)(x) = f(h(x)) \; = \; x^2 + 1.$$
* Složením těchže funkcí „*naopak*“ ale vznikne funkce $(h \circ f)(x) = h(f(x)) \; = \; (x + 1)^2$.

`Poznámka`: Nepříjemné je, že v některých oblastech matematiky (například v algebře při skládání zobrazení) se setkáme s *právě opačným* zápisem skládání, kdy se místo $S \circ R$ píše $R \cdot S$ nebo jen RS. Proto je si vždy dobré slovně ujasnit, které pořadí skládaných relací máme na mysli. My zde zásadně budeme používat pořadí $S \circ R$.

Tvrzení 8.2. *Nechť $R \subseteq A \times B$ a $S \subseteq B \times C$ jsou binární relace. Pak inverzí složené relace $S \circ R$ je relace $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$.*

`Důkaz`: Zopakujme si, co nám říkají výše uvedené definice:

$$\begin{aligned} R^{-1} &= \{(b, a) \mid (a, b) \in R\} \\ S^{-1} &= \{(c, b) \mid (b, c) \in S\} \\ (S \circ R)^{-1} &= \{(c, a) \mid \textit{existuje } b \in B \text{ takové, že } (a, b) \in R, (b, c) \in S\} \end{aligned}$$

Dosazením z prvních dvou vztahů do třetího vzniká

$$(S \circ R)^{-1} \; = \; \{(c, a) \mid \textit{existuje } b \in B \text{ takové, že } (b, a) \in R^{-1}, (c, b) \in S^{-1}\}$$

a to již podle Definice 8.1 ihned implikuje požadovaný závěr

$$(S \circ R)^{-1} = R^{-1} \circ S^{-1}.$$

□

8.2 Praktické použití skládání

Podívejme se nyní, jak se skládání relací přirozeně objevuje v práci s relačními databázemi. (Dá se zjednodušeně říci, že právě v operátoru skládání tabulkových relací tkví hlavní smysl relačních databází...)

Příklad 8.3. *Skládání v relační databázi studentů, jejich předmětů a fakult.*

Mějme dvě binární relace – jednu R přiřazující studentům MU kódy jejich zapsaných předmětů, druhou S přiřazující kódy předmětů jejich mateřským fakultám. Malý výsek z těchto relací může v tabulkové reprezentaci vypadat třeba následovně.

R :

student (učo)	předmět (kód)
121334	MA010
133935	M4135
133935	IA102
155878	M1050
155878	IB000

S :

předmět (kód)	fakulta MU
MA010	FI
IB000	FI
IA102	FI
M1050	PřF
M4135	PřF

Jak z těchto „tabulkových“ relací zjistíme, kteří studenti mají zapsané předměty na kterých fakultách (třeba na FI)?

Jedná se jednoduše o *složení relací* $S \circ R$. V našem příkladě tabulkové reprezentace vyjde výsledek:

$S \circ R$:

student (učo)	fakulta MU
121334	FI
133935	FI
133935	PřF
155878	FI
155878	PřF

□

Zobecněné skládání relací

V praktických použitích relačních tabulek povětšinou nevystačíme jen s binárními relacemi, takže je přirozené se ptát, jestli lze podobně skládat i více-ární relace. Odpověď je snadná – lze to a ani nepotřebujeme novou definici, vystačíme s tou, kterou už máme výše uvedenou.

`Definice`: (***skládání relací vyšší arity***):
Mějme relace $T \subseteq K_1 \times K_2 \times \cdots \times K_k$ a $U \subseteq L_1 \times L_2 \times \cdots \times L_\ell$, přičemž pro nějaké $m < \min(k, \ell)$ platí $L_1 = K_{k-m+1}, L_2 = K_{k-m+2}, \ldots, L_m = K_k$. Pak relaci T lze *složit* s relací U na zvolených m složkách $L_1, \ldots, L_m$ („překrytí“) s *použitím Definice 8.1* takto:

* Položme $A = K_1 \times \cdots \times K_{k-m}$, $B = L_1 \times \cdots \times L_m$ a $C = L_{m+1} \times \cdots \times L_\ell$.
* Příslušné relace pak jsou $R = \{(\vec{a}, \vec{b}) \in A \times B \mid (a_1, \ldots a_{k-m}, b_1, \ldots b_m) \in T\}$ a $S = \{(\vec{b}, \vec{c}) \in B \times C \mid (b_1, \ldots b_m, c_{m+1}, \ldots c_\ell) \in U\}$.
* Nakonec přirozeně položme $U \circ_m T \simeq S \circ R$, takže vyjde

$$U \circ_m T = \left\{(\vec{a}, \vec{c}) \mid \text{ ex. } \vec{b} \in B, \text{ že } (a_1, \ldots a_{k-m}, b_1, \ldots b_m) \in T \text{ a } (b_1, \ldots b_m, c_{m+1}, \ldots c_\ell) \in U\right\}.$$

Schematicky pro snadnější orientaci ve složkách našich relací:

$$\begin{aligned} T \subseteq \quad & K_1 \times \cdots \times K_{k-m} \times \quad K_{k-m+1} \times \cdots \times K_k \\ U \subseteq \quad & \qquad\qquad\qquad\quad L_1 \quad \times \cdots \times L_m \quad \times L_{m+1} \times \cdots \times L_\ell \\ U \circ_m T \subseteq \quad & \underbrace{K_1 \times \cdots \times K_{k-m}}_{A} \times \quad \underbrace{\qquad\qquad\qquad}_{B} \quad \times \underbrace{L_{m+1} \times \cdots \times L_\ell}_{C} \end{aligned}$$

Opět je nejjednodušší si koncept skládání vícečetných relací ilustrovat příkladem.

Příklad 8.4. *Skládání v relační databázi pasažérů a letů u leteckých společností.*

Podívejme se na příklad hypotetické rezervace letů pro cestující, relace T. Jak známo (tzv. codeshare), letecké společnosti si mezi sebou „dělí“ místa v letadlech, takže různé lety (podle kódů) jsou ve skutečnosti realizovány stejným letadlem jedné ze společností. To zase ukazuje relace U.

T :

pasažér	datum	let
Petr	5.11.	OK535
Pavel	6.11.	OK535
Jan	5.11.	AF2378
Josef	5.11.	DL5457
Alena	6.11.	AF2378

U :

datum	let	letadlo
5.11.	OK535	ČSA
5.11.	AF2378	ČSA
5.11.	DL5457	ČSA
6.11.	OK535	AirFrance
6.11.	AF2378	AirFrance

Ptáme-li se nyní, setkají se Petr a Josef na palubě stejného letadla? Případně, čí letadlo to bude? Odpovědi nám dá složení relací $U \circ_2 T$, jak je popsáno výše.

$U \circ_2 T$:

pasažér	letadlo
Petr	ČSA
Josef	ČSA
Pavel	AirFrance
...	...

Zkuste se zamyslet, lze tyto dvě relace skládat ještě jinak? Co by pak bylo významem? □

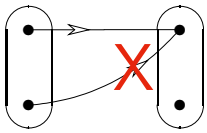
8.3 Vlastnosti funkcí

Funkce je intuitivně chápána jako předpis, který každé hodnotě (či hodnotám v případě více proměnných) levé strany přiřadí jednoznačně hodnotu pravé strany, tzv. *funkční hodnotu* neboli *výsledek*. Avšak v nejobecnější podobě pojem funkce nemusí být nijak svázaný s konkrétním předpisem jejího výpočtu; tento obecnější náhled je obzvláště potřebný v informatice, kdy se budete setkávat i s funkcemi, které *nejsou algoritmicky vyčíslitelné*.

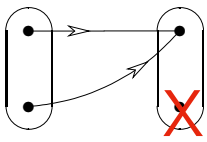
Připomeňme si, že funkce je podle Definice 3.12 speciálním případem relace, mající pro každou hodnotu levé strany jedinou (funkční) hodnotu pravé strany. V tomto oddíle si uvedeme několik vlastností specifických pro funkce.

Definice 8.5. ***Funkce*** (případně parciální funkce) $f : A \to B$ je

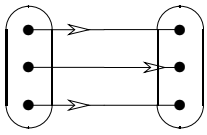
* *injektivní* (nebo také *prostá*) právě když pro každé $x, y \in A$, $x \neq y$ platí $f(x) \neq f(y)$;



* *surjektivní* (nebo také „*na*“) právě když pro každé $y \in B$ existuje $x \in A$ takové, že $f(x) = y$;



* *bijektivní* (vzáj. *jednoznačná*) právě když je injektivní a *současně* surjektivní.



`Komentář:` Následují jednoduché ukázky vlastností funkcí.

* Funkce *plus* : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ je surjektivní, ale není prostá.
* Funkce $g : \mathbb{Z} \to \mathbb{N}$ daná předpisem

$$g(x) = \begin{cases} -2x - 1 & \text{jestliže } x < 0, \\ 2x & \text{jinak} \end{cases}$$

je bijektivní.

* Funkce $\emptyset : \emptyset \to \emptyset$ je bijektivní.
* Funkce $\emptyset : \emptyset \to \{a, b\}$ je injektivní, ale není surjektivní.

Příklad 8.6. *Dokázali byste nalézt jednoduše (tj. „hezky“) vypočitatelnou bijektivní funkci* $\mathbb{N} \to \mathbb{N} \times \mathbb{N}$*?*

V řešení si problém nejprve zkusíme představit vhodným geometrickým modelem (není to nutné, ale velmi dobré pro pochopení). Množina $\mathbb{N} \times \mathbb{N}$ nechť je představována všemi jednotkovými čtverečky v prvním kvadrantu roviny. Celý tento kvadrant postupně vyplňujeme po krocích většími čtverci z levého dolní rohu, tj. postupně o rozměrech 1×1, 2×2, 3×3, atd. Co nám v každém kroku i přibude (postupujeme od $i = 0$)? Bude to pásek $2i + 1$ jednotkových čtverečků ve tvaru $\urcorner$. Pokud tyto pásky narovnáme a seřadíme za sebou, dostaneme jednostranně nekonečný pás jednotkových čtverečků, což znamená pro nás množinu $\mathbb{N}$. Kýžená bijekce je na světě!

Nyní zbývá formální stránka věci – zapsat tuto bijekci předpisem, čili vzorcem. K tomu si připomeneme, že $\mathbb{N} = \{0, 1, 2, 3, \dots\}$. Jak si sami můžete snadno ověřit, předpis výpočtu hodnoty $f(n)$, kde $f : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ je výše geometricky popsaná bijekce, je následující:

* Nalezneme největší k takové, že $k^2 \leq n$ $\quad(k := \lfloor \sqrt{n} \rfloor)$.
* Pro $n - k^2 \leq k$ máme $f(n) := (k, n - k^2)$,
  kdežto pro $n - k^2 > k$ je $f(n) := (k^2 + 2k - n, k)$. □

Příklad 8.7. *Pro jaké hodnoty parametrů* a, b *je funkce* $f(x) = ax^3 + 3x^2 + bx + 1$ *z* $\mathbb{R}$ *do* $\mathbb{R}$ *injektivní či surjektivní?*

Nejprve rozebereme případ $a = 0$, kdy platí $f(0) = f(-b/3) = 1$. Pro $b \neq 0$ je $-b/3 \neq 0$, kdežto pro $b = 0$ zase platí $f(1) = f(-1) = 4$. Tudíž pro $a = 0$ funkce f nemůže být nikdy injektivní. Zároveň pro $a = 0$ nemůže být ani surjektivní, neboť pro každé x platí, že $f(x) = 3x^2 + bx + 1 \geq 1 - b^2/12$.

Nechť tedy $a > 0$. Potom $\lim_{x \to -\infty} f(x) = -\infty$ a $\lim_{x \to \infty} f(x) = \infty$ a vzhledem ke spojitosti funkce f musí nabývat všech reálných hodnot a je surjektivní. Totéž platí pro $a < 0$. Zodpovězení otázky, kdy je f injektivní, již vyžaduje některé základní poznatky matematické analýzy. Ze školní znalosti průběhu polynomické funkce plyne, že ta je prostá právě tehdy, když nemá žádný lokální extrém. V případě kubického polynomu to konkrétně znamená, že první derivace nesmí být nulová ve dvou různých reálných bodech. Vyjádřeno příslušnou kvadratickou rovnicí, $f'(x) = 3ax^2 + 6x + b = 0$, dostáváme z jejího determinantu podmínku $36 - 12ab \leq 0$.

Závěr: Funkce f je surjektivní, právě když $a \neq 0$. Funkce f je injektivní, právě když $a \neq 0$ a $36 - 12ab \leq 0$. □

Inverze funkce

Inverze funkce je dána definicí inverze relace z Oddílu 8.1.

`Komentář:` Příklady inverzí pro běžné funkce:

* Inverzí *bijektivní* funkce $f(x) = x + 1$ na $\mathbb{Z}$ je funkce $f^{-1}(x) = x - 1$.
* Inverzí *prosté* funkce $f(x) = e^x$ na $\mathbb{R}$ je *parciální* funkce $f^{-1}(x) = \ln x$.
* Funkce $g(x) = x \bmod 3$ *není prostá* na $\mathbb{N}$, a proto její inverzí je „*jen*“ relace $g^{-1} = \{(a, b) \mid a = b \bmod 3\}$. Konkrétně $g^{-1} = \{(0,0), (0,3), (0,6), \dots, (1,1), (1,4), \dots, (2,2), (2,5), \dots\}$.

K pojmu inverze funkce se vztahují následující základní a jednoduché poznatky:

Tvrzení 8.8. *Mějme funkci $f : A \to B$. Pak její inverzní relace f^{-1} je*

a) parciální funkce právě když f je prostá,

b) funkce právě když f je bijektivní.

`Důkaz` vyplývá přímo z definic funkce a inverze relace. Zkuste si sami. □

Tvrzení 8.9. *Je-li inverzní relace f^{-1} funkce f taktéž parciální funkcí, tak platí $f^{-1}(f(x)) = x$ pro všechna x z definičního oboru.*

`Důkaz` vyplývá přímo z definic funkce a inverze relace. □

8.4 Skládání funkcí, permutace

Obdobně jako u inverze, i skládání funkcí je dáno definicí skládání relací z Oddílu 8.1. Jelikož však definice skládání relací je poměrně složitá, je vhodné si uvést alternativní jednoduchou definici skládání pouze pro funkce.

`Fakt`: Mějme funkce (zobrazení) $f : A \to B$ a $g : B \to C$. Pak jejich *složením* coby relací v tomto pořadí vznikne zobrazení $(g \circ f) : A \to C$ definované

$$(g \circ f)(x) := g(f(x)) \,.$$

`Komentář:`

* Jak například na běžné kalkulačce vypočteme hodnotu funkce $\sin^2 x$? Složíme (v tomto pořadí) „elementární“ funkce $f(x) = \sin x$ a $g(x) = x^2$.
* Jak bychom na „elementární“ funkce rozložili aritmetický výraz $2\log(x^2 + 1)$? Ve správném pořadí složíme funkce $f_1(x) = x^2$, $f_2(x) = x + 1$, $f_3(x) = \log x$ a $f_4(x) = 2x$.
* A jak bychom obdobně vyjádřili složením funkcí aritmetický výraz $\sin x + \cos x$? Opět je odpověď přímočará, vezmeme „elementární“ funkce $g_1(x) = \sin x$ a $g_2(x) = \cos x$, a pak je „složíme“ další funkcí $h(x, y) = x + y$. Vidíme však, že takto pojaté „skládání“ už nezapadá hladce do našeho zjednodušeného formalismu skládání relací.

  Pro nedostatek prostoru si skládání funkcí s více parametry nedefinujeme, ale sami vidíte, že obdobné skládání se v programátorské praxi vyskytuje doslova „na každém rohu“ a ani se nad tím nepozastavujeme.

Skládání permutací

Po zbytek tohoto oddílu se zaměříme na permutace coby speciální případ (bijektivních) zobrazení, neboť dobře ilustrují problematiku skládání bez zbytečných technických komplikací a zároveň se jedná o užitečnou oblast diskrétní matematiky.

`Značení`: Nechť *permutace* π množiny $\{1, 2, \ldots, n\}$ je určena seřazením jejích prvků jako $(p_1, p_2, \ldots, p_n)$. Pak π je zároveň *bijektivním zobrazením* $\{1, \ldots, n\} \to \{1, \ldots, n\}$ definovaným předpisem $\pi(i) = p_i$. Abychom postihli obě tyto podoby permutace zároveň, budeme také jako pomůcku používat obvyklý rozšířený zápis $\begin{pmatrix} 1 & 2 & \ldots & n \\ p_1 & p_2 & \ldots & p_n \end{pmatrix}$.

Permutace lze *skládat jako relace* (funkce) podle Definice 8.1 a platí následující:

Tvrzení 8.10. *Mějme permutace π a σ množiny $\{1, 2, \ldots, n\}$. Pak jejich složení $\sigma \circ \pi$ je opět permutací množiny $\{1, 2, \ldots, n\}$.*

`Důkaz` vyplývá snadno aplikací definice bijektivní funkce. □

`Poznámka`: Všechny permutace množiny $\{1, 2, \ldots, n\}$ spolu s operací skládání tvoří grupu, zvanou symetrická grupa S_n. Permutační grupy (podgrupy symetrické grupy) jsou velice důležité v algebře, neboť každá konečná grupa je vlastně isomorfní některé permutační grupě.

`Komentář:` Příkladem permutace vyskytujícím se v programátorské praxi je třeba zobrazení $i \mapsto (i + 1) \mod n$ ("inkrement"). Často se třeba lze setkat (aniž si to mnohdy uvědomujeme) s permutacemi při indexaci prvků polí.

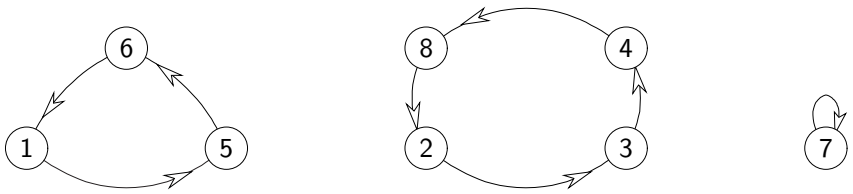
V kontextu pohledu na funkce a jejich skládání coby relací si zavedeme jiný, názornější, způsob zápisu permutací – pomocí jejich cyklů.

`Definice`: Nechť π je permutace na množině A. *Cyklem v π* rozumíme posloupnost $\langle a_1, a_2, \ldots, a_k \rangle$ různých prvků A takovou, že $\pi(a_i) = a_{i+1}$ pro $i = 1, 2, \ldots, k - 1$ a $\pi(a_k) = a_1$.

Jak název napovídá, v zápise cyklu $\langle a_1, a_2, \ldots, a_k \rangle$ není důležité, kterým prvkem začneme, ale jen dodržení cyklického pořadí. Cyklus v permutaci může mít i jen jeden prvek (zobrazený na sebe).

`Komentář:` Nakreslete si (vámi zvolenou) permutaci π obrázkem, ve kterém vedete šipku vždy od prvku i k prvku $\pi(i)$. Pak uvidíte, že cykly dle naší definice jsou právě cykly tvořené šipkami ve vašem obrázku. S tímto grafickým zobrazením pro vás nebude problém pochopit následující látku.

