



ZADÁNÍ BAKALÁ SKÉ PRÁCE

Název:	Implementace násobení na neasociativních (nekomutativních) algebrách
Student:	Matyáš Hollmann
Vedoucí:	RNDr. Ji ina Scholtzová, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2018/19

Pokyny pro vypracování

Algebra je vektorový prostor s p idanou operací vektorového násobení. Násobení v neasociativní algeb e nemusí spl ovat asociativitu. V neasociativní matematice je mnoho problému nevy ešených, již jen proto, že výpo ty jsou pracné a nelze použít dostupných nástroj , jako Mathematica atp. Proto by se hodila aplikace, která by t chto výpo t byla schopná. Navrhn te a implementujte takovou aplikaci pro jednu konkrétní algebru - alternativní nil superalgebru s jedním lichým generátorem.

1. Seznamte se s neasociativními algebami a s lánky souvisejícími s tímto konkrétním zadáním, viz seznam literatury.
2. Seznamte se s aplikacemi, které pracují s neasociativními algebami.
3. Navrhn te vhodný algoritmický postup pro násobení prvk alternativní nil superalgebry s jedním lichým generátorem, je-li daná báze a tabulka násobení na prvcích báze.
4. Vytvo te a otestujte aplikaci, která spo ítá sou in dvou prvk ze superalgebry a vyjád í ho jako lineární kombinaci prvk její báze.

Seznam odborné literatury

- [1] J. Scholtzová, N. Zhukavets: The free alternative nil-superalgebra of index 3 on one odd generator, Journal of Algebra and Its Applications, Vol. 9, No. 2 (2010), 209-?222.
- [2] J. Scholtzová, N. Zhukavets: The free alternative nil-superalgebra of index n on one odd generator, Communications in Algebra, Vol. 42, No. 1 (2014), 96-107.
- [3] I. Shestakov, N. Zhukavets: The free alternative superalgebra on one odd generator, Internat. J. Algebra Comp., Vol. 17 , No. 5/6 (2007), 1215-1247.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
d kan

V Praze dne 8. b ezna 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
KATEDRA TEORETICKÉ INFORMATIKY



Bakalářská práce

Implementace násobení na neasociativních (nekomutativních) algebrách

Matyáš Hollmann

Vedoucí práce: RNDr. Jiřina Scholtzová, Ph.D.

16. května 2017

Poděkování

V první řadě bych chtěl poděkovat vedoucí mé bakalářské práce, RNDr. Jiřině Scholtzové, Ph.D., za rady a pomoc v průběhu tvorby této práce, nemohl jsem si přát lepší vedoucí. Dále bych chtěl poděkovat své rodině za finanční a emocionální podporu v průběhu celého studia na FIT ČVUT.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2017

.....

České vysoké učení technické v Praze
Fakulta informačních technologií

© 2017 Matyáš Hollmann. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Hollmann, Matyáš. *Implementace násobení na neasociativních (nekomutativních) algebrách*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce uvede čtenáře do problematiky neasociativních algeber a superalgeber, což jsou matematické struktury (množiny s nějakými operacemi na nich). Představí také koncept volných (super)algeber, které definují identity a superidentity variet. Poté se soustředí na jednu volnou alternativní nil_n superalgebru s jedním lichým generátorem, pro kterou byla v této práci vyvinuta interaktivní aplikace pro počítání s prvky báze této superalgebry. Aplikace také umí v této superalgebře vyhodnotit nil_n superidentitu pro pevně stanovené n . Aplikace podporuje výstup ve formátu *Wolfram Mathematica* a pro pohodlí uživatele i výstup ve formátu *LaTeX*.

Klíčová slova alternativní nil_n superalgebra, superidentita, generátor, neasociativní, algebra, *Wolfram Mathematica* *WSTP*

Abstract

This bachelor's thesis introduces the reader to the subject of nonassociative algebras and superalgebras, which are mathematical structures (sets equipped with some operations on them). It also presents to the reader the concept of free (super)algebras that define identities and superidentities of varieties. The second part of this thesis focuses on one alternative nil_n superalgebra on one

odd generator for which an interactive computer program has been developed. This program can evaluate any arithmetic expression in said superalgebra. For a fixed n it can also evaluate the nil_n superidentity in this superalgebra. The program supports output in the *Wolfram Mathematica* format and for greater user convenience even in *LaTeX*.

Keywords alternative nil_n superalgebra, superidentity, generator, nonassociative, algebra, *Wolfram Mathematica WSTP*

Obsah

Seznam zdrojových kódů	xvii
Úvod	1
1 Matematický úvod do problematiky	3
1.1 Úvod do lineární a obecné algebry	3
1.2 Algebry nad tělesem	7
1.3 Podalgebry a homomorfismy algeber	10
1.4 Polynomy, variety algeber a volné algebry	12
1.5 Superalgebry	15
1.6 Alternativní $\mathcal{N}il_n$ -superalgebra s jedním lichým generátorem .	18
2 Analýza a návrh	23
2.1 Rešerše stávajících řešení	23
2.2 Vlastní návrh	25
3 Implementace vlastního návrhu	35
3.1 Spojení programu s kernelem <i>Wolframu Mathematica</i> pomocí <i>WSTP</i>	35
3.2 Popis tříd programu a konfiguračního souboru	38
3.3 Instalace programu	42
3.4 Spuštění a ovládání programu	43
4 Testování programu a získané výsledky	45
4.1 Testování programu	45
4.2 Nalezené chyby v publikovaných článcích	46
Závěr	49
A Uživatelská příručka	51

A.1 Instalace	51
A.2 Ovládání programu	52
A.3 Příklady použití	53
B Seznam použitých zkratk	55
C Obsah přiloženého CD	57
Literatura	59

Seznam obrázků

1.1	Ilustrace univerzální vlastnosti volné algebry	15
3.1	UML diagram tříd programu	41

Seznam tabulek

1.1	Tabulka násobení imaginárních jednotek oktonionů	10
2.1	Tokeny rozpoznávané lexikálním analyzátozem	31
2.2	Příklady zápisu násobků prvků báze v navrženém formátu	32

Seznam zdrojových kódů

- 3.1 Ukázka inicializace spojení s kernelem *Wolframu Mathematica*
v jazyce *C++* 36
- 3.2 Ukázka volání funkcí *Wolframu Mathematica* přes *WSTP* v ja-
zyce *C++* 37

Úvod

Od poloviny minulého století se matematici zabývají neasociativními matematickými strukturami (například algebry). Algebra je vektorový prostor s přidanou operací násobení vektorů. Jedním ze stále nevyřešených problémů v teorii neasociativních algeber je konstrukce báze volné alternativní algebry s n generátory. Zatím se podařilo popsat pouze její podprostory [1]. Cílem této práce je vytvořit aplikaci pro počítání s prvky alternativní nil superalgebry s jedním lichým generátorem, která je popsána v sekci 1.6. Program bude umět vyhodnotit libovolný (v praxi je délka výrazu limitována výpočetními možnostmi stroje, na kterém tento program spouštíme) aritmetický výraz složený z prvků dané superalgebry a operace vektorového součinu prvků této superalgebry. Program bude dále schopný pro pevně zvolené n vyhodnotit nil_n superidentitu, uvedenou v sekci 1.5. Aplikace bude „terminálová“ s jednoduchým uživatelským rozhraním, kde si uživatel bude moci zvolit formát výstupu (*LaTeX* nebo *Wolfram Mathematica*). Po vytvoření a otestování správnosti vyvinutého programu bude použit pro ověření výpočtů, vedoucích ke konstrukci báze alternativní nil_n superalgebry s jedním lichým generátorem, publikovaných v článcích [2], [3] a [1].

Tato bakalářská práce je rozdělena do 4 různě obsáhlých kapitol. Čtenář bude nejdříve seznámen s matematickými pojmy týkajícími se číselných těles, algeber a identit, superalgeber a superidentit, volných algeber a volných superalgeber. Na konci první kapitoly bude popsána alternativní nil_n superalgebra s jedním lichým generátorem. V druhé kapitole poté seznámíme čtenáře s návrhem naší aplikace pro práci s prvky této superalgebry. Třetí kapitola se bude zabývat implementací tohoto návrhu, kde bude mimo jiné podrobně popsáno vytvoření spojení s kernelem *Wolfram Mathematica* pomocí *WSTP*. Tato část může sloužit jako návod pro ostatní vývojáře, kteří potřebují využívat funkcionalitu *Wolfram Mathematica* v nějakém svém externím programu. Poslední kapitola se bude týkat testování vyvinuté aplikace a použití této aplikace k ověření výpočtů zveřejněných v článcích [2], [3] a [1].

Matematický úvod do problematiky

V této kapitole definujeme základní matematické pojmy použité v této práci. Čerpali jsme především z přednášek [4] a skript [5] k předmětu BI-LIN, handoutů k předmětu MI-MPI [6], disertační práce naší vedoucí [1] a článků [2], [3]. Ostatní zdroje jsou uvedeny explicitně v místě, kde byly použity.

1.1 Úvod do lineární a obecné algebry

Uspořádanou dvojici (M, \circ) , kde M je libovolná neprázdná množina a binární operace $\circ : M \times M \rightarrow M$, nazýváme **grupou**, pokud pro operaci \circ platí:

- $\forall x, y, z \in M : x \circ (y \circ z) = (x \circ y) \circ z$, (asociativní zákon)
- $\exists e \in M, \forall x \in M : e \circ x = x \circ e = x$, (existence neutrálního prvku)
- $\forall x \in M, \exists x^{-1} \in M : x \circ x^{-1} = x^{-1} \circ x = e$. (existence inverzního prvku)

Pokud navíc platí:

- $\forall x, y \in M : x \circ y = y \circ x$, (komutativní zákon)

nazýváme (M, \circ) **komutativní** (Abelovskou) **grupou**.

Uspořádanou trojici $(M, +, \cdot)$, kde M je libovolná neprázdná množina a dvě binární operace $+$: $M \times M \rightarrow M$, \cdot : $M \times M \rightarrow M$, nazýváme **okruhem**, pokud platí:

- $(M, +)$ je komutativní (Abelovská) grupa,
- $\forall x, y \in M : x \cdot y \in M$, (uzavřenost množiny M vůči operaci \cdot)
- $\forall x, y, z \in M : x \cdot (y \cdot z) = (x \cdot y) \cdot z$. (asociativní zákon)

- Platí distributivita násobení vůči sčítání:
 $\forall a, b, c \in M : a \cdot (b + c) = a \cdot b + a \cdot c,$ (levý distributivní zákon)
 $\forall a, b, c \in M : (b + c) \cdot a = b \cdot a + c \cdot a.$ (pravý distributivní zákon)

Dále okruh $(M, +, \cdot)$, kde navíc platí:

- $\exists e \in M, \forall x \in M : e \cdot x = x \cdot e = x,$ (existence neutrálního prvku)
- $\forall x \in M \setminus \{0\}, \exists x^{-1} \in M : x \cdot x^{-1} = x^{-1} \cdot x = e,$ (existence inverze)

nazýváme **tělesem** a značíme ho \mathbb{T} . Uspořádanou dvojici $(M, +)$ nazýváme **aditivní grupou** tělesa \mathbb{T} a neutrální prvek této grupy značíme symbolem $\mathbf{0}$. Uspořádaná dvojice (M, \cdot) je díky splnění výše uvedených podmínek grupou a nazýváme ji **multiplikativní grupou** tělesa \mathbb{T} , neutrální prvek této grupy značíme symbolem $\mathbf{1}$ a nazýváme **jednotkou**.

