9 Jednoduchý deklarativní jazyk

Pokračujeme 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; nyní si ukážeme (ještě) přesnější přístup založený na myšlenkách funkcionálního programování.

$$
\begin{array}{lcllcll}
f(3) & \mapsto & \textbf{if } 3 \textbf{ then } 3 * f(3-1) \textbf{ else } 1 & \mapsto & 3 * f(3-1) & \mapsto \\
3 * f(2) & \mapsto & 3 * (\textbf{if } 2 \textbf{ then } 2 * f(2-1) \textbf{ else } 1) & \mapsto & 3 * (2 * f(2-1)) & \mapsto \\
3 * (2 * f(1)) & \mapsto & 3 * (2 * (\textbf{if } 1 \textbf{ then } 1 * f(1-1) \textbf{ else } 1)) & \mapsto & 3*(2*(1* f(1-1))) & \mapsto \\
3*(2*(1*f(0))) & \mapsto & 3*(2*(1*(\textbf{if } 0 \textbf{ then } 0 * f(0-1) \textbf{ else } 1))) & \mapsto & 3 * (2 * (1 * 1)) & \mapsto \\
3 * (2 * 1) & \mapsto & 3 * 2 & \mapsto & 6 &
\end{array}
$$

9 Jednoduchý deklarativní jazyk

Pokračujeme 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; nyní si ukážeme (ještě) přesnější přístup založený na myšlenkách funkcionálního programování.

$$
\begin{array}{lcllcllcl}
f(3) & \mapsto & \textbf{if } 3 \textbf{ then } 3 * f(3-1) \textbf{ else } 1 & \mapsto & 3 * f(3-1) & \mapsto \\
3 * f(2) & \mapsto & 3 * (\textbf{if } 2 \textbf{ then } 2 * f(2-1) \textbf{ else } 1) & \mapsto & 3 * (2 * f(2-1)) & \mapsto \\
3 * (2 * f(1)) & \mapsto & 3 * (2 * (\textbf{if } 1 \textbf{ then } 1 * f(1-1) \textbf{ else } 1)) & \mapsto & 3*(2*(1* f(1-1))) & \mapsto \\
3*(2*(1*f(0))) & \mapsto & 3*(2*(1*(\textbf{if } 0 \textbf{ then } 0 * f(0-1) \textbf{ else } 1))) & \mapsto & 3 * (2 * (1 * 1)) & \mapsto \\
3 * (2 * 1) & \mapsto & 3 * 2 & \mapsto & 6 &
\end{array}
$$

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 k otázce, jak se máme přesvědčit, že program funguje „správně“?

O „správnosti“ programů, podruhé

Vraťme se k otázce, jak se máme přesvědčit, že program funguje „správně“?

- V některých případech, jak už jsme zmínili, je třeba mít naprostou jistotu, že program funguje tak jak má, například v řídících systémech, na nichž závisí lidské životy.

O „správnosti“ programů, podruhé

Vraťme se k otázce, jak se máme přesvědčit, že program funguje „správně“?

- V některých případech, jak už jsme zmínili, je třeba mít naprostou jistotu, že program funguje tak jak má, například v řídících systémech, na nichž závisí lidské životy.
  V takovém případě je jedinou „dostatečně spolehlivou“ možností podat formální matematický důkaz chování algoritmu.

O „správnosti“ programů, podruhé

Vraťme se k otázce, jak se máme přesvědčit, že program funguje „správně“?

- V některých případech, jak už jsme zmínili, je třeba mít naprostou jistotu, že program funguje tak jak má, například v řídících systémech, na nichž závisí lidské životy.
  V takovém případě je jedinou „dostatečně spolehlivou“ možností podat formální matematický důkaz chování algoritmu.
- A co tedy důkazy vlastností symbolicky zapsaných (procedurálních) algoritmů z Lekce 8? Všimli jste si, co v nich bylo problematickým bodem?

O „správnosti“ programů, podruhé

Vraťme se k otázce, jak se máme přesvědčit, že program funguje „správně“?

