Karlova Univerzita v Praze
Fakulta Filosofie

Katedra Logiky

Obor Logika

Bakalářská práce

Bezestrojová charakterizace polynomiálně počitatelných funkcí

Machine-Free Characterization of Polynomially Computable Functions

Autor:
Michal Profeld

Vedoucí práce:
Doc. RNDr. Vítězslav Švejdar, CSc.

15. srpna 2016

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně, že jsem řádně citoval všechny použité prameny a literaturu a že práce nebyla využita v rámci jiného vysokoškolského studia či k získání jiného nebo stejného titulu.

V Praze dne

.....

Podpis

.....

Abstrakt

Tato bakalářská práce se zabývá sestavením Matematického systému. Tento systém je pečlivě vypracovaný, tak aby byl uzavřený na funkce, které v něm figurují. Je vytvořen tak, aby pokryl funkce určitého růstu. Konkrétně funkce, o kterých můžeme říct, že operují v polynomiálním čase na Turingové stroji. Platí tedy, že náš systém obsahuje všechny funkce, které na Turingových strojích běží v polynomálním čase, nebo v čase rychlejším a žádné jiné funkce neobsahuje. Tvorba tohoto matematického systému byla ovlivněna především prací Samuela R. Busse [1]

Abstract

This work is focused into constructing mathematical structure. This structure is closed under it's operations. Structure was developed to contain all functions of certain growth rate. To be More specific functions with polynomial growth rate. We can say that our structure contains all functions that have growth rate slower or equal to polynomial growth rate and no other function. Development of our structure was influenced mostly by work of Samuel R. Buss [1]

Obsah

1 Úvodem

Polynomiální funkce jsou poměrně dobře známy, jak mezi matematiky, logiky, tak i mezi programátory a IT experty. Zatímco pro matematiky je na polynomiálních funkcích nejzajmavější slavný problém "P = NP", tak pro lidi z IT sféry je polynomiálnost jakousi mezí efektivnosti pro algoritmus. Předem bych chtěl upozornit, že v této práci se pokusím problému "P = NP"plně vyhnout. Naopak hledisko polynomiálnosti, jako meze použitelnosti funkce bude dosti aktuální.

Zjednodušeně by se dalo říci, že většina programů, které se píší, operují v polynomiálním čase. Což znamená, že dané programy běží velice rychle i pro poměrně veliké vstupy. Počet kroků v takovémto programu je maximálně n^c pro předem danou konstantu c a délku vstupu n. Přesná definice pomocí Turingova stroje se nachází v další kapitole. Nicméně definice polynomiálního času není závislá na Turingově stroji a dá se vztáhnout i k jiným výpočetním nástrojům, například programovacím jazykům. Co je ovšem mnohem zajímavější, polynomiální čas lze definovat i zcela bezestrojově. Tento poměrně překavapivý výsledek je základem této bakalářské práce.

Tato práce bude postupovat souběžně s knihou [2], nicméně formálně se bude držet spíše práce [1] a bude lehce inspirována prací [3]. Zezačátku se tedy definuje množina jednoduchých funkcí, spolu se skládáním funkcí. Z této množiny se odvodí pojem základně polynomiální podmínky a ukáže se jejich uzavřenost na základní logické operace. Dále prohlásíme, že tyto podmínky nám definují množinu základně polynomiálních funkcí. Tato množina je však stále dosti malá. Proto přidáme odvození pomocí rekurze. Tyto rekurze definujeme dvě a dokážeme jejich zaměnitelnost. Z nově vzniklého systému odvodíme polynomiální podmínky, pomocí délkově omezených kvantifikátorů. Dále si odvodíme nějpoužívanější jednoduché polynomiální funkce a časem se dostaneme k funkcím složitějším. S tímto již poměrně silným systémem si ukážeme možnosti kódování. Konkrétně si definujeme kódování posloupností a práci s nimi. Pro kódování posloupností budeme používat Gödelova čísla. Definujeme tedy funkce jako přidání čísla do posloupnosti, cut, nebo truncate. Tyto posloupnosti jsou použity v důkazu, že naše množina polynomiálních funkcí opravdu obsahuje všechny polynomiální funkce a že naopak neobsahuje nic navíc. O tomto důkazu se pouze zmíníme.

2 Počáteční strojová definice

Definice 2.1. Polynomiální čas
Polynomiální čas je třída všech problémů, jež jsou řešitelné pomocí deterministického Turingova stroje, tak že jsou polynomiálně omezeny pomocí délky daného vstupu.

Takováto definice nám však mnoho neřekne a pro lepši pochopení je třeba odvodit takzvanou Big $\mathcal{O}$ notaci, jež nám pomůže k pochopení termínu "polynomiálně omezit v závislosti na vstupu". Použiji obecně známou definici pro big $\mathcal{O}$ notaci.

Definice 2.2. Big $\mathcal{O}$ notace
Jsou dány funkce $f(n)$ a $g(n)$, pak řekneme, že $f(n)$ je nejvýše řádu $g(n)$, psáno $f(n) = \mathcal{O}(g(n))$, jestliže:

$$(\exists c \in \mathrm{R})\ (\exists n_0 \in \mathrm{N})\ (\forall n \geq n_0) : f(n) \leq c * g(n)$$

Dalším termínem, který je třeba odvodit je Turingův stroj, který nám bude sloužit jako nástroj pro zpracovávání zmíněných problémů. Následující definice je obecně známa. Bohužel nemohu přesně citovat zdroje, jelikož znalosti pochází z více přednášek na ČVUT, například[6].

Definice 2.3. Turingův stroj
Je sedmicí $M = \{Q, \Gamma, b, \Sigma, \delta, q_0, F\}$, kde platí:

- Q je neprázdná, konečná množina stavů.
- Γ je neprázdná, konečná množina symbolů na pásce.
- $b \in \Gamma$ je prázdný symbol.
- $\Sigma \subseteq (\Gamma - b)$ je množina výstupních symbolů.
- $\delta : (Q - F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- δ je přechodovou funkcí a L, R jsou posuny vlevo, vpravo.
- q_0 je počáteční stav.
- $F \subseteq Q$ je množina přijmajících stavů.
- Pokud není δ definována pro současný stav a symbol na pásce, pak se stroj zastaví.
- Pokud se stroj zastaví v jednom z přijímaných stavů, pak je počáteční stav pásky přijat.

Rád bych dodal, že nezáleží, jestli používáme jednopáskový, nebo vícepáskový Turingův stroj. Jelikož jsou na sebe převeditelné a převod má složitost x^2, takže i po převodu zůstaneme v polynomiálním čase. Nyní se vraťme k naší

definici. V této práci se budeme snažit dospět k nejmenší třídě funkcí, jež obsahuje všechny problémy řešitelné v polynomiálním čase. Tedy pro každou z funkcí v této třídě musí existovat polynom který ji omezuje. Jinými slovy je funkce polynomiální, pokud pro vstup libovolné délky n potřebuje na Turingově stroji maximálně n^c operací, kde c je předem zvolená libovolná konstanta.

3 Množina základních funkcí

Nejdříve definujeme množinu základních funkcí, které nám budou sloužit jako odrazový můstek pro definování funkcí, které jsou složitější. Součástí množiny základních funkcí bude i skládání funkcí, tak jak ho známe. Množina základních funkcí je částečně převzata z knihy [1].

Definice 3.1. Definice základních funkcí

(1) $$\mathrm{z} : \underline{x} \longmapsto 0 \text{ (Konstantní nulová funkce)}$$

(2) $$\mathrm{i}_j^n : \underline{x} \longmapsto x_j \text{ (Projekce proměnných)}$$

(3) $$\mathrm{s}(x) : \underline{x} \longmapsto x + 1 \text{ (Funkce následníka)}$$

(4) $$\mathrm{asr}(x) : \underline{x} \longmapsto \lfloor x/2 \rfloor \text{ (Bitový posun vpravo)}$$

(5) $$\mathrm{asl}(x) : \underline{x} \longmapsto x \cdot 2 \text{ (Bitový posun vlevo)}$$

(6) $$x \leq y : \underline{x} \longmapsto \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{if } x > y \end{cases} \text{ (Menší rovno)}$$

(7) $$\mathrm{Choice}(x, y, z) : \underline{x} \longmapsto \begin{cases} y & \text{if } x > 0 \\ z & \text{if } x = 0 \end{cases} \text{ (Výběrová funkce)}$$

(8) $$|x_1 + \cdots + x_n| : \underline{x} \longmapsto \lceil \log_2(x_1) \rceil + \cdots + \lceil \log_2(x_n) \rceil \text{ (Binární délka čísla)}$$

Dále si definujme skládání, které vezme m funkcí k proměnných a složí je do jedné funkce f' k proměnných. Uvnitř to vypadá tak, že je každé z m funkcí předáno všech k proměnných a funkce si vybere ty proměnné, jež potřebuje. Výsledky jednotlivých funkcí jsou pak dosazeny do funkce f', která je zpracuje a vrátí jedno číslo, jako výsledek. Skládání funkcí je inspirováno [4].