Pokud lze opakovaným sčítáním jednotky multiplikativní grupy tělesa \mathbb{T} dosáhnout neutrálního prvku aditivní grupy tohoto tělesa, nazýváme nejmenší takové číslo **charakteristikou** tělesa \mathbb{T} , viz [7]. Pokud takové číslo neexistuje, říkáme, že těleso \mathbb{T} je charakteristiky 0. Dále v textu budeme uvažovat pouze tělesa charakteristiky 0. Jako příklady charakteristik některých základních těles uveďme konečná tělesa \mathbb{Z}_p , která mají charakteristiku p a reálná čísla \mathbb{R} , která jsou nekonečným tělesem a mají charakteristiku 0.

Mějme okruh $(O, +, \cdot)$ a $I \subseteq O$ jeho neprázdnou podmnožinu, pak I nazýváme (oboustranným) **ideálem** okruhu O , pokud platí:

- $\forall x \in I, \forall o \in O : x \cdot o \in I,$ (pravý ideál)
- $\forall x \in I, \forall o \in O : o \cdot x \in I,$ (levý ideál)
- $\forall a, b \in I : a - b \in I.$ (uzavřenost vůči sčítání)

Dále platí, že průnik libovolného neprázdného systému ideálů v okruhu O je opět ideál v okruhu O , viz [8]. Nechť $(O, +, \cdot)$ je okruh a M je libovolná podmnožina množiny O , potom průnik všech ideálů v O , které obsahují množinu M , je ideál v O , který nazýváme **ideálem generovaným množinou** M a značíme ho $\langle M \rangle$. Množinu M nazýváme **systémem generátorů ideálu** $\langle M \rangle$ a její prvky **generátory** tohoto ideálu.

Mějme libovolné těleso \mathbb{T} s neutrálním prvkem $\mathbf{0}$ a jednotkou $\mathbf{1}$. Prvky tělesa \mathbb{T} nazýváme **skaláry** a uspořádanou n -tici $v = (x_1, x_2, \dots, x_n)$, kde $x_i \in \mathbb{T}$, nazýváme **vektorem**. Na množině vektorů V definujme dvě binární operace, sčítání vektorů $\oplus : V \times V \rightarrow V$ a násobení vektoru skalárem $\odot : \mathbb{T} \times V \rightarrow V$, pro které platí:

- $\forall v = (x_1, x_2, \dots, x_n) \in V, \forall \alpha \in \mathbb{T} : \alpha \odot v = (\alpha x_1, \alpha x_2, \dots, \alpha x_n)$,
(násobení vektoru skalárem)
- $\forall u, v \in V : u \oplus v = v \oplus u$, (komutativita sčítání vektorů)
- $\forall u, v, w \in V : (u \oplus v) \oplus w = u \oplus (v \oplus w)$, (asociativita sčítání vektorů)
- $\forall \alpha, \beta \in \mathbb{T}, \forall v \in V : \alpha \odot (\beta \odot v) = (\alpha\beta) \odot v$,
(asociativita pro násobení skalárem)
- $\forall \alpha \in \mathbb{T}, \forall u, v \in V : \alpha \odot (u \oplus v) = (\alpha \odot u) \oplus (\alpha \odot v)$,
(distributivita násobení skalárem vůči sčítání vektorů)
- $\forall \alpha, \beta \in \mathbb{T}, \forall v \in V : (\alpha + \beta) \odot v = (\alpha \odot v) \oplus (\beta \odot v)$,
(distributivita násobení skalárem vůči sčítání skalárů z tělesa \mathbb{T})
- $\forall v \in V : 1 \odot v = v \odot 1 = v$, (násobení vektoru jednotkou tělesa \mathbb{T})
- $\exists \theta \in V, \forall v \in V : 0 \odot v = \theta$, (existence nulového vektoru)

pak V nazýváme **vektorovým prostorem nad tělesem \mathbb{T}** . Vektor θ z poslední vlastnosti nazýváme **nulovým vektorem** a platí:

$$\forall v \in V : \theta \oplus v = v \oplus \theta = v.$$

Neprázdňnou podmnožinu P vektorového prostoru V nazveme **vektorovým podprostorem**, pokud platí:

- $\forall x, y \in P : x \oplus y \in P$,
- $\forall \alpha \in \mathbb{T}, \forall x \in P : \alpha \odot x \in P$.

Požadujeme tedy, aby množina P byla podmnožinou množiny V a aby byla uzavřená vůči oběma operacím vektorového prostoru V . Tento vztah pak značíme $P \subset V$.

Mějme vektory v, v_1, \dots, v_n z vektorového prostoru V . Říkáme, že vektor v je **lineární kombinací** vektorů v_1, \dots, v_n , právě když existují skaláry $\alpha_1, \dots, \alpha_n \in \mathbb{T}$ takové, že $v = \sum_{i=1}^n \alpha_i v_i$. Skaláry $\alpha_1, \dots, \alpha_n$ nazýváme **koefficienty lineární kombinace**. Pokud jsou všechny koefficienty lineární kombinace nulové, nazýváme tuto lineární kombinaci **triviální**. V opačném případě jde o lineární kombinaci **netriviální**.

Nechť (v_1, \dots, v_n) je soubor vektorů z V . Pokud je pouze triviální kombinace tohoto souboru rovna nulovému vektoru θ , je tento soubor vektorů **lineárně**

nezávislý (LN). V opačném případě je tento soubor vektorů **lineárně závislý** (LZ).

Nechť (v_1, \dots, v_n) je soubor vektorů z V . Množinu všech lineárních kombinací tohoto souboru nazýváme **lineárním obalem souboru** (v_1, \dots, v_n) a značíme ji $\langle v_1, \dots, v_n \rangle$.

O lineárně nezávislém souboru vektorů B z vektorového prostoru V řekneme, že je **bází** vektorového prostoru V , pokud generuje celý vektorový prostor V , tj. $\langle B \rangle = V$.

Nechť M a N jsou dvě libovolné množiny. **Zobrazením** $f : M \rightarrow N$ z množiny M do množiny N je jakýkoliv předpis, který každému prvku z množiny M přiřadí nejvýše jeden prvek z množiny N .

Nechť jsou P a Q dva vektorové prostory nad stejným tělesem \mathbb{T} a zobrazení $f : P \rightarrow Q$. Zobrazení f nazveme **lineární** právě pokud platí:

- $\forall x, y \in P : f(x \oplus_P y) = f(x) \oplus_Q f(y)$, (aditivita)
- $\forall \alpha \in \mathbb{T}, \forall x \in P : f(\alpha \odot_P x) = \alpha \odot_Q f(x)$, (homogenita)

kde operace $\{\oplus_P, \odot_P\}$ jsou z vektorového prostoru P a operace $\{\oplus_Q, \odot_Q\}$ jsou z vektorového prostoru Q .

Nechť jsou P, R a Q vektorové prostory nad stejným tělesem \mathbb{T} a zobrazení $f : P \times R \rightarrow Q$. Zobrazení f nazveme **bilineární**, pokud je lineární v obou svých parametrech, tj.:

- pro libovolné pevně dané $p \in P, \forall \alpha \in \mathbb{T}, \forall x, y \in R :$

$$\begin{aligned} f(p, \alpha \odot_R x) &= \alpha \odot_Q f(p, x), \\ f(p, x \oplus_R y) &= f(p, x) \oplus_Q f(p, y), \end{aligned}$$

- pro libovolné pevně dané $r \in R, \forall \alpha \in \mathbb{T}, \forall x, y \in P :$

$$\begin{aligned} f(\alpha \odot_P x, r) &= \alpha \odot_Q f(x, r), \\ f(x \oplus_P y, r) &= f(x, r) \oplus_Q f(y, r), \end{aligned}$$

kde operace $\{\oplus_P, \odot_P\}$ jsou z vektorového prostoru P , operace $\{\oplus_Q, \odot_Q\}$ jsou z vektorového prostoru Q a operace $\{\oplus_R, \odot_R\}$ jsou z vektorového prostoru R .

1.2 Algebry nad tělesem

Algebra A je vektorový prostor nad tělesem \mathbb{T} , který má definované bilineární zobrazení $\cdot : A \times A \rightarrow A$, s vlastností distributivity, tj. $\forall \alpha, \beta \in \mathbb{T}, \forall u, v, w \in A$, platí:

- $u \cdot (\alpha \odot v \oplus \beta \odot w) = \alpha \odot (u \cdot v) \oplus \beta \odot (u \cdot w)$,
- $(\alpha \odot u \oplus \beta \odot v) \cdot w = \alpha \odot (u \cdot w) \oplus \beta \odot (v \cdot w)$.

Toto bilineární zobrazení nazýváme (vektorovým) **součinem**.

Poznámka k notaci: pokud bude z kontextu zřejmé, zda-li mluvíme o sčítání vektorů, nebo o sčítání skalárů, budeme v dalším textu pro tuto operaci jednotně používat symbol $+$. Pro násobení dvou vektorů, násobení dvou skalárů a násobení vektoru skalárem budeme jednotně používat symbol \cdot . Pokud to bude z kontextu zřejmé, pro násobení dvou vektorů vynecháme tento symbol úplně. Například pro vektory x, y bude od této chvíle zápis xy značit jejich vektorový součin $x \cdot y$.

Asociátor algebry A je trilineární zobrazení $(\cdot, \cdot, \cdot) : A \times A \times A \rightarrow A$, které je definováno vztahem $(x, y, z) = (xy)z - x(yz)$, pro $x, y, z \in A$.

Jestliže pro všechny trojice prvků x, y, z algebry A platí, že $(x, y, z) = 0$, nazýváme tuto algebru **asociativní**. Není-li algebra A asociativní, říkáme, že je **neasociativní**. Definujeme také slabší podmínku, než je asociativita, tzv. alternativitu:

- $(x, x, y) = 0, \quad x, y \in A, \quad$ **(levá alternativita)**
- $(x, y, y) = 0, \quad x, y \in A. \quad$ **(pravá alternativita)**

Jestliže pro všechny dvojice prvků x, y algebry A platí levá a pravá alternativita, nazýváme tuto algebru **alternativní**.

Komutátor algebry A je bilineární zobrazení $[\cdot, \cdot] : A \times A \rightarrow A$, které je definováno vztahem $[x, y] = xy - yx$, pro $x, y \in A$.

Jestliže pro všechny dvojice prvků x, y algebry A platí: $[x, y] = 0$, nazýváme tuto algebru **komutativní**. Není-li algebra A komutativní, říkáme, že je **nekomutativní**.

O algebře řekneme, že je **nil-algebrou** indexu n , jestliže pro každý prvek $x \in A$ platí $x^n = 0$ pro nějaké přirozené číslo n . V případě neasociativních algeber toto musí platit pro součet všech možných uzávorkování výrazu x^n .

Jordanův součin algebry A je bilineární zobrazení $A \times A \rightarrow A$, které je definováno vztahem $x \circ y = xy + yx$, pro $x, y \in A$. Jordanův součin charakterizuje tzv. Jordanovy algebry, kterými se ale v této práci nebudeme zabývat.

Výše zmíněné vlastnosti algeber budeme potřebovat později v sekci 1.6.