- V některých případech, jak už jsme zmínili, je třeba mít naprostou jistotu, že program funguje tak jak má, například v řídících systémech, na nichž závisí lidské životy.
  V takovém případě je jedinou „dostatečně spolehlivou“ možností podat formální matematický důkaz chování algoritmu.
- 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.

O „správnosti“ programů, podruhé

Vraťme se k otázce, jak se máme přesvědčit, že program funguje „správně“?

- V některých případech, jak už jsme zmínili, je třeba mít naprostou jistotu, že program funguje tak jak má, například v řídících systémech, na nichž závisí lidské životy.
  V takovém případě je jedinou „dostatečně spolehlivou“ možností podat formální matematický důkaz chování algoritmu.
- 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 IB000).

- Nechť $Var = \{x, y, z, \ldots\}$ je spočetná množina *proměnných*.

9.1 Popis jednoduchého deklarativního jazyka

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

- Nechť $Var = \{x, y, z, \ldots\}$ je spočetná množina *proměnných*.
- Nechť $Num = \{\mathbf{0}, \mathbf{1}, \ldots \mathbf{52}, \ldots \mathbf{397}, \ldots\}$ je množina všech dekadických zápisů přirozených čísel.

9.1 Popis jednoduchého deklarativního jazyka

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

- Nechť $Var = \{x, y, z, \ldots\}$ je spočetná množina *proměnných*.
- Nechť $Num = \{\mathbf{0}, \mathbf{1}, \ldots \mathbf{52}, \ldots \mathbf{397}, \ldots\}$ je množina všech dekadických zápisů přirozených čísel.
- Nechť $Fun = \{f, g, h, \ldots\}$ 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.

9.1 Popis jednoduchého deklarativního jazyka

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

- Nechť $Var = \{x, y, z, \ldots\}$ je spočetná množina *proměnných*.
- Nechť $Num = \{\mathbf{0}, \mathbf{1}, \ldots \mathbf{52}, \ldots \mathbf{397}, \ldots\}$ je množina všech dekadických zápisů přirozených čísel.
- Nechť $Fun = \{f, g, h, \ldots\}$ 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:

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

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

Poznámka: Takováto specifikace syntaxe je abstraktní v tom smyslu, že se nezabývá tím, jak výrazy jednoznačně zapsat do řádku jako posloupnost symbolů. Je na nás, abychom napsali dostatečně mnoho závorek a případně stanovili prioritu operátorů tak, aby bylo zcela jasné, jak daný výraz podle uvedené rovnice vznikl.

Poznámka: Takováto specifikace syntaxe je abstraktní v tom smyslu, že se nezabývá tím, jak výrazy jednoznačně zapsat do řádku jako posloupnost symbolů. Je na nás, abychom napsali dostatečně mnoho závorek a případně stanovili prioritu operátorů tak, aby bylo zcela jasné, jak daný výraz podle uvedené rovnice vznikl.

(Ve smyslu Lekce 6 tato induktivní definice není jednoznačná. To nám však nebude v Definici 9.2 vadit.)

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

- $\mathbf{254}$
- $\mathbf{2 + 3 * 4}$

Poznámka: Takováto specifikace syntaxe je abstraktní v tom smyslu, že se nezabývá tím, jak výrazy jednoznačně zapsat do řádku jako posloupnost symbolů. Je na nás, abychom napsali dostatečně mnoho závorek a případně stanovili prioritu operátorů tak, aby bylo zcela jasné, jak daný výraz podle uvedené rovnice vznikl.

(Ve smyslu Lekce 6 tato induktivní definice není jednoznačná. To nám však nebude v Definici 9.2 vadit.)