Definice 3.2. Skládání funkcí
Nechť jsou $f_1, \ldots, f_m$ funkcemi k proměnných. Pak složením funkcí dostaneme funkci f', která je funkcí k proměnných:

$$f' \circ [f_1, \ldots, f_m](x_1, \ldots, x_k) = f'(f_1(x_1, \ldots, x_k), \ldots, f_m(x_1, \ldots, x_k))$$

Je třeba poznamenat, že pro $m = 1$ budeme zapisovat zjednodušeně $f' \circ f_1$ $(x_1, \ldots, x_k)$. Dále se dohodněme, že ne vždy musíme za funkci psát všechny proměnné, například psát $(\mathrm{s} \circ \mathrm{z})(x)$ je naprosto zbytečné, přestože se jedná o funkce jedné proměnné (funkce z je doplněná o jalovou proměnnou). Další věc, kterou bych rád zmínil, je používání funkce pro přidání jalové proměnné. Tato funkce je používána tak, aby všechny funkce měly stejnou aritu. Její používání budeme brát jako samozřejmost a nebudeme se jí při odvozování zabývat. Dále bych se rád zmínil o používání závorek, při skládání více funkcí. Pokud jsou jednotlivá skládání čteny zleva doprava, pak můžeme závorky vynechat. Poslední věcí, kterou bych rád zmínil je, že bych se dále rád vyhnul u skládání funkcí formalitám a občas budu psát $\mathrm{s}(x)$ místo $(\mathrm{s} \circ \mathrm{i}_1^n)(x)$

Definice 3.3. Množina základních funkcí
Definujme nyní množinu základních funkcí, jako nejmenší množinu, která obsahuje základní funkce a je uzavřená na jejich skládání.

3.1 Rozbor množiny základních funkcí

3.1.1 Konstantní nulová funkce z

V této práci budeme pracovat výhradně s oborem přirozených čísel (značme N). Je tedy nutné, aby naše množina základních funkcí měla do N přístup. Právě k tomu nám slouží funkce z, která bude naším vstupem do struktury přirozených čísel. Funkce z dává pochopitelně přístup k číslu 0 z N.

3.1.2 Výběrová funkce

Nulová funkce nám dává přístup do množiny přirozených čísel. Pro práci s funkcemi však potřebujemi i jinou množinu, ve které se budeme pohybovat. Jedná se o množinu vstupních parametrů funkce. K parametrům funkce se dostaneme pomocí naší výběrové funkce. Nyní je na čase odvodit si čtyři triviální funkce pro práci se vstupními parametry. Rozhodl jsem se však odvodit pouze dvě z těchto funkcí, jelikož ostatní jsou odvozeny v knize [4].

Věta 3.4. *Přidání jalové proměnné je v množině základních funkcí*

Důkaz. Důkaz je inspirován zdrojem [4]. Nechť f je funkce n proměnných, kde $n \geq 0$. Definujme funkci g která má $n+1$ proměnných a vrací na všech vstupech ekvivalentní výsledky s funkcí f takto:

$$g(x_1, \ldots, x_n) = f(\mathrm{i}_0^n, \ldots, \mathrm{i}_n^n)$$

□

Později budeme v této práci pro zjednodušení používat pouze jména proměnných, například místo i_2^n z $x_1, \ldots, x_n$ budeme jednoduše psát proměnnou x_2.

3.1.3 Funkce následníka

Nyní jsme schopni pracovat s parametry funkce a máme přístup do $\mathbb{N}$. Je tedy potřeba odvodit si v $\mathbb{N}$ funkci, která nám umožní pohyb a tím přístup k libovolné konstantě z $\mathbb{N}$.

Věta 3.5. *Přístup k libovolné konstantě z $\mathbb{N}$ je v množině základních funkcí*

Důkaz. Nechť je tedy c libovolná konstanta z $\mathbb{N}$. Uvažujme nyní složení funkcí:

$$c = \underbrace{\mathrm{s} \circ \ldots \circ \mathrm{s}}_{c\text{-krát}} \circ \mathrm{z}$$

Čímž dostaneme funkci generující konstantu c. □

Je třeba poznamenat, že musíme odvodit více než jednu funkci, tedy pro každou konstantu odvodit samostatnou funkci. Zřetězení c následnických funkcí můžeme použít také na proměnnou, místo konstantní nuly. Výsledkem bude funkce přičítající konstantu c k proměnné x.

Věta 3.6. *0 Přičtení konstanty $(x + c)$ je v množině základních funkcí*

Důkaz. Nechť c je libovolná konstanta a nechť x je vstupní proměnná, pak funkci $x + c$ definujme takto:

$$f(x) = (\underbrace{\mathrm{s} \circ \ldots \circ \mathrm{s}}_{c\text{-krát}} \circ \mathrm{i}_1^1)(x)$$

□

Opět berme v potaz, že jsme pro každou konstantu odvodili jednu funkci. Nyní odvoďme další z triviálních funkcí a to konkrétně funkci pro dosazení konstanty do funkce. Důkaz je inspirován zdrojem [4].

Věta 3.7. *Dosazení konstanty c funkce místo proměnné x ve funkci* $g(\underline{x})$ *neboli* $\text{sub}(c, x, f(\underline{x}))$ *je v množině základních funkcí*

Důkaz. Nechť g je funkce n proměnných, kde $n>0$. Definujme funkci f která má $n-1$, protože je za proměnnou x_j dosazena konstanta c takto:

$$\begin{aligned}\text{sub}(c, x_j, g(x_0, \ldots, x_j, \ldots, x_n)) &= g(\text{i}_0^n, \ldots, \text{i}_{j-1}^n, c, \text{i}_{j+1}^n, \ldots, \text{i}_n^n) \\ &= f(x_0, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)\end{aligned}$$

□

3.1.4 Bitové posuny

Dalšími funkcemi jsou bitové posuny, které slouží k manipulaci s čísly, konkrétně s jejich délkou. V sekci definice jsme se dohodli, že budeme čísla reprezentovat binárně. Což znamená, že operace dělení a násobení dvěma se chová vhodně vzhledem k číslům. Bitový posun vpravo tedy odstraní z čísla poslední cifru (nejméně signifikantní bit) a bitový posun vlevo přidá jednu nulu, jako poslední cifru čísla. Dalo by se také říci, že posuny zkrátí/prodlouží číslo o jednu binární cifru. Dohodněme se nyní, že budeme bitové posuny čísla x o jeden bit vlevo/vpravo značit takto: $\text{asl}(x)/\text{asr}(x)$

Věta 3.8. *Funkce identity* - $\text{Id}(x)$ *je v množině základních funkcí*

Důkaz. Nechť x je proměnná. Funkce identita je definována takto:

$$\text{asr}(\text{asl}(1))$$

□

Věta 3.9. *Bitový posun vlevo*$(x \times 2^c)$ *a vpravo*$(\frac{x}{2^c})$ *o c bitů je v množině základních funkcí*

Důkaz. Odvození probíhá pomocí složení c funkcí stejně jako v sekci 3.1.3 □

Dále značit posun čísla x o konstantních c bitů takto: "$x \ll c$" / "$x \ll c$"

3.1.5 Větvící funkce (funkce menší rovno a funkce choice)

Předchozí funkce sloužily k manipulaci s číslem, případně zajišťovaly přístup do $\mathbb{N}$. Větvící funkce jsou trochu jiné a umožňují totiž více deterministických možností rozvětvení a opětovného sloučení funkce. Funkce choice(x, y, z) nám funkci rozvětví na dvě možné větve. Naopak funkce $x \leq y$ sloučí dvě větve funkce do jedné. Dále budeme definovat funkci pro porovnání rovnosti dvou čísel. Použijeme funkci Choice$(x \leq y, y \leq x, 0)$, která představuje $x \leq y \wedge y \leq x$, což si ukážeme ke konci kapitoly.