1.2.1 Příklady algeber

Většina algeber, se kterými jsme se setkali v předmětu BI-LIN, byly asociativní a komutativní. Nekomutativní algebrou je pak například algebra čtvercových matic $n \times n$ s operací násobení matic, kde matice $B \cdot C$ nemusí být stejná jako matice $C \cdot B$. To si ukážeme na matici 2×2 nad tělesem reálných čísel.

$$\begin{aligned} B &= \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \\ C &= \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \\ [B, C] &= BC - CB = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix} - \begin{pmatrix} 23 & 34 \\ 31 & 46 \end{pmatrix} \\ &= \begin{pmatrix} -4 & -12 \\ 12 & -4 \end{pmatrix} \neq \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{aligned}$$

A tedy algebra matic 2×2 není komutativní.

Dalším příkladem algebry mohou být **kvaterniony**, které byly prvně popsány Williamem Rowanem Hamiltonem v roce 1843. Hamilton se nejdříve věnoval komplexním číslům, které reprezentoval jako uspořádanou dvojici reálných čísel a definoval pro ně operace sčítání a násobení. Poté se snažil rozšířit komplexní čísla přidáním druhého imaginárního prvku a pomocí těchto trojic chtěl reprezentovat rotace v prostoru, obdobně jako se komplexní čísla dají použít pro reprezentaci rotací v rovině. Největším problémem bylo stanovit, čemu se bude rovnat součin dvou různých imaginárních prvků. Lze ukázat, že tento problém nemá řešení a je nutné přidat ještě třetí imaginární prvek, viz [9]. Název kvaterniony pochází z anglického slova pro čtveřici. O historii kvaternionů jsem čerpal ze zdroje [9]. Kvaterniony jsou rozšířením tělesa komplexních čísel, ke kterému se přidají další dvě imaginární jednotky j, k . Obecný tvar kvaternionu je tedy

$$a + bi + cj + dk,$$

kde $a, b, c, d \in \mathbb{R}$ a symboly i, j, k jsou imaginární jednotky a pro jejich násobení platí následující pravidla:

- $i^2 = j^2 = k^2 = -1$,
- $ij = k, \quad ji = -k$,

- $jk = i, \quad kj = -i,$
- $ki = j, \quad ik = -j.$

Přidáme-li ještě pravidla:

- $\forall a \in \mathbb{R} : ai = ia, \quad aj = ja, \quad ak = ka,$

máme algebru kvaternionů nad tělesem \mathbb{R} . Z pravidel pro násobení imaginárních jednotek je zřejmé, že algebra kvaternionů je nekomutativní. Kvaterniony se značí \mathbb{H} a jejich bází je množina $\{1, i, j, k\}$.

Krátce po popsání kvaternionů Hamiltonem přišel John Thomas Graves v prosinci roku 1843 s návrhem rozšířit kvaterniony na novou matematickou strukturu s jedním reálným prvkem a sedmi imaginárními prvky. Tuto algebru nazýval oktávy. Tento svůj nápad však nezveřejnil ihned a nakonec ho předběhl jeho současník Arthur Cayley, který v roce 1845 zveřejnil článek pojednávající o **oktonionech**. Oktoniony se pak po něm někdy také nazývají Cayleyho čísla. O historii oktonionů jsem čerpal ze zdroje [10]. Oktoniony jsou rozšířením kvaternionů a mají sedm imaginárních jednotek značených $i_0, i_1, i_2, i_3, i_4, i_5, i_6$. Obecný oktonion se dá zapsat ve tvaru:

$$a + bi_0 + ci_1 + di_2 + ei_3 + fi_4 + gi_5 + hi_6,$$

kde $a, b, c, d, e, f, g, h \in \mathbb{R}$. Následující uspořádané trojice imaginárních jednotek

$$i_0, i_1, i_3,$$

$$i_1, i_2, i_4,$$

$$i_2, i_3, i_5,$$

$$i_3, i_4, i_6,$$

$$i_4, i_5, i_0,$$

$$i_5, i_6, i_1,$$

$$i_6, i_0, i_2,$$

se chovají jako uspořádaná trojice imaginárních jednotek kvaternionů i, j, k , viz [11]. Pro snazší orientaci při výpočtech se také často používá **tabulka násobení**. Následující tabulka násobení 1.1 udává na pozici (k, l) výsledek násobení prvku v k -tém řádku prvkem v l -tém sloupci.

Tabulka 1.1: Tabulka násobení imaginárních jednotek oktonionů

\cdot	i_0	i_1	i_2	i_3	i_4	i_5	i_6
i_0	-1	i_3	i_6	$-i_1$	i_5	$-i_4$	$-i_2$
i_1	$-i_3$	-1	i_4	i_0	$-i_2$	i_6	$-i_5$
i_2	$-i_6$	$-i_4$	-1	i_5	i_1	$-i_3$	i_0
i_3	i_1	$-i_0$	$-i_5$	-1	i_6	i_2	$-i_4$
i_4	$-i_5$	i_2	$-i_1$	$-i_6$	-1	i_0	i_3
i_5	i_4	$-i_6$	i_3	$-i_2$	$-i_0$	-1	i_1
i_6	i_2	i_5	$-i_0$	i_4	$-i_3$	$-i_1$	-1

Oktoniony se značí \mathbb{O} . Násobení oktonionů není komutativní ani asociativní, je pouze alternativní. Je tedy prvním příkladem alternativní algebry, která je zároveň neasociativní a nekomutativní algebrou.

$$\begin{aligned} [i_0, i_1] &= i_0 i_1 - i_1 i_0 = i_3 - (-i_3) \\ &= 2i_3 \neq 0. \end{aligned} \quad (\text{nekomutativita})$$

$$\begin{aligned} (i_0, i_5, i_2) &= (i_0 i_5) i_2 - i_0 (i_5 i_2) = -i_4 i_2 - i_0 i_3 = i_1 + i_1 \\ &= 2i_1 \neq 0. \end{aligned} \quad (\text{neasociativita})$$

1.3 Podalgebry a homomorfismy algeber

Tato sekce pojednává o podalgebrách, homomorfismech algeber a jejich vlastnostech ve vztahu k ideálům. Informace uvedené v této sekci byly převzaty ze zdroje [1].

Podalgebra B algebry A je vektorový podprostor uzavřený vůči vektorovému součinu: $BB \subset B$ (pro všechny prvky $x, y \in B$ platí, že jejich součin xy také patří do B). (Oboustranný) **ideál** I algebry A je podalgebra uzavřená vůči násobení prvky $z \in A$, tj.:

$$AI + IA \subseteq I.$$

Ideály 0 a A algebry A nazýváme **triviálními** ideály.

V teorii neasociativních algeber dále definujeme dvě speciální podmnožiny algebry A :

Jádro $N(A)$ algebry A (někdy se také nazývá asociativní centrum) je množina všech prvků $z \in A$, které se chovají asociativně vůči libovolné dvojici prvků $a, b \in A$ ve smyslu, že $(z, a, b) = (a, z, b) = (a, b, z) = 0$, tj.:

$$N(A) = \{z \in A \mid (z, A, A) = (A, z, A) = (A, A, z) = 0\},$$

kde (\cdot, \cdot, \cdot) je asociátor algebry A .

Centrum $Z(A)$ algebry A je množina všech prvků $z \in A$, které se chovají komutativně a asociativně vůči všem prvkům algebry A , tj.:

$$Z(A) = \{z \in N(A) \mid [z, A] = 0\},$$

kde $[\cdot, \cdot]$ je komutátor algebry A . Poznamenejme, že $N(A)$ je asociativní podalgebra A a $Z(A)$ je komutativní a asociativní podalgebra A . Také platí, že $Z(A) \subseteq N(A)$.

Homomorfismus algeber $\varphi : A \rightarrow A'$ je lineárním zobrazením, které zachovává strukturu algebry (vektorový součin). Tj. splňuje navíc rovnost:

$$\varphi(xy) = \varphi(x)\varphi(y), \quad \forall x, y \in A.$$

Množina

$$\text{Im } \varphi = \{\varphi(a) \mid a \in A\}$$

je homomorfním **obrazem** algebry A a **jádro** homomorfismu φ je množina:

$$\text{Ker } \varphi = \{a \in A \mid \varphi(a) = 0\}.$$

Pokud je φ prostým homomorfismem, říkáme, že algebra A je **vnořená** v A' . Bijektivní homomorfismus algeber nazýváme **izomorfismem algeber**.

Nechť I je ideál algebry A , potom zobrazení $A \rightarrow A/I$ takové, že $a \mapsto a + I$, nazýváme **přirozeným** (někdy také kanonickým) homomorfismem algeber.

Nechť A, A' jsou dvě libovolné algebry. Nechť I je ideálem algebry A , $\varphi : A \rightarrow A'$ homomorfismem algeber a $\pi : A \rightarrow A/I$ přirozeným homomorfismem algeber. Poté existuje jednoznačně určený homomorfismus $\varphi' : A/I \rightarrow A'$ takový, že $\varphi'(a + I) = \varphi(a)$. Pokud je homomorfismus φ na (surjektivní) a jeho jádro je ideálem I , tj. $\text{Ker } \varphi = I$, poté je zobrazení φ' navíc i izomorfismem algeber.

Nechť $\varphi : A \rightarrow A'$ je homomorfismem algeber, $B \subseteq A$ podalgebrou algebry A a $I_2 \subset I_1$ jsou ideály algebry A , poté platí:

- $\text{Ker } \varphi$ je ideálem A , $\text{Im } \varphi$ je podalgebrou A' a

$$A/\text{Ker } \varphi \simeq \text{Im } \varphi \text{ (množiny jsou izomorfní),}$$

- $I_1 \cap B$ je ideálem B , $I_1 + B$ je podalgebrou A , I_1 je ideálem $I_1 + B$ a

$$(I_1 + B)/I_1 \simeq B/(I_1 \cap B) \text{ (množiny jsou izomorfní),}$$

- pro dva ideály $I_2 \subset I_1$ algebry A , I_1/I_2 je ideál algebry A/I_2 a

$$(A/I_2)/(I_1/I_2) \simeq A/I_1 \text{ (množiny jsou izomorfní).}$$

1.4 Polynomy, variety algeber a volné algebry

Výraz $a_n x^n + \dots + a_2 x^2 + a_1 x + a_0$, kde $a_0, a_1, \dots, a_n \in \mathbb{T}$, nazýváme **polynomem jedné proměnné** nad číselným tělesem \mathbb{T} , prvky a_0, a_1, \dots, a_n nazýváme **koeficienty** tohoto polynomu a symbol x nazýváme **proměnnou** tohoto polynomu. Každý člen x^i , kde $i = 0, 1, \dots, n$, nazýváme **monomem**. Množinu všech polynomů v proměnné x nad tělesem \mathbb{T} značíme $\mathbb{T}[x]$.

Nechť polynom $p \in \mathbb{T}[x]$ má koeficienty a_0, a_1, \dots, a_n . **Stupeň** polynomu p je největší index k takový, že $a_k \neq 0$. Pokud jsou všechny koeficienty nulové, stupeň definujeme rovný -1 a polynom p nazýváme **nulovým polynomem**. Stupeň polynomu p značíme $\deg(p) = n$ a říkáme, že polynom p je stupně n .

Definici polynomů jedné proměnné můžeme zobecnit na více proměnných. Mějme množinu proměnných $X = \{x_1, x_2, \dots, x_n\}$, kterou odteď budeme brát jako uspořádanou. Výraz $x_{i_1}^{k_1} \cdot x_{i_2}^{k_2} \cdots x_{i_l}^{k_l}$, kde $i_1, \dots, i_l \in \{1, \dots, n\}$, $k_1, \dots, k_l \in \mathbb{N}_0$, nazýváme obecně **monomem** (nekomutativním) a budeme ho označovat M . U monomů více proměnných definujeme **multistupeň**, který udává kolikrát jsou jednotlivé proměnné x_1, \dots, x_n v tomto monomu zastoupeny. Značíme ho $\text{mdeg}(M) = (m_1, \dots, m_n)$, kde m_i je součet všech mocnin proměnné $x_i \in X$ v daném monomu M .