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

- $\mathbf{254}$
- $\mathbf{2} + \mathbf{3} * \mathbf{4}$
- $f(\mathbf{2}) \div g(\mathbf{5})$
- $f(\mathbf{2} + x, g(y, \mathbf{3} * y))$

Poznámka: Takováto specifikace syntaxe je abstraktní v tom smyslu, že se nezabývá tím, jak výrazy jednoznačně zapsat do řádku jako posloupnost symbolů. Je na nás, abychom napsali dostatečně mnoho závorek a případně stanovili prioritu operátorů tak, aby bylo zcela jasné, jak daný výraz podle uvedené rovnice vznikl.

(Ve smyslu Lekce 6 tato induktivní definice není jednoznačná. To nám však nebude v Definici 9.2 vadit.)

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

- $\mathbf{254}$
- $\mathbf{2} + \mathbf{3} * \mathbf{4}$
- $f(\mathbf{2}) \div g(\mathbf{5})$
- $f(\mathbf{2} + x, g(y, \mathbf{3} * y))$
- **if** $x - \mathbf{1}$ **then** $(\mathbf{2} + f(y))$ **else** $g(x, x)$

(Vyhodnocení podmínky v „if“ testuje nenulovost argumentu.)

Poznámka: Takováto specifikace syntaxe je abstraktní v tom smyslu, že se nezabývá tím, jak výrazy jednoznačně zapsat do řádku jako posloupnost symbolů. Je na nás, abychom napsali dostatečně mnoho závorek a případně stanovili prioritu operátorů tak, aby bylo zcela jasné, jak daný výraz podle uvedené rovnice vznikl.

(Ve smyslu Lekce 6 tato induktivní definice není jednoznačná. To nám však nebude v Definici 9.2 vadit.)

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

- $\mathbf{254}$
- $\mathbf{2} + \mathbf{3} * \mathbf{4}$
- $f(\mathbf{2}) \div g(\mathbf{5})$
- $f(\mathbf{2} + x, g(y, \mathbf{3} * y))$
- **if** $x - \mathbf{1}$ **then** $(\mathbf{2} + f(y))$ **else** $g(x, x)$

(Vyhodnocení podmínky v „if“ testuje nenulovost argumentu.)

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

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_1 \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_n \end{array},$$

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

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

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_1 \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_n \end{array},$$

kde pro každé $1 \leq i \leq n$ platí, že $f_i \in Fun$ je funkce arity a_i, $x_1, \ldots, x_{a_i} \in Var$ 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.

- $f(x) \;=\;$ **if** x **then** $x * f(x-1)$ **else** 1

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

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_1 \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_n \end{array},$$

kde pro každé $1 \leq i \leq n$ platí, že $f_i \in \mathit{Fun}$ je funkce arity a_i, $x_1, \ldots, x_{a_i} \in \mathit{Var}$ 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.

- $f(x) \;=\; \textbf{if } x \textbf{ then } x * f(x-1) \textbf{ else } 1$
- $\begin{array}{rcl} f(x) & = & g(x-\mathbf{1}, x) \\ g(x,y) & = & \textbf{if } x \textbf{ then } f(y) \textbf{ else } \mathbf{3} \end{array}$
  (Jak uvidíme formálně později, konvencí našich výpočtů je neužívat záporná čísla, místo toho $0-1$ *„=“* 0*.)*

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

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_1 \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_n \end{array},$$

kde pro každé $1 \leq i \leq n$ platí, že $f_i \in \mathit{Fun}$ je funkce arity a_i, $x_1, \ldots, x_{a_i} \in \mathit{Var}$ 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.

- $f(x) \;=\; \textbf{if } x \textbf{ then } x * f(x-1) \textbf{ else } 1$
- $\begin{array}{rcl} f(x) & = & g(x-\mathbf{1}, x) \\ g(x,y) & = & \textbf{if } x \textbf{ then } f(y) \textbf{ else } \mathbf{3} \end{array}$
  (Jak uvidíme formálně později, konvencí našich výpočtů je neužívat záporná čísla, místo toho $0-1$ *„=“* 0*.)*
- $g(x,y) \;=\; \textbf{if } x - y \textbf{ then } x \textbf{ else } y$

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

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_1 \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_n \end{array},$$

kde pro každé $1 \leq i \leq n$ platí, že $f_i \in \mathit{Fun}$ je funkce arity a_i, $x_1, \ldots, x_{a_i} \in \mathit{Var}$ 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.

