NSWI183: Sémantika programů

2. Indukce

Jan Kofroň

MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

Department of
Distributed and
Dependable
Systems

Indukce

- Jeden ze základních způsobů dokazování platnosti matematických tvrzení
- První zmínky o použití tohoto principu sahají až k Platónovi (370 př.n.l.) a Euklidovi (300 př.n.l.)
- Existuje několik typů indukce:
  - matematická indukce (stepwise induction)
  - úplná indukce (complete induction)
  - fundovaná indukce (well-founded induction)

Matematická indukce

Základní princip spočívá v tom, že dokážeme:

1. Tvrzení platí pro první (nultý) prvek (indukční předpoklad)
2. Když tvrzení platí pro prvek n, pak platí pro prvek $n + 1$ (indukční krok)

Neboli:

$$(F[0] \land \forall n.\, (F[n] \to F[n+1])) \to \forall n.\, F[n]$$

Součástí axiomů Peanovy a Presburgerovy aritmetiky

PŘÍKLAD

Předpokládejme rozšíření Peanovy aritmetiky o následující axiomy:

$$\forall x.\, x^0 = 1$$
$$\forall x, y.\, x^{y+1} = x^y \cdot x$$
$$\forall x, z.\, exp_3(x, 0, z) = z$$
$$\forall x, y, z.\, exp_3(x, y+1, z) = exp_3(x, y, x \cdot z)$$

Chceme dokázat následující formuli:

$$\forall x, y.\, exp_3(x, y, 1) = x^y$$

Příklad

$$\forall x, y.\, exp_3(x, y, 1) = x^y$$

- Zdá se, že pro indukci je lepší zvolit y než x vzhledem k axiomům o exp_3
- Budeme tedy dokazovat, že platí následující formule $F[y]$:

$$F[y] : \forall x.\, exp_3(x, y, 1) = x^y$$

PŘÍKLAD

$$F[y] : \forall x.\, exp_3(x, y, 1) = x^y$$

- Jako indukční předpoklad tedy dokážeme formuli $F[0] : \forall x.\, exp_3(x, 0, 1) = x^0$
- Podle axiomů je $x^0 = 1$ a $exp_3(x, 0, 1) = 1$, tedy máme první část hotovou
- Předpokládejme nyní, že formule $F[n]$ platí pro libovolné n a dokažme, že:

$$F[n+1] : \forall x.\, exp_3(x, n+1, 1) = x^{n+1}$$

PŘÍKLAD

$$F[n+1] : \forall x.\, exp_3(x, n+1, 1) = x^{n+1}$$

- Z axiomů dostáváme, že $exp_3(x, n+1, 1) = exp_3(x, n, x \cdot 1)$
- Indukční hypotézu ale nemůžeme použít ani na levou ani na pravou stranu rovnosti a dalším rozvinutím axiomů se pravděpodobně nedostaneme k cíli
- Rovnost ale zjevně platí, co je tedy špatně? Měli jsme pro indukci zvolit x místo y?

Příklad

- Někdy je třeba dokazované tvrzení zesílit, aby se dala aplikovat indukce
- Dokážeme tedy o něco silnější tvrzení, které zřejmě dokazuje původní formuli:

$$\forall x, y, z.\, exp_3(x, y, z) = x^y \cdot z$$

Příklad

$$\forall x, y, z.\, exp_3(x, y, z) = x^y \cdot z$$

- Opět zvolíme y jako indukční proměnnou a budeme dokazovat formuli $F[y]$:

$$F[y] : \forall x, z.\, exp_3(x, y, z) = x^y \cdot z$$

PŘÍKLAD

$$F[y] : \forall x, z.\, exp_3(x, y, z) = x^y \cdot z$$

- Jako indukční předpoklad tedy dokážeme formuli $F[0] : \forall x, z.\, exp_3(x, 0, z) = x^0 \cdot z$
- Z axiomů víme, že $exp_3(x, 0, z) = z$ a že $x^0 \cdot z = 1 \cdot z = z$, což nám dokazuje indukční předpoklad
- Předpokládejme nyní, že formule $F[n]$ platí pro libovolné n a dokažme, že:

$$F[n+1] : \forall x, z.\, exp_3(x, n+1, z) = x^{n+1} \cdot z$$

Příklad

$$F[n+1] : \forall x, z'.\, exp_3(x, n+1, z') = x^{n+1} \cdot z'$$

$$\begin{aligned} exp_3(x, n+1, z') &= exp_3(x, n, x \cdot z') && \text{(z axiomů)} \\ &= x^n \cdot (x \cdot z') && \text{(IH, } z \to x \cdot z') \\ &= x^{n+1} \cdot z' && \text{(z axiomů)} \end{aligned}$$