Například uvažujme následující dva nekomutativní monomy dvou proměnných z množiny proměnných $X = \{x, y\}$. První monom je:

$$p_1(x, y) = x^2 y^5 x^3 y,$$

jehož multistupeň je $\text{mdeg}(p_1) = (5, 6)$. Druhý monom je následující:

$$p_2(x, y) = x^5 y^6,$$

jehož multistupeň je stejný jako multistupeň monomu p_1 , tedy $\text{mdeg}(p_2) = (5, 6)$. Mohlo by se zdát, že druhý monom je jen kompaktněji zapsaný první monom, ale není tomu tak, kvůli nekomutativitě definují p_1 a p_2 dva různé monomy.

Nekomutativní **polynom více proměnných** $p(x_1, \dots, x_n)$ nad tělesem \mathbb{T} je konečná lineární kombinace monomů více proměnných s koeficienty polynomu z tělesa \mathbb{T} , tj.:

$$p(x_1, \dots, x_n) = \sum_i \alpha_i M_i,$$

kde $\alpha_i \in \mathbb{T}$, M_i je nějaký nekomutativní monom proměnných x_1, x_2, \dots, x_n . Množinu všech polynomů v proměnných $X = \{x_1, \dots, x_n\}$ nad tělesem \mathbb{T} značíme $\mathbb{T}[X]$.

Pokud je polynom navíc neasociativní, je potřeba jeho monomy navíc uzavřít, aby byl zápis tohoto polynomu jednoznačný. Například následující dva polynomy dvou proměnných z množiny proměnných $X = \{x, y\}$:

$$\begin{aligned} p_1(x, y) &= 5(x^2)y + 8x(y^2) - (xy)x, \\ p_2(x, y) &= 5x(xy) + 8(xy)y - (xy)x, \end{aligned}$$

jsou různé, i když mají stejné koeficienty a stejné multistupně svých monomů. Multistupeň monomu nebere v potaz případnou neasociativitu daného monomu.

Pokud mají všechny monomy polynomu stejný multistupeň, nazýváme takový polynom **homogenním**. Například následující polynom dvou proměnných z množiny proměnných $X = \{x, y\}$:

$$p(x, y) = x^2y + 8x(yx) + (yx)x,$$

je homogenní, jelikož multistupně všech jeho monomů jsou rovny vektoru $(2, 1)$.

Homogenní polynom n proměnných, který má multistupně všech svých monomů rovny vektoru $(1, 1, \dots, 1)$, nazýváme **multilineárním**. Například následující polynom dvou proměnných z množiny proměnných $X = \{x, y\}$:

$$p(x, y) = xy + yx,$$

je multilineární, jelikož multistupně všech jeho monomů jsou rovny vektoru $(1, 1)$.

Nechť A je algebra nad tělesem \mathbb{T} a polynom $p(x_1, \dots, x_n) \in \mathbb{T}[X]$, pro který platí:

$$\forall a_1, \dots, a_n \in A : p(a_1, \dots, a_n) = 0,$$

poté nazýváme takový polynom **identitou** algebry A a o algebře A říkáme, že splňuje identitu $p(x_1, \dots, x_n)$. Již dříve jsme uvedli například komutativní, asociativní či alternativní identitu.

Množinu neasociativních polynomů můžeme také definovat následujícím induktivním postupem: mějme množinu polynomů I z $\mathbb{T}[X]$. Potom množinu všech algeber splňujících všechny identity z množiny I , nazýváme **varietou** algeber nad tělesem \mathbb{T} , která je definována množinou identit I . Algebry z variety \mathcal{V} nazýváme také \mathcal{V} -algebrami. Jako příklad uvedeme algebru oktonionů, která je alternativní algebrou a patří tedy do variety *Alt*-algeber a ta, jak jsme již dříve uvedli, je definovaná dvěma identitami (levou a pravou alternativitou). Dalším příkladem může být algebra komplexních čísel, která je komutativní algebrou a patří tedy do variety *Com*-algeber.

Uvažujme pevně zvolenou množinu proměnných X . Přidejme do této množiny otevírací a uzavírací závorky a získáme množinu $X^* = X \cup \{(\, , \,)\}$. Induktivně definujme množinu $K[X]$ posloupností prvků z X^* , které nazýváme **neasociativními slovy** prvků množiny X , následujícími pravidly:

$$\begin{aligned}x &\in K[X], & \forall x \in X, \\x_1 x_2 &, & \forall x_1, x_2 \in X, \\x(u) &, & \forall x \in X, \forall u \in K[X], \\(u)x &, & \forall x \in X, \forall u \in K[X], \\(u)(v) &, & \forall u, v \in K[X].\end{aligned}$$

Dále definujeme násobení na této množině $\cdot : K[X] \times K[X] \rightarrow K[X]$:

$$\begin{aligned}x_1 \cdot x_2 &= x_1 x_2, & \forall x_1, x_2 \in X, \\x_1 \cdot u &= x_1(u), & \forall x_1 \in X, \forall u \in K[X], \\u \cdot x_2 &= (u)x_2, & \forall x_2 \in X, \forall u \in K[X], \\u \cdot v &= (u)(v), & \forall u, v \in K[X].\end{aligned}$$

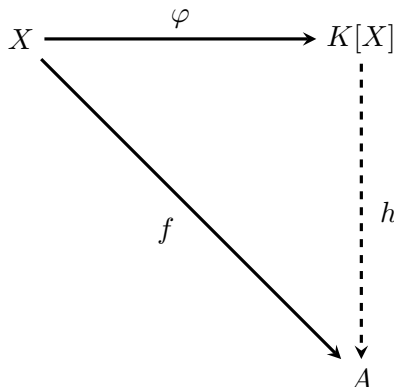
Nechť \mathbb{T} je číselné těleso a $\mathbb{T}[X]$ množina lineárních kombinací prvků z $K[X]$ s koeficienty z \mathbb{T} . Rozšíříme-li operaci násobení definovanou na množině $K[X]$ na množinu $\mathbb{T}[X]$ pomocí pravidla:

$$\left(\sum_i \alpha_i u_i\right) \cdot \left(\sum_j \beta_j v_j\right) = \sum_{ij} \alpha_i \beta_j (u_i \cdot v_j), \quad \alpha_i, \beta_j \in \mathbb{T}, \quad u_i, v_j \in K[X],$$

nazýváme tuto operaci vektorovým součinem na množině $\mathbb{T}[X]$. Množinu $\mathbb{T}[X]$ s operací vektorového součinu na této množině nazýváme **volnou neasociativní algebrou** nad tělesem \mathbb{T} **generovanou** množinou X . Prvky množiny X nazýváme **generátory** této algebry.

Každá volná algebra je definována následující univerzální vlastností: nechť A je libovolná algebra, $K[X]$ volná neasociativní algebra generovaná množinou X a $\varphi : X \rightarrow K[X]$ libovolné zobrazení. Poté pro každé zobrazení $f : X \rightarrow A$ z množiny generátorů X do algebry A platí, že toto zobrazení lze jednoznačně rozšířit na homomorfismus algeber $h : K[X] \rightarrow A$, viz [1]. Znázornění této univerzální vlastnosti lze vidět na obrázku 1.1.

Obrázek 1.1: Ilustrace univerzální vlastnosti volné algebry



Nechť I_{Alt} je ideálem volné algebry $\mathbb{T}[X]$, který je generovaný levými stranami identit alternativity, tj. dvěma asociátory:

$$(f_1, f_1, f_2), (f_1, f_2, f_2), \text{ kde } f_1, f_2 \in \mathbb{T}[X],$$

potom **volnou alternativní** algebrou $Alt[X]$, generovanou množinou proměnných X , je algebra:

$$Alt[X] = \mathbb{T}[X]/I_{Alt}.$$

Univerzální vlastnost volné algebry stále platí, protože pro alternativní algebru A nad tělesem \mathbb{T} a zobrazení $f : X \rightarrow A$, existuje jednoznačně určený homomorfismus algeber $\varphi_{Alt} : Alt[X] \rightarrow A$ takový, že platí $\varphi_{Alt}(x) = f(x)$ pro všechny $x \in X$.

Obdobně definujeme **volnou alternativní nil_n-algebru** $Alt-Nil_n[X]$:

$$Alt-Nil_n[X] = Alt[X]/I_{Nil_n},$$

kde I_{Nil_n} je ideálem volné alternativní algebry $Alt[X]$, který je generovaný prvky:

$$\sum_{\pi \in Sym(n)} (\dots (f_{\pi(1)} f_{\pi(2)}) f_{\pi(3)}) \dots f_{\pi(n)}, \text{ pro } f_1, \dots, f_n \in Alt[X],$$

kde $Sym(n)$ je symetrická grupa (množina všech permutací n -prvkové množiny $\{1, 2, \dots, n\}$ s operací skládání permutací).

1.5 Superalgebry

Algebru A nad tělesem \mathbb{T} , kterou můžeme zapsat jako $A = A_0 + A_1$ (tzv. \mathbb{Z}_2 -graduace), kde platí:

- $\forall x \in A_0, \forall y \in A_0 : x \cdot y \in A_0,$
- $\forall x \in A_0, \forall y \in A_1 : x \cdot y \in A_1,$
- $\forall x \in A_1, \forall y \in A_0 : x \cdot y \in A_1,$
- $\forall x \in A_1, \forall y \in A_1 : x \cdot y \in A_0,$

nazýváme **superalgebrou** nad tělesem \mathbb{T} . A_0 nazýváme **sudou částí** superalgebry A a tato část je podalgebrou superalgebry A . A_1 nazýváme **lichou částí** superalgebry A a není to algebra sama o sobě, protože není uzavřená vůči operaci \cdot (vektorový součin). Dále definujeme **paritu** prvku x superalgebry A jako \bar{x} a pokud x náleží A_0 , nazýváme x **sudým prvkem**, jehož znaménko je $\bar{x} = 0$. Obdobně pokud x náleží A_1 , nazýváme ho **lichým prvkem**, jehož znaménko je $\bar{x} = 1$. Jako příklad superalgebry uveďme algebru komplexních čísel, kterou lze zapsat jako: $\mathbb{C} = \mathbb{R} + \mathbb{R}i$, sudou částí této superalgebry jsou reálná čísla a lichou částí jsou čistě komplexní čísla (reálná složka je nulová).

Definice podalgebry, ideálu algebry, homomorfismu/izomorfismu algeber platí též v superalgebrách, jediná podmínka je zachování \mathbb{Z}_2 gradace, tedy zachování struktury superalgebry, podrobněji uvedeno v [1].

Nechť \mathcal{V} je varieta algeber nad tělesem \mathbb{T} definovaná množinou multilineárních polynomiálních identit I (každý homogenní polynom v tělese charakteristiky 0 lze linearizovat, postup je uveden např. v [1]). Z každé multilineární identity z množiny I lze získat tzv. **superidentitu** podle Kaplanského principu, viz [1]. Tento princip říká, že vždy, když jsou v identitě prohozené dvě liché proměnné (proměnné polynomu si seřadíme podle uspořádání množiny proměnných X), tak se znaménko příslušného monomu změní. Superalgebru splňující všechny superidentitu vzniklé touto transformací z identit z množiny I nazýváme **\mathcal{V} -superalgebrou**.