- $f(x) \;=\; \textbf{if } x \textbf{ then } x * f(x-1) \textbf{ else } 1$
- $\begin{array}{rcl} f(x) & = & g(x-\mathbf{1}, x) \\ g(x,y) & = & \textbf{if } x \textbf{ then } f(y) \textbf{ else } \mathbf{3} \end{array}$
  (Jak uvidíme formálně později, konvencí našich výpočtů je neužívat záporná čísla, místo toho $0-1$ „=“ 0.)
- $g(x,y) \;=\; \textbf{if } x - y \textbf{ then } x \textbf{ else } y$
- $f(x) \;=\; f(x)$

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

$$\begin{array}{rcl} f_1(x_1, \cdots, x_{a_1}) & = & E_1 \\ \vdots & & \vdots \\ f_n(x_1, \cdots, x_{a_n}) & = & E_n \end{array},$$

kde pro každé $1 \leq i \leq n$ platí, že $f_i \in \mathit{Fun}$ je funkce arity a_i, $x_1, \ldots, x_{a_i} \in \mathit{Var}$ 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.

- $f(x) \;=\; \textbf{if } x \textbf{ then } x * f(x-1) \textbf{ else } 1$
- $f(x) \;=\; g(x-1, x)$
  $g(x,y) \;=\; \textbf{if } x \textbf{ then } f(y) \textbf{ else } 3$
  (Jak uvidíme formálně později, konvencí našich výpočtů je neužívat záporná čísla, místo toho $0-1$ „=“ 0.)
- $g(x,y) \;=\; \textbf{if } x - y \textbf{ then } x \textbf{ else } y$
- $f(x) \;=\; f(x)$
  (Nezapisuje toto náhodou „nekonečnou smyčku“?)

9.2 Formalizace pojmu „výpočet“

Za výpočet (nad Δ) budeme považovat posloupnost úprav výrazů, které jsou „postaveny“ na naší uvažované deklaraci Δ. To je formálně podchyceno následujícíma dvěma definicema.

Definice: Buď Δ 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.

9.2 Formalizace pojmu „výpočet“

Za výpočet (nad Δ) budeme považovat posloupnost úprav výrazů, které jsou „postaveny“ na naší uvažované deklaraci Δ. To je formálně podchyceno následujícíma dvěma definicema.

Definice: Buď Δ 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.

Fakt: Množinu $Exp(\Delta)$ lze definovat také induktivně:

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

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.

9.2 Formalizace pojmu „výpočet“

Za výpočet (nad Δ) budeme považovat posloupnost úprav výrazů, které jsou „postaveny“ na naší uvažované deklaraci Δ. To je formálně podchyceno následujícíma dvěma definicema.

Definice: Buď Δ 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.

Fakt: Množinu $Exp(\Delta)$ lze definovat také induktivně:

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

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.

Definice 9.2. Výpočet a krok výpočtu v našem deklarativním jazyce. Funkci „krok výpočtu“ $\mapsto : Exp(\Delta) \to Exp(\Delta)$ definujeme induktivně takto; místo $\mapsto(E) = F$ budeme psát $\boldsymbol{E \mapsto F}$.

- $\mathbf{n} \mapsto \mathbf{n}$ pro každé $\mathbf{n} \in Num$.

Definice 9.2. Výpočet a krok výpočtu v našem deklarativním jazyce. Funkci „krok výpočtu“ $\mapsto : Exp(\Delta) \to Exp(\Delta)$ definujeme induktivně takto; místo $\mapsto(E) = F$ budeme psát $\boldsymbol{E \mapsto F}$.

