9 Jednoduchý deklarativní jazyk

Nyní se vracíme a pokračujeme dále v tématu předchozí lekce, tj. budeme se zabývat matematickým dokazováním vlastností a správnosti algoritmů. Třebaže mnohým mohla přijít už Lekce 8 více než dost formální, není tomu úplně tak; v následujícím si ukážeme (ještě) přesnější přístup založený na myšlenkách funkcionálního programování. Zároveň si tak i procvičíme správné chápání a používání matematických definic.

$$
\begin{array}{llllll}
f(\mathbf{3}) & \mapsto & \text{if } \mathbf{3} \text{ then } \mathbf{3} * f(\mathbf{3}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi} & \mapsto & \mathbf{3} * f(\mathbf{3}-\mathbf{1}) & \mapsto \\
\mathbf{3} * f(\mathbf{2}) & \mapsto & \mathbf{3} * (\text{if } \mathbf{2} \text{ then } \mathbf{2} * f(\mathbf{2}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi}) & \mapsto & \mathbf{3} * (\mathbf{2} * f(\mathbf{2}-\mathbf{1})) & \mapsto \\
\mathbf{3} * (\mathbf{2} * f(\mathbf{1})) & \mapsto & \mathbf{3} * (\mathbf{2} * (\text{if } \mathbf{1} \text{ then } \mathbf{1} * f(\mathbf{1}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi})) & \mapsto & \mathbf{3} * (\mathbf{2} * (\mathbf{1} * f(\mathbf{1}-\mathbf{1}))) & \mapsto \\
\mathbf{3} * (\mathbf{2} * (\mathbf{1} * f(\mathbf{0}))) & \mapsto & \mathbf{3} * (\mathbf{2} * (\mathbf{1} * (\text{if } \mathbf{0} \text{ then } \mathbf{0} * f(\mathbf{0}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi}))) & \mapsto & \mathbf{3} * (\mathbf{2} * (\mathbf{1} * \mathbf{1})) & \mapsto \\
\mathbf{3} * (\mathbf{2} * \mathbf{1}) & \mapsto & \mathbf{3} * \mathbf{2} & \mapsto & \mathbf{6} &
\end{array}
$$

□

Stručný přehled lekce

* Zavedení jednoduchého deklarativního jazyka, jeho formální přínos.
* Formalizace pojmu „výpočet“ z hlediska našeho jazyka.
* Několik konkrétních zápisů algoritmů a jejich vyhodnocení / důkaz.

O „správnosti“ programů, podruhé

Vraťme se zpět – jak se máme přesvědčit, že program funguje „správně“?

- V případech, kdy je třeba mít naprostou jistotu správné funkce, je jedinou „dostatečně spolehlivou“ možností podat formální matematický důkaz chování algoritmu.
  - Úplné matematické důkazy pro svou složitost dokáží obsáhnout pouze „malé“ algoritmy.
  - Přesto je jejich přesné dokazování důležité, neboť právě malé algoritmy obvykle tvoří stavební kameny rozsáhlejších systémů (kde již pak nastupují jiné metody verifikace).

- A co tedy důkazy vlastností symbolicky zapsaných (procedurálních) algoritmů z Lekce 8? Všimli jste si, co v nich bylo problematickým bodem?
  - Náš procedurální zápis algoritmu totiž přesně nedefinuje, co je to „elementární krok“ výpočtu – to je sice většinou docela zřejmé, někdy však může hlavní problém nastat právě zde.
  - Sice by bylo možné použít k definici některý z přesných teoretických modelů výpočtu jako je *Turingův stroj* (nebo třeba i některý vhodný z reálných programovacích jazyků), avšak pak by se formální důkazy staly velmi složitými.
- Vhodnějším řešením (pro potřeby formálního dokazování) se jeví příklon k „funkcionálnímu“ zápisu algoritmů pomocí matematicky zcela přesných *deklarací*.

9.1 Popis jednoduchého deklarativního jazyka

Definice 9.1. Deklarativní programovací jazyk (pro přednášky FI: IB000).