Uvedeme zde pár příkladů identit algebry A a z nich vytvořených superidentit superalgebry B , tyto superidentitu byly převzaty z [1] a použijeme v následující sekci 1.6.

Superasociativita

- $\forall x, y, z \in A : (xy)z - x(yz) = 0,$ (asociativita)
- $\forall x, y, z \in B : (xy)z - x(yz) = 0.$ (**superasociativita**)

Asociativita a superasociativita jsou totožné, protože zde nebyly prohozeny žádné prvky. Superalgebry splňující superasociativitu nazýváme asociativními superalgebrami a značíme je *Assoc*-superalgebry.

Superkomutativita

- $\forall x, y \in A : xy - yx = 0$, (komutativita)
- $\forall x, y \in B : xy - (-1)^{\overline{xy}}yx = 0$. (**superkomutativita**)

Komutativita a superkomutativita se liší ve znaménku u členu yx (pokud jsou x, y liché prvky), protože zde bylo prohozeno x s y . Superalgebry splňující superkomutativitu nazýváme komutativními superalgebry a značíme je $\mathcal{C}om$ -superalgebry.

Dále definujeme **superkomutátor** superalgebry B , který je bilineárním zobrazením $[\cdot, \cdot]_s : B \times B \rightarrow B$, které je definováno vztahem:

$$[x, y]_s = xy - (-1)^{\overline{xy}}yx, \text{ pro } x, y \in B.$$

Superalternativita

- $\forall x, y, z \in A : (x, y, z) + (y, x, z) = 0$, (linearizovaná levá alternativita)
- $\forall x, y, z \in A : (x, y, z) + (x, z, y) = 0$, (linearizovaná pravá alternativita)
- $\forall x, y, z \in B : (x, y, z) + (-1)^{\overline{xy}}(y, x, z) = 0$, (**levá superalternativita**)
- $\forall x, y, z \in B : (x, y, z) + (-1)^{\overline{yz}}(x, z, y) = 0$. (**pravá superalternativita**)

Superalgebry splňující levou i pravou superalternativitu nazýváme alternativními superalgebry a značíme je $\mathcal{A}lt$ -superalgebry.

Super-Jordanův součin

- $\forall x, y \in A : x \circ y = xy + yx = 0$, (Jordanův součin)
- $\forall x, y \in B : x \circ_s y = xy + (-1)^{\overline{xy}}yx = 0$. (**super-Jordanův součin**)

Nil_n superidentita vznikla linearizací identity $x^n = 0$. Na zvoleném uzávorkování linearizované formy identity nezáleží, protože nil-algebry jsou asociativní v mocninách, viz [1]. Proto budeme používat přirozené levé uzávorkování.

- $\forall x \in A : \sum_{\pi \in Sym(n)} (\dots (x_{\pi(1)}x_{\pi(2)})x_{\pi(3)}) \dots x_{\pi(n)} = 0$, (nil_n identita)
- $\forall x \in B : \sum_{\pi \in Sym(n)} \text{sign}_{\text{odd}}(\pi) (\dots (x_{\pi(1)}x_{\pi(2)})x_{\pi(3)}) \dots x_{\pi(n)} = 0$, (**nil_n superidentita**)

kde $Sym(n)$ je symetrická grupa a $\text{sign}_{\text{odd}}(\pi)$ je počet prohozených lichých prvků modulo 2 v permutaci π . Superalgebry splňující nil_n superidentitu nazýváme **nil_n**-superalgebry a značíme je $\mathcal{N}il_n$ -superalgebry. Alternativní superalgebry splňující nil_n superidentitu nazýváme **alternativními nil_n**-superalgebry a značíme je $\mathcal{A}lt\text{-}\mathcal{N}il_n$ -superalgebry.

1.6 Alternativní $\mathcal{N}il_n$ -superalgebra s jedním lichým generátorem

V disertační práci naší vedoucí řešili hledání báze $\mathcal{A}lt\text{-}\mathcal{N}il_n$ superalgebry s jedním lichým generátorem. Tato superalgebra je definována jako podalgebra superalgebry $\mathcal{A}lt$, jejíž báze byla popsána I. Shestakovem a N. Zhukavets např. v [12]. V následujícím textu budeme používat následující označení celočíselných indexů bez dalšího komentáře: i, j, k, m, n .

Nechť $\mathcal{A} = \mathcal{A}lt[\emptyset; x]$ je volná alternativní superalgebra generovaná jedním lichým generátorem x , která je definována induktivně:

$$x^{[1]} = x, \quad x^{[i+1]} = [x^{[i]}, x]_s, \quad i > 0,$$

a označme si

$$t = x^{[2]}, \quad z^{[k]} = [x^{[k]}, t], \quad u^{[k]} = x^{[k]} \circ_s x^{[3]}, \quad k > 1.$$

- Prvky

$$\begin{aligned} t^m x^\sigma, \quad m + \sigma \geq 1, \quad t^m (x^{[k+2]} x^\sigma), \\ t^m (u^{[4k+\varepsilon]} x^\sigma), \quad t^m (z^{[4k+\varepsilon]} x^\sigma), \end{aligned} \quad (1.1)$$

kde $k > 0$, $m \geq 0$, $\varepsilon, \sigma \in \{0, 1\}$, tvoří bázi superalgebry \mathcal{A} .

- Pro všechny $k > 0$ platí:

$$z^{[4k-1]} = z^{[4k-2]} = u^{[4k-1]} = 0, \quad u^{[2]} = 0, \quad u^{[4k+2]} = -tz^{[4k+1]}.$$

- Jádro superalgebry \mathcal{A} je rovno následujícímu ideálu:

$$\langle u^{[k]}, z^{[k]} \mid k > 1 \rangle.$$

Centrum superalgebry \mathcal{A} je rovno následujícímu vektorovému prostoru:

$$\langle t^m z^{[k]}, t^m (2z^{[k]} x - u^{[k]}) \mid m \geq 0, \quad k > 2 \rangle.$$

Dále v textu budeme psát prvky báze $t^m u^{[k]} x^\sigma$, $t^m z^{[k]} x^\sigma$ bez uzávorkování. Definujeme také nulové a záporné mocniny prvku t jako $t^0 = 1$ a $t^k = 0$ pro $\forall k < 0$.

1.6.1 Tabulka násobení prvků báze \mathcal{A}

Tabulka násobení prvků báze superalgebry \mathcal{A} je dána následujícími pravidly: pro libovolné $m, n \geq 0$, $k > 2$, $\varepsilon, \sigma \in \{0, 1\}$, $n + \sigma \geq 1$, platí, že prvky:

$$t^m u^{[k]} x^\sigma, \quad t^m z^{[k]} x^\sigma,$$

vynulují všechny ostatní prvky báze (1.1) kromě prvků prvního typu. Dále platí:

$$\begin{aligned} (t^m u^{[k]} x^\varepsilon)(t^n x^\sigma) &= \begin{cases} t^{m+n} u^{[k]} x^{\varepsilon+\sigma}, & \varepsilon + \sigma < 2, \\ \frac{1}{2} t^{m+n+1} u^{[k]}, & \varepsilon + \sigma = 2, \end{cases} \\ (t^m z^{[k]} x^\varepsilon)(t^n x^\sigma) &= \begin{cases} t^{m+n} z^{[k]} x^{\varepsilon+\sigma}, & \varepsilon + \sigma < 2, \\ \frac{1}{2} t^{m+n+1} z^{[k]}, & \varepsilon + \sigma = 2, \end{cases} \end{aligned}$$

$$(t^n x)(t^m u^{[k]} x^\varepsilon) = (-1)^{k+1} \begin{cases} t^{m+n} (u^{[k]} x - 2tz^{[k]}), & \varepsilon = 0, \\ t^{m+n+1} (\frac{1}{2} u^{[k]} - 2z^{[k]} x), & \varepsilon = 1, \end{cases}$$

$$(t^n x)(t^m z^{[k]} x^\varepsilon) = (-1)^k \begin{cases} t^{m+n} z^{[k]} x, & \varepsilon = 0, \\ \frac{1}{2} t^{m+n+1} z^{[k]}, & \varepsilon = 1. \end{cases}$$

$$\begin{aligned} t^m (t^n x^\sigma) &= t^{m+n} x^\sigma, \\ (t^m x) t^n &= t^{m+n} x - n t^{m+n-1} x^{[3]}, \\ (t^m x)(t^n x) &= \frac{1}{2} t^{m+n+1} - n t^{m+n-1} (x^{[3]} x) + \frac{m+2n}{3} t^{m+n-1} x^{[4]} \\ &\quad + \frac{m(m+2n-1)}{6} t^{m+n-2} z^{[4]}. \end{aligned}$$

$$\begin{aligned} t^m (t^n (x^{[k]} x^\varepsilon)) &= t^{m+n} (x^{[k]} x^\varepsilon), \\ (t^m x)(t^n x^{[k]}) &= (-1)^k \left(t^{m+n} ((x^{[k]} x) - x^{[k+1]}) \right. \\ &\quad \left. - t^{m+n-1} \left(\frac{n}{2} u^{[k]} + \frac{2m+n}{6} z^{[k+1]} \right) \right), \\ (t^m x)(t^n (x^{[k]} x)) &= (-1)^k \left(\frac{1}{2} t^{m+n+1} x^{[k]} + t^{m+n} \left(\frac{2}{3} x^{[k+2]} - x^{[k+1]} x \right. \right. \\ &\quad \left. \left. + \frac{2m+4n+1}{6} z^{[k]} \right) - t^{m+n-1} \left(\frac{n}{2} u^{[k]} x - \frac{m+2n}{6} u^{[k+1]} \right. \right. \\ &\quad \left. \left. + \frac{2m+n}{6} z^{[k+1]} x - \frac{m}{6} z^{[k+2]} \right) \right). \end{aligned}$$

$$\begin{aligned} (t^n x^{[k]}) t^m &= t^{m+n} x^{[k]} + m t^{m+n-1} z^{[k]}, \\ (t^n x^{[k]})(t^m x) &= t^{m+n} (x^{[k]} x) + t^{m+n-1} \left(m z^{[k]} x + \frac{2m+n}{3} z^{[k+1]} \right), \\ (t^n (x^{[k]} x)) t^m &= t^{m+n} (x^{[k]} x) - m t^{m+n-1} \left(\frac{1}{2} u^{[k]} - z^{[k]} x - \frac{1}{2} z^{[k+1]} \right), \\ (t^n (x^{[k]} x))(t^m x) &= \frac{1}{2} t^{m+n+1} x^{[k]} + t^{m+n} \left(\frac{1}{3} x^{[k+2]} + \frac{7m+2n+2}{6} z^{[k]} \right) \\ &\quad - t^{m+n-1} \left(\frac{m}{2} u^{[k]} x - \frac{2m+n}{6} u^{[k+1]} + \frac{m+2n}{6} z^{[k+1]} x \right. \\ &\quad \left. - \frac{2m+n}{6} z^{[k+2]} \right). \end{aligned}$$

$$\begin{aligned}
(t^m x^{[i]})(t^n x^{[j]}) &= \frac{1}{2}(-1)^{c(j+1)} t^{m+n} (u^{[i+j-3]} + \delta_j t z^{[i+j-4]} \\
&\quad - (-1)^j z^{[i+j-2]}), \\
(t^m x^{[i]})(t^n (x^{[j]} x)) &= \frac{1}{2}(-1)^{c(j+1)} t^{m+n} (u^{[i+j-3]} x + \delta_j t (z^{[i+j-4]} x) \\
&\quad - (-1)^j z^{[i+j-2]} x - \frac{2}{3} z^{[i+j-1]}), \\
(t^m (x^{[i]} x))(t^n x^{[j]}) &= \frac{1}{2}(-1)^{c(j)} t^{m+n} (u^{[i+j-3]} x + (-1)^j u^{[i+j-2]} \\
&\quad + \delta_j t (z^{[i+j-4]} x) + (-1)^j \delta_{j-1} t z^{[i+j-3]} \\
&\quad - (-1)^j z^{[i+j-2]} x - \frac{1}{3} z^{[i+j-1]}), \\
(t^m (x^{[i]} x))(t^n (x^{[j]} x)) &= \frac{1}{2}(-1)^{c(j)} t^{m+n} (\frac{1}{2} t u^{[i+j-3]} + (-1)^j u^{[i+j-2]} x \\
&\quad - \frac{1}{3} u^{[i+j-1]} + \frac{1}{2} \delta_j t^2 z^{[i+j-4]} - \delta_{j-1} t (z^{[i+j-3]} x) \\
&\quad - (\frac{5}{6} \delta_j - \frac{1}{2}) t z^{[i+j-2]} + \frac{1}{3} z^{[i+j-1]} x - \frac{1}{3} (-1)^j z^{[i+j]}),
\end{aligned}$$

kde $c(j) = \frac{j(j-1)}{2}$ a $\delta_j = 1 + (-1)^j$.