Věta 3.10. *číselné porovnání($x = y$) je v množině základních funkcí*

Důkaz. Nechť jsou x a y proměnné, pak definujme funkci f, která vrátí jedničku pokud proměnné x a y reprezentují stejné číslo. Pokud nereprezentují stejné číslo, pak vrací nulu.

$$f = \text{Choice}(x \leq y, y \leq x, 0)$$

□

3.1.6 Binární délková funkce

Tato funkce slouží jako přístup k délce proměnných a budeme ji používat především v sekci s délkovou rekurzí, jelikož délka proměnných a hloubka rekurze spolu silně souvisí. Nicméně tím se budeme zabývat až v další kapitole.

3.2 Základně polynomiální podmínky

Definice 3.11. Charakteristická funkce
Definujme charakteristickou funkci f množiny A v množině B tak, že pro libovolné x z B platí:

$$f(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if Jinak} \end{cases}$$

Definice 3.12. Základně polynomiální množina a základně polynomiální podmínka
Definujme základně polynomiální množinu, jako libovolnou množinu, jejíž charakteristická funkce je v množině základních funkcí. Takovouto charakteristickou funkci budeme nazývat základně polynomiální podmínkou. O polynomiálních podmínkách budeme tvrdit, že definují základně polynomiální funkce. Například základně polynomiální podmínka:

$$x < 3$$

definuje v přirozených číslech základně polynomiální množinu:

$$\{\ x \mid x < 3\} = \{0, 1, 2\}$$

Dohodněme se, že budeme dále značit základně polynomiální podmínky pomocí malých písmen řecké abecedy (např. φ nebo ψ).

3.2.1 Uzavřenost základně polynomiálních podmínek na výrokové logické operace

V této sekci budeme odvozovat logické operace. Je třeba poznamenat, že základně polynomiální podmínky vrací logické hodnoty, tedy 0 a 1. Nemusíme tedy mít strach, že by logická operace dostala na vstupu číslo větší než 1.

Lemma 3.13. *Uzavřenost základně polynomiálních podmínek na negaci($\neg\varphi$)*

Důkaz. Inspirováno prací [1]. Nechť φ je základní polynomiální podmínkou. Pak základní polynomiální podmínku ψ, jež reprezentuje její negaci, odvodíme takto:

$$\psi = \varphi \leq 0$$

□

Lemma 3.14. *Uzavřenost základně polynomiálních podmínek na konjunkci ($\varphi \wedge \psi$)*

Důkaz. Inspirováno prací [1]. Nechť φ a ψ jsou základní polynomiální podmínky. Pak základní polynomiální podmínku χ, jež reprezentuje jejich konjunkci, odvodíme takto:

$$\chi = \text{Choice}(\varphi, \psi, 0)$$

□

Lemma 3.15. *Uzavřenost základně polynomiálních podmínek na disjunkci ($\varphi \vee \psi$)*

Důkaz. Nechť φ a ψ jsou základní polynomiální podmínky. Pak základní polynomiální podmínku χ, jež reprezentuje jejich disjunkci, odvodíme takto:

$$\chi = \neg(\neg\varphi \wedge \neg\psi)$$

□

Věta 3.16. *Uzavřenost na logické operace*
Tím, že jsme dokázali uzavřenost na negaci a konjunkci jsme dokázali uzavřenost na všechny logické operace, jelikož všechny ostatní operace jsou z nich odvoditelné. Formální důkaz nebudu uvádět, nicméně nastíním postup důkazu. Každý logický operátor lze popsat pomocí tabulky hodnot. Takovouto tabulku jsme pak schopni popsat, jelikož se jedná o disjunkci logicky platných řádek. Každou řádku tabulky lze popsat pomocí konjukcí "buněk" v daném řádku. Dále jsme schopni popsat "buňku" pomocí použití/nepoužití negace a proměnné, která se k řádku vztahuje. Například tedy $\neg x$ pro 0 a x pro 1.

4 Definice rekurzívních schémat

Nejdříve definujeme 2 typy rekurze a dokážeme jejich ekvivalenci, poté vytvoříme systém pomocí základních funkcí, jejich skládání a omezené rekurze. O tomto systému se dozvíme, že jeho uzávěr je roven právě množině polynomiálně počitatelných funkcí.

4.1 Pojmy a proměnné k rekurzivním schématům

Dříve, než uvedeme definice, vysvětlíme pojmy a proměnné, které se k definicím budou vztahovat.

Proměnná x v rekurzivních schématech

Nechť je $\underline{x}$ množina proměnných, která obsahuje vstupní parametry funkce.

Proměnná y v rekurzivních schématech

Proměnná y jsou spojena s hloubkou rekurze u rekurzivních funkcí. Snažme se tedy proměnnou y volit tak aby odpovídala hloubce potřebné rekurze. V některých případech však budeme nuceni volit hloubku rekurze větší, než je nutno.

Prostorové omezení

Prostorové omezení nám říká, že růst počtu cifer výsledného čísla je omezený nějakým polynomem.

Časové omezení

Časové omezení nám říká, že růst hloubky rekurze v průběhu výpočtu je omezený nějakým polynomem.

4.2 Definice rekuzivních schémat

Definujme dva typy polynomiálni rekurze.

Definice 4.1. Délková rekurze
Jedná se o stejnou definici, jakou má práce pana Macha[3]. Nechť g je libovolná funkce $g : \mathrm{N}^k \to \mathrm{N}$ pro $k \leq 0$ a f je libovolná funkce $f : \mathrm{N}^{k+2} \to \mathrm{N}$pro $k \leq 0$. Dále nechť q je libovolný vhodný polynom. Pak funkce $f : \mathrm{N}^k \to \mathrm{N}$ je odvozena omezenou rekurzí z funkcí f,g s časovým omezením $|y|$ a

prostorovým omezením pomocí polynomu q, jestliže pro libovolné $|y|$ platí následující:

- $f(\underline{x}, y) = \tau(\underline{x}, |y|)$
- $\tau(\underline{x}, 0) = g(\underline{x})$
- $\tau(\underline{x}, n+1) = h(\underline{x}, n, \tau(\underline{x}, n))$

A navíc:
$\forall n \leq |y|(|\tau(\underline{x}, n)| \leq q(|\underline{x}|, |y|))$

Definice 4.2. Polynomiální rekurze
Jedná se o stejnou definici, jakou má práce pana Macha[3]. Nechť g je libovolná funkce $g : \mathrm{N}^k \to \mathrm{N}$ pro $k \leq 0$ a f je libovolná funkce $f : \mathrm{N}^{k+2} \to \mathrm{N}$ pro $k \leq 0$. Dále nechť p je libovolný vhodný polynom. Pak funkce $f : \mathrm{N}^k \to \mathrm{N}$ je odvozena polynomiální rekurzí z funkcí f,g s časovým omezením pomocí polynomu q, jestliže platí následující:

- $f(\underline{x}, y) = h(\underline{x}, y, f(\underline{x}, \lfloor y/2 \rfloor))$
- $f(\underline{x}, 0) = g(\underline{x})$

A navíc:
$|f(\underline{x}, y)| \leq q(|\underline{x}|, |y|)$

4.3 Pomocné funkce

Při důkazu ekvivalence délkově rekurziních funkcí budeme potřebovat odvození některých funkcí. Začněme tedy s pokusem odvodit bitový posun vlevo, což by se nám nemělo povést, protože se jedná o exponenciální funkci.

Teorém 4.3. *$x_1 \ll x_2$ lze odvodit pomocí délkové rekurze pomocí délkové rekurze s dosazením*

Důkaz. Všimněme si, že se jedná o funkci $x_1 * 2^{x_2}$. To je ale funkce exponenciální a tedy nemůže být omezena žádnou polynomiální funkcí. Naše odvození funkce $x_1 \ll x_2$ tedy selhalo. □

Zkusme tedy bitový posun vlevo trochu poupravit, aby nám šel odvodit. Následují funkce provede za každou cifru čísla x_2 posun čísla x_1 o jedna vlevo.