- $\mathbf{n} \mapsto \mathbf{n}$ pro každé $\mathbf{n} \in Num$.
- Pro $\boldsymbol{E \equiv (E_1)}$ definujeme krok výpočtu takto:
  - Jestliže $E_1 \mapsto F \in Num$, pak $(E_1) \mapsto F$.
  - Jestliže $E_1 \mapsto F \notin Num$, pak $(E_1) \mapsto (F)$.

Definice 9.2. Výpočet a krok výpočtu v našem deklarativním jazyce. Funkci „krok výpočtu“ $\mapsto : Exp(\Delta) \to Exp(\Delta)$ definujeme induktivně takto; místo $\mapsto(E) = F$ budeme psát $\boldsymbol{E \mapsto F}$.

- $\mathbf{n} \mapsto \mathbf{n}$ pro každé $\mathbf{n} \in Num$.
- Pro $\boldsymbol{E \equiv (E_1)}$ definujeme krok výpočtu takto:
  - Jestliže $E_1 \mapsto F \in Num$, pak $(E_1) \mapsto F$.
  - Jestliže $E_1 \mapsto F \notin Num$, pak $(E_1) \mapsto (F)$.
- Pro $\boldsymbol{E \equiv E_1 + E_2}$ 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 $E_1 \notin Num$, pak $E_1 + E_2 \mapsto F + E_2$, kde $E_1 \mapsto F$.
  - Jestliže $E_1 \in Num$ a $E_2 \notin Num$, pak $E_1 + E_2 \mapsto E_1 + F$, kde $E_2 \mapsto F$.

- 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 $E_1 \notin Num$, pak $E_1 - E_2 \mapsto F - E_2$, kde $E_1 \mapsto F$.
  - Jestliže $E_1 \in Num$ a $E_2 \notin Num$, pak $E_1 - E_2 \mapsto E_1 - F$, kde $E_2 \mapsto F$.

- Pro $\underline{E \equiv E_1 - E_2}$ 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 $E_1 \notin Num$, pak $E_1 - E_2 \mapsto F - E_2$, kde $E_1 \mapsto F$.
  - Jestliže $E_1 \in Num$ a $E_2 \notin Num$, pak $E_1 - E_2 \mapsto E_1 - F$, kde $E_2 \mapsto F$.
- Pro $\underline{E \equiv E_1 * E_2}$ 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 $E_1 \notin Num$, pak $E_1 * E_2 \mapsto F * E_2$, kde $E_1 \mapsto F$.
  - Jestliže $E_1 \in Num$ a $E_2 \notin Num$, pak $E_1 * E_2 \mapsto E_1 * F$, kde $E_2 \mapsto F$.

- 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 $E_1 \notin Num$, pak $E_1 - E_2 \mapsto F - E_2$, kde $E_1 \mapsto F$.
  - Jestliže $E_1 \in Num$ a $E_2 \notin Num$, pak $E_1 - E_2 \mapsto E_1 - F$, kde $E_2 \mapsto F$.
- 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 $E_1 \notin Num$, pak $E_1 * E_2 \mapsto F * E_2$, kde $E_1 \mapsto F$.
  - Jestliže $E_1 \in Num$ a $E_2 \notin Num$, pak $E_1 * E_2 \mapsto E_1 * F$, kde $E_2 \mapsto F$.
- 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 $E_1 \notin Num$, pak $E_1 \div E_2 \mapsto F \div E_2$, kde $E_1 \mapsto F$.
  - Jestliže $E_1 \in Num$ a $E_2 \notin Num$, pak $E_1 \div E_2 \mapsto E_1 \div F$, kde $E_2 \mapsto F$.

- Pro $E \equiv$ **if** E_1 **then** E_2 **else** E_3 definujeme krok výpočtu takto:
  - Jestliže $E_1 \in Num$ a $E_1 \equiv \mathbf{0}$, pak **if** E_1 **then** E_2 **else** $E_3 \mapsto E_3$.
  - Jestliže $E_1 \in Num$ a $E_1 \not\equiv \mathbf{0}$, pak **if** E_1 **then** E_2 **else** $E_3 \mapsto E_2$.
  - Jestliže $E_1 \notin Num$, pak
    if E_1 **then** E_2 **else** $E_3 \mapsto$ **if** F **then** E_2 **else** E_3, kde $E_1 \mapsto F$.