Poznamenejme, že pro $\forall k \geq 3, \forall m, n \geq 0$ platí:

$$\begin{aligned}
(t^n x^{[3]})(t^m x^{[k]}) &= \frac{1}{2}(-1)^k t^{m+n} (u^{[k]} - z^{[k+1]}), \\
(t^n x^{[3]})(t^m (x^{[k]} x)) &= \frac{1}{2}(-1)^k t^{m+n} (u^{[k]} x - z^{[k+1]} x + \frac{2}{3} z^{[k+2]}), \\
(t^n x^{[4]})(t^m x^{[k]}) &= \frac{1}{2} t^{m+n} (u^{[k+1]} + 2t z^{[k]} + z^{[k+2]}), \\
(t^n x^{[4]})(t^m (x^{[k]} x)) &= \frac{1}{2} t^{m+n} (u^{[k+1]} x + 2z^{[k]} x + z^{[k+2]} x + \frac{2}{3} z^{[k+3]}).
\end{aligned}$$

Použitím této tabulky násobení lze snadno vypočítat:

$$(x, x, x) = \frac{1}{2} x^3 \neq 0.$$

Tedy tato alternativní superalgebra \mathcal{A} není asociativní.

1.6.2 Báze volné alternativní $\mathcal{N}il_n$ -superalgebry s jedním lichým generátorem

V této podsececi zmíníme bázi volné alternativní $\mathcal{N}il_n$ -superalgebry s jedním lichým generátorem $\mathcal{B}_n = \mathcal{A}lt\text{-}\mathcal{N}il[\emptyset; x]$. Konstrukce této báze byla podrobně popsána v [1].

Z [1] víme, že pokud je A alternativní algebrou nad tělesem charakteristiky 0 a I_n je nějaký podprostor A generovaný prvky $a^n, \forall a \in A$, tak je poté I_n ideálem algebry A . Toto tvrzení nyní rozšíříme na superalgebry.

Nechť $\mathcal{A} = \mathcal{A}[\emptyset; x]$ je volná alternativní superalgebra s jedním lichým generátorem x s bází popsanou v (1.1). Nechť \mathcal{I}_n je ideál superalgebry \mathcal{A} generovaný

prvky $W_n(u_1, u_2, \dots, u_n)$, $u_1, \dots, u_n \in \mathcal{A}_0 \cup \mathcal{A}_1$, kde

$$W_n(u_1, u_2, \dots, u_n) = \sum_{\sigma \in \text{Sym}(n)} \text{sign}_{\text{odd}}(\sigma) (\dots ((u_{\sigma(1)} u_{\sigma(2)}) u_{\sigma(3)}) \dots) u_{\sigma(n)},$$

poté je \mathcal{I}_n ideálem superalgebry \mathcal{A} , viz [1]. Superalgebra $\mathcal{A}/\mathcal{I}_n$ je poté volnou alternativní $\mathcal{N}il_n$ -superalgebrou s jedním lichým generátorem x a tuto superalgebru budeme značit $\mathcal{B}_n = \text{Alt-}\mathcal{N}il_n[\emptyset; x]$.

Poznamenejme, že pro každý lichý prvek $y \in \mathcal{B}_n$ a libovolné prvky $u_i \in \mathcal{B}_n$ platí:

$$W_n(y, y, u_3, \dots, u_n) = 0.$$

Navíc pro všechny permutace $\sigma \in \text{Sym}(n)$ platí:

$$W_n(u_{\sigma(1)}, u_{\sigma(2)}, \dots, u_{\sigma(n)}) = \text{sign}_{\text{odd}}(\sigma) W_n(u_1, u_2, \dots, u_n).$$

Uvažujme nyní ideál $\mathcal{I}_n \subset \mathcal{A}$ generovaný $W_n(u_1, u_2, \dots, u_n)$, $u_1, u_2, \dots, u_n \in \mathcal{A}_0 \cup \mathcal{A}_1$, kde

$$W_n(u_1, u_2, \dots, u_n) = \sum_{\sigma \in \text{Sym}(n)} \text{sign}_{\text{odd}}(\sigma) (\dots ((u_{\sigma(1)} u_{\sigma(2)}) u_{\sigma(3)}) \dots) u_{\sigma(n)}.$$

Prvky:

$$\begin{aligned} & t^{m+n} x^\sigma, \\ & t^{n-1} x - \frac{n-1}{2} t^{n-2} x^{[3]}, \\ & t^{n-2} x^{[k+3]} + \frac{n-2}{2} t^{n-3} (u^{[k+2]} + z^{[k+3]}), \quad t^{n-2} (x^{[k+2]} x) + \frac{n-2}{3} t^{n-3} z^{[k+3]}, \\ & t^{m+n-1} (x^{[k+2]} x^\sigma), \\ & t^{m+n-3} (u^{[4k+\varepsilon]} x^\sigma), \quad m + \sigma \geq 1, \\ & t^{m+n-3} (z^{[4k+\varepsilon]} x^\sigma), \quad m + \sigma \geq 1, \end{aligned}$$

kde $k > 0$, $m \geq 0$; $\varepsilon, \sigma \in \{0, 1\}$, tvoří bázi ideálu \mathcal{I}_n , podrobné výpočty byly zveřejněny v [1] a my je ověříme naším programem v kapitole 4.

Superalgebra $\mathcal{B}_n = \mathcal{A}/\mathcal{I}_n$ je poté generována prvky:

$$\begin{aligned} & t^i x^\sigma, \quad 0 \leq i < n, \quad i + \sigma > 0, \\ & t^i (x^{[k]} x^\sigma), \quad 0 \leq i < n - 2, \\ & t^i u^{[4m+\varepsilon]} x^\sigma, \quad t^i z^{[4m+\varepsilon]} x^\sigma, \quad 0 \leq i < n - 2, \quad i + \sigma < n - 2, \end{aligned} \quad (1.2)$$

kde $k > 2$, $m > 0$, $\varepsilon, \sigma \in \{0, 1\}$. Všechny ostatní prvky z (1.1) jsou rovny 0, kromě prvků:

$$\begin{aligned}t^{n-1}x &= \frac{n-1}{2}t^{n-2}x^{[3]}, \\t^{n-2}x^{[k+3]} &= -\frac{n-2}{2}t^{n-3}(u^{[k+2]} + z^{[k+3]}), \\t^{n-2}(x^{[k+2]}x) &= -\frac{n-2}{3}t^{n-3}z^{[k+3]}.\end{aligned}$$

Prvky z (1.2) tvoří bázi superalgebry \mathcal{B}_n .

Tabulka násobení prvků báze \mathcal{B}_n je dána tabulkou násobení prvků báze \mathcal{A} modulo ideál \mathcal{I}_n .

V následující kapitole se zaměříme na analýzu a návrh aplikace schopné vyhodnocovat aritmetické výrazy v superalgebře popsané v 1.6. Tato aplikace bude také pro pevně stanovené n umět také vyhodnocovat nil_n superidentitu uvedenou v 1.5. Aplikaci poté využijeme k ověření správnosti výpočtů zveřejněných v disertační práci [1] v kapitole, která popisovala konstrukci báze (1.2) volné alternativní $\mathcal{N}il_n$ -superalgebry s jedním lichým generátorem. Výsledky tohoto ověření uvedeme v kapitole 4.

Analýza a návrh

V této kapitole nejdříve popíšeme již existující aplikace pro práci s neasociativními/nekomutativními matematickými strukturami a zmíníme jejich výhody a nevýhody. Poté nastíníme vlastní návrh aplikace pro práci s prvky neasociativní (alternativní) nil-superalgebry s jedním lichým generátorem, která byla popsána v předcházející kapitole.

2.1 Rešerše stávajících řešení

Studiem neasociativních algeber se na světě nezabývá mnoho lidí, a tudíž není žádným překvapením, že programů pro práci s nimi je velmi málo. Dle našeho vlastního hledání a sbírky aplikací pro symbolické výpočty uveřejněnou ACM SIGSAM [13] (skupina odborníků zabývající se využitím počítačů při symbolických výpočtech, odnož profesního sdružení počítačových odborníků) je veřejně k dispozici pouze jediný počítačový program pro práci s neasociativními algebry, a tím je *Albert*. Popsáním programu *Albert* se budeme zabývat v následující podsekci. V závěru této sekce ještě představíme univerzální matematický software pro symbolické výpočty, *Wolfram Mathematica*, který jsme se rozhodli využít pro zjednodušování výrazů v naší aplikaci.

2.1.1 Albert

Albert je interaktivní počítačový program pro práci s neasociativními algebry. Informace o programu *Albert* byly převzaty z úvodního článku [14] napsaného tvůrci tohoto programu při jeho vydání. Program nese své jméno po vynikajícím americkém matematikovi 20. století Abrahamu Adrianu Albertovi [15], který se skoro celý svůj život věnoval studiu matematických struktur a byl také průkopníkem teorie neasociativních algeber. Tento program je implementován v jazyce *C* a byl vytvořen v roce 1990 na univerzitě v Clemsonu v Jižní Karolíně pány Davidem Jacobsem a Sekharem Muddannou. *Albert* řeší rozpoznávání neasociativních polynomiálních identit algeber nad konečnými

tělesy \mathbb{Z}_p . To znamená, že do *Albertu* zadáme množinu homogenních identit, o kterých víme, že je daná algebra splňuje, množinu generátorů a zvolíme číselné těleso nad kterým chceme algebru zkonstruovat. Program je bohužel omezen na práci s pouze konečnými tělesy \mathbb{Z}_p , kde p je prvočíslo menší nebo rovno 251. Po zadání těchto parametrů *Albert* sestrojí volnou algebru nad zvoleným číselným tělesem definovanou zadanými identitami. Tím je program inicializován a připraven k použití. Poté můžeme do *Albertu* zadat polynom, o kterém chceme vědět, zda-li je identitou. *Albert* tento polynom vyhodnotí ve zkonstruované volné algebře, a pokud je zde roven 0, tak nám vrátí kladnou odpověď, že zadaný polynom je identitou variety algeber nad námi zvoleným tělesem definované množinou zadaných identit. Cílem této práce je ale vyvinout nástroj pro práci s prvky superalgebry nad tělesem charakteristiky 0 a to nám *Albert* bohužel nedovolí, proto je potřeba se zabývat i jinými možnostmi řešení tohoto problému.