Například permutaci $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 5 & 3 & 4 & 8 & 6 & 1 & 7 & 2 \end{pmatrix}$ si lze obrázkem nakreslit takto:



Reprezentace permutací jejich cykly

Věta 8.11. *Každou permutaci π na konečné množině A lze zapsat jako složení cyklů na disjunktních podmnožinách (rozkladu) A.*

`Důkaz`: Vezmeme libovolný prvek $a_1 \in A$ a iterujeme zobrazení $a_2 = \pi(a_1)$, $a_3 = \pi(a_2)$, atd., až se dostaneme „zpět" k $a_{k+1} = \pi(a_k) = a_1$. Proč tento proces skončí? Protože A je konečná a tudíž ke zopakování některého prvku a_{k+1} musí dojít. Nadto je π prostá, a proto nemůže nastat $\pi(a_k) = a_j$ pro $j > 1$. Takto získáme první cyklus $\langle a_1, \ldots, a_k \rangle$.

Induktivně pokračujeme s hledáním dalších cyklů ve zbylé množině $A' = A \setminus \{a_1, \ldots, a_k\}$, dokud nezůstane prázdná. V tomto indukčním kroku si musíme uvědomit, že π omezené na nosnou množinu A' je stále permutací podle definice (neboli žádná prvek z A' se nezobrazí do $\{a_1, \ldots, a_k\}$). □

`Značení` permutace jejími cykly: Nechť se permutace π podle Věty 8.11 skládá z cyklů $\langle a_1, \ldots, a_k\rangle$, $\langle b_1, \ldots, b_l\rangle$ až třeba $\langle z_1, \ldots, z_m\rangle$. Pak zapíšeme

$$\pi = \big(\, \langle a_1, \ldots, a_k\rangle\, \langle b_1, \ldots, b_l\rangle \ldots \langle z_1, \ldots, z_m\rangle \,\big)\,.$$

`Komentář:` Primitivní pseudonáhodné generátory v počítačích iterují z náhodného počátku permutaci danou vztahem $i \mapsto (i + p) \mod q$. Je pochopitelné, že tato permutace nesmí obsahovat krátké cykly, lépe řečeno, měla by se skládat z jediného (dlouhého) cyklu. (Pro úplnost, jedná se o permutaci množiny $\{0, 1, \ldots, q-1\}$).

Příklad 8.12. *Ukázka skládání permutací daných svými cykly.*

Vezměme 7-prvkovou permutaci zadanou seřazením $\pi = (3, 4, 5, 6, 7, 1, 2)$, neboli rozšířeně zapsanou jako $\left(\begin{smallmatrix}1&2&3&4&5&6&7\\3&4&5&6&7&1&2\end{smallmatrix}\right)$. Ta se skládá z jediného cyklu $\langle 1, 3, 5, 7, 2, 4, 6\rangle$. Jiná permutace daná seřazením $\sigma = (5, 3, 4, 2, 6, 1, 7)$, neboli $\left(\begin{smallmatrix}1&2&3&4&5&6&7\\5&3&4&2&6&1&7\end{smallmatrix}\right)$, se rozkládá na tři cykly $\langle 1, 5, 6\rangle$, $\langle 2, 3, 4\rangle$ a $\langle 7\rangle$. Nyní určíme složení $\sigma \circ \pi$ těchto dvou permutací (už přímo v zápisu cykly):

$$\big(\langle 1, 5, 6\rangle\langle 2, 3, 4\rangle\langle 7\rangle\big) \circ \big(\langle 1, 3, 5, 7, 2, 4, 6\rangle\big) = \big(\langle 1, 4\rangle\langle 2\rangle\langle 3, 6, 5, 7\rangle\big)$$

(Nezapomínejme, že první se ve složení aplikuje pravá permutace!)
Postup skládání jsme použili následovný: 1 se zobrazí v permutaci vpravo na 3 a pak vlevo na 4. Následně 4 se zobrazí na 6 a pak na 1. Tím „uzavřeme“ první cyklus $\langle 1, 4\rangle$. Dále se 2 zobrazí na 4 a pak hned zpět na 2, tj. má samostatný cyklus. Zbylý cyklus $\langle 3, 6, 5, 7\rangle$ určíme analogicky. □

Rozšiřující studium

Mějte na paměti, že na pojmech množin a relací jsou vystavěny prakticky všechny datové struktury používané v dnešní informatice. Explicitně toto můžete vidět na relačních databázích, ale i v mnoha jiných implicitních výskytech. A dalším důležitým matematickým datovým typem odvozeným z binárních relací jsou pak grafy probírané od Lekce 9, na nichž v různé míře staví většina základních algoritmů, které se budete učit.

S formalizací pojmu funkce a jejími vlastnostmi se zatím setkáváte spíše v matematice, avšak například na bijektivní funkce v informatice narazíte hned při zpracování dat při volbě klíče apod. Našim cílem bylo ukázat práci s funkcemi v jejich abstraktní podobě, tj. bez vazby na nějaký konkrétní analytické vzoreček jako v matematice. Takové abstraktní pojetí je bližší tomu, jak relace a funkce využívá informatika.

9 Pojem grafu

Úvod

Po relacích se náš výklad upře směrem k další základní struktuře diskrétní matematiky, grafu. Třebaže grafy (pozor, nepleťte si je s grafem funkce) jsou jen jedním z mnoha typů objektů v diskrétní matematice a vlastně pouze speciálním případem binárních relací, vydobyly si svou užitečností a názorností (a to především ve vztahu k informatice) důležité místo na slunci. Dá se tak bez nadsázky říci, že teorie grafů je asi nejvýznamnější součástí soudobé diskrétní matematiky, a proto se jí budeme věnovat po dvě následující lekce.

Neformálně řečeno, graf se skládá z vrcholů, které si představíme jako nakreslené „puntíky“, a z hran, které spojují dvojice vrcholů mezi sebou. Své důležité místo si grafy získaly především dobře vyváženou kombinací vlastností – snadno pochopitelným názorným nakreslením a zároveň jednoduchým zpracováním na počítačích. Díky těmto vlastnostem se grafy prosadily jako vhodný matematický model pro popis různých vztahů mezi daty a objekty.

Cíle

Definujeme, co je graf a jaké jsou nejzákladnější grafové pojmy (třeba hrany a stupně, podgrafy, souvislost). Klademe důraz na to, aby se čtenář naučil grafy uchopit a účelně s nimi pracovat a také aby správně viděl „stejnost“ (isomorfismus) grafů. Poté se věnujeme nejjednoduššímu druhu grafů, totiž stromům. Jedná se vesměs o pojmy, které se hojně vyskytují v informatických aplikacích grafů a především pak ve většině základních algoritmů, které se student informatiky učí.

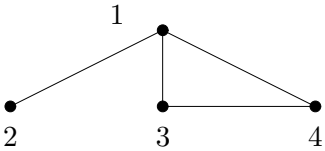
9.1 Definice grafu

Hned na úvod přistoupíme k formální definici grafu. Bude se jednat o definici tzv. *jednoduchého neorientovaného* grafu, který budeme považovat za základní, pokud neřekneme jinak. Svým způsobem navazujeme na reprezentace relací v Oddíle 6.1.

Definice 9.1. ***Graf*** je uspořádaná dvojice $G = (V, E)$,
kde V je konečná množina *vrcholů* a E je množina *hran* – množina vybraných dvouprvkových podmnožin množiny vrcholů; tj. $E \subseteq \binom{V}{2}$.

`Značení`: Hranu mezi vrcholy u a v píšeme jako $\{u, v\}$, nebo zkráceně uv. Vrcholy spojené hranou jsou *sousední* a hrana uv *vychází* z vrcholů u a v. Na množinu vrcholů známého grafu G odkazujeme jako na $V(G)$, na množinu hran $E(G)$.

`Komentář:` Grafy se často zadávají přímo názorným obrázkem, jinak je lze formálně zadat výčtem vrcholů a výčtem hran. Například:



$$V = \{1, 2, 3, 4\}, \quad E = \Big\{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{3, 4\}\Big\}$$

Na graf se lze dívat také jako na symetrickou ireflexivní relaci, kde hrany tvoří právě dvojice prvků z této relace. Pro další výklad je žádoucí si uvědomit, že v základním podání jsou vrcholy grafu vždy „bezejmenné“ (neformálně máme jen ty puntíky, které žádné jméno nemusí mít přiřazené). Jména vrcholům grafu přiřazujeme jen pro účely zápisu či popisu tohoto grafu a můžeme je volit libovolně.

Stupně vrcholů v grafu

Máme-li graf, často nás zajímá, kolik z kterého vrcholu vychází hran-spojnic, neboli kolik má vrchol „sousedů“. Proto jedním z prvních definovaným pojmů bude stupeň vrcholu v grafu.

Definice 9.2. ***Stupněm vrcholu*** v v grafu G
rozumíme počet hran vycházejících z v. Stupeň v v grafu G značíme $d_G(v)$.

`Komentář:` Slovo „vycházející“ zde nenaznačuje žádný směr; je totiž obecnou konvencí u neorientovaných grafů říkat, že hrana vychází z obou svých konců zároveň.

Například v nakreslené ukázce jsou stupně přímo zapsány u vrcholů.

`Definice`: Graf je *d-regulární*, pokud všechny jeho vrcholy mají stejný stupeň d.

`Značení`: *Nejvyšší* stupeň v grafu G značíme $\Delta(G)$ a *nejnižší* $\delta(G)$.

Věta 9.3. *Součet stupňů v grafu je vždy sudý, roven dvojnásobku počtu hran.*

`Důkaz`. Při sčítání stupňů vrcholů v grafu započítáme každou hranu dvakrát – jednou za každý její konec. Proto také výsledek vyjde sudý. □

Příklad 9.4. *Zodpovězme následující otázky:*
a) Kolik hran má graf se 17 *vrcholy stupňů* 4*?*
b) Existují dva „různé“ grafy se 6 *vrcholy stupňů* 2*?*

V otázce (a) je součet stupňů $17 \cdot 4 = 68$ a podle Věty 9.3 je počet hran $68/2 = 34$.
V (b) existují dva velmi odlišné grafy uvedené vlastnosti, snadno je nakreslíme takto:

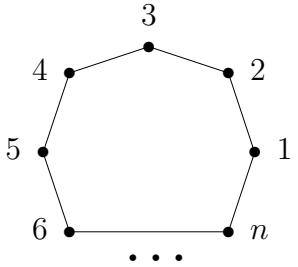
V dalším textu si zavedeme některé základní názvy, které nám umožní podobně jednoduché grafy snadno a stejně názorně popisovat slovy místo obrázků. □

Běžné typy grafů

Pro snadnější vyjadřování je zvykem některé běžné typy grafů nazývat popisnými jmény. Jde čistě o věc konvence a autoři se mohou v některých názvech lišit (i přicházet s novými názvy), avšak následujících pět názvů patří k všeobecným základům teorie grafů.

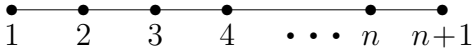
Kružnice délky n má $n \geq 3$ různých vrcholů spojených „do jednoho cyklu“ n hranami:

C_n



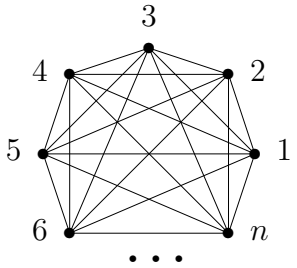
Cesta délky $n \geq 0$ má $n+1$ různých vrcholů spojených „za sebou“ n hranami:

P_n



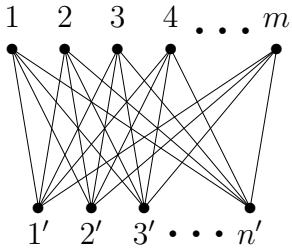
Úplný graf na $n \geq 1$ vrcholech má n různých vrcholů spojených po všech dvojicích (tj. celkem $\binom{n}{2}$ hran):

K_n



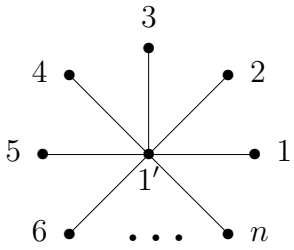
Úplný bipartitní graf na $m \geq 1$ a $n \geq 1$ vrcholech má $m+n$ vrcholů ve dvou skupinách (partitách), přičemž hranami jsou spojeny všechny $m \cdot n$ dvojice z různých skupin:

$K_{m,n}$



Hvězda s $n \geq 1$ ***rameny*** je zvláštní název pro úplný bipartitní graf $K_{1,n}$:

S_n



`Definice`: Formálně nechť *kružnice* délky $n \geq 3$ je graf C_n, kde $V(C_n) = \{1, 2, \ldots, n\}$ a $E(C_n) = \big\{\{i, i+1\} : 1 \leq i \leq n\big\} \cup \big\{\{n, 1\}\big\}$. Nechť *cesta* délky $n \geq 0$ je graf P_n, kde $V(P_n) = \{1, 2, \ldots, n+1\}$ a $E(P_n) = \big\{\{i, i+1\} : 1 \leq i \leq n+1\big\}$. Nechť *úplný graf* na $n \geq 1$ vrcholech je K_n, kde $V(K_n) = \{1, 2, \ldots, n\}$ a $E(K_n) = \big\{\{i, j\} : 1 \leq i < j \leq n\big\}$. Nechť

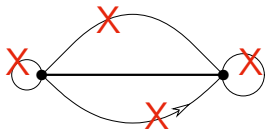
úplný bipartitní graf na $m \geq 1$ a $n \geq 1$ vrcholech je $K_{m,n}$, kde $V(K_{m,n}) = \{1, 2, \ldots, m, m+1, \ldots, m+n\}$ a $E(K_{m,n}) = \big\{\{i,j\} : 1 \leq i \leq m,\ m+1 \leq j \leq m+n\big\}$.

Příklad 9.5. *Zodpovězte si sami následující snadné otázky:*

* Pro jakou hodnotu n je úplný graf K_n zároveň cestou?
* Pro jakou hodnotu n je úplný graf K_n zároveň kružnicí?
* Pro jaké hodnoty $m, n > 0$ je úplný bipartitní graf $K_{m,n}$ zároveň kružnicí?
* Kolik hran musíte přidat do kružnice délky 6, aby vznikl úplný graf na 6 vrcholech?
* Pro jaké hodnoty $m, n > 0$ úplný bipartitní graf $K_{m,n}$ neobsahuje žádnou kružnici? □

Zmínka o zobecněných grafech

`Komentář:` Všimněme si, že v definici grafu (Def. 9.1) vůbec neuvažujeme možnosti vícenásobných hran (mezi stejnou dvojicí vrcholů) a tzv. „smyček“ (hrana se stejným jedním koncem) – takovému zobecnění by se říkalo *multigraf*. Také prozatím nepřisuzujeme hranám žádný směr.



V Oddíle 9.4 si však ještě zavedeme *orientované grafy*, které každé hraně přiřazují jistý směr.

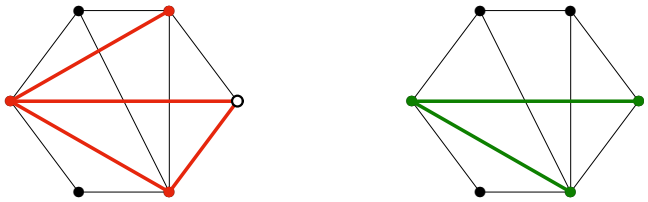
9.2 Podgrafy a isomorfismus

Dva základní nástroje pro práci s grafy jsou následující; možnost popisovat „část grafu“ (podobně jako podmnožinu množiny, avšak je nutno se vyvarovat nekorektních situací) a poznávat „stejnost“ dvou grafů.

`Definice:` *Podgrafem* grafu G rozumíme libovolný graf H na podmnožině vrcholů $V(H) \subseteq V(G)$, který má za hrany *libovolnou* podmnožinu hran grafu G majících oba vrcholy ve $V(H)$.

Píšeme $H \subseteq G$, tj. stejně jako množinová inkluze (ale význam je trochu jiný).

`Komentář:` Na následujícím obrázku vidíme zvýrazněné podmnožiny vrcholů hran. Proč se vlevo nejedná o podgraf? Obrázek vpravo už podgrafem je.

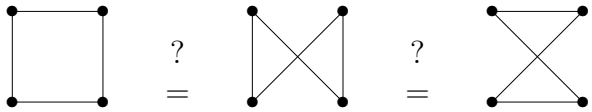


`Definice:` *Indukovaným podgrafem* je podgraf $H \subseteq G$ takový, který obsahuje *všechny hrany* grafu G mezi dvojicemi vrcholů z $V(H)$.

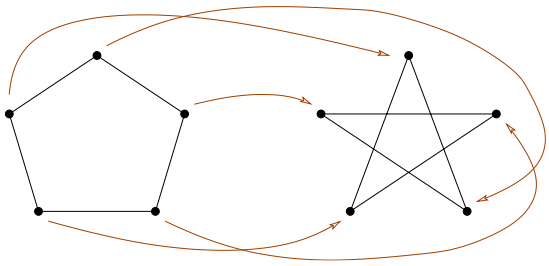
„Stejnost“ grafů

Pozorný čtenář si možná již při čtení předchozího oddílu položil otázku: Co když vezmeme jeden graf (třeba kružnici délky 4) a nakreslíme nebo zapíšeme jej jednou tak, podruhé zase jinak – *je to stále tentýž graf nebo ne*? Viz obrázky dole.

Přísně formálně řečeno, každé další nakreslení jistého grafu, třeba této kružnice C_4, je jiným grafem, ale přitom bychom rádi řekli, že různá nakreslení téhož grafu jsou „stále stejná“. Pro tuto stejnost grafů se vžil pojem *isomorfní grafy*.



Definice 9.6. ***Isomorfismus*** $\simeq$ grafů G a H
je bijektivní (*vzájemně jednoznačné*) zobrazení $f : V(G) \to V(H)$, pro které platí, že každá dvojice vrcholů $u, v \in V(G)$ je spojená hranou v G právě tehdy, když je dvojice $f(u), f(v)$ spojená hranou v H.



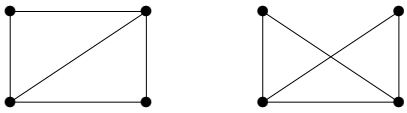
Grafy G a H jsou *isomorfní*, pokud mezi nimi existuje isomorfismus. Píšeme $G \simeq H$.

Vlastnosti isomorfismu

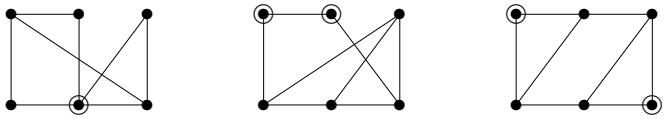
`Fakt`: Mějme isomorfismus f grafů G a H. Pak platí následující

* G a H mají stejný počet hran,
* f zobrazuje na sebe vrcholy stejných stupňů, tj. $d_G(v) = d_H(f(v))$.

`Komentář:`



U výše zakreslených dvou grafů objevíme isomorfismus velmi snadno – podíváme se, jak si odpovídají vrcholy stejných stupňů.