- Pro $E \equiv$ **if** E_1 **then** E_2 **else** E_3 definujeme krok výpočtu takto:
  - Jestliže $E_1 \in Num$ a $E_1 \equiv \mathbf{0}$, pak **if** E_1 **then** E_2 **else** $E_3 \mapsto E_3$.
  - Jestliže $E_1 \in Num$ a $E_1 \not\equiv \mathbf{0}$, pak **if** E_1 **then** E_2 **else** $E_3 \mapsto E_2$.
  - Jestliže $E_1 \notin Num$, pak
    if E_1 **then** E_2 **else** $E_3 \mapsto$ **if** F **then** E_2 **else** E_3, kde $E_1 \mapsto F$.
- Pro $E \equiv f(E_1, \cdots, E_k)$ 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 $E_i \mapsto F$.

- Pro $\boldsymbol{E \equiv \textbf{if } E_1 \textbf{ then } E_2 \textbf{ else } E_3}$ definujeme krok výpočtu takto:
  - Jestliže $E_1 \in Num$ a $E_1 \equiv \mathbf{0}$, pak **if** E_1 **then** E_2 **else** $E_3 \mapsto E_3$.
  - Jestliže $E_1 \in Num$ a $E_1 \not\equiv \mathbf{0}$, pak **if** E_1 **then** E_2 **else** $E_3 \mapsto E_2$.
  - Jestliže $E_1 \notin Num$, pak
    if E_1 **then** E_2 **else** $E_3 \mapsto$ **if** F **then** E_2 **else** E_3, kde $E_1 \mapsto F$.
- 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 $E_i \mapsto F$.

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

Tato rozsáhlá a důležitá definice si zaslouží několik podstatných připomínek.

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný, neboli všechna záporná čísla jsou nahrazena nulou.

Tato rozsáhlá a důležitá definice si zaslouží několik podstatných připomínek.

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný, neboli všechna záporná čísla jsou nahrazena nulou.
  - Výsledek dělení je vždy celočíselný, počítáme jeho dolní celou část.

Tato rozsáhlá a důležitá definice si zaslouží několik podstatných připomínek.

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný, neboli všechna záporná čísla jsou nahrazena nulou.
  - Výsledek dělení je vždy celočíselný, počítáme jeho dolní celou část.
  - Dělení nulou je definováno (není chybou), výsledkem je opět nula.

Tato rozsáhlá a důležitá definice si zaslouží několik podstatných připomínek.

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný, neboli všechna záporná čísla 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.

Tato rozsáhlá a důležitá definice si zaslouží několik podstatných připomínek.

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný, neboli všechna záporná čísla 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.
  Mimo jiné je takto „definována“ nejvyšší priorita vyhodnocení uzávorkovaného výrazu.

Tato rozsáhlá a důležitá definice si zaslouží několik podstatných připomínek.

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný, neboli všechna záporná čísla 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.
  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.

Tato rozsáhlá a důležitá definice si zaslouží několik podstatných připomínek.

- Za prvé si dobře povšimněte některých „aritmetických“ aspektů výpočtu.
  - Výsledek odečítání je vždy nezáporný, neboli všechna záporná čísla 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.
  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 se však z $\mapsto$ stává pouhá relace. My se touto možností nebudeme zabývat.