- Nechť $Var = \{x, y, z, \dots\}$ je spočetná množina *proměnných*.
- Nechť $Num = \{\mathbf{0}, \mathbf{1}, \dots \mathbf{52}, \dots \mathbf{397}, \dots\}$ je množina všech dekadických zápisů přirozených čísel.
- Nechť $Fun = \{f, g, h, \dots\}$ je spočetná množina *funkčních symbolů*. Ke každému $f \in Fun$ je přiřazeno číslo $a \in \mathbb{N}$, které nazýváme *arita* f.
  Dále předpokládáme, že pro každé $a \in \mathbb{N}$ existuje nekonečně mnoho $f \in Fun$ s aritou a.
- Množina **výrazů** Exp je (induktivně) definována následující abstraktní syntaktickou rovnicí:

$$\begin{array}{rcl} E & ::= & x \mid \mathbf{n} \\ & \mid & E_1 + E_2 \mid E_1 - E_2 \mid E_1 * E_2 \mid E_1 \div E_2 \mid (E_1) \\ & \mid & f(E_1, \cdots, E_a) \\ & \mid & \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \text{ fi} \end{array}$$

V uvedené rovnici je $x \in Var$, $\mathbf{n} \in Num$, $f \in Fun$ a $a \in \mathbb{N}$ je arita f.

Poznámka: Dobře si povšimněte, že ve smyslu Lekce 6 tato induktivní Definice 9.1 není jednoznačná. Neboli jeden výraz, jako třeba $a + 1 - 2 * 3$ lze odvodit několika různými způsoby. Praktickým důsledkem pak je, že není zřejmá „priorita operací“, což lze na druhou stranu ale vždy snadno napravit přidáním dostatečného počtu závorek. □

Tuto volnost syntaxe ponecháváme z čistě pragmatického důvodu snadnějšího méně formálního zápisu výrazů. Nakonec bude priorita operací přesně určena Definicí 9.2. □

Pro lepší pochopení uvádíme několik příkladů výrazů (Exp) z definice.

- $\mathbf{254}$
- $\mathbf{2 + 3 * 4}$ □
- $f(\mathbf{2}) \div g(\mathbf{5})$
- $f(\mathbf{2} + x, g(y, \mathbf{3} * y))$ □
- if $x - \mathbf{1}$ then $(\mathbf{2} + f(y))$ else $g(x, x)$ fi (čtěme „if $x - 1 \neq 0 \ldots$“)

Co je deklarace

Definice: *Deklarace* (v jazyce Definice 9.1) je konečný systém rovnic tvaru

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_{f_1} \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_{f_n} \end{array},$$

kde pro každé $1 \leq i \leq n$ platí, že $f_i \in Fun$ je funkce arity a_i, že $x_1, \ldots, x_{a_i} \in Var$ jsou proměnné a E_i je výraz, v němž se mohou vyskytovat pouze proměnné $x_1, \ldots, x_{a_i}$ a funkční symboly $f_1, \ldots, f_n$. □

Opět uvádíme pro osvětení několik příkladů deklarací z naší definice.

- $f(x) = \text{if } x \text{ then } x * f(x - \mathbf{1}) \text{ else } \mathbf{1} \text{ fi}$ □
- $\begin{array}{rcl} f(x) & = & g(x - \mathbf{1}, x) \\ g(x, y) & = & \text{if } x \text{ then } f(y) \text{ else } \mathbf{3} \text{ fi} \end{array}$

  Později uvidíme, proč a jak je konvencí našich výpočtů tvrdit $0 - 1$ "=" 0. □
- $g(x, y) = \text{if } x - y \text{ then } x \text{ else } y \text{ fi}$ □
- $f(x) = f(x)$ □ (Nezapisuje toto náhodou „nekonečnou smyčku"?)

9.2 Formalizace pojmu „výpočet“

Za výpočet budeme považovat posloupnost úprav výrazů, které jsou „postaveny“ na naší uvažované deklaraci Δ. To je formálně podchyceno v následujících dvou definicích, které si porovnejte také s neformálním pojetím algoritmů jako konečných posloupností elementárních kroků v Oddíle 1.4. □

- Pro připomenutí... *Deklarace* je konečný systém rovnic tvaru

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_{f_1} \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_{f_n} \end{array} .\square$$

Definice: Nechť Δ je deklarace. Symbolem $Exp(\Delta)$ označíme množinu všech výrazů E, které splňují zároveň tyto dvě podmínky:

- E neobsahuje žádnou proměnnou.□
- Jestliže E obsahuje funkční symbol f, pak f byl v Δ deklarován.