Lemma 4.4. *$x_1 \ll |x_2|$ lze odvodit pomocí délkové rekurze*

Důkaz. Nyní máme lépe zvolené vstupní parametry a funkce by měla jít omezit polynomem:

- $\underline{x} = \{x_1, x_2\}$
- $y = x_2$
- $g(\underline{x}) = \underline{x}$
- $h(\underline{x}, n, \tau(\underline{x}, n)) = \tau(\underline{x}, n) \ll 1$

Opět zkusme rozdělit funkci na dvě funkce a obě omezit polynomem. $P_1 = x_1$ a $p_2 = |2^{|x_2|}| = |x_2|$. Neboli $q = |x_1| + |x_2| = |\underline{x}|$ □

Stejným způsobem provedeme odvození funkce pro posun vlevo. Funkce tedy za každou cifru čísla x_2 posun čísla x_1 o jedna vpravo.

Lemma 4.5. *$x_1 \gg |x_2|$ lze odvodit pomocí délkové rekurze*

Důkaz.

- $\underline{x} = \{x_1, x_2\}$
- $y = x_2$
- $g(\underline{x}) = \underline{x}$
- $h(\underline{x}, n, \tau(\underline{x}, n)) = \tau(\underline{x}, n) \gg 1$

Vzhledem k tomu, že délku čísla x_1 zmenšujeme, tak můžeme omezit polynomem $q = (|\underline{x}|)$. □

Další z pomocných funkcí je zbytek po dělení dvěma. Tato funkce je shodná s funkcí, jež přečte poslední bit. Při odvození této funkce smažeme z x poslední bit a nahradíme ho nulou. Poté porovnáme s původním číslem x. Pokud se jedná o stejné číslo, pak byla na konci čísla x nula. Jinak se na konci čísla nachází jednička.

Lemma 4.6. *Zbytek po dělení dvěma ($x \,\%\, 2$) lze odvodit pomocí délkové rekurze*

Důkaz. Odvodíme funkci pro modulo dvěma

$$x \,\%\, 2 = \begin{cases} 0 & \text{if } \text{asl}(\text{asr}(f(x))) = x \\ 1 & \text{if } \text{Jinak} \end{cases}$$

□

Omezme polynomem $q = 1$
Dále budeme potřebovat funkci pro odečtení jedničky. Ta by měla pracovat tak, že všechny nuly na konci čísla změní v jedničky a zároveň změní první jedničku od konce v nulu.

Lemma 4.7. *$x \dot{-} 1$ lze odvodit pomocí délkové rekurze*

Důkaz. Nejdříve si připravíme funkci f, která smaže všechny nuly na konci čísla.

- $\underline{x} = \{x\}$
- $y = x$
- $g(\underline{x}) = \underline{x}$
- $h(\underline{x}, n, \tau(\underline{x}, n)) = \begin{cases} \text{asr}(\tau(\underline{x}, n)) & \text{if } \tau(\underline{x}, n)[0] = 0 \\ \tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Vzhledem k tomu, že délku čísla x_1 zmenšujeme, tak můžeme omezit polynomem $q = |x_1|$.

Dále si odvodíme funkci $\dot{-}1$, tak že jedničku na konci čísla přepíšeme na nulu a doplníme adekvátní počet nul pomocí rekurze. Za zmínku stojí fakt, že takováto funkce nijak neovlivní číslo 0.

- $\underline{x} = \{x\}$
- $y = x$
- $g(\underline{x}) = \text{asl}(\text{asr}(f(x)))$
- $h(\underline{x}, n, \tau(\underline{x}, n)) = \begin{cases} asl(\tau(\underline{x}, n)) & \text{if } asl(\tau(\underline{x}, n)) \leq x \\ \tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Vzhledem k tomu, že délku čísla x_1 zmenšujeme, tak můžeme omezit polynomem $q = |x_1|$. □

Předposlední z pomocných funkcí je jakési délkové odčítání, které je užitečné a extrémně jednoduché na odvození.

Lemma 4.8. *$x_1 - |x_2|$ lze odvodit pomocí délkové rekurze*

Důkaz. Funkce $x_1 - |x_2|$ představuje rozdíl délky čísla x_1 a čísla x_2. Podmínkou pro správné fungování této funkce je $x_1 \leq |x_2|$. Je tedy třeba tuto podmínku ověřit při každém použití.

- $\underline{x} = \{x_1, x_2\}$
- $y = x_2$
- $g(\underline{x}) = x_1$
- $h(\underline{x}, n, \tau(\underline{x}, n)) = \tau(\underline{x}, n) \dot{-} 1$

Vzhledem k tomu, že délku čísla $|x_1|$ zmenšujeme, tak můžeme omezit polynomem $q = (|\underline{x}|)$. □

Poslední z pomocných funkcí je vylepšení posunu vpravo. Odvození probíhá tak, že rekurze aplikuje posun vpravo, dokud $x_1 - |tau(\underline{x}, n)| < x_2$, tedy dokud byla proměnná posunuta rekurzí o méně, než x_2 bitů. Jakmile byla posunuta dostatečně, pak rekurze přestane pracovat.

Lemma 4.9. *$x_1 \gg x_2$ lze odvodit pomocí délkové rekurze*

Důkaz. Funkce $x_1 \gg x_2$ představuje vylepšení funkce $x_1 \gg |x_2|$, kterou jsme odvodili dříve.

- $\underline{x} = \{x_1, x_2\}$
- $y = x_1$
- $g(\underline{x}) = x_1$
- $h(\underline{x}, n, \tau(\underline{x}, n)) = \begin{cases} \text{asr}(tau(\underline{x}, n)) & \text{if } x_1 - |tau(\underline{x}, n)| < x_2 \\ tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Vzhledem k tomu, že délku čísla $|x_1|$ zmenšujeme, tak můžeme omezit polynomem $q = (|\underline{x}|)$. □

5 Ekvivalence délkově rekurzivních schémat

V této sekci ukážeme rovnost rekurzivních schémat. Při odvozování budeme značit odvozované schéma a jeho proměnné s aspostrofy, aby se mohli ve funkcích vyznat. Navíc budeme používat funkci projekce trochu nekorektně (bereme pře projekci $\underline{x}/\underline{x}'$ jako jednu proměnnou), nicméně nesprávné použití ušetří dosti místa a přehlednosti.

Věta 5.1. *4.1 $\longmapsto$ 4.2*

Důkaz. V této sekci si ukážeme 4.1 $\longmapsto$ 4.2. Důkaz bude probíhat pomocí indukce. Nejdříve si zvolíme proměnné $\underline{x}'$ a y'(y' pro nultý krok rekurze). Dále si definujeme funkce g' a h'. Všimněme si, že můžeme použít g a h z odvození pomocí rekurzivního schématu 4.1. Funkce g' a h' budou tedy pouze upravovat parametry a dosazovat je do funkcí g a h.

- $\underline{x}' = \underline{x}$,
- $y' = y$ (v nultém kroku rekurze)
- $g'(\underline{x}') = g(\mathrm{i}_1^1)$
- $h'(\underline{x}', y', f'(\underline{x}', \lfloor y'/2 \rfloor)) = h(\mathrm{i}_1^3, |\mathrm{i}_2^3|, \mathrm{i}_3^3)$

Nejdříve si dokažme ekvivalenci $f'(\underline{x}', 0) = f(\underline{x}, 0)$

$$f'(\underline{x}', 0) = \ g'(\underline{x}') = g(\mathrm{i}_1^1) = g(\underline{x}') = g(\underline{x}) = \tau(\underline{x}, 0) = f(\underline{x}, 0)$$

Rád bych nyní probral jednotlivé rovnosti v této ekvivalenci. První a druhá rovnost plyne z definice plynomální funkce. Třetí rovnost plyne z toho, jak jsme si nastavili funkci g'. Čtvrtá rovnost plyne z definice projekce proměnných. Pátá rovnost pak plyne z definice funkce $\underline{x}'$. Šestá a sedmá rovnost plyne z definice délkové rekurze. Tímto jsme dokázali bázi indukce.
Dále si dokažme ekvivalenci $f(\underline{x}', y') = f(\underline{x}, y)$

$$\begin{aligned} f'(\underline{x}', y') &= h'(\underline{x}', y', f'(\underline{x}', \lfloor y'/2 \rfloor)) = h(\mathrm{i}_1^3, |\mathrm{i}_2^3|, \mathrm{i}_3^3) = h(\underline{x}', |y'|, \tau'(\underline{x}', n')) \\ &= h(\underline{x}, n, \tau(\underline{x}, n)) = \tau(\underline{x}, n+1) = f(\underline{x}, y) \end{aligned}$$