9.3 Příklady výpočtů

Příklad 9.3. *Ukážeme si několik ilustrativních „výpočtů“ nad různými deklaracemi.*

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

$$
\begin{array}{llllll}
f(3) & \mapsto & \textbf{if 3 then } 3*f(3-1) \textbf{ else } 1 & \mapsto & 3*f(3-1) & \mapsto \\
3*f(2) & \mapsto & 3*(\textbf{if 2 then } 2*f(2-1) \textbf{ else } 1) & \mapsto & 3*(2*f(2-1)) & \mapsto \\
3*(2*f(1)) & \mapsto & 3*(2*(\textbf{if 1 then } 1*f(1-1) \textbf{ else } 1)) & \mapsto & 3*(2*(1*f(1-1))) & \mapsto \\
3*(2*(1*f(0))) & \mapsto & 3*(2*(1*(\textbf{if 0 then } 0*f(0-1) \textbf{ else } 1))) & \mapsto & 3*(2*(1*1)) & \mapsto \\
3*(2*1) & \mapsto & 3*2 & \mapsto & 6\,. &
\end{array}
$$

9.3 Příklady výpočtů

Příklad 9.3. *Ukážeme si několik ilustrativních „výpočtů“ nad různými deklaracemi.*

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

$$
\begin{array}{lcllcl}
f(3) & \mapsto & \textbf{if 3 then } 3*f(3-1) \textbf{ else } 1 & \mapsto & 3*f(3-1) & \mapsto \\
3*f(2) & \mapsto & 3*(\textbf{if 2 then } 2*f(2-1) \textbf{ else } 1) & \mapsto & 3*(2*f(2-1)) & \mapsto \\
3*(2*f(1)) & \mapsto & 3*(2*(\textbf{if 1 then } 1*f(1-1) \textbf{ else } 1)) & \mapsto & 3*(2*(1*f(1-1))) & \mapsto \\
3*(2*(1*f(0))) & \mapsto & 3*(2*(1*(\textbf{if 0 then } 0*f(0-1) \textbf{ else } 1))) & \mapsto & 3*(2*(1*1)) & \mapsto \\
3*(2*1) & \mapsto & 3*2 & \mapsto & 6\,. &
\end{array}
$$

Uvažme deklaraci $f(x) = g(x-1, x)$ a $g(x, y) = \textbf{if } x \textbf{ then } f(y) \textbf{ else } 3$. Pak například $f(3) \mapsto^* f(3)$, neboť

$$f(3) \mapsto g(3-1, 3) \mapsto g(2, 3) \mapsto \textbf{if 2 then } f(3) \textbf{ else } 3 \mapsto f(3)\,.$$

9.3 Příklady výpočtů

Příklad 9.3. *Ukážeme si několik ilustrativních „výpočtů“ nad různými deklaracemi.*

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

$$
\begin{array}{llllll}
f(3) & \mapsto & \textbf{if 3 then } 3*f(3-1) \textbf{ else } 1 & \mapsto & 3*f(3-1) & \mapsto \\
3*f(2) & \mapsto & 3*(\textbf{if 2 then } 2*f(2-1) \textbf{ else } 1) & \mapsto & 3*(2*f(2-1)) & \mapsto \\
3*(2*f(1)) & \mapsto & 3*(2*(\textbf{if 1 then } 1*f(1-1) \textbf{ else } 1)) & \mapsto & 3*(2*(1*f(1-1))) & \mapsto \\
3*(2*(1*f(0))) & \mapsto & 3*(2*(1*(\textbf{if 0 then } 0*f(0-1) \textbf{ else } 1))) & \mapsto & 3*(2*(1*1)) & \mapsto \\
3*(2*1) & \mapsto & 3*2 & \mapsto & 6\,. &
\end{array}
$$

Uvažme deklaraci $f(x) = g(x-1, x)$ a $g(x, y) = \textbf{if } x \textbf{ then } f(y) \textbf{ else } 3$. Pak například $f(3) \mapsto^* f(3)$, neboť

$$f(3) \mapsto g(3-1, 3) \mapsto g(2, 3) \mapsto \textbf{if 2 then } f(3) \textbf{ else } 3 \mapsto f(3)\,.$$

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

Ale $f(f(2+3)) \mapsto f(f(5)) \mapsto f(f(5))$. □

Důkaz správnosti programu

Příklad 9.4. *Pro ukázku uvažme deklaraci Δ obsahující pouze rovnici*

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

Věta. Pro každé $n \in \mathbb{N}_0$ platí $f(\mathbf{n}) \mapsto^* \mathbf{m}$, kde $\mathbf{m} \equiv n!$.