2.1.2 Wolfram Mathematica

Wolfram Mathematica je univerzální matematický software, který zvládá symbolické i numerické výpočty, vizualizaci informací a mnoho dalšího. Je vyvíjen společností *Wolfram Research*, kterou založil Stephen Wolfram v roce 1987. První verze programu *Wolfram Mathematica* spatřila světlo světa v roce 1988 [16]. Tento program bohužel stále nepodporuje práci s neasociativními strukturami, nabízí pouze nekomutativní, ale asociativní, násobení matematických objektů, více viz oficiální dokumentace této funkce [17]. Velká výhoda tohoto programu ale spočívá v jeho modularitě, lze například vytvořit vlastní modul, který bude provádět požadované operace a po spuštění *Wolframu Mathematica* si ho může každý uživatel načíst a využívat jím přidanou funkcionalitu. Bohužel ale zatím žádný univerzální modul pro práci s neasociativními algebrami nebyl vytvořen, což se snad v budoucnu změní. *Wolfram Mathematica* také nabízí možnost připojení externího programu k jeho kernelu (kernel *Wolframu Mathematica* je komunikační rozhraní umožňující vyhodnocovat příkazy v jazyce *Wolfram Language*) a volání jeho funkcí, což my využijeme pro finální zjednodušování výrazů (koeficientů a exponentů) a převodu výsledků do formátu *LaTeX*.

2.1.3 Shrnutí současného stavu

Cílem práce je vytvořit aplikaci pro počítání v alternativní nil-superalgebře, popsané v 1.6, nad nekonečným tělesem charakteristiky 0. *Albert* se pro náš účel nehodí hned ze dvou důvodů. Za prvé pracuje pouze s algebrami nad konečnými číselnými tělesy prvočíselné charakteristiky a za druhé nezobrazuje žádné mezivýsledky, ani postup svých výpočetních operací. Zobrazí nám pouze výsledek, zda-li je zadaný polynom identitou či nikoliv. My ale potřebujeme nástroj schopný vyhodnocovat obecné výrazy v nějaké superalgebře.

Wolfram Mathematica vůbec nepodporuje práci s neasociativními matematickými strukturami a zatím není k dispozici žádný uživatelský modul, který by tuto chybějící funkcionalitu rozšiřoval. Proto nelze použít žádné již existující řešení a je potřeba vytvořit novou specializovanou aplikaci, která bude pracovat pouze s prvky této jedné superalgebry.

2.2 Vlastní návrh

V této části představíme návrh nové aplikace pro počítání v alternativní nil-superalgebře s jedním lichým generátorem, popsané v 1.6. V té sekci ji budeme nazývat pouze superalgebra. Nejdříve si stanovíme, co od vyvíjeného programu požadujeme za funkcionalitu, poté popíšeme jakým způsobem plánujeme tyto funkcionality splnit a na závěr uvedeme diskuzi možných technologií a popis gramatiky syntaktického analyzátoru vstupu.

2.2.1 Funkcionalita

- Program bude umět vyhodnocovat aritmetické výrazy v této superalgebře, aritmetické výrazy se budou moci skládat z prvků superalgebry, závorek a operace vektorového součinu prvků superalgebry. Program bude dále umět vyhodnocovat superidentitu nil_n , popsanou v 1.5, pro pevně stanovené n . Definice konkrétní superidentity (pro různá n) bude načítána ze souboru určeného formátu. Výsledek všech výpočtů bude vyjádřen jako lineární kombinace prvků báze této superalgebry.
- Program bude umět načíst tabulku násobení prvků báze ze souboru určeného formátu a tuto tabulku poté použije pro vyhodnocování operace vektorového součinu prvků této superalgebry.
- Program bude obsahovat interaktivní mód (pro zadávání příkazů z terminálu), v tomto módu bude uživateli k dispozici textové menu, které mu mimo zadávání příkazů umožní také nastavení chování programu. Grafické uživatelské rozhraní není požadováno.
- Program umožní uživateli zvolit si ze dvou formátů výstupu. K dispozici bude výstup ve formátu *Wolfram Mathematica* a nebo ve formátu *LaTeX*.

V následující části popíšeme, jak bude zajištěno splnění těchto funkcionalit.

Načtení tabulky násobení

Aby program mohl vůbec začít vyhodnocovat výrazy, je potřeba načíst tabulku násobení prvků báze. Tabulka bude načtena ze souboru určeného formátu. Je tedy nutné vytvořit gramatiku pro tento formát souborů a syntaktický

analyzátor tohoto formátu. Na začátku programu bude tato tabulka násobení tímto syntaktickým analyzátozem načtena a uložena do operační paměti.

Vyhodnocování aritmetických výrazů

Pro vyhodnocování aritmetických výrazů musíme nejdříve rozdělít („tokenizovat“) vstup na prvky algebry a ostatní symboly. Při vyhodnocování výrazu nám budou největší problém dělat závorky, protože je nutné je vyhodnocovat od těch nejnvnitřnějších. Toto lze vyřešit například pomocí zásobníku následujícím algoritmem:

- Pokud máme na vstupu uzavírací závorku, v cyklu vyndáváme prvky z vrcholu zásobníku („pop“), dokud nenarazíme na otevírací závorku, poté tuto sekvenci symbolů mezi závorkami vyhodnotíme a výsledek vložíme na vrchol zásobníku („push“), v případě nenalezení otevírací závorky či výskytu nějaké chyby při vyhodnocování tohoto výrazu vypíšeme chybu a ukončíme zpracování vstupu.
- Pokud dostaneme na vstupu jiný symbol než je uzavírací závorka, tak tento symbol vložíme na vrchol zásobníku a pokračujeme dále.
- Pokud nenastala žádná chyba v průběhu zpracování vstupu, tak máme na konci na zásobníku pouze prvky superalgebry a symboly vektorového součinu.
- Každý prvek superalgebry je lineární kombinací prvků báze této superalgebry, lze si ho představit jako vektor násobků prvků báze superalgebry. Pro výpočet vektorového součinu nám tedy stačí vynásobit každý prvek z vektoru na levé straně s každým prvkem vektoru na pravé straně, pravidla pro tyto násobení jsou obsaženy v tabulce násobení prvků báze. Tato algebra je nekomutativní, je proto důležité zachovat pořadí činitelů součinu. Poté výsledek této operace můžeme zjednodušit (sečíst koeficienty u stejných prvků báze a odstranit nulové prvky). Výsledek tohoto násobení znovu vložíme na vrchol zásobníku jako jeden prvek superalgebry.
- Pro vyhodnocení celého aritmetického výrazu vyndáváme prvky ze dna zásobníku (zásobník s přístupem na oba konce může být třeba oboustranně zřetěžený spojový seznam) a pamatujeme si vždy předchozí prvek (levý operand), když narazíme na symbol násobení a levý operand existuje, přepneme se do módu násobení, pokud neexistuje, vypíšeme chybu a skončíme. Když jsme v módu násobení a na dně zásobníku je nějaký prvek, tak vypočítáme vektorový součin těchto dvou prvků a výsledek si uložíme jako případný levý operand pro následující násobení. Takto postupujeme dokud není zásobník prázdný, poté máme výsledek v levém operandu.

Vyhodnocení výrazu by se dalo realizovat i odebráním prvků z vrcholu zásobníku, stejně jako to je provedeno při párování závorek, ale jelikož je tato algebra neasociativní, tak vyhodnocování výrazu zespoda zásobníku dává uživateli možnost usnadnit si práci při zápisu součinu více jak 2 prvků. Například zápis výrazu: $x_1 \times x_2 \times x_3 \cdots x_n$, kde x_i jsou prvky superalgebry a \times je operace vektorového součinu, není jednoznačný, protože výsledek záleží na jeho uzávorkování. Pokud chce uživatel vyhodnotit tento výraz v přirozeném levém uzávorkování $(\cdots((x_1 \times x_2) \times x_3) \cdots) \times x_n$, může ho zapsat bez jakýchkoliv závorek a díky vyhodnocování zespoda zásobníku dostane přesně toto uzávorkování. Pokud by ale uživatel chtěl vyhodnotit tento výraz v přirozeném pravém uzávorkování $x_1 \times (\cdots(x_{n-2} \times (x_{n-1} \times x_n)) \cdots)$, může ho celý uzavřít do jednoho páru závorek a výraz se vyhodnotí jako vnitřek závorek od vrcholu zásobníku a uživatel dostane přesně toto pravé uzávorkování bez nutnosti vypisovat všechny závorky manuálně. Samozřejmě pokud uživatel zapíše výraz jednoznačně uzávorkovaný, nezáleží na tom, z jaké strany zásobníku ho budeme vyhodnocovat, uživatel dostane v obou případech stejný výsledek.

Vyhodnocování nil_n superidentity

Obecný tvar nil_n superidentity je popsán v sekci 1.5. Náš program bude umět vyhodnotit tuto superidentitu vždy pouze pro pevně stanovené n . Předpis této superidentity pro pevně zvolené n bude uložen v souboru určeného formátu. Pro tento formát je potřeba vytvořit gramatiku a syntaktický analyzátor této gramatiky. Pro vyhodnocení nil_n superidentity musíme nejdřív její předpis načíst z daného souboru. Po načtení superidentity bude už n zafixované a můžeme od uživatele načíst n prvků superalgebry a udělat substituci těchto prvků do předpisu superidentity. Tento výsledek poté můžeme vyhodnotit jako jakýkoliv jiný výraz v této superalgebře postupem popsaným výše v 2.2.1.

Zajištění formátovaného výstupu

Pro výstup do formátu *LaTeX* nebo *Wolfram Mathematica* je potřeba vytvořit funkce, které budou interní data transformovat do vybraného formátu. Výstup do formátu *Wolfram Mathematica* je snadný, protože je to interní formát našeho programu a používáme ho pro úpravy koeficientů a exponentů. Výstup do formátu *LaTeX* lze zajistit voláním funkce *Wolframu Mathematica TeXForm*, viz [18].

2.2.2 Volba technologií

V této podsekci popíšeme různé možné způsoby implementace návrhu, jejich výhody a nevýhody a na závěr zmíníme, jaký programovací jazyk a jaké knihovny byly zvoleny pro implementaci naší aplikace.

Matematická knihovna

Matematickou knihovnu budeme využívat pro úpravu aritmetických výrazů (koeficientů a exponentů prvků superalgebry). Od knihovny požadujeme práci s proměnnými (možnost přidání omezujících podmínek pro proměnné je vítána), vyhodnocování standardních matematických funkcí a především zjednodušování aritmetických výrazů.

První možností bylo použít matematickou knihovnu *ExprTk* [19] pro jazyk *C++*. Tato knihovna podporuje vyhodnocování aritmetických výrazů i s proměnnými a běžnými matematickými funkcemi (před vyhodnocením je ale potřeba přiřadit proměnným číselnou hodnotu). Knihovna je při své práci efektivní. Velký problém ale spočívá v tom, že nepodporuje zjednodušování výrazů s proměnnými (do kterých nebylo dosazeno) a také, že tato knihovna je pouze pro jazyk *C++*.