Naopak v této trojici grafů (se stejnými počty vrcholů i hran) žádné dva nejsou isomorfní. Proč? Ten vlevo má vrchol stupně 4, čímž se od obou zbylých liší. Prostřední graf pak má jediné dva vrcholy stupně 2 spojené hranou, kdežto v pravém takové dva vrcholy spojené nejsou (isomorfismus by je však i s hranou musel zachovat).

Příklad 9.7. *Jsou následující dva grafy isomorfní?*

Pokud mezi nakreslenými dvěma grafy hledáme isomorfismus, nejprve se podíváme, zda mají stejný počet vrcholů a hran. Mají. Pak se podíváme na stupně vrcholů a zjistíme, že oba mají stejnou posloupnost stupňů $2, 2, 2, 2, 3, 3$. Takže ani takto jsme mezi nimi nerozlišili a mohou (nemusejí!) být isomorfní. Dále tedy nezbývá, než zkoušet všechny přípustné možnosti zobrazení isomorfismu z levého grafu do pravého.

Na levém grafu si pro ulehčení všimněme, že oba vrcholy stupně tři jsou si symetrické, proto si bez újmy na obecnosti můžeme vybrat, že nejlevější vrchol prvního grafu, označme jej 1, se zobrazí na nejlevější vrchol $1'$ v druhém grafu (taky stupně tři). Očíslujme zbylé vrcholy prvního grafu $2, \ldots, 6$ v kladném smyslu, jak je ukázáno na následujícím obrázku. Druhý vrchol stupně tři, označený 4, se musí zobrazit na analogický vrchol druhého grafu (pravý spodní). Pak je již jasně vidět, že další sousedé $2, 6$ vrcholu 1 se zobrazí na analogické sousedy $2', 6'$ vrcholu $1'$ v druhém grafu, a stejně je to i se zbylými vrcholy $3, 5$. Výsledný isomorfismus vypadá v odpovídajícím značení vrcholů takto:

□

Abychom mohli s isomorfismem grafů přirozeně pracovat, je potřeba následující fakt:

Věta 9.8. *Relace „být isomorfní“ $\simeq$ na třídě všech grafů je ekvivalencí.*

`Důkaz`. Relace $\simeq$ je reflexivní, protože graf je isomorfní sám sobě identickým zobrazením. Relace je také symetrická, neboť bijektivní zobrazení lze jednoznačně obrátit. Tranzitivita $\simeq$ se snadno dokáže skládáním zobrazení–isomorfismů. □

Důsledkem je, že všechny možné grafy se rozpadnou na *třídy isomorfismu*. V praxi pak, pokud mluvíme o *grafu*, myslíme tím obvykle jeho *celou třídu isomorfismu*, tj. nezáleží nám na konkrétní prezentaci grafu.

`Komentář:` Je uvedený přístup, tj. zaměňování konkrétního grafu za celou jeho třídu isomorfismu, v matematice neobvyklý? Ne, například už v geometrii jste říkali „čtverec o straně 2“ či „jednotkový kruh“ a podobně, aniž jste měli na mysli konkrétní obrázek, nýbrž celou třídu všech těchto shodných objektů.

Další (pod)grafové pojmy

`Definice`: Mějme libovolný graf G.

- Podgrafu $H \subseteq G$, který je isomorfní nějaké kružnici, říkáme *kružnice v G*.

* Speciálně říkáme *trojúhelník* kružnici délky 3.
* Podgrafu $H \subseteq G$, který je isomorfní nějaké cestě, říkáme *cesta v G*.
* Podgrafu $H \subseteq G$, který je isomorfní nějakému úplnému grafu, říkáme *klika v G*. (Někdy se za kliku považuje pouze takový úplný podgraf, který je maximální vzhledem k uspořádání inkluzí.)
* Podmnožině vrcholů $X \subseteq V(G)$, mezi kterými nevedou v G vůbec žádné hrany, říkáme *nezávislá množina X v G*.
* Indukovanému podgrafu $H \subseteq G$, který je isomorfní nějaké kružnici, říkáme *indukovaná kružnice v G*.

`Komentář:` Uvažujme následující ukázky grafů:

První z ukázaných grafů například neobsahuje žádný trojúhelník, ale obsahuje kružnici délky 4, dokonce indukovanou. Druhý graf trojúhelník obsahuje a kružnici délky 4 taktéž. První graf obsahuje cestu délky 4 na vrcholech $1, 2, 3, 4, 5$, ale ta není indukovaná. Indukovaná cesta délky 4 v něm je třeba $2, 3, 4, 5, 6$. Druhý graf tyto cesty také obsahuje, ale naopak žádná z nich není indukovaná. První graf má největší kliku velikosti 2 – jedinou hranu, kdežto druhý graf má větší kliku na vrcholech $3, 4, 5$. Největší nezávislá množina u obou grafů má 3 vrcholy $2, 4, 6$.

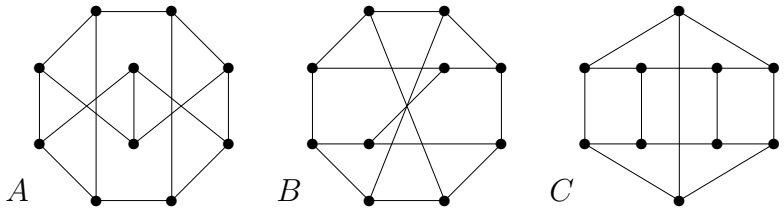
Jak poznat neisomorfní grafy

`Fakt`: Mějme isomorfismus f grafů G a H. Pokud G obsahuje podgraf F, pak H také musí obsahovat podgraf isomorfní F. Obecněji lze tvrdit, že počet podgrafů v grafu G isomorfních zvolenému F je vždy roven takovému počtu v grafu H.

Příklad 9.9. *Jsou následující dva grafy isomorfní?*

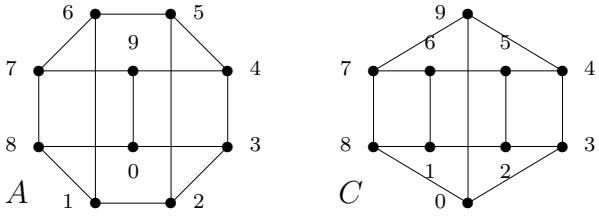
Postupovat budeme jako v Příkladě 9.7, nejprve ověříme, že oba grafy mají stejně mnoho vrcholů i stejnou posloupnost stupňů $2, 2, 2, 2, 3, 3$. Pokud se však budeme snažit najít mezi nimi isomorfismus, něco stále nebude vycházet... Co nám tedy v nalezení isomorfismu brání? Podívejme se, že v druhém grafu oba vrcholy stupně tři mají svého společného souseda, tvoří s ním trojúhelník. V prvním grafu tomu tak není, první graf dokonce nemá žádný trojúhelník. Proto zadané dva grafy nejsou isomorfní. □

Příklad 9.10. *Najděte všechny isomorfní dvojice grafů v následujících obrázcích tří 10-vrcholových grafů. Isomorfní dvojice odpovídajícím způsobem očíslujte, u neisomorfních toto zdůvodněte.*



Postupujme systematicky: Všechny tři grafy mají po 10 vrcholech a všechny vrcholy stupňů 3. Takto jsme je tedy nijak nerozlišili. Podívejme se třeba na trojúhelníky v grafech – opět si nepomůžeme, neboť žádný z nich trojúhelníky neobsahuje. Co se tedy podívat na obsažené kružnice délky 4? Graf C jich jasně obsahuje pět, graf A po chvíli zkoumání také, ale v grafu B najdeme i při vší snaze jen tři kružnice délky 4. (Obdobného rozdílu si můžeme všimnout, pokud se zaměříme na kružnice C_5, zkuste si to sami.)

Takže co z dosavadního zkoumání plyne? Graf B nemůže být isomorfní žádnému z A, C. Nyní tedy zbývá najít (očekávaný) isomorfismus mezi grafy A a C. To se nám skutečně podaří poměrně snadno - stačí „prohozením“ prostředních dvou vrcholů u grafu A získat lepší obrázek



a odpovídající bijekce je na pohled zřejmá. Pro větší názornost jsme bijekci potvrzující isomorfismus zakreslených grafů explicitně vyznačili odpovídajícím číslováním vrcholů. □

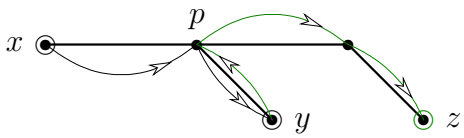
`Poznámka`: Výše uvedené příklady nám ukazují některé cesty, jak poznat (tj. najít nebo vyloučit) isomorfismus dvou grafů. Ty však ne vždy musí fungovat. Čtenář se může ptát, kde tedy najde nějaký univerzální postup pro nalezení isomorfismu? Bohužel vás musíme zklamat, žádný rozumný univerzální postup není znám a zatím platí, že jediná vždy fungující cesta pro nalezení či nenalezení isomorfismu mezi dvěma grafy je ve stylu *vyzkoušejte všechny možnosti* bijekcí mezi vrcholy těchto grafů. (Těch je, jak známo, až $n!$.) Avšak heuristické algoritmické přístupy pracující se stupni vrcholů samotných a jejich sousedů a případně s malými podgrafy jsou v praxi velmi efektivní pro rozhodování isomorfismu.

9.3 Souvislost, komponenty a vzdálenost

Důležitou globální vlastností grafů je *souvislost*, tedy možnost se v nich pohybovat odkudkoliv kamkoliv podél jeho hran, neboli po cestách v grafu. Tuto vlastnost si nyní upřesníme.

Tvrzení 9.11. *Mějme relaci $\sim$ na množině vrcholů $V(G)$ libovolného grafu G takovou, že pro dva vrcholy $x \sim y$ právě když existuje v G cesta začínající v x a končící v y. Pak $\sim$ je relací ekvivalence.*

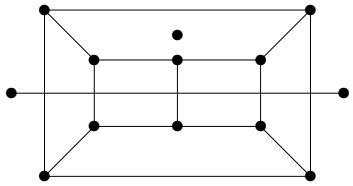
`Důkaz`. Relace $\sim$ je reflexivní, neboť každý vrchol je spojený sám se sebou cestou délky 0. Symetrická je také, protože cestu z x do y snadno v neorientovaném grafu obrátíme na cestu z y do x. Důkaz tranzitivity však není takto triviální—pokud vezmeme cestu z x do y a cestu z y do z, tak se tyto dvě cesty mohou protínat i jinde než v y a nelze je prostě „navázat“ na sebe.



Pro důkaz tranzitivity si označme P cestu z x do y a Q cestu z y do z. Pokud označíme $P' \subseteq P$ tu část první cesty z x do prvního vrcholu p v průniku s Q (tj. $p \in V(P) \cap V(Q)$) a označíme $Q' \subseteq Q$ zbytek druhé cesty od p do z, tak $P' \cup Q'$ vždy je cestou z x do z. □

Definice 9.12. ***Komponentami souvislosti*** grafu G nazveme třídy ekvivalence výše popsané (Tvrz. 9.11) relace $\sim$ na $V(G)$. Jinak se také *komponentami souvislosti myslí podgrafy* indukované na těchto třídách ekvivalence.

`Komentář:` Podívejte se, kolik komponent souvislosti má tento graf:

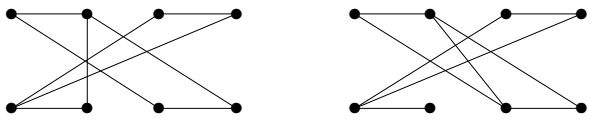


Vidíte v obrázku všechny tři komponenty? Jedna z nich je izolovaným vrcholem, druhá hranou (tj. grafem isomorfním K_2) a třetí je to zbývající.

`Definice`: Graf G je *souvislý* pokud je G tvořený nejvýše jednou komponentou souvislosti, tj. pokud každé dva vrcholy G jsou *spojené cestou*.

`Poznámka`: Prázdný graf je souvislý a má 0 komponent.

`Komentář:` Který z těchto dvou grafů je souvislý?



Příklad 9.13. *Dokažme si, že každý souvislý jednoduchý 2-regulární graf G je kružnicí (tj. isomorfní některé kružnici C_n z Oddílu 9.1).*

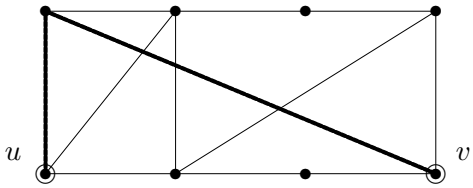
Nechť e je hranou mezi vrcholy u, v grafu G a vezměme graf $G' = G - e$ vzniklý odebráním hrany e. Pokud by u a v náležely v G' různým komponentám souvislosti, tyto komponenty by každá měla lichý součet stupňů, což nelze podle Věty 9.3. Proto u a v jsou spojeny cestou v G', nechť vrcholy této cesty jsou po řadě značeny $u_1 = u, u_2, \ldots, u_k = v$. Toto značení nám nyní udává isomorfismus zobrazující vrchol u_i na vrchol i z definice kružnice C_k. Nyní už zbývá jen drobnost; dokázat, že jiné vrcholy než $u_1, \ldots, u_k$ v grafu G nejsou. Pokud bychom měli další vrchol x, pak x je spojen cestou Q do u a první vrchol z $V(Q) \cap \{u_1, \ldots, u_k\}$ by měl stupeň větší než 2, spor. □

Grafová vzdálenost

Mimo předchozí otázku souvislosti grafu, tj. existence jakékoliv cesty mezi dvojicemi vrcholů, je častou úlohou se dostat z místa na místo tou nejkratší cestou. Ve zjednodušeném podání nás zajímá jen počet hran, neboli považujeme každou hranu grafu za jednotkově dlouhou, a definujeme následující.

Definice 9.14. ***Vzdálenost*** $d_G(u,v)$ dvou vrcholů u, v v grafu G je dána délkou nejkratší cesty mezi u a v v G. Pokud cesta mezi u, v neexistuje, je vzdálenost definována $d_G(u,v) = \infty$.

`Komentář:` Neformálně řečeno, vzdálenost mezi u, v je rovna *nejmenšímu počtu hran*, které musíme překonat, pokud se chceme dostat z u do v. Speciálně vždy platí $d_G(u,u) = 0$ a dále $d_G(u,v) = \infty$ právě když u, v patří různým komponentám souvislosti.



Jaká je ve výše zakresleném grafu vzdálenost $d_G(u,v)$? Ano, snadno určíme hodnotu $d_G(u,v) = 2$, viz vyznačená cesta délky 2.
Najdete ale v témže grafu nějakou dvojici vrcholů se vzdáleností 3?

`Fakt`: V neorientovaném grafu je vzdálenost symetrická, tj. $d_G(u,v) = d_G(v,u)$.

Lema 9.15. *Vzdálenost v grafech splňuje trojúhelníkovou nerovnost:*

$$\forall u, v, w \in V(G) \;:\; d_G(u,v) + d_G(v,w) \geq d_G(u,w)\,.$$

`Důkaz`. Postupujeme podobně jako v důkaze Tvrzení 9.11 – pokud máme cesty P, P' mezi u, v a mezi v, w, tak existuje cesta $Q \subseteq P \cup P'$ mezi u, w, jež má zřejmě délku nejvýše $d_G(u,v) + d_G(v,w)$. Skutečná vzdálenost mezi u, w pak už může být jen menší. □

Příklad 9.16. *Proč každý 3-regulární graf na 12 vrcholech musí obsahovat dvojici vrcholů se vzdáleností 3?*

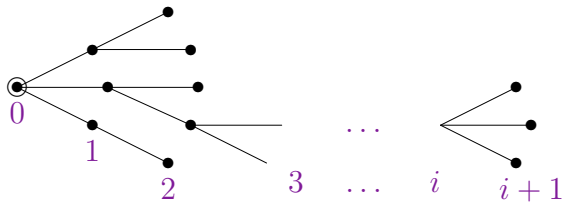
Dokažme požadované tvrzení sporem. Nechť v je libovolný vrchol našeho grafu G. Pak v má tři sousedy a, b, c (tj. ve vzdálenosti 1 od v). Podle definice vzdálenosti je každý vrchol ve vzdálenosti 2 od v sousedem některého z a, b, c. Avšak a má za jednoho souseda v a mimo něj má už jen dva další sousedy. Stejně tak b, c. Tudíž v G existuje nejvýše šest vrcholů mimo v, kteří jsou sousedé jednoho z a, b, c. Celkem tak napočítáme (od v) jen $1+3+6 = 10$ vrcholů a některý z 12 vrcholů grafu G musí být ve větší vzdálenosti než 2, spor. Zakreslete si sami tuto úvahu obrázkem. □

Jednoduché zjištění vzdálenosti

Jak nejsnadněji určíme vzdálenost v grafu? Jednoduše postupujeme od prvního z vrcholů „do šířky“ (jakoby v soustředných kruzích) a počítáme si kroky, až narazíme na druhý z vrcholů. Formalizovat tento postup můžeme následovně (a jak se později dozvíte, jedná se o obecné schéma prohledávání grafu do šířky).

Algoritmus 9.17. ***Určení vzdáleností z vrcholu u grafu G.***
Pro daný souvislý graf G a jeho vrchol u určíme vzdálenosti $d(u,x)$ do každého vrcholu $x \in V(G)$ následujícím postupem.

1. Na začátku položíme $d(u,u) := 0$.
2. Pro $i = 0, 1, 2, \ldots$, přesněji dokud nejsou určeny všechny vzdálenosti, provádíme:
   Pro každou hranu $xy \in E(G)$ takovou, že $d(u,x) = i$ a $d(u,y)$ je dosud neurčená, položíme $d(u,y) := i + 1$.



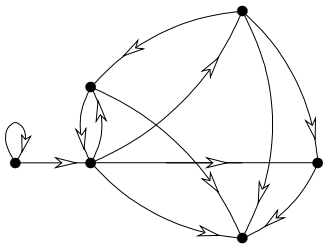
`Důkaz` správnosti algoritmu: Cílem je dokázat, že pro hodnoty určené Algoritmem 9.17 platí $d(u,v) = d_G(u,v)$ podle Definice 9.14. Předně si uvědomíme, že v souvislém konečném grafu je vzdálenost vždy konečná, neboli existuje přirozené i takové, že $d_G(u,v) = i$. Proto stačí dokázat indukcí podle $i \geq 0$, že pro každý vrchol x takový, že $d_G(u,x) \leq i$, skutečně platí $d(u,x) = d_G(u,x)$.

Pro bázi indukce, tj. pro $i = 0$ a $x = u$ to je zřejmé. Nechť nyní naše tvrzení platí pro nějaké přirozené i a vezměme libovolný vrchol y takový, že $d_G(u,y) \leq i + 1$. Pokud $d_G(u,y) \leq i$, jsme hotovi podle indukčního předpokladu. Jinak $d_G(u,y) = i+1$ a existuje cesta $P \subseteq G$ s konci u a y délky $i+1$. Nechť x je sousedem y na cestě P. Pak $d_G(u,x) = i$ podle definice vzdálenosti a tudíž $d(u,x) = i$ podle indukčního předpokladu. Tudíž v našem algoritmu určíme $d(u,y) := i + 1 = d_G(u,y)$. □

`Poznámka`: Vedle jednotkové grafové vzdálenosti podle Definice 9.14 nás často v praktických aplikacích zajímá zobecněné pojetí, ve kterém nemají všechny hrany stejnou délku. Formálně tak můžeme studovat obecnou vzdálenost v grafech, jejichž hrany jsou váženy nezápornými reálnými čísly (*délkami*). K výpočtu takto zobecněné vzdálenosti už nestačí jednoduchý postup Algoritmu 9.17, ale obracíme se k trochu sofistikovanějším algoritmům jako například k Dijkstrově algoritmu. To je už téma nad rámec tohoto matematického předmětu.