Opět probereme rovnosti, první z rovností plyne z definice polynomiální funkce. Druhá rovnost plyne z naší definice funkce h. Třetí rovnost vyplývá z definice projekce proměnných. Další rovnost je složitější. Rovnost prvních argumentů plyne z naší definice $\underline{x}'$. Třetí argumenty jsou si rovny díky indukčnímu předpokladu. Rovnost $n = |y|'$ na současné hloubce rekuze je potřeba dokázat indukcí. Nebudu dokazovat formálně, nicméně myšlenka je taková, že na nejvyšší úrovni rekurze platí $n = |y'|$, díky naší definici y'. Máme tedy bázi indukce. Nechť se tedy rovnají na k-té úrovni indukce. na $k+1$-té úrovni indukce by mělo platit $n - 1 = ||\lfloor y'/2 \rfloor||$, což očividně platí, takže $n = |y'|$ platí na všech úrovních rekurze. Vraťme se nyní k dokazování rovností. Poslední dvě rovnosti plynou z definice délkové rekurze. Všimněme si, že jsme v důkazu použili dvě indukce. Rovnost délkových omezení je triviální. □

Lemma 5.2. *$f'''(x_1, x_2)$ zkrátí číslo x na délku y lze odvodit pomocí délkové rekurze*

Důkaz.

- $\underline{x} = \{x_1, x_2\}$
- $y = x_1$
- $g(\underline{x}) = x_1$
- $h(\underline{x}, n, \tau(\underline{x}, n)) = \begin{cases} \mathrm{asr}(tau(\underline{x}, n)) & \text{if } tau(\underline{x}, n) > x_2 \\ tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Vzhledem k tomu, že délku čísla $|x_1|$ zmenšujeme, tak můžeme omezit polynomem $q = (|\underline{x}|)$. □

Věta 5.3. *4.2 $\longmapsto$ 4.1*

Důkaz bude probíhat pomocí indukce. Nejdříve si zvolíme proměnné $\underline{x}'$ a $y'(y'$ pro nultý krok rekurze). Dále si definujeme funkce g' a h'. Všimněme si, že můžeme použít g a h z odvození pomocí rekurzivního schématu 4.2. Funkce g' a h' budou tedy pouze upravovat parametry a dosazovat je do funkcí g a h.

Důkaz.

- $\underline{x}' = \underline{x}, y$ (kde se jedná o y v nultém kroku rekurze)
- $y' = y$ (v nultém kroku rekurze)
- $g'(\underline{x}, y) = g(\mathrm{i}_1^1)$
- $h'(\underline{x}, y, y', f'(\underline{x}', \lfloor y'/2 \rfloor)) =$

$$f'(\underline{x}', 0) = \tau'(\underline{x}', 0) \;=\; g'(\underline{x}') = g'(\underline{x}, y) = g(\mathrm{i}_1^1) = g(\underline{x}) = f(\underline{x}, 0)$$

Rád bych nyní probral jednotlivé rovnosti v této ekvivalenci. První a druhá rovnost plyne z definice délkové rekuze. Třetí rovnost plyne z toho, jak jsme si definovali x. Čtvrtá rovnost plyne z toho, jak jsme si nastavili funkci g'. Pátá rovnost plyne z definice projekce proměných. Šestá rovnost plyne z definice délkové rekurze. Tímto jsme dokázali bázi indukce.

$$\begin{aligned} f'(\underline{x}', y, y') &= \tau'(\underline{x}, y, n' + 1) = h'(\underline{x}, y, n', \tau'(\underline{x}, y, n')) = h(\mathrm{i}_1^4, f'''(\mathrm{i}_2^4, \mathrm{i}_3^4), \mathrm{i}_4^4) \\ &= h(\underline{x}, f'''(y, n'), \tau(\underline{x}, n)) = h(\underline{x}, y, f(\underline{x}, \lfloor y/2 \rfloor)) = f(\underline{x}, y) \end{aligned}$$

Opět proberme rovnosti. První rovnost plyne z naší definice x a definice délkové rekurze. Druhá rovnost vyplývá z definice délkové rekurze. Třetí z rovnost vyplývá z toho jak jsme si definovali funkci h. Čtvrtá rovnost vyplývá z definice funkce projekce. Další rovnost je složitější. Rovnost prvních argumentů je triviální. Druhé argumenty jsou výsledkem definice funkce f'''. Rovnost třetích argumentů vyplývá z indukčního předpokladu. Poslední dvě rovnosti vyplývají z definice plynomiální funkce. Rovnost délkových omezení je triviální.

□

Věta 5.4. *Polynomiální rekurze lze napsat v následující formě.*
$f(\underline{x}, 2 * y) = h_1(\underline{x}, y, f(\underline{x}, y))$
$f(\underline{x}, 2 * y + 1) = h_2(\underline{x}, y, f(\underline{x}, y))$
$f(\underline{x}, 0) = g(\underline{x})$
A navíc:
$|f(\underline{x}, y)| \leq q(|\underline{x}|, |y|)$

Důkaz. Důkaz je triviální (podobný dvěma předešlým důkazům, nicméně trochu snažší). □

6 Polynomiální funkce

Definice 6.1. Množina základních funkcí s rekurzí
Definujme množinu základních funkcí s rekurzí, jako nejmenší rozšíření množiny základních funkcí, která je uzavřená na schéma polynomiální rekurze.

Definice 6.2. Polynomiální množina a polynomiální podmínka
Definujme polynomiální množinu, jako libovolnou množinu, jejíž charakteristická funkce je v množině základních funkcí s rekurzí. Takovouto charakteristickou funkci budeme nazývat polynomiální podmínkou. O polynomiálních podmínkách budeme tvrdit, že definují základně polynomiální funkce.

Věta 6.3. *Polynomiální podmínky a délkově omezená kvantifikace*

Důkaz. Definujme délkově omezenou kvantifikaci nad základní polynomiální podmínkou ψ obsahující proměnnou x, jako jednu z formulí ve tvaru:

- $(\forall y \leq |x|)(\psi(y))$
- $(\forall y < |x|)(\psi(y))$
- $(\exists y \leq |x|)(\psi(y))$
- $(\exists y < |x|)(\psi(y))$

Nyní dokážeme, že polynomiální podmínky jsou uzavřeny délkově omezenou kvantifikaci. Pro jednoduchost provedu důkaz pro první z tvarů délkově omezené kvantifikace. Ostatní tvary délkově omezené kvantifikace mají podobný důkaz. Dokazujme tedy pro $f(2z)$ a $f(2z + 1) = f(\forall y \leq |x|)(\psi(y))$. Odvoďme tedy funkci $f(2z)$, tak že pro ψ, kde je y substituováno hodnotou proměnné z (funkce sub umí substituovat pouze konstantou, nicméně odvození funkce, která substituuje hodnotou proměnné by nebylo obtížné). Výslednou hodnotu dosaďme do konjunkce spolu s hodnotou funkce $f(z)$. Na úrovni rekurze $|z|$ zavolá funkce $|z - 1|$, čímž vznikne konkunkce $|x| + 1$ funkcí ψ, kde je za y postupně dosazeno 0 až $|x|$, což jsme chtěli.

$f(0) = 1$
$f(2z) = f(z) \wedge \text{sub}(|2z|, y, \psi)$
$f(2z+1) = f(z) \wedge \text{sub}(|2z|, y, \psi)$

Omezme pomocí $q = 1$ □

7 Odvození funkcí s použitím rekurze

V této sekci se pokusíme odvodit nejpoužívanjší polynomiální funkce a některé z funkcí, které jsem shledal jako zajímavé.

7.1 Přípravné funkce

Před tím, než si odvodíme polynomální funkce je však třeba si odvodit jednodušší funkce, které nám pomohou k odvození dálších funkcí. První z funkcí, které budeme odvozovat je funkce pro přístup k $y-$tému bitu zprava, s tím že se budeme držet programátorského kodexu a budeme číslovat od nuly. Funkci odvodíme tak, že porovnáme číslo x bez posledních y bitů a číslo x bez posledních $y+1$ bitů nakonci doplněné o nulu nakonec. Pokud se čísla rovnají, pak je y-tý bit nula, jinak je y-tý bit jednička.