Výrazy nad deklarací

Definice: Nechť Δ je deklarace. Symbolem $Exp(\Delta)$ označíme množinu všech výrazů E, které splňují zároveň tyto dvě podmínky:

- E neobsahuje žádnou proměnnou.
- Jestliže E obsahuje funkční symbol f, pak f byl v Δ deklarován.

Dívejte se na množinu $Exp(\Delta)$ jako na soubor „platných vstupů“ (jako u programu) pro deklaraci Δ, nad kterými bude prováděn výpočet.

Fakt: Množinu $Exp(\Delta)$ lze podle Definice 9.1 zadat také induktivně:

$$\begin{array}{rcl} E & ::= & \mathbf{n} \mid (E_1) \mid E_1 + E_2 \mid E_1 - E_2 \mid E_1 * E_2 \mid E_1 \div E_2 \\ & \mid & f(E_1, \cdots, E_a) \mid \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \text{ fi} \end{array}$$

V uvedené zápise je $\mathbf{n} \in Num$, $f \in Fun$ je funkční symbol deklarovaný v Δ a $a \in \mathbb{N}$ je arita tohoto f.

Jednotlivý krok výpočtu

Definice 9.2. Výpočet a krok výpočtu v našem deklarativním jazyce. Relaci „krok výpočtu“ $\mapsto \subseteq Exp(\Delta) \times Exp(\Delta)$ definujeme induktivně; místo $(E, F) \in \mapsto$ budeme psát $\boldsymbol{E \mapsto F}$, neboť se pro naše potřeby bude zhusta jednat o funkci.

i) $\mathbf{n} \mapsto \mathbf{n}$ pro každé $\mathbf{n} \in Num$. □

ii) Pro $\underline{\boldsymbol{E \equiv (E_1)}}$ definujeme krok výpočtu takto:

- Jestliže $E_1 \mapsto F \in Num$, pak $(E_1) \mapsto F$.
- Jestliže $E_1 \mapsto F \notin Num$, pak $(E_1) \mapsto (F)$. □

iii) Pro $\underline{\boldsymbol{E \equiv E_1 + E_2}}$ definujeme krok výpočtu takto:

- Jestliže $E_1, E_2 \in Num$, pak $E_1 + E_2 \mapsto \mathbf{z}$, kde $\mathbf{z}$ je dekadický zápis čísla $E_1 + E_2$.
- Jestliže $E_1 \notin Num$ a $E_1 \mapsto F$, pak $E_1 + E_2 \mapsto F + E_2$.
- Jestliže $E_1 \in Num$ a $E_2 \notin Num$ a $E_2 \mapsto F$, pak $E_1 + E_2 \mapsto E_1 + F$.

iv) Pro $\underline{\boldsymbol{E \equiv E_1 - E_2}}$ definujeme krok výpočtu takto:

- Jestliže $E_1, E_2 \in Num$, pak $E_1 - E_2 \mapsto \mathbf{z}$, kde $\mathbf{z}$ je dekadický zápis čísla $\max\{0, E_1 - E_2\}$. (Pozor na nezápornost výsledku odčítání!)
- Jestliže $E_1 \notin Num$ a $E_1 \mapsto F$, pak $E_1 - E_2 \mapsto F - E_2$.
- Jestliže $E_1 \in Num$ a $E_2 \notin Num$ a $E_2 \mapsto F$, pak $E_1 - E_2 \mapsto E_1 - F$. □

v) Pro $\underline{\boldsymbol{E \equiv E_1 * E_2}}$ definujeme krok výpočtu takto:

- Jestliže $E_1, E_2 \in Num$, pak $E_1 * E_2 \mapsto \mathbf{z}$, kde $\mathbf{z}$ je dekadický zápis čísla $E_1 * E_2$.
- Jestliže $E_1 \notin Num$ a $E_1 \mapsto F$, pak $E_1 * E_2 \mapsto F * E_2$.
- Jestliže $E_1 \in Num$ a $E_2 \notin Num$ a $E_2 \mapsto F$, pak $E_1 * E_2 \mapsto E_1 * F$. □

vi) Pro $\underline{\boldsymbol{E \equiv E_1 \div E_2}}$ definujeme krok výpočtu takto:

- Jestliže $E_1, E_2 \in Num$, pak $E_1 \div E_2 \mapsto \mathbf{z}$, kde $\mathbf{z}$ je dekadický zápis (celé části) čísla $\lfloor E_1/E_2 \rfloor$. Pokud $E_2 \equiv \mathbf{0}$, je $\mathbf{z} \equiv \mathbf{0}$ (dělení nulou!).
- Jestliže $E_1 \notin Num$ a kde $E_1 \mapsto F$, pak $E_1 \div E_2 \mapsto F \div E_2$.
- Jestliže $E_1 \in Num$ a $E_2 \notin Num$ a $E_2 \mapsto F$, pak $E_1 \div E_2 \mapsto E_1 \div F$.

vii) Pro $\boldsymbol{E \equiv \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \text{ fi}}$ definujeme krok výpočtu takto:

- Jestliže $E_1 \in Num$ a $E_1 \equiv \mathbf{0}$, pak if E_1 then E_2 else E_3 fi $\mapsto (E_3)$.
- Jestliže $E_1 \in Num$ a $E_1 \not\equiv \mathbf{0}$, pak if E_1 then E_2 else E_3 fi $\mapsto (E_2)$.
- Jestliže $E_1 \notin Num$ a $E_1 \mapsto F$, pak
  if E_1 then E_2 else E_3 fi $\mapsto$ if F then E_2 else E_3 fi. □

viii) Pro $\boldsymbol{E \equiv f(E_1, \cdots, E_k)}$ definujeme krok výpočtu takto:

- Jestliže $E_1, \cdots, E_k \in Num$, pak $f(E_1, \cdots, E_k) \mapsto (E_f(x_1 \restriction E_1, \cdots, x_k \restriction E_k))$
- Jinak $f(E_1, \cdots, E_k) \mapsto f(E_1, \cdots, E_{i-1}, F, E_{i+1}, \cdots, E_k)$, kde i je nejmenší index pro který platí $E_i \notin Num$ a $E_i \mapsto F$. □

V zápise jednotlivých bodů vždy platí, že $E_1, E_2, \cdots \in Exp(\Delta)$. Znak $\equiv$ zde znamená „textovou“ rovnost na množině výrazů Exp.

Při nejednoznačnosti vždy aplikujeme první použitelné pravidlo naší definice na prvním použitelném místě zleva. □

Definice: *Reflexivní a tranzitivní uzávěr* relace $\mapsto$ značíme $\mapsto^*$ („*výpočet*“).

Poznámky ke kroku výpočtu

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný – záporná jsou nahrazena nulou.□
  - Výsledek dělení je vždy celočíselný, počítáme jeho dolní celou část. □
  - Dělení nulou je definováno (není chybou), výsledkem je opět nula. □
- Další připomínka se týká pořadí vyhodnocování ve výrazu — to je přesně dáno pořadím pravidel v Definici 9.2, neboli vždy aplikujeme první pravidlo, které aktuálně lze použít na výraz E, a to na prvním možném místě zleva.□ Mimo jiné je takto „definována“ nejvyšší priorita vyhodnocení uzávorkovaného výrazu. □
- Uvědomte si dobře, že definice výpočetního kroku $\mapsto$ je (poněkud skrytě) rekurzivní. Třeba krok $(\mathbf{2} * \mathbf{1}) \mapsto \mathbf{2}$ je ve skutečnosti jediným krokem, přestože „vyvolá“ použití dvou pravidel v sobě – vyhodnocení součinu i odstranění závorek. □
- Ještě si uveďme, že naše definice připouští jistý nedeterminismus: Je možné mít v deklaraci Δ zadaných více rovnic pro tutéž funkci $f()$, pak však $\mapsto$ přestává být funkcí. My se touto možností nebudeme zabývat.