9.4 Základní pojmy orientovaných grafů

Ve většině aplikací grafů se v základu soustředíme na neorientované grafy, ale pro některé úlohy je vhodné či přímo nezbytné vědět, kterým „směrem“ hrana grafu vede. Požadavek explicitně vyjádřit *směr hrany* přirozeně vede na následující definici orientovaného grafu, ve kterém hrany jsou *uspořádané* dvojice vrcholů. (V obrázcích kreslíme orientované hrany se šipkami.)

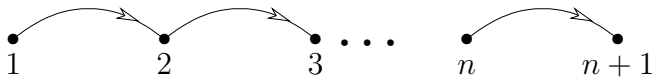


Definice 9.18. ***Orientovaný graf*** je uspořádaná dvojice $D = (V, E)$, kde V je množina vrcholů a $E \subseteq V \times V$ je množina hran – tj. podmnožina uspořádaných dvojic vrcholů.
Pojmy *podgrafu* a *isomorfismu* z Oddílu 9.2 se přirozeně přenášejí na orientované grafy.

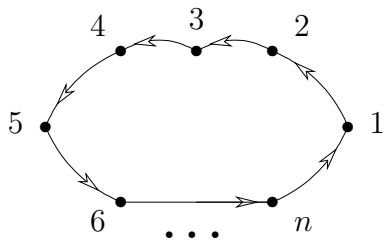
`Značení`: Hrana (u, v) (zvaná také *šipka*) v orientovaném grafu D *začíná* ve vrcholu u a *končí* ve (míří do) vrcholu v. Opačná hrana (v, u) je různá od (u, v).
Speciálně hrana tvaru (u, u) se nazývá *orientovaná smyčka*.

`Komentář:` Orientované grafy odpovídají relacím, které nemusí být symetrické. Mezi stejnou dvojicí vrcholů mohou tudíž existovat dvě hrany v obou směrech, nebo jen kterákoliv jedna z nich nebo žádná. Všimněte si také, že orientované grafy implicitně odlišujeme od obyčejných použitím písmene D místo dřívějších G, H.

Například *orientovaná cesta* délky $n \geq 0$ je následujícím grafem na $n + 1$ vrcholech



a *orientovaná kružnice* (také *cyklus*) délky $n \geq 1$ vypadá takto:



`Definice`: Počet hran začínajících ve vrcholu u orientovaného grafu D nazveme *výstupním stupněm* $d_D^{out}(u)$ a počet hran končících v u nazveme *vstupním stupněm* $d_D^{in}(u)$.

`Komentář:` Součet všech výstupních stupňů je přirozeně roven součtu všech vstupních stupňů orientovaného grafu. Důkaz viz Věta 9.3.

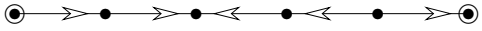
`Definice`: *Symetrizací* orientovaného grafu D rozumíme neorientovaný graf G vzniklý „zapomenutím směru hran“ v D, přesněji $V(G) = V(D)$ a $uv \in E(G)$ právě když $\{(u, v), (v, u)\} \cap E(D) \neq \emptyset$.

Souvislost na orientovaných grafech

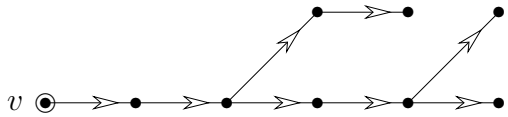
Pojem orientované souvislosti grafu D je natolik fundamentálně odlišný od neorientovaného případu (což je dáno právě jeho „směrovostí“), že si zaslouží samostatnou diskusi i v našem zběžném pohledu na orientované grafy. Uvedeme si odstupňovaně tři základní pohledy na orientovanou souvislost:

- *Slabá souvislost.* Jedná se o tradiční *souvislost na symetrizaci* grafu D

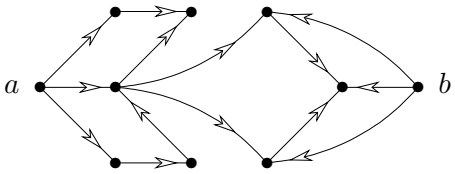
`Komentář:` Zjednodušeně a názorně se dá říci, že při cestování grafem „zapomeneme“ směr šipek. Na obrázku:



- *Dosažitelnost (směrem „ven“).* Orientovaný graf D je *dosažitelný směrem ven*, pokud v něm existuje vrchol $v \in V(D)$ takový, že každý vrchol $x \in V(D)$ je dosažitelný orientovanou cestou z v.

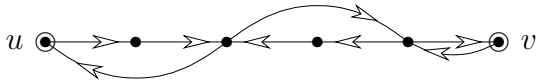


`Komentář:` Podrobným zkoumáním následujícího obrázku zjistíme, že jeho graf není dosažitelný směrem ven, neboť chybí možnost dosáhnout vrchol b úplně vpravo. Na druhou stranu po vypuštění b je zbylý graf dosažitelný ven z vrcholu a vlevo.



Všimněte si, že definici dosažitelnosti lze snadno „obrátit“, avšak touto další možností se v našem krátkém přehledu nebudeme zabývat.

- *Silná souvislost.* V nejsilnější verzi vyžadujeme současnou existenci spojení (cest) v obou směrech mezi dvojicí vrcholů.

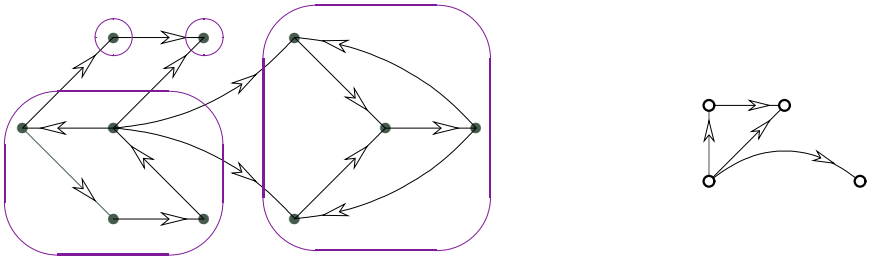


Tvrzení 9.19. *Nechť $\approx$ je binární relace na vrcholové množině $V(D)$ orientovaného grafu D taková, že $u \approx v$ právě když existuje dvojice orientovaných cest – jedna z u do v a druhá z v do u v grafu D. Pak $\approx$ je relace ekvivalence.*

`Důkaz` (náznak). Postupujeme podobně jako v důkaze Tvrzení 9.11. □

Definice 9.20. ***Silné komponenty*** orientovaného grafu D jsou třídy ekvivalence relace $\approx$ uvedené v Tvrzení 9.19. Orientovaný graf D je *silně souvislý* pokud má nejvýše jednu silnou komponentu.

`Komentář:` Pro ilustraci si mírně upravíme dříve prezentovaný orientovaný graf tak, že bude dosažitelný z nejlevějšího vrcholu. Je výsledek silně souvislý?



Ne, na obrázku jsou vyznačené jeho 4 silné komponenty. Vpravo zároveň uvádíme pro ilustraci obrázek *kondenzace* silných komponent tohoto grafu, což je acyklický orientovaný graf s vrcholy reprezentujícími zmíněné silné komponenty a směry hran mezi nimi.

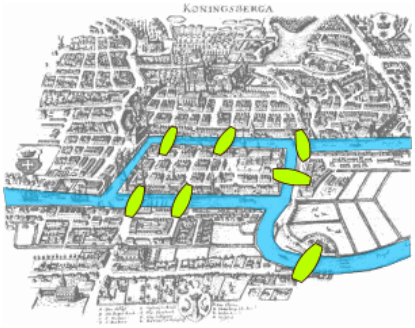
Pro zvídavé čtenáře poznamenáváme, že si mohou definici silné komponenty porovnat s jádrem předuspořádání v Oddíle 7.3. Jde o hodně podobné koncepty, ale vidíte i jeden drobný (skutečně ne až tak podstatný) rozdíl?

9.5 Dodatek: 7 mostů jedním tahem

Pravděpodobně *nejstarší zaznamenaný výsledek teorie grafů* pochází od Leonharda Eulera – jedná se o slavný problém 7-mi mostů v Královci / Königsbergu / dnešním Kaliningradě. Tento problém, či spíše matematickou hříčku, o kreslení grafů „jedním tahem“ zařazujeme na závěr lekce především z důvodů historických. Eulerovo jednoduché řešení má některé zajímavé a užitečné důsledky v jiných oblastech kombinatoriky, ty však přesahují rámec našeho úvodního textu.

O jaký problém se 7-mi mosty se tehdy v Königsbergu 18-tého století jednalo?

Příklad 9.21. *Je možné při jedné procházce suchou nohou přejít po každém ze sedmi vyznačených mostů v Königsbergu právě jednou?*



Geniálně jednoduché řešení otázky představíme ve zbytku tohoto oddílu. □

Sled a tah v grafu

Pro přesnou formulaci řešení uvedeného příkladu využijeme následující nové pojmy.

`Definice`: *Sledem* délky n v grafu G rozumíme posloupnost vrcholů a hran

$$(v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n)\,,$$

ve které vždy hrana e_i má koncové vrcholy v_{i-1}, v_i.

> `Komentář:` Všimněte si, že sled je vlastně jakákoliv procházka po hranách grafu z u do v. Příkladem sledu může být průchod IP paketu internetem (včetně cyklení).

`Definice`: *Tah* je sled v grafu bez opakování hran. *Uzavřený tah* je tahem, který končí ve vrcholu, ve kterém začal. *Otevřený tah* je tahem, který končí v jiném vrcholu, než ve kterém začal.

> `Komentář:` Jistě znáte dětskou hříčku s „kreslením domečku jedním tahem“... Ano, to je v podstatě totéž, co *tah* v grafu (kterým „kreslíme“ hrany našeho grafu).

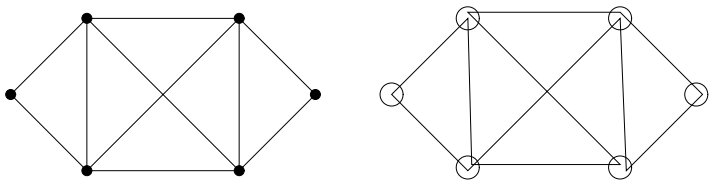
`Fakt`: Cesta je přesně otevřený tah bez opakování vrcholů. Kružnice je přesně uzavřený tah bez opakování vrcholů. (Srovnejte si toto s definicemi Oddílu 9.1.)

Eulerovské grafy

Slibované řešení Příkladu 9.21 od Leonharda Eulera zní takto:

Věta 9.22. *Graf G lze pokrýt (nakreslit) jedním uzavřeným tahem právě když G je souvislý a všechny vrcholy v G jsou sudého stupně.*

`Komentář:` Znění této věty lze velmi názorně ilustrovat následujícím obrázkem grafu a příslušného uzavřeného tahu pokrývajícího všechny hrany.



A jak je tomu v Příkladu 9.21? Zde nejprve nakreslíme příslušný (multi)graf, ve kterém vrcholy jsou jednotlivé kusy země oddělené vodou (tj. dva říční ostrovy a dva břehy):

Jaké jsou stupně vrcholů tohoto grafy? Je to $3, 3, 3, 5$, neboli všech 7 hran–mostů města Königsbergu nelze dle Věty 9.22 pokrýt jedním uzavřeným tahem (ani otevřeným tahem, viz Důsledek 9.23).

Důsledek 9.23. *Graf G lze pokrýt (nakreslit) jedním otevřeným tahem právě když G je souvislý a všechny vrcholy v G až na dva jsou sudého stupně.*

`Důkaz`: Dokazujeme oba směry ekvivalence Věty 9.22. Pokud lze G pokrýt jedním uzavřeným tahem, tak je zřejmě G souvislý a navíc má každý stupeň sudý, neboť uzavřený tah každým průchodem vrcholem pokryje dvě hrany.

Naopak zvolíme mezi všemi uzavřenými tahy T obsaženými v G ten (jeden z) nejdelší. Tvrdíme, že T obsahuje všechny hrany grafu G.

- Pro spor vezměme graf $G' = G - E(T)$, o kterém předpokládejme, že je neprázdný. Jelikož G' má taktéž všechny stupně sudé, je (z indukčního předpokladu) libovolná jeho hranově-neprázdná komponenta $C \subseteq G'$ pokrytá jedním uzavřeným tahem T_C.
- Vzhledem k souvislosti grafu G každá komponenta $C \subseteq G'$ protíná náš tah T v některém vrchole w, a tudíž lze oba tahy T_C a T „propojit přes w“. To je spor s naším předpokladem nejdelšího možného T, neboť $T \cup T_C$ je delším tahem v G. □

`Důkaz` důsledku: Nechť u, v jsou dva vrcholy grafu G mající lichý stupeň, neboli dva (předpokládané) konce otevřeného tahu pro G. Do G nyní přidáme nový vrchol w spojený hranami s u a v. Tím jsme náš případ převedli na předchozí případ grafu se všemi sudými stupni. □

Rozšiřující studium

Grafy můžeme v informatice potkat doslova na každém kroku, mimo jiné hojně už v základních kurzech algoritmizace. Nejen že grafy jsou základem mnoha programátorských datových struktur, ale představují i vhodný model pro mnoho praktických problémů, z nichž některé ochutnáte již v příští lekci. Celkově je zdejších pár lekcí teorie grafů jen lehkým

úvodem do celé rozsáhlé oblasti, přičemž na FI MU lze pokračovat v jejím cíleném studiu v předmětech MA010 a MA015.

Rozsáhlý matematický úvod do teorie grafů je zahrnut ve skvělé knize Kapitoly z diskrétní matematiky autorů Jiřího Matouška a Jaroslava Nešetřila. Vřele ji doporučujeme jako doplňkový studijní zdroj všem, kteří chtějí lépe pochopit grafy z jejich matematické stránky.

10 Stromy a kostry grafů

Úvod

Na rozdíl od předchozí lekce, která se zabývala grafy z obecného a také trochu povrchního pohledu, se nyní soustředíme na jednu konkrétní nepříliš obtížnou oblast, které se budeme věnovat do větší hloubky. Jde o problematiku stromů, neboli acyklických souvislých grafů, představujících nejjednodušší podobu grafů. Na stromech si budeme ilustrovat argumentaci a důkazy o grafech a také jeden z historicky základních diskrétních optimalizačních problémů – problém minimální kostry (tj. minimálně ohodnoceného stromu na dané množině vrcholů).

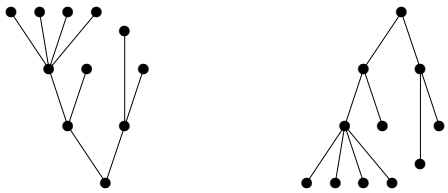
Cíle

Rozebereme si různé charakteristické vlastnosti stromů a rozebereme si důkazy těchto vlastností. Zmíníme také kořenové stromy. Poté si definujeme problém minimální kostry a ukážeme jeho efektivní řešení.

10.1 Stromy – grafy bez kružnic

Podrobné studium některých užitečných vlastností grafů si pro zjednodušení ukážeme na tom prakticky nejjednodušším typu grafu – na *stromech*, jež jsou mimo jiné základem mnoha datových typů používaných v informatice.

`Komentář:` Začněme ilustračními obrázky stromů. Povšimněte si přitom jedné zvláštnosti, totiž že v informatice stromy typicky rostou „shora dolů“...



Definice 10.1. ***Strom*** je (jednoduchý) souvislý graf T bez kružnic.

`Komentář:` Obecněji *les* je pak graf bez kružnic (opět jednoduchý, ale nemusí být souvislý). Komponenty souvislosti lesa jsou stromy. Jeden vrchol bez hran a prázdný graf jsou také stromy. Grafy bez kružnic také obecně nazýváme *acyklické*.

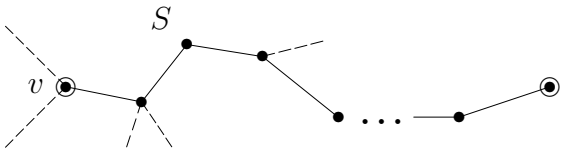
Vlastnosti stromů

Přehled základních vlastností stromů je pro nás zároveň příležitostí si ukázat několik nových hezkých matematických důkazů a naučit se správně zdůvodňovat v oblasti grafů.

Tvrzení 10.2. *Strom s více než jedním vrcholem obsahuje vrchol stupně* 1.

`Důkaz`: Vezmeme libovolný strom T a v něm libovolný vrchol v. Jelikož souvislý graf s více než jedním vrcholem nemá vrchol stupně 0, zvolený vrchol v má incidentní hranu. Zvolme (sestrojme) nyní nejdelší možnou cestu S v T začínající ve v: S začne libovolnou hranou vycházející z v; v každém dalším vrcholu u cesty S, který má stupeň větší než 1, pokračuje S další hranou. Pokud by tomu tak nebylo, není S nejdelší možná nebo další

hrana vede do některého předchozího vrcholu cesty S. Tím bychom ale získali *kružnici*, což ve stromě nelze.



Proto cesta S nutně skončí v nějakém vrcholu stupně 1 v T. □

`Komentář:` Uvedený důkaz lze zapsat výrazně stručněji (ale poněkud tím utrpí jeho pochopitelnost): Rovnou na začátku lze zvolit S jako libovolnou nejdelší možnou cestu ve stromě T a podívat se, kam by případně vedly další hrany z koncového vrcholu S.

Zamyslete se navíc, proč v každém stromě s více než jedním vrcholem jsou alespoň dva vrcholy stupně 1 (odpověď je skrytá už v předchozím důkaze). Zároveň si odpovězte, jestli lze tvrdit, že každý strom s více než jedním vrcholem obsahuje tři vrcholy stupně 1.

`Definice`: Každý jeho vrchol stromu stupně 1 nazveme *listem* stromu.

Věta 10.3. *Strom na n vrcholech má přesně $n-1$ hran pro $n \geq 1$.*

`Důkaz`: Toto tvrzení dokážeme indukcí podle n. Strom s jedním vrcholem má přesně $0 = n - 1$ hran.