Lemma 7.1. *$x[y]$ je v množině základních funkcí s rekurzí*

Důkaz. Nechť x je číslo, ve kterém chceme znát velikost y-tého bitu. Dohodněme se, že podle programátorské tradice budeme číslovat bity od nultého bitu. Nejdříve odvoďme funkci g, která nám vrátí výsledek pro $y \leq |x|$

$$g(x,y) : \begin{cases} 0 & \text{if } (x \gg y) = \text{asl}(x \gg \text{s}(y)) \\ 1 & \text{if Jinak} \end{cases}$$

□

Za zmínku stojí, že funkce funguje i pro $y > |x|$, jelikož porovnáváme dvě nuly a proto funkce vrátí nulu, což sme chtěli. Další z funkcí, kterou budeme odvozovat, bude konkatenační funkce. Tato funkce provede konkatenaci čísla s bitem. Dalo by se tedy říci, že provede konkatenaci čísla x_1 s číslem jedna, nebo nula. Pokud je x_2 větší než jedna, pak považujeme x_2 za číslo jedna. Tuto funkci oceníme především při kódování posloupností v další kapitole. Dovození probíhá pomocí bitového pusunu vlevo o jeden bit a přičtení jedničky pro $x_2 > 0$.

Lemma 7.2. $\text{add}(x_1, x_2)$ *je v množině základních funkcí s rekurzí*

Důkaz. Odvoďme si funkci $\text{add}(x_1, x_2)$ která na konec binárního zápisu čísla x_1 doplní číslici x_2, kde x_2 je brána jako číslice jedna, nebo nula.

$$f(x,y) : \begin{cases} \text{asl}(x_1) & \text{if } x_2 = 0 \\ \text{s}(\text{asl}(x_1)) & \text{if Jinak} \end{cases}$$

Omezme polynomem $q = |x_1| + 1$ □

Poslední z pomocných funkcí je funkce pro reverzi bitů. Tato funkce provede reverzi bitů v čísle. Nicméně pro některá čísla by se některé bity ztratily. Právě z tohoto důvodu je tato funkce binární s tím, že jako druhý argument je libovolné číslo. Toto je číslo je funkcí umístěno na začátek čísla (bez revertování) a může tak zamezit ztrátě nul. Je třeba poznamenat, že zvolení nuly, jako druhého argumentu neumístí před číslo nic.

Lemma 7.3. $\text{rev}(x_1, x_2)$ *je v množině základních funkcí s rekurzí*

Důkaz. Definujeme funkci, která bude sloužit, jako reverze bitů čísla. Máme navíc možnost přidat x_2 před reverzi, což se nám hodí, pokud hrozí ztráta nulových bitů na konci čísla.

- $\underline{x} = x;2$
- $y = |x|$
- $g(\underline{x}) = x_2$
- $h(\underline{x}, y, f(\underline{x}, \lfloor y/2 \rfloor)) = \text{add}(f(\underline{x}, \lfloor y/2 \rfloor), x_1[y])$

Omezme polynomem $q = |x_1| + |x_2|$ □

7.2 Aritmetické funkce

Dále odvodíme aritmetické funkce, tedy ty polynoimální funkce, jež používáme nejčastěji. Jmenovitě se jedná o funkce plus, mínus, krát, děleno a modulo $(+, -, *, /, \%)$. Začneme s funcí sčítání. Odvození kopíruje školní algoritmus pro sčítání (algoritmus sčítání pod sebou) kde přechod na vyšší bit čísla x_2 je reprezentován jako $2x_2$ (nebo $2x_2 + 1$) a carry bit(takzvané držení jedničky) je reprezentován funkcí s.

Věta 7.4. $x_1 + x_2$ *je v množině základních funkcí s rekurzí*

Důkaz. Odvodíme funkci pro sčítání dvou čísel, která bude využívat klasický školní sčítací algoritmus.

$$x_1 + 2(x_2) = \begin{cases} \text{asl}(\text{asr}(x_1) + x_2) & \text{if } x_1 \,\%\, 2 = 0 \\ \text{s}(\text{asl}(\text{asr}(x_1) + x_2)) & \text{if } x_1 \,\%\, 2 = 1 \end{cases}$$

$$x_1 + 2(x_2) + 1 = \begin{cases} \mathrm{s}(\mathrm{asl}(\mathrm{asr}(x_1) + x_2)) & \text{if } x_1 \,\%\, 2 = 0 \\ \mathrm{s}(\mathrm{s}(\mathrm{asl}(\mathrm{asr}(x_1) + x_2))) & \text{if } x_1 \,\%\, 2 = 1 \end{cases}$$

$x_1 + 0 = x_1$

Omezme polynomem $q = |x_1| + |x_2|$ □

Dále si odvodíme funkce, pro součin dvou čísel. Je třeba poznamenat, že s pomocí polynomiální rekurze a funkce pro sčítání je odvození násobící funkce velice intuitivní a jednuduché. Stačí uvážit, že $x_1 * 2x_2 = 2 * x_1 * (x_2)$ a $x_1 * (2x_2 + 1) = (x_1 * 2x_2) + x_1$.

Věta 7.5. *$x_1 * x_2$ je v množině základních funkcí s rekurzí*

Důkaz. Odvodíme funkci pro násobení dvou čísel.

$x_1 * 2(x_2) = \mathrm{asr}(x_1 * x_2)$
$x_1 * (2x_2 + 1) = \mathrm{asr}(x_1 * x_2) + x_1$
$x_1 * 0 = 0$

Omezme polynomem $q = |x_1| + |x_2|$

□

Další z aritmetických funkcí je funkce pro rozdíl dvou čísel. Její odvození nebudu uvádět, jelikož školní algoritmy pro sčítání a odčítání fungují podobně.

Věta 7.6. *$x_1 - x_2$ je v množině základních funkcí s rekurzí*

Důkaz. Odvození mínusové funkce je velice podobné odvození funkce pro sčítání. Nicméně se při odvození používá funkce $\dot{-}1$. □

Jako předposlední z aritmetických funkci máme pro zbytek při dělení, neboli funkce modulo. Odvození funkce je velice prosté, jelikož vynásobení čísla dvěma znamená i vynásobení zbytku dvěma. Stejně zvětšení čísla o jedna znamená zvětšení zbytku o jedna.

Věta 7.7. *$x_1 \,\%\, x_2$ je v množině základních funkcí s rekurzí*

Důkaz. Nejdříve si odvodíme funkci g:
$$g(x_1, x_2) = \begin{cases} x_1 & \text{if } x_1 < x_2 \\ x_1 - x_2 & \text{if Jinak} \end{cases}$$

Nyní odvoďme funkci modulo:

$2x_1 \,\%\, x_2 = g(\mathrm{asl}(x_1 \,\%\, x_2), x_2)$

$(2x_1 + 1) \,\%\, x_2 = g(\mathrm{s}(\mathrm{asl}(x_1) + x_2))$
$0 \,\%\, x_2 = 0$

Omezme polynomem $q = |x_2|$ □

Poslední z aritmetických funkcí je funkce pro dělení dvou čísel. Odvození pracuje s tím, že pokud vynásobím číslo dvěma, pak je vynásoben dvěma i výsledek dělení a zbytek po dělení. Pro $(2x_1)/x_2$ tedy stačí sečíst dvojnásobek dosavadního výsledku po dělění $(2*(x/y))$ a číslo jedna pokud je dvojnásobek dosavadního zbytku větší než x_2 (pokud $2(x \,\%\, y) > x_2$). Pro $(2x_1 + 1)/x_2$ je algoritmus stejný, jen dvojnásobek dosavadního zbytku zvětšíme o jedna.

Věta 7.8. *$x_1 \,/\, x_2$ je v množině základních funkcí s rekurzí*

Důkaz. Odvodíme si funkci pro dělení:

$(2x_1 + 1) \,/\, x_2 = 2(x \,/\, y) + ((2(x \,\%\, y) + 1) \,/\, y)$
$(2x_1) \,/\, x_2 = 2(x \,/\, y) + ((2(x \,\%\, y)) \,/\, y)$
$0 \,/\, x_2 = 0$

Omezme polynomem $q = |x_1|$
K funkci pro dělení je však potřeba dodat, že nesmí být děleno nulou. Dělení nulou není definováno a bude dávat nesmyslné výsledky.

