

1 Základní formalismy: Důkaz a Algoritmus

Studium informatiky neznámá jen „naučit se nějaký programovací jazyk“, nýbrž zahrnuje celý soubor dalších relevantních předmětů, mezi nimiž najdeme i matematicko-teoretické (formální) základy moderní informatiky.

Právě odborný nadhled nad celou informatikou včetně nezbytné formální teorie odliší řadového „programátora“, kterých jsou dnes spousty i bez VŠ vzdělání, od skutečného a mnohem lépe placeného IT experta. □

Stručný přehled lekce

- * Pochopení formálního zápisu a významu matematických tvrzení (vět) a jejich důkazů.
- * Rozbor logické struktury matematických vět, potřebné úrovně formality a diskuze konstruktivnosti důkazů.
- * Správný formální zápis postupu – algoritmu, ve světle matematických formalismů.

1.1 Úvod do matematického dokazování

Matematika (a tudíž i teoretická informatika jako její součást) se vyznačuje **velmi přísnými** formálními požadavky na korektnost argumentace. □

- 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**. □

O potřebné úrovni formality matematických důkazů a o běžných důkazových technikách se dozvíme dále v této a příští lekci. . .

Nyní si prostě celou problematiku uvedeme názornými příklady.

Příklad 1.2. 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 $2k$.
- **Liché** číslo je celé číslo nedělitelné 2, tj. tvaru $2k + 1$. \square

Důkaz postupuje v následujících **formálních krocích**:

| tvrzení | zdůvodnění |
|----------------------------------|------------------------------------------------------|
| 1) $a = 2k + 1$, k celé | předpoklad |
| 2) $b = 2l + 1$, l celé | předpoklad \square |
| 3) $x = a + b = 2k + 2l + 1 + 1$ | 1,2) a komutativita sčítání (axiom) \square |
| 4) $x = 2(k + l) + 2 \cdot 1$ | 3) a distributivnost násobení \square |
| 5) $x = 2(k + l + 1)$ | 4) a opět distributivnost násobení \square |
| 6) $x = 2m$, m celé | 5) a $m = k + l + 1$ je celé číslo (axiom) \square |

Příklad 1.3. Dokažte následující tvrzení:

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

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

- Necht' $z = \frac{x+y}{2} = x + \frac{y-x}{2} = y - \frac{y-x}{2}$.
- Číslo z je racionální, neboť x a y jsou racionální.
- Platí $z > x$, neboť $\frac{y-x}{2} > 0$.
- Dále platí $z < y$, opět neboť $\frac{y-x}{2} > 0$.
- Celkem $x < z < y$. \square

\square

Poznámka. Alternativní formulace Věty z Příkladu 1.3 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.

1.2 Význam matematických vět

- První krok formálního důkazu je uvědomit si, **co říká věta**, která se má dokázat; tedy co je **předpoklad** a co **závěr** dokazovaného tvrzení.

Pravdivost takového tvrzení pak je třeba chápat v následujícím významu:

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

- Příklady běžné formulace matematických vět:
 - * Konečná množina má konečně mnoho podmnožin.□
 - * $\sin^2(\alpha) + \cos^2(\alpha) = 1$.□
 - * Graf je rovinný jestliže neobsahuje podrozdělení K_5 nebo $K_{3,3}$.□
- Často pomůže pouhé rozepsání definic pojmů, které se v dané větě vyskytují.
- Všimněte si také, jaký je správný logický význam matematického tvrzení vyloveného touto formou **implikace** („jestliže . . . , pak . . .“).
Především, pokud **předpoklady nejsou splněny** nebo jsou sporné, tak celé **tvrzení je platné** bez ohledu na pravdivost závěru!

Příklad 1.4. 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 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. □

1.3 Struktura matematických vět a důkazů

Jak „moc formální“ mají správné matematické důkazy vlastně být?

- Záleží na tom, 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. □

A jak na správný matematický důkaz mohu přijít?

- No. . . , □nalézání matematických důkazů je tvůrčí činnost, která není vůbec snadná a vyžaduje od vás přímo „umělecké“ vlohy. Přesto se jí alespoň trochu musíte naučit.