Předpokládejme platnost tvrzení pro libovolné přirozené $n := i \geq 1$. Nechť T je nyní libovolný strom na $n := i+1$ vrcholech. Podle Tvrzení 10.2 obsahuje T vrchol v stupně 1. Označme $T' = T - v$ graf vzniklý z T odebráním vrcholu v a jedné jeho hrany. Pak T' je také souvislý graf bez kružnic, a tudíž strom na $n - 1 = i$ vrcholech. Dle indukčního předpokladu T' má $i - 1 = n - 2$ hran, a proto T má $n - 2 + 1 = n - 1$ hran. □

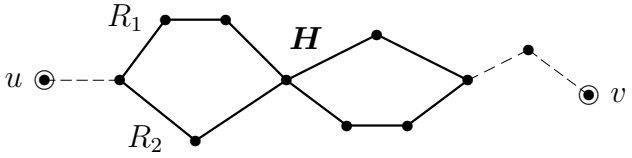
Příklad 10.4. *Les G má 2015 vrcholů a 4 souvislé komponenty. Kolik má hran?*

Pokud počty vrcholů jednotlivých komponent lesa G po řadě označíme n_1, n_2, n_3, n_4, tak máme $n_1 + n_2 + n_3 + n_4 = 2015$. Každá komponenta G je strom podle definice lesa. Zároveň podle Věty 10.3 má i-tá komponenta přesně $n_i - 1$ hran. V součtu má G celkem $(n_1 - 1) + (n_2 - 1) + (n_3 - 1) + (n_4 - 1) = 2015 - 4 = 2011$ hran. □

Cesty ve stromech

Věta 10.5. *Mezi každými dvěma vrcholy stromu vede právě jediná cesta.*

`Důkaz`:



Jelikož strom T je souvislý dle definice, mezi libovolnými dvěma vrcholy u, v vede nějaká cesta. Pokud by existovaly dvě různé cesty R_1, R_2 mezi u a v, tak bychom vzali jejich symetrický rozdíl, podgraf $H = R_1 \Delta R_2$ s neprázdnou množinou hran, kde H zřejmě má všechny stupně sudé. Na druhou stranu se však podgraf stromu musí opět skládat z komponent stromů, a tudíž obsahovat vrchol stupně 1 podle Tvrzení 10.2, což je spor. Proto cesta mezi u a v existuje jen jedna. □

`Komentář:` Uvedený důkaz Věty 10.5 může působit poněkud „tajemně a neprůhledně“ (ve srovnání s předchozími přímočařejšími důkazy). Vždyť to přeci vypadá docela jasně, že ze dvou cest mezi u a v složíme dohromady nějakou(?) kružnici. Avšak po hlubším zamyšlení můžete sami zjistit, že správně zapsat, kde ta kružnice v $R_1 \cup R_2$ je, není vůbec lehké a vedlo by to k delšímu a ne moc průhlednějšímu důkazu.

Důsledek 10.6. *Přidáním jedné nové hrany do stromu vznikne právě jedna kružnice.*

`Důkaz`: Nechť mezi vrcholy u, v ve stromu T není hrana. Přidáním hrany $e = uv$ vznikne právě jedna kružnice z e a jediné cesty mezi u, v v T podle Věty 10.5. □

Alternativní charakterizace stromů

Z předchozích tvrzení vyplývá následující alternativní charakterizace stromů, která ukazuje důležitost jich samotných i jako tzv. *koster* obecných grafů (viz Oddíl 10.2).

Na dané množině vrcholů je (vzhledem k inkluzi množin hran) *strom*

- *minimální souvislý* graf (plyne z Věty 10.5)
- a zároveň *maximální acyklický* graf (plyne z Důsledku 10.6).

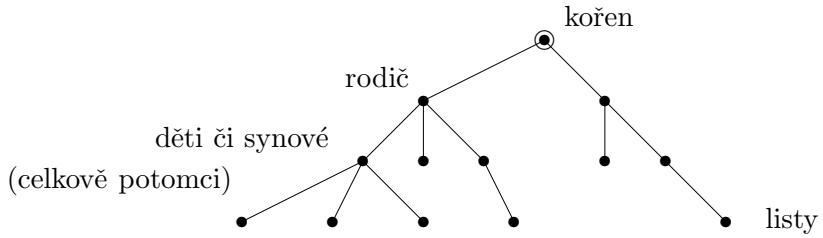
`Komentář:` Jen tak mimochodem, kolik dokážete nalézt neisomorfních stromů na 4 nebo 5 vrcholech? Vidíte, že jich není mnoho? Nakreslete si je všechny.

Kořenový strom

Mimo samotnou teorii grafů se častěji setkáte s poněkud odlišným pojetím stromu, ve kterém má význačné místo jeden speciálně vybraný vrchol stromu, zvaný *kořen* (viz třeba mnohé datové struktury v algoritmech).

`Definice`: Strom T s vyznačeným vrcholem $r \in V(T)$, zkráceně zapsaný dvojicí (T, r), nazýváme *kořenovým stromem* s *kořenem* r. Vrchol p nazveme *potomkem* vrcholu q, pokud (ta jediná) cesta z kořene r do p obsahuje (neboli vede přes) q. V kořenovém stromu nazveme *listem* každý vrchol, který nemá potomky.

`Komentář:` Každý vrchol kořenového stromu je *potomkem* kořene. V přirozeně přeneseném významu se u kořenových stromů používají pojmy rodič, děti/synové, sourozenci, předchůdce, následník, atd. Zběžná ilustrace použití těchto pojmů je na následujícím schématu stromu.



`Definice`: *Výška kořenového stromu* je rovna největší vzdálenosti z jeho kořene do některého listu.

`Poznámka`: Všimněte si, že význam pojmu „list“ se může lišit mezi stromem a kořenovým stromem. V extrémním případě stromu sestávajícího pouze z kořene je tento kořen (stupně 0) zároveň listem, ale jinak každý list kořenového stromu má nutně stupeň 1. V opačném směru zase platí, že kořen stupně 1, který je nazýván listem v obyčejném stromu, není listem příslušného kořenového stromu (má přece jednoho potomka).

10.2 Problém minimální kostry

V návaznosti na učivo o stromech se podíváme na tradiční a široce studovaný problém nalezení minimálního souvislého podgrafu (stromu) v daném grafu – této úloze se říká *minimální kostra* neboli MST (z anglického minimum spanning tree). V jakém smyslu však chápat slovo „minimální“ u kostry? Na jedné straně to odkazuje na fakt, že kostra souvislého grafu je vždy strom, coby inkluzí minimální souvislý graf na daných vrcholech podle učiva Oddílu 10.1. Přesně definováno:

`Definice`: Podgraf $T \subseteq G$ souvislého grafu G se nazývá *kostrou*, pokud

* T je stromem a
* $V(T) = V(G)$, neboli T *propojuje všechny vrcholy* G.

Na druhou stranu navíc požadujeme, že nalezená kostra má mít v součtu co nejmenší *celkovou délku*. Co však toto znamená? Podle Věty 10.3 má každý strom na daných n vrcholech stejně hran, přesně $n-1$. Tudíž aby úloha minimalizace délky (či váhy) kostry vůbec dávala smysl, budeme se věnovat grafům s „obecně dlouhými“ hranami:

`Definice`: Vážený graf je graf G spolu s ohodnocením w hran reálnými čísly $w : E(G) \to \mathbb{R}$. Váženému také někdy říkáme *ohodnocený*.

Vahou (celkovou délkou) kostry $T \subseteq G$ váženého souvislého grafu (G, w) rozumíme

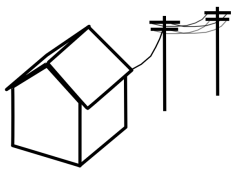
$$d_G^w(T) := \sum\nolimits_{e \in E(T)} w(e)\,,$$

tj. součet vah všech hran této kostry.

Definice 10.7. ***Problém minimální kostry (MST)*** ve váženém souvislém grafu (G, w) hledá kostru $T \subseteq G$ s nejmenší možnou vahou (přes všechny kostry grafu G).

`Komentář:` Problém minimální kostry je ve skutečnosti historicky úzce svázán s jižní Moravou a Brnem, konkrétně s elektrifikací jihomoravských vesnic ve dvacátých letech! Právě na základě tohoto praktického optimalizačního problému brněnský matematik Otakar Borůvka jako první v matematické literatuře zformuloval a podal řešení problému minimální kostry v roce 1926.

Ve výzkumu minimálních koster pokračoval i velmi dobře známý český matematik Vojtěch Jarník, s publikací v roce 1930 (viz Algoritmus 10.10). První ne-českou publikací na toto téma je pak až Kruskalův hladový algoritmus z roku 1956 (viz Algoritmus 10.8).



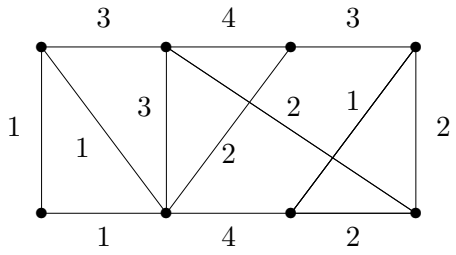
Řešení minimální kostry

Následující postup tzv. hladového nalezení minimální kostry pochází od Kruskala. Sice nejde ani o nejstarší publikovaný postup (viz Borůvka a Jarník výše), ani o nejvhodnější algoritmus k praktické implementaci, má svou hlubokou teoretickou hodnotu, a proto jej uvádíme první a podrobně.

Algoritmus 10.8. ***Hladový postup pro minimální kostru grafu*** (G, w)**.**
Mějme dán souvislý vážený graf G s ohodnocením hran w.

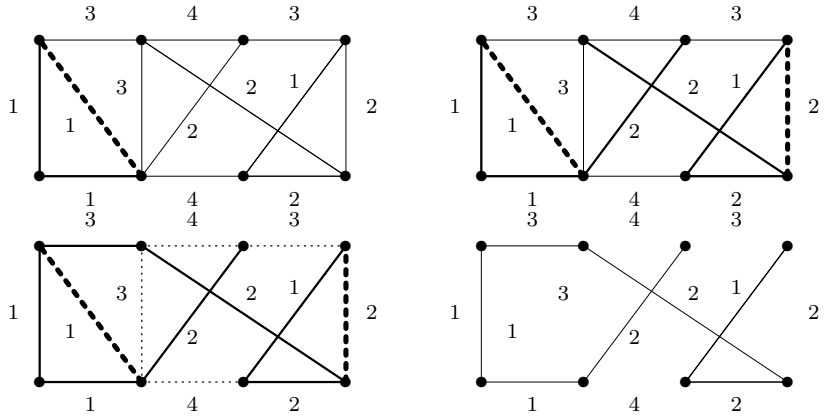
1. Seřadíme všechny hrany G jako $E(G) = (e_1, e_2, \ldots, e_m)$ tak, že $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$.
2. Inicializujeme prázdnou kostru $T = (V(G), \emptyset)$.
3. Po řadě pro $i = 1, 2, \ldots, m$ provedeme následující:
   - Pokud $T + e_i$ *nevytváří kružnici*, tak $E(T) \leftarrow E(T) \cup \{e_i\}$.
     (Neboli pokud e_i spojuje různé komponenty souvislosti dosavadního T.)
4. Na konci T obsahuje minimální kostru grafu G (případně jednu z několika takových).

`Komentář:` Pro ilustraci si ukážeme postup hladového algoritmu pro vyhledání kostry následujícího grafu:



Hrany si nejprve seřadíme podle jejich vah $1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4$.

V obrázku průběhu algoritmu používáme tlusté čáry pro vybrané hrany kostry a tečkované čáry pro „zahozené“ hrany. Hrany teď postupně přidáváme do kostry či zahazujeme...



Získáme tak minimální kostru velikosti $1 + 2 + 2 + 3 + 1 + 1 + 2 = 12$, která je v tomto případě (náhodou) cestou, na posledním obrázku vpravo.

Poznamenáváme, že při jiném seřazení hran stejné váhy by kostra mohla vyjít jinak, ale vždy bude mít stejnou váhu 12.

Věta 10.9. *Hladový postup korektně spočítá minimální kostru váženého grafu (G, w).*

`Důkaz` (náznak): Pro spor předpokládejme, že T_1 je kostra spočítaná Algoritmem 10.8 a T_2 nějaká minimální kostra, kde $d^w_G(T_2) < d^w_G(T_1)$ a rozdíl $|E(T_1)\Delta E(T_2)|$ je nejmenší. Nechť i je nejmenší index takový, že $e_i \in E(T_1)\Delta E(T_2)$. Pak nutně $e_i \in E(T_1) \setminus E(T_2)$ (proč?), a tudíž $T_2 + e_i$ podle Důsledku 10.6 obsahuje kružnici procházející také hranou e_j pro $j > i$. Potom však $T_3 = (T_2 + e_i) \setminus \{e_j\}$ je další kostrou mající váhu $d^w_G(T_2) + w(e_i) - w(e_j) \geq d^w_G(T_2)$, a proto $w(e_i) = w(e_j)$. Tudíž T_3 je minimální kostra „bližší“ T_1 ve smyslu symetrického rozdílu, což je spor s volbou T_2. □

Jarníkův (Primův) algoritmus

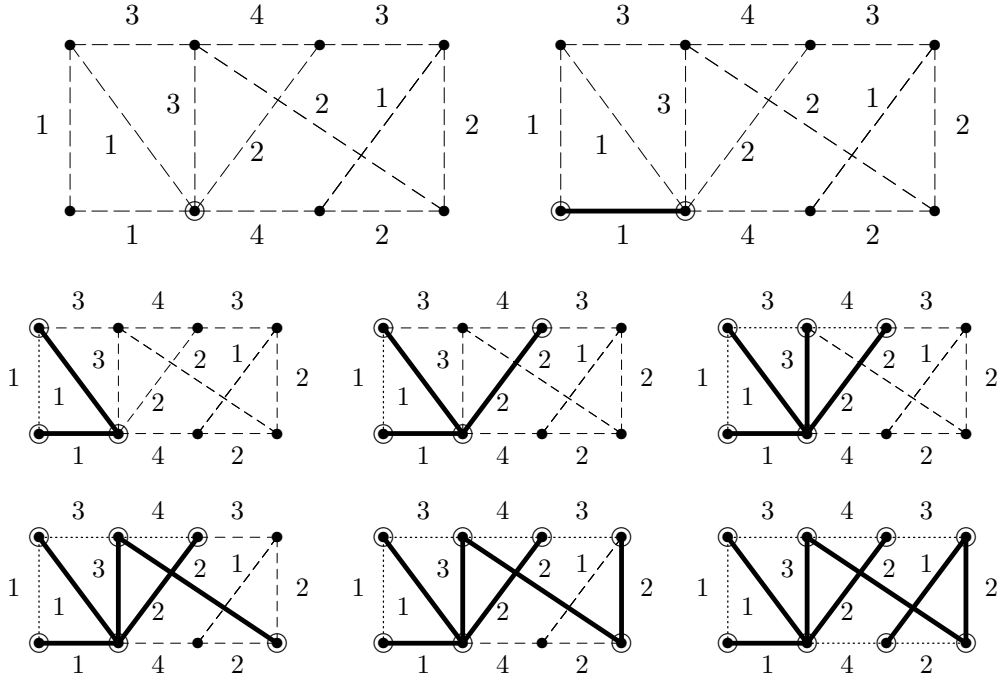
Ač koncepčně velmi jednoduchý, má Algoritmus 10.8 některé problematické implementační detaily, pro které je mnohem častěji používán následující algoritmus (často je připisován Američanu Primovi, ale mnohem dříve publikován Vojtěchem Jarníkem v roce 1930), vycházející z prostého prohledávání do šířky (viz také Algoritmus 9.17).

Algoritmus 10.10. ***Hledání minimální kostry ve váženém grafu*** (G, w)**.**
Opět mějme dán souvislý vážený graf G s ohodnocením hran w.

1. Na začátku zvolíme libovolný vrchol $u \in V(G)$ a podstrom $T := (\{u\}, \emptyset)$.
2. Dokud $V(T) \neq V(G)$, provádíme: Nalezneme hranu $f = uv \in E(G)$ nejmenší váhy s jedním koncem u ve $V(T)$ a druhým v ve $V(G) \setminus V(T)$ a položíme $T := T + v + f$.

   Formálně přesněji tento krok popíšeme následovně.
   - Nechť $X = \{uv \in E(G) : u \in V(T), v \in V(G) \setminus V(T)\}$ a zvolme $f = uv \in X$ minimalizující váhu $w(f)$.
   - Položme $V(T) := V(T) \cup \{v\}$ a $E(T) := E(T) \cup \{f\}$.
3. Nyní je T minimální kostrou grafu G.

`Komentář:` Následuje stručná ukázka průběhu Jarníkova algoritmu, kde kostru opět vyznačujeme tučnými čarami a zpracované vrcholy kroužky kolem.



Již bez důkazu si na závěr našeho pojednání o kostrách uvedeme:

Věta 10.11. *Algoritmus 10.10 korektně spočítá minimální kostru váženého grafu (G, w).*

10.3 Dodatek: Výběr různých reprezentantů

S grafy je úzce svázán i tento klasický kombinatorický problém, vybírající různé reprezentanty z daného systému množin (které typicky nejsou disjunktní a tudíž není jasné, zda reprezetanti dvou množin skutečně budou různí).

`Definice`: Nechť $M_1, M_2, \ldots, M_k$ jsou neprázdné množiny. *Systémem různých reprezentantů* množin $M_1, M_2, \ldots, M_k$ nazýváme posloupnost *různých* prvků $(x_1, x_2, \ldots, x_k)$ takových, že $x_i \in M_i$ pro $i = 1, 2, \ldots, k$.

Teoretické řešení problému výběru různých reprezentantů je plně popsané následující charakterizací jeho existence (známou jako Hallova věta):

Věta 10.12. *Nechť $M_1, M_2, \ldots, M_k$ jsou neprázdné množiny. Pro tyto množiny existuje systém různých reprezentantů, právě když platí*

$$\forall J \subseteq \{1, 2, \ldots, k\} : \left|\bigcup_{j \in J} M_j\right| \geq |J|\,, \tag{1}$$

neboli pokud sjednocení libovolné skupiny z těchto množin má alespoň tolik prvků, kolik množin je sjednoceno.

`Komentář:` Všimněte si dobře, že v jednom směru je platnost Věty 10.12 zřejmá; neboli pokud systém různých reprezentantů existuje, tak podmínka (1) jasně musí platit.

Co nám tedy Věta 10.12 prakticky říká? Jednoduše, pokud systém různých reprezentantů existuje, stačí jej najít či uhodnout. A pokud neexistuje, stačí nalézt vhodnou množinu J porušující podmínku (1) Věty 10.12. Popisům jako tato věta se říká *dobré charakterizace*, neboť nám poskytují buď snadno ověřitelné řešení našeho problému, nebo snadno uvěřitelný důvod, proč řešení nemůže existovat.

Párování grafu a reprezentanti

Přiřazení reprezentantů množinám lze chápat jako hrany grafu, ve kterém na jedné straně jsou naše množiny $M_1, M_2, \ldots, M_k$ z Věty 10.12 a na druhé straně jsou jejich všechny prvky. Takto vypadajícím grafům říkáme *bipartitní* (formálně je bipartitní graf libovolným podgrafem vhodného úplného bipartitního grafu $K_{m,n}$).

Požadavek na jedinečnost a různost reprezentantů pak můžeme vyjádřit touto definicí:

`Definice`: *Párování* v (nyní bipartitním) grafu G je podmnožina hran $M \subseteq E(G)$ taková, že žádné dvě hrany z M nesdílejí koncový vrchol.