□

7.3 Zajímavé funkce

V této sekci se podíváme na funkce, které mně přišli zajímavé tím, že obsahují exponenciální funkci. To je zajímavé z toho hlediska, že všechny problémy, které jsou exponenciálně náročné, nejsou zpravidla polynomiální. Nicméně, všechny funkce které se nám povede odvodit pracují v polynomiálním čase i přesto, že obsahují ve svém předpisu funkci mocniny. První z takovýchto funkcí bude x_1 na délku x_2. Odvození probíhá tak, že začneme s jedničkou a každou cifru čísla x_2 vynásobíme dvěma. V odvozování této funkce jsem se nechal inspirovat prací[3].

Věta 7.9. *$2^{|x_1|}$ je v množině základních funkcí s rekurzí*

Důkaz.

- $\underline{x} = \{x_1\}$
- $y = x_1$
- $g(\underline{x}) : 1$

- $h(\underline{x}, y, f(\underline{x}, \lfloor y/2 \rfloor)) : \tau(\underline{x}, n) * 2$

Omezme polynomem $q = |x_1| + 2$. □

Další ze zajímavých funkcí je takzvaná funkce "Smash". Tato funkce dokonce obsahuje násobení uvnitř exponenciální funkce a mohlo by se zdát, že se jedná o funkci exponenciální. Jedná se však o funkci polynomiální. Všimněme si, že $2^{|x_1|*|x_2|} = 2^{|x_1|^{|x_2|}}$. Použijeme tedy odvozenou funkci $2^{|x_1|}$. Začneme s opět s jedničkou a za kažkou cifru čísla x_2 vynásobíme mezivýslek číslem $2^{|x_1|}$. V odvozození této funkce jsem se nechal inspirovat prací[3].

Věta 7.10. *$Smash(2^{|x_1|*|x_2|})$ je v množině základních funkcí s rekurzí*

Důkaz.

- $\underline{x} = \{x_1, x_2\}$
- $y = x_2$
- $g(\underline{x}) : 1$
- $h(\underline{x}, y, f(\underline{x}, \lfloor y/2 \rfloor)) : \tau(\underline{x}, n) * 2^{|x_1|}$

Omezme polynomem $q = (|x_1| + 2) * (|x_2| + 2)$. □

Další z odvozovaných funkcí je několikrát zmíněná exponenciální funkce. O této funkci je jasné, že nepatří do množiny polynomiálních funkcí a důkaz musí tedy někde selhat. Odvození ukazuje, kde přesně důkaz selže.

Věta 7.11. *$x_1^{x_2}$ je v množině základních funkcí s rekurzí*

Důkaz. Pokud se podíváme na funkci, tak je jasné, že počet růstu cifer je exponenciální a nepůjde tedy omezit polynomem. □

8 Kódování posloupností

Pro kódování funkcí budeme potřebovat kódovací funkci, dekódovací funkci a případně ještě další funkce pro manipulaci se zakódovanými posloupnostmi. Abychom mohly tyto funkce odvodit, tak nejříve potřebujeme zvolit způsob kódovaní. My budeme kódovat posloupnosti pomocí takzvaných Gödelových čísel.

Definice 8.1. Gödelovo číslo pro posloupnost
Nechť $M = \{x_0, \ldots, x_n\}$ je množina proměnných, na které můžeme definovat uspořádání pomocí indexů čísel. Gödelovo číslo pro množinu M, na které jsme si definovali upořádání pomocí indexů, dostaneme takto. Vezmeme konkatenaci binarních reprezentací čísel x_0 až x_n s tím, že jednotlivá číla oddělíme čárkami (čísla konkatenujeme pomocí našeho uspořádání). Výsledkem bude řetězec nul, čárek a jedniček. Vezměme reverzi tohoto řetězce a prověďme simultální substituci 10 za 0, 11 za 1 a 00 . Výsledek můžeme přečíst jako binární číslo. Toto číslo prohlašme za Gödelovo číslo pro posloupnost $< x_0, \ldots, x_n >$.

Dále budeme potřebovat takzvanou prázdnou posloupnost, pro kterou si vyhradíme číslo nula.

Definice 8.2. Prázdná posloupnost <>
Definujme prázdnou posloupnost: $<>= 0$

Pro definování kódování bude pochopitelně potřebovat funkci která převede číslo na tvar Gödelova čísla. K tomu si nejdříve odvoďme dvojitou simultální substituci. Ta jde po cifrách čísla x_1 a za každou nalezenou nulu přidá k mezivýsledku jedničku a nulu nakonec. Za každou jedničku přidá dvě jedničky nakonec.

Věta 8.3. *Dvojitá simultální substituce(pro tvorbu Gödelova čísla) je v množině základních funkcí s rekurzí*

Důkaz. Provede simultální substuci 10 za 0 a 11 za 1. výslednou funkci značme $\text{subs}(x_1)$

- $\underline{x} = \{x_1\}$
- $y = \text{asl}(x_1)$
- $g(x) : 0$
- $h(x) : \begin{cases} \text{add}(\text{add}(\tau(\underline{x}, n), 1), 0) & \text{if } x_2[n-1] = 0 \\ \text{add}(\text{add}(\tau(\underline{x}, n), 1), 1) & \text{if Jinak} \end{cases}$

Omezme polynomem $q = |x_1 * 2|$.
Tím jsme vytvořili funkci $f(x_1)$, pro jistotu nyní odvoďme $f'(x_1)$, která bude zabezpečená na nulu.

$$f'(x_1) = \begin{cases} 01 & \text{if } x_1 \equiv 0 \\ f(x_1) & \text{if Jinak} \end{cases}$$

□

Dále budeme pořebovat rozšiřování již vzniklých posloupností. Právě na to si tedy odvoďme funkci, pro přidání proměnné do posloupnosti. Odvození probíhá tak, že na proměnnou x_0 použijeme simultální substituci a přidáme čárku (dvě nuly) pomocí funkce add. Poté vezmeme posloupnost do níž chceme vkládat a převedeme jí do nereverziního stavu, pomocí reverze. Dále vezmeme naši x_0 s čárkou a přidáme naši nakonec posloupnost v nereverziním stavu. To provedeme tak že x_0 s čárkou posuneme vlevo o délku naší posloupnosti a tuto posloupnost přičteme. Vznikne nám nerevertovaná verze naší rošířené posloupnosti a proto nám stačí použít reverzi čímž dostaneme výsledek.

Věta 8.4. *Přídání proměnné do posloupnosti* $x_0* < x_1, \ldots, x_n >$ *je v množině základních funkcí s rekurzí*

Důkaz. Nechť $< x_1, \ldots, x_n >$ je proměnná $n-1$ proměnných. Definujme přidání proměnné x_0 do této posloupnosti (na první místo v posloupnosti):

$$f(x_0, < x_1, \ldots, x_n >) =$$

$$\text{rev}((\text{add}(\text{add}(\text{subs}(x_0), 0), 0) << | < x_1, \ldots, x_n > |) + \text{rev}(< x_1, \ldots, x_n >))$$

Tím jsme odvodili funkci $f(x_0, < x_1, \ldots, x_n >)$, která funguje, pokud $< x_1, \ldots, x_n > \neq <>$. Odvoďme tedy funkci $f'(x_0, < x_1, \ldots, x_n >)$, která bude fungovat vždy.

$$f'(x_0, < x_1, \ldots, x_n >) : \begin{cases} \text{rev}(\text{subs}(x_0)) & \text{if } < x_1, \ldots, x_n >=<> \\ f(x_0, < x_1, \ldots, x_n >) & \text{if Jinak} \end{cases}$$

□

Pokud dostaneme posloupnost, pak by bylo dobré vědět, kolik členů tato posloupnost obsahuje. Odvoďme tedy pro tento účel funkci len. Odvození probíhá tak, že kontrujeme sudé cifry. Pokud se sudá cifra rovná nule a cifra před ní také, pak přičteme jedničku k počtu členů

Věta 8.5. *Počet členů posloupnosti* - $\text{len}(< x_1, \ldots, x_n >)$ *je v množině základních funkcí s rekurzí*

Důkaz.
Definujme $f'(\underline{x}, y) = \begin{cases} 1 & \text{if } (< x_1, \ldots, x_n > [y] = 0 \wedge < x_1, \ldots, x_n > [y+1] = 0 \\ 0 & \text{if Jinak} \end{cases}$

- $\underline{x} = \{< x_1, \ldots, x_n >\}$
- $y = \text{asl}(< x_1, \ldots, x_n >)$
- $g(x) : 1$

- $h(x) = \begin{cases} \tau(\underline{x}, n) + f'(\underline{x}, n-1) & \text{if } x_1[n-1] \,\%\, 2 = 0 \\ \tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Omezme polynomem $q = |x_0|$.