9.3 Příklady výpočtů a důkazů

Příklad 9.3. *Ukážeme si několik ilustrativních „výpočtů“ nad různými deklaracemi. Všimněte si, že při úpravách výrazů si dovolujeme (oproti formální definici) pro zpřehlednění vynechávat zdvojené a vnější závorky.*

- Uvažme deklaraci $f(x) = \text{if } x \text{ then } x * f(x-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi}$. Pak $f(\mathbf{3}) \mapsto^* \mathbf{6}$, neboť

$$
\begin{array}{lll}
f(\mathbf{3}) & \mapsto \text{if } \mathbf{3} \text{ then } \mathbf{3} * f(\mathbf{3}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi} & \mapsto \mathbf{3} * f(\mathbf{3}-\mathbf{1}) \mapsto \\
\mathbf{3} * f(\mathbf{2}) & \mapsto \mathbf{3} * (\text{if } \mathbf{2} \text{ then } \mathbf{2} * f(\mathbf{2}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi}) & \mapsto \mathbf{3} * (\mathbf{2} * f(\mathbf{2}-\mathbf{1})) \mapsto \\
\mathbf{3} * (\mathbf{2} * f(\mathbf{1})) & \mapsto \mathbf{3} * (\mathbf{2} * (\text{if } \mathbf{1} \text{ then } \mathbf{1} * f(\mathbf{1}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi})) & \mapsto \mathbf{3} * (\mathbf{2} * (\mathbf{1} * f(\mathbf{1}-\mathbf{1}))) \mapsto \\
\mathbf{3} * (\mathbf{2} * (\mathbf{1} * f(\mathbf{0}))) & \mapsto \mathbf{3} * (\mathbf{2} * (\mathbf{1} * (\text{if } \mathbf{0} \text{ then } \mathbf{0} * f(\mathbf{0}-\mathbf{1}) \text{ else } \mathbf{1} \text{ fi}))) & \mapsto \mathbf{3} * (\mathbf{2} * (\mathbf{1} * \mathbf{1})) \mapsto \\
\mathbf{3} * (\mathbf{2} * \mathbf{1}) & \mapsto \mathbf{3} * \mathbf{2} & \mapsto \mathbf{6}\,.\ \square
\end{array}
$$

- Uvažme deklaraci $f(x) = g(x-\mathbf{1}, x)$ a $g(x,y) = \text{if } x \text{ then } f(y) \text{ else } \mathbf{3} \text{ fi}$. Pak

$$f(\mathbf{3}) \mapsto g(\mathbf{3}-\mathbf{1}, \mathbf{3}) \mapsto g(\mathbf{2}, \mathbf{3}) \mapsto \text{if } \mathbf{2} \text{ then } f(\mathbf{3}) \text{ else } \mathbf{3} \text{ fi} \mapsto f(\mathbf{3})\,.\ \square$$

- Uvažme deklaraci $f(x) = f(x)$. Pak pro každé $\mathbf{n} \in Num$ platí

$$f(\mathbf{n}) \mapsto f(\mathbf{n})$$

a podobně $f(f(\mathbf{n})) \mapsto f(f(\mathbf{n}))$. Ale $f(f(\mathbf{2}+\mathbf{3})) \mapsto f(f(\mathbf{5})) \mapsto f(f(\mathbf{5}))$. $\square$

O pořadí provádění operací ve výrazu

Příklad 9.4. *Jak bude přesně vyhodnocen následující výraz?*

$$\mathbf{1 + 2 - 3 + 4 - 5}\square$$

Řešení: Klíčem k řešení tohoto je správné pochopení Definice 9.2. Prvním použitelným pravidlem našeho deklarativního jazyka je to pro $E \equiv E_1 + E_2$. Je použito na prvním místě zleva, tj. pro $E_1 \equiv 1$ a $E_2 \equiv 2 - 3 + 4 - 5$. Podle definice je tedy nutno upravit $E_1 \in Num$ i $E_2 \notin Num$, neboli definici aplikovat (rekurzivně) na výraz E_2. $\square$