Uvažujme systém množin $M_1, \ldots, M_k$ a označme $p_1, \ldots, p_m$ všechny prvky ve sjednocení $M_1 \cup \cdots \cup M_k$. Definujeme si bipartitní graf G na množině vrcholů $\{1, 2, \ldots, k\} \cup \{p_1, \ldots, p_m\}$, který je tvořen hranami $\{i, p_j\}$ pro všechny dvojice i, j, pro které $p_j \in M_i$. Pak platí:

Tvrzení 10.13. *Množiny $M_1, \ldots, M_k$ mají systém různých reprezentantů právě tehdy, když v grafu G existuje párování pokrývající všechny vrcholy $\{1, 2, \ldots, k\}$.*

`Důkaz`: Nechť $(x_1, \ldots, x_k)$ je systém různých reprezentantů množin $M_1, \ldots, M_k$. Pak $M = \{\{i, x_i\} : i = 1, \ldots, k\}$ je požadované párování v G. Naopak párování pokrývající všechny vrcholy $\{1, 2, \ldots, k\}$ v G přímo určuje jednoho reprezentanta každé množiny a tito reprezentanti jsou různí, neboť žádné dvě hrany párování nesdílejí koncový vrchol. □

`Poznámka`: Pro nalezení největšího párování v bipartitním grafu existují rychlé postupy (algoritmy), související s tzv. toky v sítích. Tyto postupy se také dají využít (podle uvedeného tvrzení) k hledání systémů různých reprezentantů.

10.4 Příklady použití grafů

Závěrem si v našem studijním textu nastíníme některé základní motivace pro zavedení a použití grafů při popisu a řešení problémů například v informatice.

Příklad 10.14. *Ukázky některých problémů „ze života“ popsatelných grafy. Podotýkáme, že tyto ukázky jsou často velmi zjednodušené (pro jejich lepší přístupnost širokému okruhu čtenářů), ale to neubírá jejich motivačnímu potenciálu.*

- Vyjádření mezilidských vztahů – „mají se rádi“, „kamarádí se“, „nesnesou jeden druhého“, apod:

  Zde jednotlivé osoby tvoří vrcholy grafu a vztahy jsou hranami (často neorientované, ale i orientace je přípustná). Všimněme si coby zajímavosti, jak tento model přirozeně preferuje „párový“ pohled na vztahy – hrany přece spojují jen dvojice vrcholů. Třebaže například vztah „kamarádí se“ může být obecně platný pro větší skupinky lidí než dvojice, stejně se obvykle vyjadřuje klikou v grafu (každí dva v našem družstvu jsou dobří kamarádi...). Tato obecně pojímaná tendence vyjadřovat i *složitější vztahy těmi párovými* je vodou na mlýn použití teorie grafů jako téměř univerzálního vyjadřovacího prostředku v podobných případech.

  Na druhou stranu i teorie grafů disponuje pojmem tzv. *hypergrafu* umožňujícího použití hran libovolné arity (počtu koncových vrcholů), ale rozsah výskytu hypergrafů v teorii i aplikacích je oproti grafům vskutku zanedbatelný.

- Vyjádření závislostí mezi objekty nebo procesy:

  Představme si situace, ve kterých jednotlivé entity (modelované jako vrcholy) závisí na výstupech jiných entit a naopak poskytují výstupy další entitám. Typickým příkladem mohou být závislosti jednotlivých kroků výrobního nebo rozhodovacího procesu. Ty pak vedou k definici *orientovaného grafu* na dané množině vrcholů/entit, tj. použití hran „se šipkami“. Všimněme si, že závislosti často bývají časového charakteru (přičemž směr závislosti je implicitně jasný) a pak je nezbytnou doplňkovou podmínkou *vyloučení výskytu orientovaných cyklů* v modelovém grafu. Na druhou stranu existují i situace, kdy cyklické závislosti jsou dovoleny a mají svůj význam.

  Pro ještě jednu ukázku závislostí z běžného života informatika se podíváme na správu balíčků softwaru například v Linuxových distribucích. V tom případě jsou jednotlivé balíčky vrcholy grafu, jejich vyžadované závislosti popisují odchozí hrany a jejich poskytované vlastnosti jsou příchozími hranami grafu závislostí. Korektní instalace zvoleného balíčku pak řeší problém zahrnutí všech dalších vrcholů „dosažitelných“ ze zvoleného. Vše je navíc komplikováno správou verzí balíčků, ale to už je mimo rámec našeho úvodního slova.

- Modelování technických či dopravních sítí grafy:

V takových případech bývají vrcholy grafu jednotlivá technická zařízení jako třeba rozvodny, routery, křižovatky a podobně, kdežto hrany jsou tvořeny spojnicemi/vedením mezi vrcholy. Často se zde setkáváme s orientovanými grafy a obecně *multigrafy*.

- Vizualizace vztahů a závislostí pro lidského pozorovatele:

  Nejen při řešení cvičných příkladů v naší učebnici, ale i v mnoha reálných aplikacích využívajících grafy jako modely, je velmi potřebné tyto grafy *vizualizovat* (tj. hezky nakreslit) pro lidského pozorovatele. Jedná se obecně o poměrně obtížný úkol, který přesahuje hranice našeho textu. □

Zpracování grafu počítačem

Mějme jednoduchý graf G na n vrcholech a značme vrcholy jednoduše čísly $V(G) = \{0, 1, \ldots, n-1\}$. Pro počítačovou implementaci grafu G se nabízejí dva základní způsoby, které budeme implicitně využívat i v některých algoritmech následujících lekcí.

- Implementace *maticí sousednosti* $\boldsymbol{G}$, tj. dvourozměrným polem, ve kterém hodnota $\boldsymbol{G}[i, j] = 1$ reprezentuje hranu mezi vrcholy i a j.
- Implementace *výčtem sousedů*, tj. systémem množin, který každému vrcholu i grafu udává seznam jeho sousedů. Všimněte si, že pořadí sousedů může být jakékoliv.

`Poznámka`: Dávejte si pozor na symetrii hran v implementaci! To znamená, že pokud uložíte hranu $\boldsymbol{G}[i, j] = 1$, tak musíte zároveň uložit i hranu $\boldsymbol{G}[j, i] = 1$, jinak se dočkáte v algoritmech nepříjemných překvapení. Totéž se týká i seznamů sousedů, ať už jsou implementovány jakkoliv.

`Komentář:` Implementace maticí sousednosti je hezká svou jednoduchostí, avšak velmi nehezká svou paměťovou náročností. Druhá možnost se pak mnohem lépe hodí pro grafy s relativně malým počtem hran, což nastává ve většině praktických aplikací. (Navíc je implementace výčtem sousedů vhodná i pro multigrafy.) Ke grafům lze do zvláštních složek přidat také *ohodnocení vrcholů a hran* libovolnými čísly či značkami...

Rozšiřující studium

Náš kurz poskytuje jen skutečně minimalistický úvod do informaticky zaměřené teorie grafů. Mnohem více se o matematické stránce teorie grafů můžete dozvědět samostudiem, třeba knihy Kapitoly z diskrétní matematiky autorů Jiřího Matouška a Jaroslava Nešetřila, či v pozdějším navazujícím kurzu MA010 na FI MU.

Vedle toho, jak již bylo předesláno, se s informatickou stránkou grafů potkáte velmi široce v prakticky všech kurzech návrhu algoritmů, třeba na FI MU hned v druhém semestru v IB002. Mimo jiné se tak brzy dozvíte o základních postupech prohledávání grafu do šířky a do hloubky, o efektivním nalézání souvislých a silných komponent, o základních algoritmech pro problém hledání nejkratší cesty v ohodnoceném grafu a podobně.

11 Formalizace a důkazy pro algoritmy

Úvod

Po předchozí převážně matematické látce se náš výklad obrací bezprostředně k informatice. Mnozí z vás si asi již všimli, že umění programovat není zdaleka jen o tom naučit se syntaxi programovacího jazyka, ale především o schopnosti vytvářet a správně formálně zapisovat algoritmy. Přitom třeba situace, kdy programátorem zapsaný kód ve skutečnosti počítá něco trochu jiného, než si autor představuje, je snad nejčastější programátorskou chybou – o to zákeřnější, že ji žádný „chytrý“ překladač nemůže odhalit.

Proto již na počátku studia informatiky je žádoucí klást důraz na správné chápání zápisu algoritmů i na přesné důkazy jejich vlastností a správnosti. Tyto poznatky by měly základem toho, aby si čtenář jako programátor uměl po sobě své algoritmy „přečíst“ a ověřit jejich skutečnou správnost na lokální úrovni.

Cíle

Bude zaveden způsob formálního zápisu algoritmů pro potřeby dalšího výkladu, nezávisle na konkrétních programovacích jazycích. Na tomto formalismu pak bude ukazováno správné chápání chování algoritmů a příklady důkazů na konkrétních „malých“ algoritmech. Nejdůležitější technikou důkazů bude matematická indukce. Na druhou stranu je třeba dodat, že uváděné algoritmy jsou pouze bezvýznamné ilustrativní ukázky pro cvičení důkazů a není úkolem tohoto textu učit čtenáře návrhu algoritmů.

11.1 Formální popis algoritmu

Před samotným závěrem našeho matematického kurzu si položme klíčovou otázku, co je vlastně algoritmus? Když se na tím zamyslíte, asi zjistíte, že to není tak jednoduché přesně říci. Neformálně je algoritmus něčím jako kuchařským receptem, podle kterého ze surovin (vstupů) uvaříme chutné jídlo (očekávaný výstup/výsledek). Říci toto matematicky přesně je však natolik obtížný úkol, že si zde můžeme podat jen docela zjednodušenou (či naivní?) odpověď, přesto však dostatečnou pro zamýšlenou demonstraci matematických důkazů pro běžné algoritmy.

`Poznámka`: Za definici algoritmu je obecně přijímána tzv. *Church–Turingova teze* tvrdící, že všechny algoritmy lze „simulovat“ na Turingově stroji. Jedná se sice o přesnou, ale značně nepraktickou definici. Mimo Turingova stroje existují i jiné *matematické modely výpočtů*, jako třeba stroj RAM, který je abstrakcí skutečného strojového kódu, nebo také třeba tzv. neprocedurální (neimperativní) modely zahrnující funkcionální a logické programování.

Konvence 11.1. Zjednodušeně zde *algoritmem* rozumíme konečnou posloupnost elementárních výpočetních *kroků*, ve které každý další krok *vhodně*[1] využívá (neboli závisí na) vstupní údaje či hodnoty vypočtené v předchozích krocích. Tuto závislost přitom pojímáme zcela obecně nejen na operandy, ale i na vykonávané instrukce v krocích.

Pro zápis algoritmu a jeho zpřehlednění a zkrácení využíváme *řídící konstrukce* – podmíněná větvení a cykly. Při jejich použití však je třeba dát dobrý pozor, aby byla naplněna podmínka skončení algoritmu.

[1] Zvídaví studenti si mohou na tomto místě uvědomit, že ve slůvku „vhodně“ se skrývá celá hloubka Church–Turingovy teze. V žádném případě tak nelze mechanicky bez zamyšlení obracet, že by každá posloupnost kroků atd ... byla algoritmem ve smyslu této teze (viz také Lekce 12).

`Komentář:` Vidíte, jak blízké si jsou konstruktivní matematické důkazy a algoritmy v našem pojetí? Jedná se nakonec o jeden ze záměrů našeho přístupu...

Příklad 11.2. *Zápis algoritmu pro výpočet průměru daného pole* `a[]` *s* `n` *prvky.*

Algoritmus.

- Inicializujeme `sum` $\leftarrow$ `0`;
- postupně pro `i=0,1,2,...,n-1` provedeme
  - `sum` $\leftarrow$ `sum+a[i]`;
- vypíšeme podíl `(sum/n)`. □

Ve „vyšší úrovni“ formálnosti (s jasnějším vyznačením *elementárních kroků* a *řídících struktur* algoritmu) se totéž dá zapsat jako:

Algoritmus 11.3. ***Průměr*** z daného pole `a[]` s `n` prvky.

```
input pole a[] délky n ≥ 1;
sum ← 0;
foreach  i ← 0,1,2,...,n-1  do
    sum ← sum+a[i];
done
res ← sum/n;
output res .
```

Symbolický zápis algoritmů

`Značení`. Pro potřeby symbolického formálního zápisu algoritmů v předmětu FI: IB000 si zavedeme následující pravidla:

- *Proměnné* nebudeme deklarovat ani typovat, pole odlišíme závorkami `p[]`.
- *Přiřazení* hodnoty zapisujeme `a` $\leftarrow$ `b`, případně `a := b`, ale nikdy ne `a=b`.
- Jako elementární operace je možné použít jakékoliv *aritmetické výrazy* v běžném matematickém zápise. Rozsahem a přesností čísel se zde nezabýváme.
- Podmíněné *větvení* uvedeme klíčovými slovy `if ... then ... else ... fi`, kde `else` větev lze vynechat (a někdy, na jednom řádku, i `fi`).
- Pevný *cyklus* uvedeme klíčovými slovy `foreach ... do ... done`, kde část za `foreach` musí obsahovat *předem danou* množinu hodnot pro přiřazování do řídící proměnné.
- *Podmíněný cyklus* uvedeme klíčovými slovy `while ... do ... done`. Zde se může za `while` vyskytovat jakákoliv logická podmínka.
- V zápise používáme jasné *odsazování* (zleva) podle úrovně zanoření řídících struktur (což jsou `if, foreach, while`).
- Pokud je to dostatečně jasné, elementární operace nebo podmínky můžeme i ve formálním zápise *popsat běžným jazykem*.

`Poznámka`: Známá informatická poučka (a zároveň název klasické učebnice programování od N. Wirtha) praví „*Algoritmy + Datové struktury = Programy*“. Jelikož však náš text neaspiruje být učebnicí programování ani algoritmizace, druhý sčítanec uvedené poučky záměrně pomíjíme a aspekty použití vhodných datových struktur pro uváděné algoritmy ponecháváme pro jiné kurzy studia informatiky.

Vyzbrojeni těmito pravidly symbolického zápisu a matematickým aparátem dokazování si nyní můžeme dovolit se podívat na mnohé základní (a jednoduché) algoritmy novým kritickým pohledem.

Co počítá následující algoritmus?

Příklad 11.4. *Je dán následující symbolicky zapsaný algoritmus. Co je jeho výstupem v závislosti na vstupech a, b?*

Algoritmus 11.5.

```
input a, b;
res ← 7;
foreach  i ← 1,2,...,b-1,b  do
    res  ←  res + a + 2·b + 8;
done
output res .
```

Nejprve si zkusmo vypočítáme hodnoty výsledku *res* v počátečních iteracích cyklu:

$b = 0$: $\quad res = 7,$
$b = 1$: $\quad res = 7 \;+\; a + 2b + 8,$
$b = 2$: $\quad res = 7 \;+\; (a + 2b + 8) \;+\; (a + 2b + 8), \ldots$

Co dále? Výčet hodnot naznačuje pravidelnost a závěr, že obecný výsledek po b iteracích cyklu bude mít hodnotu

$$res \;=\; 7 + b(a + 2b + 8) = ab + 2b^2 + 8b + 7\,.$$

Jak ale toto dokážeme? Nejpřesněji asi indukcí podle vstupní hodnoty b, avšak to není formálně úplně jednoduché, neboť s hodnotou b musíme pracovat jako se zcela obecným pevným parametrem a zároveň ji měnit(?) podle potřeb indukce. Přesné vyjádření důkazu výsledku cyklu proto teď přenecháme vašemu dalšímu studiu algoritmů. □

11.2 O „správnosti“ a dokazování programů

Jak se máme přesvědčit, že je daný program počítá „správně“?

* Co třeba ladění programů? Jelikož počet možných vstupních hodnot je (v principu) neohraničený, *nelze otestovat* všechna možná vstupní data.
* Situace je zvláště komplikovaná v případě paralelních, randomizovaných, interaktivních a nekončících programů (operační systémy, systémy řízení provozu apod.). Takové systémy mají *nedeterministické chování* a opakované experimenty vedou k různým výsledkům. (Nelze je tudíž ani rozumně ladit, respektive ladění poskytne jen velmi nedostatečnou záruku správného chování za jiných okolností...)
* V některých případech je však třeba mít naprostou jistotu, že program funguje tak jak má, případně že splňuje základní bezpečnostní požadavky.

– Pro „malé“ algoritmy je možné podat přesný matematický důkaz správnosti.
– Narůstající složitosti programových systémů a požadavky na jejich bezpečnost si pak vynucují vývoj jiných „spolehlivých“ formálních *verifikačních metod*.

`Komentář:` Mimochodem, co to vlastně znamená „počítat správně“? Uvědomte si, že abychom toto rozumně zodpověděli, je třeba mít dopředu dány explicitní specifikace očekávaného chování programu.

Ukázka formálního důkazu algoritmu

Příklad 11.6. *Je dán následující symbolicky zapsaný algoritmus. Dokažte, že jeho výsledkem je „výměna“ vstupních hodnot a, b.*

Algoritmus 11.7.
```
input a, b;
a ← a+b;
b ← a-b;
a ← a-b;
output a, b .
```

Pro správný formální důkaz si musíme nejprve uvědomit, že je třeba symbolicky odlišit od sebe proměnné `a,b` od jejich daných *vstupních hodnot*, třeba h_a, h_b. Nyní v krocích algoritmu počítáme hodnoty proměnných:

* $a = h_a,\ b = h_b,$
* $a \leftarrow a + b = h_a + h_b, \quad b = h_b,$
* $a = h_a + h_b, \quad b \leftarrow a - b = h_a + h_b - h_b = \underline{h_a},$
* $a \leftarrow a - b = h_a + h_b - h_a = \underline{h_b}, \quad b = h_a,$

V jednotlivých krocích tak jasně vidíme, že pro *kterékoliv* vstupní hodnoty h_a, h_b bude ve výsledku obsahem proměnné `a` hodnota h_b a obsahem proměnné `b` hodnota h_a. Tímto jsme s důkazem hotovi. □

11.3 Rekurzivní algoritmy

Rekurentní vztahy posloupností, stručně uvedené v Oddíle 5.1, mají svou přirozenou obdobu v *rekurzivně zapsaných algoritmech*. Zjednodušeně řečeno to jsou algoritmy, které se v průběhu výpočtu odvolávají na výsledky sebe sama pro jiné (pokud možno striktně menší) vstupní hodnoty. U takových algoritmů je zvláště důležité kontrolovat jejich správnost a také praktickou proveditelnost (časovou i paměťovou). My si tuto oblast osvětlíme několika jednoduchými příklady.

`Značení`. Pro účely symbolického zápisu rekurze zavádíme klíčová slova `function`, které uvozuje hlavičku definované funkce, a `return`, které ukončuje výpočet funkce a vrací výslednou (tzv. *návratovou*) hodnotu funkce.

Příklad 11.8. *Symbolický zápis jednoduchého rekurzivního algoritmu.*

Algoritmus .
```
function faktorial( x ):
    if x ≤ 1 then  t ← 1;
    else  t ← x · faktorial(x-1);
return t .
```

Co je výsledkem výpočtu? Jednoduše řečeno, výsledkem je *faktoriál* vstupní přirozené hodnoty x, tj. hodnota $x! = x \cdot (x-1) \cdot \cdots \cdot 2 \cdot 1$. (Záporné a necelé vstupní hodnoty x pro zjednodušení ignorujeme.) Všimněte si ještě, že algoritmus i pro vstup $x = 0$ správně vyhodnotí $0! = 1$. □

Pro jiný příklad rekurze se vrátíme k Oddílu 5.1, kde byla zmíněna známá Fibonacciho posloupnost $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$. Nyní tuto posloupnost budeme uvažovat již od jejího nultého členu, tj. jako $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$.