Máme tedy opět funkci f, která funguje pro všechny posloupnosti, kromě $<>$. Odvoďme tedy funkci f', která bude fungovat pro všechna čísla.

$$f'() : \begin{cases} 0 & \text{if } < x_1, \ldots, x_n > = <> \\ f(< x_1, \ldots, x_n >) & \text{if Jinak} \end{cases}$$

□

Další z funkcí pro manipulaci s posloupností je funkce Truncate, které nám bude sloužit ke zkrácení posloupnosti zleva. odvození probíhá tak, že si definujeme funkci f'', která zkrátí číslo o dva bity prvního čísla (jelikož první číslo je na konci, díky reverzi). Dále se aplikuje dvakrát reverze, což odstraní případnou čárku na začátku čísla. Poté je zavolána rekurze, která tuto funkci volá, dokud není délka posloupnosti o jednu menší.

Věta 8.6. *Zkrácení posloupnosti* - Truncate($< x_1, \ldots, x_n >$) *je v množině základních funkcí s rekurzí*

Důkaz.
Definujme si funkci $f''(x) = rev(rev(x >> 2))$

- $\underline{x} = \{< x_1, \ldots, x_n >\}$
- $y =< x_1, \ldots, x_n >$
- $g(x) =< x_1, \ldots, x_n >$
- $h(x) = \begin{cases} f''(\tau(\underline{x}, n)) & \text{if } \text{len}(\tau(\underline{x}, n)) = \text{len}(< x_1, \ldots, x_n >) \\ \tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Omezme polynomem $q = | < x_1, \ldots, x_n > |$.

Opět jsme odvodili funkci f, která vrací výsledek pro posloupnosti jejíž délka je větší, než 1. Odvoďme pro jistotu funkci f', která zabezpečí funkci proti prázdné posloupnosti.

$$f' = \begin{cases} f(< x_1, \ldots, x_n >) & \text{if } \text{len}(< x_1, \ldots, x_n >) > 1 \\ <> & \text{if Jinak} \end{cases}$$

□

Poslední a asi nejdůležitější funkce, je funkce, jež nám umožní přístup k n-tému členu posloupnosti. Pro tuto funkci však budeme potřebovat odvodit cut zleva a cut zprava. Odvoďme tedy cut zleva tak, že používáme funkci Truncate, dokud není posloupnost chtěné délky.

Lemma 8.7. *Zkrácení posloupnosti zleva na délku* x_1 $\mathrm{lcut}(x_1, < x_2, \ldots, x_n >)$ *je v množině základních funkcí s rekurzí*

Důkaz.

- $\underline{x} = \{x_0, x_1\}$
- $y = \mathrm{asl}(< x_2, \ldots, x_n >)$
- $g(x) : x_0$
- $h(x) = \begin{cases} \mathrm{Truncate}(\tau(\underline{x}, n)) & \text{if } \mathrm{len}(\tau(\underline{x}, n)) > x_1 \\ \tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Omezme polynomem $q = | < x_2, \ldots, x_n > |$.

□

Dále odvoďme cut zprava tak, že použijeme reverzi, aby první člen posloupnosti byl vlevo a poslední vpravo. Pak opět používáme funkci Truncate, dokud není posloupnost chtěné délky.

Lemma 8.8. *Zkrácení posloupnosti zprava na délku* x_1 $\mathrm{rcut}(x_1, < x_2, \ldots, x_n >)$ *je v množině základních funkcí s rekurzí*

Důkaz.

- $\underline{x} = \{< x_2, \ldots, x_n >, x_1\}$
- $y = \mathrm{asl}(< x_2, \ldots, x_n >)$
- $g(x) : rev(< x_2, \ldots, x_n >)$
- $h(x) = \begin{cases} \mathrm{Truncate}(\tau(\underline{x}, n)) & \text{if } \mathrm{len}(\tau(\underline{x}, n)) > x_1 \\ \tau(\underline{x}, n) & \text{if Jinak} \end{cases}$

Omezme polynomem $q = | < x_2, \ldots, x_n > |$.

Odvodili jsme funkci f, která nám dává reverzi zkrácení posloupnosti zprava. Odvoďme nyní funkci f', která nám dává požadovaný výsledek:

$$f'(x_1, < x_2, \ldots, x_n >) : \mathrm{rev}(f(x_1, < x_2, \ldots, x_n >))$$

. □

Nyní můžeme konečně odvodit funkci pro přístup k číslu z posloupnosti. To uděláme tak, že odsekneme posloupnost zleva a zprava. Nakonec použijeme reverzi.

Věta 8.9. *Vybrání čísla* x_1*-tého(číslováno od 0) z posloupnosti je v množině základních funkcí s rekurzí*

Důkaz.

$$\beta(x_1, < x_2, \ldots, x_n >) = \mathrm{rev}(\mathrm{rcut}(1, \mathrm{lcut}(x_1, < x_2, \ldots, x_n >)))$$

□

Věta 8.10. *Důkaz množina polynomiálních funkcí = P-time*
Důkaz je příliš technický a nemyslím, že by do práce přinesl něco víc, než jen opsání důkazu. Nicméně pro zvědavé čtenáře je k dispozici zde [1].

9 Závěrem

Na závěr této práce bych rád vše shrnul a nastínil její přínos a poukázal na možnost pokračování.

Tato bakalářská práce měla za úkol definovat bezestrojově množinu polynomiálních funkcí, podobně jako v Cobhamově práci[2]. To se povedlo a dokonce považuji nějaké části své práce za poměrně inovativní. Například jsem oproti zdroji[1], ze které jsem primárně vycházel, vynechal v základních definicích možnost tvorby posloupností a funkce pro tvorbu a práci s posloupnostmi jsem si vlastnoručně odvodil. Dále jsem dal čtenářům mnohem více volnosti v rekurzivních funkcích a to tak, že jsem definoval 2 různé druhy rekurze a ukázal jejich ekvivalenci. Co se týče jednotlivých funkcí, tak jsem se snažil poctivě odvodit všechny základní funkce, jež se používají v matematice a programování. Dále jsem odvodil i pár funkcí, které nepatří mezi nejpoužívanější funkce, nicméně jsou z mého hlediska zajímavé. Obecně bych svojí snahu hodnotil úspěšně, nicméně je zde mnoho prostoru, kde by bylo možné na tuto práci navázat. To by šlo například způsobem, který praktikoval Buss v [1], když dále vytvořil vlastní aritmetickou teorii se kterou dále pracoval. Nicméně nepochybuji o tom, že by se takováto definice polynomiálních funkcí mohla použít i v základě jiných matematických teorií.

REFERENCE

[1] Samuel R. Buss. Naples, Italy, 1986. University of California, Berkeley.

[2] A. Cobham. The Intrinsic Computational Difficulty of Functions, in: Proc. Logic, Methodology, and Philosophy of Science II, Y. Bar-Hillel ed., pp. 24-30. North Holland, 1965.

[3] P. Mach. Bezestrojová charakterizace funkcí počitatelných v polynomiálním čase. Předběžná verze nefinalizované ročníkové práce. Katedra logiky FF UK, 1998.

[4] ŠVEJDAR, Vítězslav. Logika: neúplnost, složitost a nutnost. Praha: Academia, 2002. ISBN 80-200-1005-X.

[5] S. Bellantoni, S. Cook. A New Recursion-Theoretic Characterization of the Polytime Functions. Computational Complexity 2:297-110, 1992.

[6] HOLUB, J. Skripta pro AAG. Praha, 2016. Dostupné také z: https://edux.fit.cvut.cz/courses/BI-AAG