Konstruktivní a existenční důkazy

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

- Důkaz Věty 1.3 je *konstruktivní*. Dokázali jsme nejen, že číslo z existuje, ale podali jsme také návod, jak ho pro dané x a y *sestrojit*.
- *Existenční* důkaz je takový, kde se prokáže existence nějakého objektu *bez toho*, aby byl podán návod na jeho konstrukci. □

Příklad 1.5. čistě *existenčního* důkazu.

Věta. *Existuje program, který vypíše na obrazovku čísla tažená ve 45. tahu sportky v roce 2010.* □

Důkaz: Existuje pouze konečně mnoho možných výsledků losování 45. tahu sportky v roce 2010. Pro každý možný výsledek *existuje* program, který tento daný výsledek vypíše na obrazovku. Mezi těmito programy je tedy jistě ten, který vypíše právě ten výsledek, který bude ve 45. tahu sportky v roce 2010 skutečně vylosován. □

To je ale „*podvod*“, že? □ A přece *není*...
Formálně správně to je prostě tak a tečka.

Fakt. Pro infromatické i další aplikované disciplíny je **důležitá konstruktivnost** důkazů vět, které se zde objevují. □

V matematice ale jsou mnohé příklady užitečných a nenahraditelných existenčních důkazů, třeba tzv. **pravděpodobnostní důkazy**.

Příklad 1.6. „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 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

$$\underbrace{2^{1-n} + \dots + 2^{1-n}}_{n^2} = \frac{n^2}{2^{n-1}} \rightarrow 0.$$

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

1.4 Formální popis algoritmu

Položme si otázku, co je vlastně algoritmus?

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. □

Konvence 1.7. Zjednodušeně zde *algotmem* rozumíme konečnou posloupnost elementárních výpočetních *kroků*, ve které každý další krok *vhodně* 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. □

Vidíte, jak blízké si jsou konstruktivní matematické důkazy a algoritmy v našem pojetí? Jedná se nakonec o jeden ze záměrů našeho přístupu. . .

Příklad 1.8. *Zápis algoritmu pro výpočet průměru z daného pole $a[]$ s n prvky.*

- Inicializujeme $sum \leftarrow 0$;
- postupně pro $i=0,1,2,\dots,n-1$ provedeme
 - * $sum \leftarrow sum+a[i]$;
- vypíšeme podíl (sum/n) . □

Ve „vyšší úrovni“ formálnosti (s jasnějším vyznačením *řídících struktur* algoritmu) se totéž dá zapsat jako:

Algoritmus 1.9. Průměr

z daného pole $a[]$ s n prvky.

```
sum  $\leftarrow$  0;
foreach i  $\leftarrow$  0,1,2,...,n-1 do
    sum  $\leftarrow$  sum+a[i];
done
res  $\leftarrow$  sum/n;
output res;
```

Značení. Pro potřeby symbolického formálního zápisu algoritmů v předmětu FI: IB000 si zavedeme následovnou konvenci:

- *Proměnné* nebudeme deklarovat ani typovat, pole odlišíme závorkami `p[]`.
- *Přiřazení* hodnoty zapisujeme `a ← b`, případně `a := b`, ale nikdy **ne** `a=b`.
- Jako elem. 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 řídicí 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 řídicích struktur (což jsou `if`, `foreach`, `while`).
- Pokud je to dostatečně jasné, elementární operace nebo podmínky můžeme i ve formálním zápise **popsat běžným jazykem**.

Malé srovnání závěrem

Jak to tedy je s vhodnou a únosnou mírou formalizace u matematických důkazů i u zápisu algoritmů? □

| | zcela formální | běžná úroveň |
|--------------|-------------------------------------------------------------|---------------------------------------------------------------------------------|
| matematika | formální rozepsání všech elem. kroků (Příklad 1.2) | strukturovaný a matem. přesný text v běžném jazyce |
| algoritmy | rozepsání všech elem. kroků ve výpočetním modelu | strukturovaný rozpis kroků (Algoritmus 1.9), i s využitím běžného jazyka |
| programování | assembler / strojový kód (kde se s ním dnes běžně setkáte?) | „vyšší“ (strukturované) programovací jazyky, například Java |

□
Pochopitelně se ve všech třech bodech obvykle držíme druhého přístupu, tedy běžné úrovně formality, pokud si specifické podmínky výslovně nevyžadují přístup nejvyšší formality...