Algoritmus 11.9. Rekurzivní výpočet členů Fibonacciho posloupnosti.
Pro dané přirozené $x \geq 0$ vypočítáme x-té Fibonacciho číslo následovně:

```
function fibonacci( x ):
    if x < 2 then  t ← x;
    else  t ←  fibonacci(x-1) + fibonacci(x-2);
return t .
```

Komentář: Správnost Algoritmu 11.9 je víceméně zřejmá z jeho přímé podoby s rekurentním vzorcem v definici Fibonacciho čísel. Algoritmus 11.9 tudíž lze prohlásit za definiční, čili referenční implementaci výpočtu Fibonacciho čísel. Zamyslete se však, jak je to s praktickou „*proveditelností*“ takového algoritmu... Vidíte (případně si zkuste naprogramovat), že čas výpočtu roste velmi rychle? Třebaže hodnotu $fibonacci(30)$ tímto algoritmem spočítáte poměrně rychle, s výpočtem $fibonacci(40)$ už budete mít větší problémy a $fibonacci(50)$ asi bude mimo vaše možnosti. To skutečně není dobrý algoritmus!

Proto si v dalším Příkladu 11.10 uvedeme poněkud (ve skutečnosti velmi výrazně) lepší algoritmus výpočtu, podobající se přirozenému lidskému postupu psaní členů posloupnosti „do řádku na papír“. Doporučujeme si oba algoritmy zkusit implementovat a mezi sebou porovnat.

Příklad 11.10. *Nerekurzivní algoritmus pro Fibonacciho čísla.*
Dokažte, že následující algoritmus pro každé přirozené n počítá tutéž hodnotu jako rekurentní funkce `fibonacci(n)` v Algoritmu 11.9 (ale mnohem mnohem rychleji).

Algoritmus .

```
input n;
b[0] ← 0;  b[1] ← 1;
foreach  i ← 2,3,...,n  do
    b[i]  ←  b[i-1]+b[i-2];
done
output b[n] .
```

Důkaz: Indukcí podle řídící proměnné i budeme dokazovat, že po i-té iteraci cyklu algoritmu bude vždy platit $b[i] = fibonacci(i)$ (a na závěr dosadíme $i = n$).

Co se týče báze indukce, tato vyplývá z úvodního přiřazení; $b[0] \leftarrow 0 = fibonacci(0)$ a $b[1] \leftarrow 1 = fibonacci(1)$. Pro libovolné $i \geq 1$ v ind. kroku předpokládáme platnost našeho indukčního předpokladu $b[j] = fibonacci(j)$ pro $j \in \{i, i-1\}$. V $(i+1)$-ní iteraci cyklu pak nastane $b[i+1] \leftarrow b[i] + b[i-1] = fibonacci(i) + fibonacci(i-1) = fibonacci(i+1)$, což je přesně podle rekurentní definice z Algoritmu 11.9. Tím je důkaz vztahu $b[j] = fibonacci(j)$ hotov pro $j = i+1$. □

11.4 Techniky důkazu indukcí

Doposud byla v našem textu matematická indukce obvykle představována ve své přímočaré formě, kdy dokazované tvrzení ihned nabízelo celočíselný parametr, podle nějž bylo potřebné indukci vést. Indukční krok pak prostě zpracoval přechod „$n = i \rightsquigarrow n = i + 1$“. To však u dokazování správnosti algoritmů často neplatí a našim cílem zde je ukázat různé možné techniky, jak správně indukci na dokazování algoritmů aplikovat. Uvidíme, jak si z nabízejících se parametrů správně vybrat a jak je případně kombinovat.

Fixace parametru

První technika důkazu prostě dopředu za některé parametry dosazuje (obecně zvolené) konstanty. Tato technika je vhodná pro případy, kdy je sice v algoritmu více parametrů, ale „zjevně“ dochází ke změně jen jednoho (nebo části) z nich a chování algoritmu ke zbylým „neměnným“ parametrům je dobře předvídatelné.

Příklad 11.11. *Mějme následující algoritmus. Co je jeho výsledkem výpočtu?*

Algoritmus .

```
function soucin(x,y):
    if x ≤ 0 then  t ← 0;
    else  t ← soucin(x-1,y)+y;
return t.
```

Sledováním průběhu rekurze v algoritmu zjistíme, že hloubka zanoření volání funkce `soucin` bude rovna nezáporné (horní celé) části `x`. Jelikož každé zanoření poté k výsledku přičte hodnotu `y`, odhadneme:

Věta. Pro každé $x, y \in \mathbb{N}$ Algoritmus 11.11 vrátí hodnotu součinu $t = x \cdot y$.

Jaký je vhodný postup k důkazu tohoto tvrzení indukcí? Je snadno vidět, že na hodnotě vstupu `y` vlastně nijak podstatně nezáleží (lze `y` *fixovat*) a důležité je sledovat `x`. Tato úvaha nás dovede k následujícímu:

`Důkaz`: Budiž $y \in \mathbb{N}$ libovolné ale pro další úvahy *pevné*. Dokazujeme tady, že pro každý vstup $x := i \in \mathbb{N}$ volání funkce `soucin(x,y)` navrátí hodnotu $i \cdot y$. Podle principu matematické indukce uplatněné na parametr i dostáváme snadno:

* V *bázi* pro vstup $x = i = 0$ provedeme první větev podmínky, tj. $t \leftarrow 0 = 0 \cdot y$.
* *Indukční krok.*
  Nechť je tvrzení známo pro $x = i \in \mathbb{N}$ a uvažujme další vstup $x := i + 1 > 0$. V tom případě se vykonává druhá větev podmínky, čili $t \leftarrow \texttt{soucin}(x - 1, y) + y$. Všimněte si, že platí $x - 1 = i$. Podle indukčního předpokladu již víme, že návratovou hodnotu volání funkce $\texttt{soucin}(i, y)$ je $i \cdot y$. Zbývá jen uvedené poznatky správně zřetězit, abychom odvodili, že návratovou hodnotu volání funkce $\texttt{soucin}(i, y)$ bude

$$t \leftarrow \texttt{soucin}(x - 1, y) + y = \texttt{soucin}(i, y) + y = i \cdot y + y = (i + 1)y = x \cdot y\,.$$

Důkaz matematickou indukcí je tímto zdárně ukončen. □

Indukce k součtu parametrů

Druhou techniku je vhodné použít především v případech, kdy se v průběhu algoritmu vždy některý parametr zmenšuje, ale pokaždé je to některý jiný parametr, takže v indukci se nelze zaměřit jen na jeden z nich. Jedná se spíše o situaci u rekurzivních algoritmů.

Příklad 11.12. *Co je výsledkem následujícího rekurzivního výpočtu?*

Algoritmus .

```
function zkombinovat(m,n):
    if m=0 ∨ n=0 then  res ← 1;
    else
        res ← zkombinovat(m-1,n) + zkombinovat(m,n-1);
    fi
return res.
```

Výše uvedený vzorec (a ostatně i název funkce) naznačuje, že funkce má co společného s kombinačními čísly a *Pascalovým trojúhelníkem*

$$\binom{a+1}{b+1} = \binom{a}{b+1} + \binom{a}{b},$$

je však třeba správně „nastavit“ význam parametrů $a, b \ldots$

> **Věta**. Pro každé parametry $m, n \in \mathbb{N}$ je výsledek výpočtu funkce `zkombinovat(m,n)` roven počtu všech m-prvkových podmnožin $(m+n)$-prvkové množiny, tedy kombinačnímu číslu $res = \binom{m+n}{m}$.

`Poznámka`: Všimněte si navíc, že náš důkaz vůbec nevyužije faktu, že dokazovaný počet podmnožin je vyjádřitelný kombinačním číslem, naopak tento vztah počtu podmnožin ke kombinačnímu číslu $\binom{m+n}{m}$ a k Pascalovu trojúhelníku vyplyne jako „boční produkt“ našeho elementárního důkazu.

`Důkaz` indukcí vzhledem k součtu parametrů $i = m + n$:

* *Báze* $i = m + n = 0$ pro $m, n \in \mathbb{N}$ znamená, že $m = n = 0$. Zde však s výhodou využijeme tzv. „*rozšíření báze*“ na všechny hraniční případy $m = 0$ nebo $n = 0$ zvlášť. V obou rozšířených případech daná podmínka algoritmu není splněna, a proto výsledek výpočtu bude počáteční `res=1`. Je toto platná odpověď?

  Kolik je prázdných podmnožin ($m = 0$) jakékoliv množiny? Jedna, $\emptyset$. Kolik je m-prvkových podmnožin ($n = 0$) m-prvkové množiny? Zase jedna, ta množina samotná. Tím je důkaz rozšířené báze indukce dokončen.

* *Indukční krok* přechází na součet $i + 1 = m + n$ pro $m, n > 0$. Nyní je podmínka algoritmu splněna a vykonají se rekurentní volání

  ```
  zkombinovat(m-1,n) + zkombinovat(m,n-1).
  ```

  Rekurentní volání se vztahují k výběru podmnožin nosné množiny, která má $m-1+n = m+n-1 = i$ prvků, například $M = \{1, 2, \ldots, i\}$. Výsledkem tedy je, podle indukčního předpokladu pro součet i, počet všech $(m-1)$-prvkových plus m-prvkových podmnožin množiny M.

Kolik nyní je m-prvkových podmnožin $(i+1)$-prvkové množiny $M' = M \cup \{i+1\}$? Pokud ze všech těchto podmnožin odebereme prvek $i+1$, dostaneme právě m-prvkové podmnožiny (z těch neobsahujících prvek $i+1$) plus $(m-1)$-prvkové podmnožiny (z těch původně obsahujících $i+1$). A to je v součtu rovno `zkombinovat(m-1,n) + zkombinovat(m,n-1)`, jak jsme měli v indukčním kroku dokázat. □

Zesílení dokazovaného tvrzení

Poslední ukázka indukčního důkazu se zabývá situací, která je jistým způsobem opačná k první ukázce fixace parametru. V nynější situaci je nám přímo zadáním jeden z parametrů zafixován na určitou hodnotu ($y = 1$), avšak sledování průběhu výpočtu vyžaduje tuto fixovanou hodnotu zpět „uvolnit“ a uvažovat ji proměnnou. V kontextu Lekce 4.4 pak vlastně dochází k *zesílení požadovaného tvrzení* v matematické indukci.

Příklad 11.13. *Zjistěte, jaká je návratová hodnota volání následující funkce* `tajemne`$(x, 1)$ *v závislosti na celočíselné vstupní hodnotě* x*.*

Algoritmus 11.14.

```
function tajemne(x,y):
    if x ≤ 0 then  t ← 0;
    else  t ← tajemne(x-1,2·y) + y;
return t.
```

Zkusíme-li si výpočet simulovat pro $x = 0, 1, 2, 3, 4 \ldots$, postupně získáme pro volání `tajemne`$(x, 1)$ návratové hodnoty $0, 1, 3, 7, 15 \ldots$. Na základě toho již není obtížné uhodnout, že návratová hodnota asi bude obecně určena vztahem $2^x - 1$. Toto tvrzení je však třeba dokázat!

`Komentář:` Jak záhy zjistíme, matematická indukce na naše tvrzení přímo „nezabírá“, ale mnohem lépe se nám povede s následujícím přirozeným zesílením dokazovaného tvrzení, které zobecňuje výsledek na proměnné hodnoty parametru y (a které také musíme uhodnout):

Věta. Pro každá přirozená x, y volání funkce `tajemne`(x, y) Algoritmu 11.14 vrátí hodnotu $(2^x - 1) \cdot y$.

`Důkaz`: Postupujeme indukcí podle $x := i \in \mathbb{N}$.

* V *bázi*, pro vstup $x = i = 0$ je navrácena hodnota $0 = (2^0 - 1) \cdot y$ bez ohledu na hodnotu y.
* *Indukční krok*. Nechť je tvrzení známo pro $x = i \in \mathbb{N}$ a uvažujme další vstup $x := i + 1 > 0$. V tom případě se vykonává druhá větev podmínky a výsledek bude roven hodnotě `tajemne`$(x-1, 2y)+y =$ `tajemne`$(i, 2y)+y$. Podle indukčního předpokladu již víme, že volání funkce `tajemne`$(i, 2y)$ navrátí $(2^i - 1) \cdot 2y$, a celkem tak platí:

$$\texttt{tajemne}(i, 2y) + y = (2^i - 1) \cdot 2y + y = 2^{i+1}y - 2y + y = (2^{i+1} - 1) \cdot y$$

Poslední výraz je roven požadovanému $(2^x - 1) \cdot y$ a důkaz matematickou indukcí je tímto ukončen. □

11.5 Dodatek: Zajímavé algoritmy aritmetiky

Pro volitelné ukázky složitějších důkazových technik pro algoritmy (s indukcí) se můžeme podívat na dva krátké a dobře známé algoritmy z oblasti aritmetiky.

Euklidův algoritmus

Již z dávných dob antiky pochází následující zajímavý a koneckonců velmi jednoduchý postup–algoritmus pro nalezení největšího společného dělitele dvou čísel.

Algoritmus 11.15. ***Euklidův pro největšího společného dělitele.***
Pro zadaná přirozená čísla p, q počítá následovně:

```
input  p, q;
while  p>0 ∧ q>0  do
    if  p>q  then  p ← p-q;
    else  q ← q-p;
done
output  p+q.
```

> **Věta**. Pro každé $p, q \in \mathbb{N}$ na vstupu algoritmus vrátí hodnotu největšího společného dělitele čísel p a q, nebo 0 pro $p = q = 0$.

`Důkaz` povedeme indukcí podle součtu $i = p + q$ vstupních hodnot. (Jak jsme psali, je to *přirozená volba* v situaci, kdy každý průchod cyklem algoritmu sníží jedno z p, q, avšak není jasné, které z nich.)

- Báze indukce pro $i = p + q = 0$ je zřejmá; cyklus algoritmu neproběhne a výsledek ihned bude 0.
- Ve skutečnosti je zase výhodné uvažovat *rozšířenou bázi*, která zahrnuje i případy, kdy jen jedno z p, q je nulové (Proč? Jedná se o všechny případy, kdy průchod cyklem neproběhne, neboli touto indukcí sledujeme *počet průchodů cyklem*.) Pak výsledek $p + q$ bude roven tomu nenulovému z obou sčítanců, což je v tomto případě zároveň jejich největší společný dělitel.
- *Indukční krok.* Uvažme vstupní hodnoty $p := h_p$ a $q := h_q$, přičemž $h_p + h_q = i + 1$ a $h_p > 0$ a $h_q > 0$ – tehdy dojde k prvnímu průchodu tělem cyklu našeho algoritmu. Předpokládejme $h_p > h_q$; poté po prvním průchodu tělem cyklu budou hodnoty $p = h_p - h_q$ a $q = h_q$, což znamená $p + q = h_p \leq h_p + h_q - 1 = i$.

  Podle indukčního předpokladu (jelikož nyní $p + q \leq i$) tudíž výsledkem algoritmu pro vstupy $p = h_p - h_q$ a $q = h_q$ bude největší společný dělitel $NSD(h_p - h_q, h_q)$. Symetricky pro $h_p \leq h_q$ algoritmus vrátí $NSD(h_p, h_q - h_p)$. (Kde $NSD()$ krátce označuje největšího společného dělitele.)

Důkaz proto bude dokončen následujícím Lematem 11.16. □

Lema 11.16. *Pro každá přirozená a, b platí $NSD(a, b) = NSD(a-b, b) = NSD(a, b-a)$.*

`Komentář:` Všimněte si, že dělitelnost je dobře definována i na záporných celých číslech.

`Důkaz`: Ověříme, že $c = NSD(a - b, b)$ je také největší společný dělitel čísel a a b (druhá část je pak symetrická).

* Jelikož číslo c dělí čísla $a-b$ a b, dělí i jejich součet $(a-b)+b=a$. Potom c je společným dělitelem a a b.
* Naopak nechť d nějaký společný dělitel čísel a a b. Pak d dělí také rozdíl $a-b$. Tedy d je společný dělitel čísel $a-b$ a b. Jelikož c je *největší* společný dělitel těchto dvou čísel, nutně d dělí c a závěr platí. □

Modulární umocňování

Dále například „zbytkové“ umocňování na velmi vysoké exponenty je podkladem známé RSA šifry. Pokud bychom takovou mocninu zkoušeli počítat hrubou silou jen opakovaným násobením základu, výsledku bychom se nikdy v rozumném čase nedočkali.

Algoritmus 11.17. ***Binární postup umocňování.***
Pro daná čísla a, b vypočteme jejich celočíselnou mocninu (omezenou na zbytkové třídy modulo m pro prevenci přetečení rozsahu celých čísel v počítači), tj. hodnotu $a^b \mod m$, následujícím postupem.

```
input a,b, m;
res ← 1;
while b > 0 do
    if b mod 2 > 0 then res ← (res·a) mod m;
    b ← ⌊b/2⌋;   a ← (a·a) mod m;
done
output res.
```

K důkazu správnosti algoritmu použijeme indukci podle délky ℓ binárního zápisu čísla b.

Věta. Algoritmus 11.17 skončí a vždy správně vypočte hodnotu $a^b \mod m$.

Důkaz: Báze indukce je pro $\ell = 1$, kdy $b = 0$ nebo $b = 1$. Přitom pro $b = 0$ se cyklus vůbec nevykoná a výsledek je $res = 1$. Pro $b = 1$ se vykoná jen jedna iterace cyklu a výsledek je $res = a \mod m$.

Nechť tvrzení platí pro všechny vstupy b délky $\ell = i \geq 1$ a uvažme vstup délky $\ell := i + 1$. Pak zřejmě $b \geq 2$ a vykonají se alespoň dvě iterace cyklu. Po první iteraci budou hodnoty proměnných po řadě

$$a_1 = a^2, \quad b_1 = \lfloor b/2 \rfloor \quad \text{a} \quad res = r_1 = (a^{b \bmod 2}) \mod m\,.$$

Tudíž délka binárního zápisu b_1 bude jen $\ell - 1 = i$ a podle indukčního předpokladu zbylé iterace algoritmu skončí s výsledkem

$$res = r_1 \cdot a_1^{b_1} \mod m = \left(a^{b \bmod 2} \cdot a^{2\lfloor b/2 \rfloor}\right) \mod m = a^b \mod m\,.$$

□

Poznámka: Zamysleli jste se, jaký mají algoritmy v tomto oddíle vlastně význam? Vždyť každý z předchozích dvou příkladů jistě sami vyřešíte jednou jednoduchou smyčkou. Podívejte se však (alespoň velmi zhruba) na počet kroků, které zde uvedené algoritmy potřebují vykonat k získání výsledku, a srovnejte si to s počty kroků oněch „jednoduchých“ algoritmů. Pro skutečně velká vstupní čísla zjistíte propastný rozdíl – s takovým zdánlivě jednoduchým algoritmem pro umocňování, třeba '`foreach i ← 1,...b do res ← res·a mod m; done`', se pro obrovské hodnoty b výsledku nikdy nedočkáte, kdežto Algoritmus 11.17 stále poběží velmi rychle. (Spočítáte, jak rychle?) Obdobně tomu bude i u dalšího Algoritmu 11.18.