Při vyhodnocování $E_2 = E' \equiv 2 - 3 + 4 - 5$ je prvním použitelným pravidlem opět to pro $E' \equiv E'_1 + E'_2$, přičemž $E'_1 \equiv 2 - 3$ a $E'_2 \equiv 4 - 5$. Podle definice je nutno v tomto místě vyhodnotit výraz $E'_1 \mapsto 0$. Celkem v prvním kroce

$$1 + 2 - 3 + 4 - 5 \;\mapsto\; 1 + \;0\; + 4 - 5.\square$$

Nakonec stejným postupem získáme:

$$1 + 0 + 4 - 5 \mapsto 1 + 0 + 0 \mapsto 1 + 0 \mapsto 1$$

$\square$

Důkaz správnosti programu

Příklad 9.5. *Pro ukázku uvažme deklaraci Δ obsahující pouze rovnici*

$$f(x) = \text{if } x \text{ then } x * f(x - \mathbf{1}) \text{ else } \mathbf{1} \text{ fi}\,.$$

Věta. Pro každé $n \in \mathbb{N}$ platí $f(\mathbf{n}) \mapsto^* \mathbf{m}$, kde $\mathbf{m} \equiv n!$.□

Důkaz povedeme indukcí podle n:

- Báze $n = 0$. Platí $f(\mathbf{0}) \mapsto \text{if } \mathbf{0} \text{ then } \mathbf{0} * f(\mathbf{0} - \mathbf{1}) \text{ else } \mathbf{1} \text{ fi} \mapsto \mathbf{1}$ a také $0! = 1$.□
- Indukční krok. Nechť $n + 1 \equiv \mathbf{k}$. Pak

  $$f(\mathbf{k}) \mapsto \text{if } \mathbf{k} \text{ then } \mathbf{k} * f(\mathbf{k} - \mathbf{1}) \text{ else } \mathbf{1} \text{ fi} \mapsto \mathbf{k} * f(\mathbf{k} - \mathbf{1}) \mapsto \mathbf{k} * f(\mathbf{w})\,,$$

  kde $\mathbf{w} \equiv k - 1 = n$. □Podle I.P. platí $f(\mathbf{w}) \mapsto^* \mathbf{u}$, kde $\mathbf{u} \equiv n!$. Proto $\mathbf{k} * f(\mathbf{w}) \mapsto^* \mathbf{k} * \mathbf{u} \mapsto \mathbf{v}$, kde $\mathbf{v} \equiv (n + 1) \cdot n! = (n + 1)!$. □

□

Vidíte, jak „hutný“ a formálně zcela přesný zápis důkazu naše formalizace umožňuje?

Důkazy „neukončenosti“ výpočtů

Fakt: Nechť Δ je deklarace. Pro každé $i \in \mathbb{N}$ definujeme relaci $\mapsto^i \subseteq Exp(\Delta) \times Exp(\Delta)$ předpisem $\mapsto^i = \underbrace{\mapsto \circ \cdots \circ \mapsto}_{i}$. Dále definitoricky klademe $\mapsto^0 = \{(E, E) \mid E \in Exp(\Delta)\}$. Pak platí $\mapsto^* = \bigcup_{i=0}^{\infty} \mapsto^i$. □

Podle předchozího faktu platí, že $E \mapsto^* F$ právě když $E \mapsto^i F$ pro nějaké $i \in \mathbb{N}$. Navíc musí existovat nejmenší i s touto vlastností. Toto pozorování bývá velmi užitečné v důkazech „neukončenosti“ výpočtů. □

Příklad 9.6. *Uvažme deklaraci* $f(x) = f(x)$.

Pak pro každé $\mathbf{n} \in Num$ platí, že neexistuje žádné $\mathbf{m} \in Num$ takové, že $f(\mathbf{n}) \mapsto^* \mathbf{m}$. □

Důkaz sporem: Předpokládejme, že existují $\mathbf{n}, \mathbf{m} \in Num$ takové, že $f(\mathbf{n}) \mapsto^* \mathbf{m}$. Pak existuje nejmenší $i \in \mathbb{N}$ takové, že $f(\mathbf{n}) \mapsto^i \mathbf{m}$. Jelikož výrazy $f(\mathbf{n})$ a $\mathbf{m}$ jsou různé, platí $i > 0$. Jelikož $\mapsto^i = \mapsto^{i-1} \circ \mapsto$ a $f(\mathbf{n}) \mapsto f(\mathbf{n})$, platí $f(\mathbf{n}) \mapsto^{i-1} \mathbf{m}$, což je spor s minimalitou i. □