- To dokazuje formuli
- Substituce $x \cdot z$ za z je korektní, protože proměnná z je univerzálně kvantifikovaná

PONAUČENÍ Z PŘÍKLADU

Pokud se nějaké tvrzení nedaří dokázat,
může být paradoxně snazší dokázat tvrzení **silnější.**

Úplná indukce

Úplná indukce využívá stejného principu jako v předchozím případě, ale může nám usnadnit dokazování

Pro Peanovu aritmetiku je definovaná následujícím schématem:

1. Předpokládejme, že tvrzení platí pro nějaké přirozené číslo n a pro všechna přirozená čísla menší než n (indukční předpoklad)
2. Dokážeme tvrzení obecně na základě indukčního předpokladu

Neboli:

$$(\forall n.\, (\forall n'.\, n' < n \to F[n']) \to F[n]) \to \forall x.\, F[x]$$

Fundovaná indukce

- Zobecněním úplné indukce (proto jsme u ní vynechali příklad)
- Využívá pojem **fundované relace:**

  Binární predikát $\prec$ nad množinou S se nazývá **fundovaná relace,** pokud *neexistuje* nekonečná posloupnost $s_1, s_2, s_3 \ldots$ prvků z S taková, že každý následující element je menší než jeho předchůdce, neboli $s_1 \succ s_2 \succ s_3 \ldots$, kde $s \prec t$ právě tehdy když $t \succ s$.

- Fundovaná indukce je pak zobecněním úplné indukce ve smyslu použití predikátu $\prec$ místo porovnání:

$$(\forall n.\, (\forall n'.\, n' \prec n \to F[n']) \to F[n]) \to \forall x.\, F[x]$$

Příklad

Uvažme následující úlohu: Máme pytel s červenými, žlutými a modrými žetony. Pokud je v pytli poslední žeton, můžeme ho vytáhnout. V opačném případě vyjmeme dva žetony a postupujeme následujícím způsobem (máme k dispozici libovolně mnoho žetonů):

1. Pokud je jeden ze dvou právě vyjmutých žetonů červený, nevložíme zpátky žádný žeton.
2. Pokud jsou oba žetony žluté, vložíme do pytle jeden žlutý a pět modrých žetonů.
3. Pokud je jeden z žetonů modrý a druhý **není** červený, vložíme do pytle deset červených.

Tento postup specifikuje všechny kombinace pro vytažené žetony. Úloha spočívá v tom rozhodnout, jestli tento proces je vždy konečný, tedy jestli bude pytel bez ohledu na počáteční obsah nakonec prázdný.

PŘÍKLAD

Při aplikaci fundované indukce je třeba nejprve definovat odpovídající relaci

Nechť trojice (#žlutých, #modrých, #červených) označuje stav pytle
(počty jednotlivých barevných žetonů uvnitř)

Pro naši úlohu definujeme $\prec_3$ jako lexikografické uspořádání takových trojic

- Například: $(11, 13, 3) \not\prec_3 (11, 9, 104)$, ale $(11, 9, 104) \prec_3 (11, 13, 3)$

Ukážeme, že pro libovolný stav pytle zbývá v algoritmu jen konečný počet kroků než skončí

PŘÍKLAD

Rozborem případů ukážeme, že pro libovolný stav pytle zbývá v algoritmu jen konečný počet kroků než skončí:

A. Pokud je pytel prázdný, algoritmus skončil
B. Pokud pytel obsahuje právě jeden žeton, tedy jeho stav je (1, 0, 0), (0, 1, 0), nebo (0, 0, 1), zbývá jeden krok algoritmu
C. Předpokládejme jako indukční předpoklad, že pro jakýkoliv stav (ž' ,m' ,č') takový, že (ž', m', č') $\prec_3$ (ž, m, č), zbývá jen konečný počet kroků algoritmu
D. Nyní předpokládejme, že stav je (ž, m, č) a vyjmeme dva žetony, dostáváme tři možné případy