Důkaz správnosti programu

Příklad 9.4. *Pro ukázku uvažme deklaraci Δ obsahující pouze rovnici*

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

Věta. Pro každé $n \in \mathbb{N}_0$ platí $f(\mathbf{n}) \mapsto^* \mathbf{m}$, kde $\mathbf{m} \equiv n!$.

Důkaz povedeme indukcí podle n:

- Báze $n = 0$. Platí $f(\mathbf{0}) \;\mapsto\; \textbf{if } \mathbf{0} \textbf{ then } \mathbf{0} * f(\mathbf{0}-\mathbf{1}) \textbf{ else } \mathbf{1} \;\mapsto\; \mathbf{1}$ a platí $0! = 1$.

Důkaz správnosti programu

Příklad 9.4. *Pro ukázku uvažme deklaraci Δ obsahující pouze rovnici*

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

Věta. Pro každé $n \in \mathbb{N}_0$ platí $f(\mathbf{n}) \mapsto^* \mathbf{m}$, kde $\mathbf{m} \equiv n!$.

Důkaz povedeme indukcí podle n:

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

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

  kde $\mathbf{w} \equiv k - 1 = n$.

Důkaz správnosti programu

Příklad 9.4. *Pro ukázku uvažme deklaraci Δ obsahující pouze rovnici*

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

Věta. Pro každé $n \in \mathbb{N}_0$ platí $f(\mathbf{n}) \mapsto^* \mathbf{m}$, kde $\mathbf{m} \equiv n!$.

Důkaz povedeme indukcí podle n:

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

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

  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 $\mathbf{v} \equiv (n+1) \cdot n! = (n+1)!$.

Důkaz správnosti programu

Příklad 9.4. *Pro ukázku uvažme deklaraci Δ obsahující pouze rovnici*

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

Věta. Pro každé $n \in \mathbb{N}_0$ platí $f(\mathbf{n}) \mapsto^* \mathbf{m}$, kde $\mathbf{m} \equiv n!$.

Důkaz povedeme indukcí podle n:

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

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

  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 přitom formálně zcela přesný zápis důkazu naše formalizace umožňuje?

Důkazy „neukončenosti“ výpočtů

Věta 9.5. *Buď Δ 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 $\mapsto^* = \bigcup_{i=0}^{\infty} \mapsto^i$.*

Důkazy „neukončenosti“ výpočtů

Věta 9.5. *Buď Δ 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 předchozí věty platí, že $E \mapsto^* F$ právě když $E \mapsto^i F$ pro nějaké $i \in \mathbb{N}_0$. Navíc musí existovat nejmenší i s touto vlastností. Toto pozorování bývá velmi užitečné v důkazech „neukončenosti“ výpočtů.

Důkazy „neukončenosti“ výpočtů

Věta 9.5. *Buď Δ 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 předchozí věty platí, že $E \mapsto^* F$ právě když $E \mapsto^i F$ pro nějaké $i \in \mathbb{N}_0$. Navíc musí existovat nejmenší i s touto vlastností. Toto pozorování bývá velmi užitečné v důkazech „neukončenosti“ výpočtů.

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

> **Věta**. Pro každé $\mathbf{n} \in Num$ platí, že neexistuje žádné $\mathbf{m} \in Num$ takové, že $f(\mathbf{n}) \mapsto^* \mathbf{m}$.

Důkazy „neukončenosti“ výpočtů

Věta 9.5. *Buď Δ 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 předchozí věty platí, že $E \mapsto^* F$ právě když $E \mapsto^i F$ pro nějaké $i \in \mathbb{N}_0$. Navíc musí existovat nejmenší i s touto vlastností. Toto pozorování bývá velmi užitečné v důkazech „neukončenosti“ výpočtů.

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

> **Věta**. 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 $i \in \mathbb{N}_0$ 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. □