Relativně rychlé odmocnění

Na závěr oddílu si ukážeme jeden netradiční krátký algoritmus a jeho analýzu a důkaz zde ponecháme otevřené. Dokážete popsat, na čem je algoritmus založen?

Algoritmus 11.18. ***Celočíselná odmocnina.***
Pro dané přirozené číslo x vypočteme dolní celou část jeho odmocniny $\lfloor\sqrt{x}\rfloor$:

```
input x;
p ← x;   res ← 0;
while  p > 0  do
    while  (res + p)^2 ≤ x  do res ← res + p;
    p ← ⌊p/2⌋;
done
output res.
```

Rozšiřující studium

Náš výklad pohlíží na algoritmy a programování tzv. procedurálním neboli imperativním paradigmatem. Vedle toho existují i jiné přístupy k programování, jako zmíněné funkcionální nebo logické. Na FI se s jinými přístupy studenti seznámí třeba v předmětu IB015 Neimperativní programování. Pro náš matematický výklad není výběr výpočetního paradigmatu to nejdůležitější.

Smyslem této lekce bylo především ukázat, že u jednoduchých algoritmů lze (a je vhodné) je matematicky formálně zapisovat i dokazovat jejich správnost. Samozřejmě je iluzorní předpokládat, že obdobné důkazy správnosti podáme i pro velké softwarové projekty čítající až miliony řádků, ale postupy a techniky naučené při ověřování jednoduchých algoritmů s úspěchem využijete i při kontrole a ladění jednotlivých kousků velkých projektů.

Mnohé další ukázky dokazování základních algoritmů najdete hned v kurzech základní algoritmizace (jako již zmíněný kurz IB002). Mimo jiné se tam naučíte další užitečné formalismy, které (ač přímo založené na stejných matematických principech dokazování a indukce) vás zbaví části starostí se správným formálním zápisem důkazu algoritmu a provedou vás tak „pohodlnější cestou“. O různých pokročilých technikách formální verifikace programů se v případě zájmu dozvíte v pokračujícím studiu.

12 Nekonečné množiny, Zastavení algoritmu

Úvod

Hloubavého čtenáře může snadno napadnout kacířská myšlenka, proč se vlastně zabýváme dokazováním správnosti algoritmů a programů, když by to přece (snad?) mohl za nás dělat automaticky počítač samotný. Bohužel to však nejde a je hlavním cílem této lekce ukázat matematické důvody proč tomu tak je.

Konkrétně si dokážeme, že nelze algoritmicky rozhodnout, ani zda se daný algoritmus na svém vstupu zastaví nebo ne. Hlavními nástroji, které použijeme, budou nekonečné množiny a důkazová technika tzv. Cantorovy diagonály, která se ve velké míře používá právě v teoretické informatice. Touto lekcí se tak krátce vymaníme ze zajetí naivní teorie množin, která v této oblasti má své nepřekonatelné limity. (Pro zvídavé – obdobně, ale mnohem složitěji, lze dokázat že ani matematické důkazy nelze obecně algoritmicky konstruovat...)

Cíle

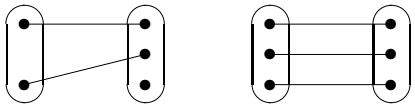
Zavedeme si ve zjednodušeném (a stále „poněkud naivním“) pohledu nekonečné množiny a na nich techniku důkazu Cantorovou diagonálou. Pak tuto klíčovou důkazovou techniku teoretické informatiky využijeme k důkazu algoritmické neřešitelnosti problému zastavení.

12.1 O nekonečných množinách a kardinalitě

Zatímco v naivní teorii jsme si velikost konečné množiny definovali jednoduše jako *počet jejích prvků* vyjádřený přirozeným číslem, pro nekonečné množiny je situace obtížná a mnohem méně intuitivní. Co nám třeba brání zavést pro velikost nekonečné množiny *symbol* ∞? Ponejvíce závažný fakt, že není „nekonečno“ jako „nekonečno“ (Věta 12.2)!

Právě proto si pro určení mohutnosti nekonečných množin musíme vypomoci následujícím příměrem: Velikosti dvou hromádek jablek dokážeme i bez počítání porovnat tak, že budeme z obou hromádek po řadě odebírat dvojice jablek (z každé jedno), až dokud první hromádka nezůstane prázdná – druhá hromádka pak je větší (nebo nejvýše rovna) té první.

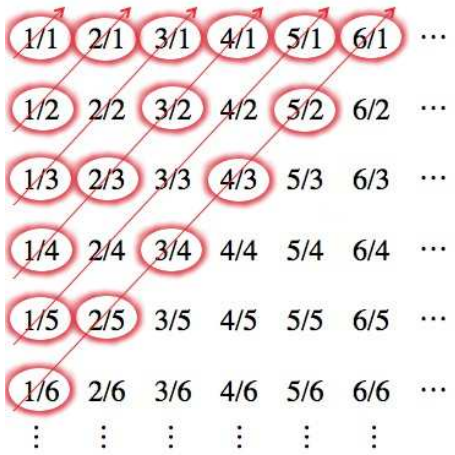
`Definice`: Množina A je „*nejvýše tak velká*“ jako množina B, právě když existuje injektivní funkce $f : A \to B$. Množiny A a B jsou „*stejně velké*“ právě když mezi nimi existuje bijekce. V případech nekonečných množin pak místo “velikosti” mluvíme formálně o jejich *kardinalitě*.



`Komentář:` Uvedená definice kardinality množin funguje korektně i pro nekonečné množiny:

* Například $\mathbb{N}$ a $\mathbb{Z}$ mají stejnou kardinalitu (jsou „stejně velké“, tzv. *spočetně nekonečné*).
* Lze snadno ukázat, že i $\mathbb{Q}$ je spočetně nekonečná, tj. existuje bijekce $f : \mathbb{N} \to \mathbb{Q}$, stejně

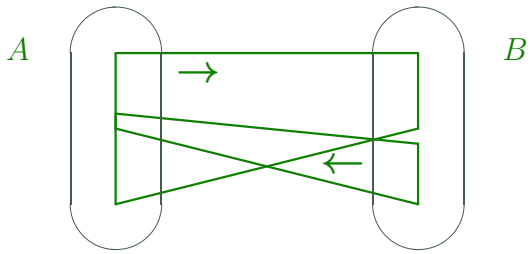
jako bijekce $h : \mathbb{N} \to \mathbb{N}^2$.



- Existují ale i nekonečné množiny, které jsou „striktně větší“ než libovolná spočetná množina (příkladem je $\mathbb{R}$ ve Větě 12.2).
- Později dokážeme, že existuje nekonečná posloupnost nekonečných množin, z nichž *každá je striktně větší než všechny předchozí*.

Pro porovnávání velikostí množin někdy s výhodou využijeme následující přirozené, ale nelehké tvrzení (bez důkazu):

Věta 12.1. *Pro libovolné dvě množiny A, B (i nekonečné) platí, že pokud existuje injekce $A \to B$ a zároveň i injekce $B \to A$, pak existuje bijekce mezi A a B.*



Cantorova diagonála, aneb kolik je reálných čísel

Prvním klíčovým poznatkem ukazujícím na neintuitivní chování nekonečných množin je následující důkaz, který dal historicky vzniknout metodě tzv. *Cantorovy diagonály*.

Věta 12.2. *Neexistuje žádné surjektivní zobrazení $g : \mathbb{N} \to \mathbb{R}$.*

Důsledek 12.3. *Neexistuje žádné injektivní (tudíž ani bijektivní) zobrazení $h : \mathbb{R} \to \mathbb{N}$. Neformálně řečeno, reálných čísel je striktně více než všech přirozených.*

`Důkaz` (Věty 12.2 sporem): Nechť takové g existuje a pro zjednodušení se omezme jen na funkční hodnoty v intervalu $(0, 1)$ tj. ponechme z každé funkční hodnoty jen její nezápornou necelou část. Podle hodnot zobrazení g si takto můžeme „uspořádat“

dekadické zápisy všech reálných čísel v intervalu $(0,1)$ po řádcích do tabulky:

$$\begin{array}{lllllllllllll} g(0) = & 0. & \mathbf{1} & 5 & 4 & 2 & 7 & 5 & 7 & 8 & 3 & 2 & 5 & \dots \\ g(1) = & 0. & & \mathbf{4} & & & & & & & & & & \dots \\ g(2) = & 0. & & & \mathbf{1} & & & & & & & & & \dots \\ g(3) = & 0. & & & & \mathbf{3} & & & & & & & & \dots \\ g(4) = & 0. & & & & & \mathbf{9} & & & & & & & \dots \\ \vdots & \vdots & & & & & & \ddots & & & & & & \end{array}$$

Nyní sestrojíme číslo $\alpha \in (0,1)$ následovně; jeho i-tá číslice za desetinnou čárkou bude 1, pokud v i-tém řádku tabulky na diagonále není 1, jinak to bude 2. V našem příkladě $\alpha = 0.21211\ldots$

Kde se naše číslo α v tabulce nachází? Nezapomeňme, že g byla surjektivní, takže α někde musí být. Konstrukce však ukazuje, že α se od každého čísla v tabulce liší na aspoň jednom desetinném místě, a to je spor. (Až na drobný technický detail s rozvojem $\ldots\bar{9}$, který pro dosažení jednoznačnosti zápisu v naší tabulce preferujeme nad konečným rozvojem takového čísla. Neboli místo 0.125 budeme do tabulky psát $0.124\bar{9}$.) □

12.2 „Naivní“ množinové paradoxy

Analogickým způsobem k Větě 12.2 lze dokázat následovné zobecnění vyjadřující se o jakékoliv množině a jí přiřazené striktně větší množině.

Věta 12.4. *Nechť M je libovolná množina. Pak existuje injektivní zobrazení $f : M \to 2^M$, ale neexistuje žádné bijektivní zobrazení $g : M \to 2^M$.*

Důkaz: Dokážeme nejprve existenci f. Stačí ale položit $f(x) = \{x\}$ pro každé $x \in M$. Pak $f : M \to 2^M$ je zjevně injektivní.

Neexistenci g dokážeme sporem. Předpokládejme tedy naopak, že existuje bijekce $g : M \to 2^M$. Uvažme množinu $K \subseteq M$ definovanou takto:

$$K = \{x \in M \mid x \notin g(x)\}.$$

Jelikož g je bijektivní a $K \in 2^M$, musí existovat $y \in M$ takové, že $g(y) = K$. Nyní rozlišíme dvě možnosti:

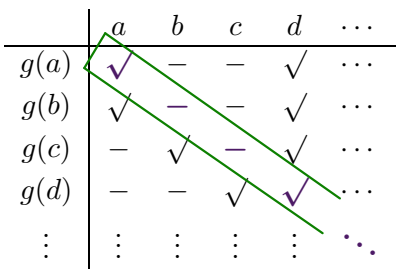
- $y \in g(y)$. Tj. $y \in K$. Pak ale $y \notin g(y)$ z definice K, což je spor.
- $y \notin g(y)$. To podle definice K znamená, že $y \in K$, tj. $y \in g(y)$, spor.

□

Komentář: Dvě navazující poznámky.

- Technika použitá v důkazech *Vět 12.2* a *12.4* se nazývá *Cantorova diagonální metoda*, nebo také zkráceně *diagonalizace*.

  Konstrukci množiny K lze znázornit pomocí následující tabulky:



Symbol $\surd$ resp. $-$ říká, že prvek uvedený v záhlaví sloupce patří resp. nepatří do množiny uvedené v záhlaví řádku. Tedy např. $d \in g(b)$ a $a \notin g(d)$. Množina K poté obsahuje ty diagonální prvky označené $-$, tj. „převrací“ význam diagonály.

- Z toho, že nekonečna mohou být „různě velká“, lze lehce odvodit řadu dalších faktů. V jistém smyslu je např. množina všech problémů větší než množina všech algoritmů (obě množiny jsou nekonečné), a proto nutně existují problémy, které *nejsou algoritmicky řešitelné*.

Cantorův paradox

Naivní teorie množin, jak jsme si ji uvedli i v tomto předmětu, trpí mnoha neduhy a nepřesnostmi, které vyplynou na povrch především při „neopatrné manipulaci“ s nekonečnými množinami. Abychom se těmto „neopatrnostem“ vyhnuli bez přílišné formalizace, dva základní z těchto paradoxů si nyní ukážeme.

Příklad 12.5. *Uvážíme-li nyní nekonečnou posloupnost množin*

$$A_1, A_2, A_3, A_4, \cdots$$

kde $A_1 = \mathbb{N}$ *a* $A_{i+1} = 2^{A_i}$ *pro každé* $i \in \mathbb{N}$, *je vidět, že všechny množiny jsou nekonečné a každá je striktně větší než libovolná předchozí.*

Kde však v tomto řazení kardinalit bude „*množina všech množin*“? Na tuto otázku, jak sami asi cítíte, nelze podat odpověď. Co to však znamená? □

∗ Takto se koncem 19. století objevil první *Cantorův paradox* nově vznikající teorie množin.

∗ Dnešní moderní vysvětlení paradoxu je jednoduché, prostě „množinu všech množin“ nelze definovat, taková v matematice neexistuje.

Brzy se však ukázalo, že je ještě *mnohem hůř*...

Russelův paradox

`Fakt`: Není pravda, že *každý soubor prvků* lze považovat za množinu.

∗ Nechť $X = \{M \mid M \text{ je množina taková, že } M \notin M\}$. Platí $X \in X$?

  – Ano. Tj. $X \in X$. Pak ale X splňuje $X \notin X$.

  – Ne. Pak X splňuje vlastnost $X \notin X$, tedy X je prvkem X, tj., $X \in X$.

∗ Obě možné odpovědi vedou ke sporu. X tedy nelze prohlásit za množinu. Jak je ale něco takového vůbec možné?

`Komentář:` Vidíte u Russelova paradoxu podobnost přístupu s Cantorovou diagonalizací? Russelův paradox se objevil začátkem 20. století a jeho „jednoduchost“ zasahující úplné základy matematiky (teorie množin) si vynutila hledání seriózního rozřešení a rozvoj formální matematické logiky.

Pro ilustraci tohoto paradoxu, slyšeli jste už, že „*v malém městečku žije holič, který holí právě ty muže městečka, kteří se sami neholí*“? Zmíněné paradoxy naivní teorie množin zatím v tomto kurzu nerozřešíme, ale zapamatujeme si, že většina matematických a informatických disciplín vystačí s „*intuitivně bezpečnými*“ *množinami*.

12.3 Algoritmická neřešitelnost problému zastavení

Výše vysvětlené myšlenky diagonalizace a principů základních paradoxů naivní teorie množin sice vypadají „velmi matematicky“. Přesto je však téměř beze změny lze aplikovat i na bytostně informatickou otázku, zda lze algoritmicky poznat, pro které vstupy se daný program vůbec zastaví. Negativní odpověď na tuto otázku je jedním z fundamentálních výsledků informatiky a přitom má překvapivě krátký a čistý důkaz diagonalizací.

`Fakt`: Uvědomme si (velmi neformálně) několik základních myšlenek.

* Program v každém programovacím jazyce je konečná posloupnost složená z *konečně mnoha* symbolů (písmena, číslice, mezery, speciální znaky, apod.) Nechť Σ je množina všech těchto symbolů. Množina všech programů je tedy jistě podmnožinou množiny $\bigcup_{i\in\mathbb{N}} \Sigma^i$, která je spočetně nekonečná. Existuje tedy bijekce f mezi množinou $\mathbb{N}$ a množinou všech programů. Pro každé $j \in \mathbb{N}$ označme symbolem P_j program $f(j)$. Pro *každý program* P tedy existuje $i \in \mathbb{N}$ takové, že $P = P_i$.
* Každý možný vstup každého možného programu lze zapsat jako *konečnou posloupnost* symbolů z konečné množiny Γ. Množina všech možných vstupů je tedy spočetně nekonečná a existuje bijekce g mezi množinou $\mathbb{N}$ a množinou všech vstupů. Pro každé $j \in \mathbb{N}$ označme symbolem V_j vstup $g(j)$.
* Předpokládejme, že *existuje program Halt*, který pro dané $i, j \in \mathbb{N}$ zastaví s výstupem *ano*/*ne* podle toho, zda P_i pro vstup V_j zastaví, nebo ne.
* Tento předpoklad dále dovedeme ke *sporu* dokazujícímu, že problém zastavení nemá algoritmické řešení.

Věta 12.6. *Neexistuje program Halt, který by pro vstup (P_i, V_j) správně rozhodl, zda se program P_i zastaví na vstupu V_j.*

`Důkaz`: Sporem uvažme program *Diag* s následujícím kódem:

```
input k;
if  Halt(k,k) = ano  then  while true do ; done
```

(Program $Diag(k)$ má na rozdíl od *Halt* jen jeden vstup k, což bude důležité.)

Fungování programu *Diag* lze znázornit za pomocí následující tabulky:

	P_0	P_1	P_2	P_3	$\cdots$
V_0	$\surd$	$-$	$-$	$\surd$	$\cdots$
V_1	$\surd$	$-$	$-$	$\surd$	$\cdots$
V_2	$-$	$\surd$	$-$	$\surd$	$\cdots$
V_3	$-$	$-$	$\surd$	$\surd$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Symbol $\surd$ resp. $-$ říká, že program uvedený v záhlaví sloupce zastaví resp. nezastaví pro vstup uvedený v záhlaví řádku. Program *Diag* „*zneguje*“ diagonálu této tabulky.

Podle našeho předpokladu (*Diag* je program a posloupnost P_i zahrnuje *všechny programy*) existuje $j \in \mathbb{N}$ takové, že $Diag = P_j$. Zastaví *Diag* pro vstup V_j?

– *Ano*. Podle kódu *Diag* pak ale tento program vstoupí do nekonečné smyčky, tedy nezastaví.

– *Ne*. Podle kódu *Diag* pak ale `if` test neuspěje, a tento program tedy zastaví.

Předpoklad existence programu *Halt* tedy vede ke sporu. □

`Komentář:` Otázkami algoritmické (ne)řešitelnosti problémů se zabývá *teorie vyčíslitelnosti*. Metoda diagonalizace se také často využívá v *teorii složitosti* k důkazu toho, že dané dvě složitostní třídy jsou různé.

Rozšiřující studium

Látka této lekce zabrousila až do teoretických hlubin matematické logiky a teorie množin. Další studium v těchto oblastech se dá očekávat hlavně u studentů specificky zaměřených teoretickým směrem (a mířících spíše do akademické než aplikační sféry), zajímajících se o matematiku samotnou nebo o teorii vyčíslitelnosti. Proto také uvedené pokročilé poznatky Lekce 12 nebudou vyžadovány u zkoušky tohoto předmětu.

Závěrem

Gratulujeme všem, kteří se naším nelehkým učebním textem „prokousali“ až sem, a přejeme mnoho úspěchů v dalším studiu informatiky.

Nakonec znovu připomínáme, že nedílnou součástí našeho studijního textu je Interaktivní osnova předmětu IB000 v IS MU a v ní přiložené online odpovědníky určené k procvičování přednesené látky.

`http://is.muni.cz/el/1433/podzim2023/IB000/index.qwarp`