Příklad

D. Nyní předpokládejme, že stav je (ž, m, č) a vyjmeme dva žetony. Dostáváme tři možné případy:

1. Jeden žeton je červený, tedy stav po odebraní je (ž–1, m, č–1), (ž, m–1, č–1), nebo (ž, m, č–2). Všechny tyto stavy jsou v relaci $\prec_3$ menší než (ž, m, č).
2. Oba odebrané žetony jsou žluté a vrátíme jeden žlutý a pět modrých zpět do pytle. Nový stav bude tedy (ž–1, m+5, č), což je v relaci $\prec_3$ menší než (ž, m, č).
3. Jeden žeton je modrý a druhý není červený a vrátíme tedy deset červených zpět. Nový stav je tedy buď (ž–1, m–1, č+10) nebo (ž, m–2, č+10). Oba tyto stavy jsou v relaci $\prec_3$ menší než (ž, m, č).

Ve všech případech můžeme aplikovat indukční předpoklad a dostáváme, že zbývá jen konečný počet kroků algoritmu. Algoritmus tedy vždy zastaví.

Příklad – co bylo těžké?

- Vhodně zvolit reprezentaci stavu
- Najít vhodnou (fundovanou) relaci
- Zbytek je pak mechanická práce

STRUKTURÁLNÍ INDUKCE

- Vhodná pro dokazování vlastností struktur, například samotných formulí, stromů, a rekurzivních datových struktur obecně
- Jako indukční předpoklad předpokládáme, že dokazované tvrzení platí pro libovolnou formuli a všechny její vlastní podformule
  - Pro stromy uvažujeme podstromy, pro obecné grafy podgrafy, …
- Protože atomy žádné podformule nemají, tvoří základní případy

PŘÍKLAD

```
@pre  0 <= l && u < |a|
@post rv <-> exists j. (l <= j && j <= u && a[j] = e)
bool LinearSearch(int[] a, int l, int u, int e) {
    for
        @ L: ???
        (int i := l; i <= u; i := i + 1)
    {
        if (a[i] = e)
            return true;
    }
    return false;
}
```

PŘÍKLAD

```
@pre  0 <= l && u < |a|
@post rv <-> exists j. (l <= j && j <= u && a[j] = e)
bool LinearSearch(int[] a, int l, int u, int e) {
    for
        @ L: ???
        (int i := l; i <= u; i := i + 1)
    {
        if (a[i] = e)
            return true;
    }
    return false;
}
```

PŘÍKLAD

```
@pre  0 <= l && u < |a|
@post rv <-> exists j. (l <= j && j <= u && a[j] = e)
bool LinearSearch(int[] a, int l, int u, int e) {
    for
        @ L: l <= i
        (int i := l; i <= u; i := i + 1)
    {
        if (a[i] = e)
            return true;
    }
    return false;
}
```

PŘÍKLAD

```
@pre  o <= l && u < |a|
@post rv <-> exists j. (l <= j && j <= u && a[j] = e)
bool LinearSearch(int[] a, int l, int u, int e) {
    for
        @ L: l <= i && o <= l && u < |a| &&
        forall j.(l <= j && j < i -> a[j] != e)
        (int i := l; i <= u; i := i + 1)
    {
        if (a[i] = e)
            return true;
    }
    return false;
}
```

Shrnutí

- Různé druhy indukce nám umožňují lépe dokazovat vlastnosti programů v různých případech
- Fundovaná indukce nevyžaduje postupovat přes celá čísla, ale používá fundovanou relaci, jejíž celočíselné porovnání je speciální případ
- Strukturální indukce umožňuje dokazovat vlastnosti samotných formulí nebo datových struktur

K ZAMYŠLENÍ

Uvažme změnu pravidel u hry s žetony. Zastaví algoritmus, pokud:

A. V kroku 1 vrátíme jeden červený žeton zpět?

B. V kroku 1 přidáme jeden modrý žeton?

C. V kroku 1 přidáme modrý žeton a v kroku 3 vrátíme jeden modrý žeton zpět, ale žádné jiné žetony?