Druhou možností bylo využít funkcionalitu programu *Wolfram Mathematica*, který nabízí řešení mnoha matematických problémů, umí pracovat s výrazy s neznámými a zjednodušovat je. Pro každou proměnnou lze navíc stanovit omezující podmínky před vyhodnocením daného výrazu. Nevýhodou této volby je ale nižší efektivita/rychlost než v případě knihovny *ExprTk*. Kromě toho ale splňuje všechny výše popsané požadavky a navíc umožňuje převést výsledky do formátu *LaTeX*, což je vítaný bonus, jelikož výstup ve formátu *LaTeX* byl jednou z požadovaných funkcionalit.

Pro využití funkcí *Wolfram Mathematica* není nutné psát vyvíjený program v jazyce *Wolfram Language*, ale lze ho napsat v nějakém podporovaném jazyce (*C/C++/C#/Java/Visual Basic .NET*) a poté použít *Wolfram Library-Link* [20] a načíst tento program do kernelu *Wolfram Mathematica* (takto se například dají importovat moduly napsané v jiném jazyce). Druhou možností je v programu použít *Wolfram Symbolic Transfer Protocol (WSTP)* [21] pro připojení tohoto programu do kernelu *Wolfram Mathematica* a využívání jeho funkcí. Nevýhody využití *Wolfram Mathematica* jsou: nižší rychlost výpočtů a také, že každý uživatel, který bude chtít náš program využívat, bude muset mít nainstalovanou a aktivovanou aplikaci *Wolfram Mathematica* na svém počítači. Další nevýhodou je bezpochyby také obtížnost zprovoznění komunikace mezi *Wolframem Mathematica* a naším programem.

Rozhodli jsme se pro *Wolfram Mathematica*, protože je to komplexní řešení a nabízí snadnou rozšiřitelnost využívaných funkcí (např. faktoriál či sumy). Splňuje všechny naše požadavky a také nám usnadní formátování výstupu do *LaTeXu*. Tato volba nám také nechává možnost rozhodnout se mezi více programovacími jazyky. Cílová skupina uživatelů tohoto programu budou matematici, kteří často mají již *Wolfram Mathematica* nainstalovaný na svém

počítači a používají ho ke své práci. Jelikož program bude specializovaný jenom pro jednu specifickou superalgebru, použijeme pro komunikaci s kernelem implementačně snazší řešení, tj. *WSTP*. Použití *LibraryLinku* by se vyplatilo v případě, že bychom například implementovali obecný nástroj pro práci s neasociativními strukturami a chtěli bychom tento nástroj přímo integrovat do *Wolfram Mathematica* jako modul, což by mohlo být zajímavým rozšířením této bakalářské práce.

Programovací jazyk

Volba programovacího jazyka byla snadná, protože pro výpočet nil_n superidentity, popsané v 1.5, je potřeba provést zhruba přibližně $n \cdot n!$ násobení prvků báze. Je tudíž potřeba zvolit jazyk, který je dostatečně rychlý. Obecně je známo, že kompilované jazyky jsou rychlejší než jazyky interpretované nebo jazyky kompilované za běhu (JIT). Z Wolframem podporovaných jazyků jsou jenom jazyky *C* a *C++* kompilované (do nativního kódu). *C++* je vlastně rozšíření jazyka *C* o podporu objektově orientovaného návrhu. Objektově orientovaný návrh usnadňuje orientaci ve zdrojovém kódu, jelikož je potřeba delší úvahy před samotným programováním. V jazyce *C++* je dále k dispozici standardní knihovna, která nabízí mnoho obecných kontejnerů, jejichž použití nám velmi usnadní vývoj této aplikace, a proto jsme se rozhodli pro programovací jazyk *C++*.

2.2.3 Lexikální analyzátor a gramatika prvků báze

Pro rozdělení vstupu na tokeny („tokenizaci“) jsme použili lexikální analyzátor *Flex* [22], který čte vstup a rozeznává v něm lexikální vzory. Tyto vzory jsou definované pomocí regulárních výrazů a ke každému vzoru je přiřazen kód v jazyce *C/C++*, který se při nalezení daného vzoru vykoná. Použití *Flexu* nám usnadní práci a případná změna detekovaných symbolů je velmi snadná. *Flex* vygeneruje ze souboru pravidel (v našem případě přibližně 100 řádků) lexikální analyzátor v jazyce *C/C++*, který lze poté zakomponovat do většího projektu. Výsledný soubor lexikálního analyzátoru je mnohonásobně větší (v tomto případě přibližně 1800 řádků kódu) a pokud bychom tento lexikální analyzátor vyvíjeli ručně, tak by pravděpodobnost, že někde uděláme chybu, byla mnohonásobně vyšší než s použitím *Flexu*.

V následující tabulce 2.1 uvádíme tokeny akceptované našim lexikálním analyzátozem a k nim odpovídající regulární výrazy, kterými jsou popsány. Sekvence znaků mezi rovnými uvozovkami {""} se testují na přesnou shodu. Znak

natých závorkách označují množinu možných znaků, které se shodují s daným pravidlem, pokud je mezi dvěma znaky v těchto závorkách znak mínus {-}, tak to udává interval a jakýkoliv znak v tomto intervalu se shoduje s tímto pravidlem. Např. [0-9] je pravidlo, které se shoduje s jakoukoli jednou číslicí od 0 do 9. Pokud za nějakým znakem či množinou znaků přidáme kvantifikátor plus {+}, tak se zde tento znak či množina znaků může libovolně krát opakovat, ale musí tu být alespoň jednou. Kvantifikátor hvězdička {*} udává, že se daný znak či množina znaků může libovolně krát opakovat, ale také tam nemusí být ani jednou. Znak svislá čára {|} je oddělovač variant, dané pravidlo se bude shodovat s jakoukoli z uvedených variant. Znak lomítka {/} se používá pro zajištění posloupnosti pravidel, vstup se musí shodovat s celým pravidlem, akorát část za lomítkem zůstane ve vstupu a použije se pro testování shody s dalšími pravidly. Např. pravidlo definované tímto regulárním výrazem: "x"/[\^] hledá ve vstupu posloupnost x^ a pokud ji najde, tak si z ní vezme jen první znak {x} a zbytek {^} nechá ve vstupním proudu pro další zpracování. V případě shody s dvěma a více pravidly *Flex* zvolí pravidlo s nejdelší shodou, pokud toto kritérium není rozhodné, tak je vybráno pravidlo z množiny shodujících se pravidel podle jejich pořadí v souboru. Bílé znaky a komentáře (zbytek řádku za symbolem {#}) jsou ignorovány. Jakékoliv jiné znaky, neshodující se s žádným pravidlem, jsou považovány za neplatný vstup a způsobí ukončení načítání vstupu.

Tabulka 2.1: Tokeny rozpoznávané lexikálním analyzátozem

Název tokenu	Regulární výraz
FUNC_DELTA	"\\delta("[0-9mnijk+/-]+)"
FUNC_C	"c("[0-9mnijk+/-]+)"
FUNC_SQRBRAC	"["["(0-9mnijk+/-)+]"
FUNC_SGN	"sgn(a_"[1-9][0-9]*)"
SYM_X_END	"x^("[]*"\\e"[]*)" "x^("[]*"\\s"[]*)" "x"
SYM_X	"x"/[\\^]
SYM_LBRACKET	"("
SYM_RBRACKET	")"
SYM_T	"t"
SYM_U	"u"
SYM_Z	"z"
SYM_IDENTITY	"W"
DELIM_ELEMENT_MUL	"**"
DELIM_MUL	"*"
DELIM_COMMA	", "
OP_POW	"^"
OP_EQUAL	"="
EXPR_PART	[0-9mnijk+/-]
IDENT_VAR	"a_"[1-9][0-9]*

Gramatika násobků prvků báze

V této podsekcí uvedeme reprezentaci jednotlivých násobků prvků báze, definované v (1.1), a gramatiku, která specifikuje formát zápisu těchto prvků, na závěr uvedeme konkrétní příklady zápisu prvků v tomto navrhovaném formátu. Násobkem prvku báze se myslí libovolný prvek báze s libovolným koeficientem (nemusí být roven 1). Rozlišujeme 4 typy bazických prvků, viz 1.1, které budeme dále označovat jako prvky prvního, druhého, třetího a čtvrtého typu. Exponent symbolu t budeme dále označovat jako t -exponent.

Gramatika násobků prvků báze vypadá následovně. Množina neterminálních symbolů je:

$$\{s, cf, a, b, c, d, e, second, third, fourth, end\},$$

množina terminálních symbolů této gramatiky se rovná množině tokenů rozpoznávaných lexikálním analyzátozem, viz tabulka 2.1, startovací symbol gramatiky je neterminální symbol s a množina přepisovacích pravidel je:

- $s \rightarrow \text{SYM_LBRACKET } a \text{ SYM_RBRACKET } | a |$
 $cf \text{ DELIM_MUL } a | cf \text{ DELIM_MUL SYM_LBRACKET } a \text{ SYM_RBRACKET},$

2. ANALÝZA A NÁVRH

- `cf` -> `SYM_LBRACKET cf SYM_RBRACKET cf | EXPR_PART cf | FUNC_DELTA cf | OP_POW SYM_LBRACKET cf SYM_RBRACKET cf | FUNC_C cf | ϵ ,`
- `a` -> `SYM_T OP_POW SYM_LBRACKET e SYM_RBRACKET b | c | SYM_T b,`
- `b` -> `SYM_LBRACKET d SYM_RBRACKET | d,`
- `c` -> `SYM_X_END | second | third | fourth,`
- `d` -> `end | second | third | fourth,`
- `second` -> `SYM_X FUNC_SQRBRAC end,`
- `third` -> `SYM_U FUNC_SQRBRAC end,`
- `fourth` -> `SYM_Z FUNC_SQRBRAC end,`
- `e` -> `EXPR_PART e | SYM_LBRACKET e SYM_RBRACKET e | ϵ ,`
- `end` -> `SYM_X_END | ϵ ,`

kde ϵ značí prázdný řetězec. Koeficienty a exponenty (t -exponent a exponent v hranatých závorkách) musí navíc být validní podle syntaxe *Wolfram Mathematica*, tato gramatika ji neřeší. Tato syntaxe je však přirozená, a tudíž ji zde ani nebudeme popisovat. Příklady zápisu konkrétních násobků prvků báze superalgebry v tomto formátu jsou uvedeny v následující tabulce 2.2.

Tabulka 2.2: Příklady zápisu násobků prvků báze v navrženém formátu

Násobek prvku báze	Zápis tohoto prvku v navrženém formátu
$-t^5x$	<code>-t^(5)x</code>
$2t^i z^{[8]}x$	<code>2*t^(i)z^[8]x</code>
$-\frac{3}{8}tu^{[i]}$	<code>-3/8*tu^[i]</code>
$mt^{m+n-1}z^{[k]}$	<code>m*t^(m + n - 1)z^[k]</code>
$(-1)^{k+1}\delta(j)t^{m+n}u^{[k]}x$	<code>(-1)^(k+1)\delta(j)*t^(m + n)u^[k] x</code>
$\frac{m(m+2n-1)}{6}t^{m+n}(x^{[k]}x)$	<code>m(m+2n-1)/(6)*t^(m+n)(x^[k]x)</code>
$\frac{(-1)^{c(j)}}{2}t^{m+n+1}z^{[i+j-4]}x$	<code>(1)/(2)(-1)^(c(j))*t^(m+n+1)(z^[i+j-4]x)</code>

Formát tabulky násobení

V této podsekcí popíšeme formát souboru tabulky násobení prvků báze superalgebry popsané v 1.6.1. Tento formát je přímočarý a vychází ze syntaxe jednotlivých prvků z předchozí podsekcí. Bílé znaky a komentáře (zbytek řádku za symbolem `{#}`) jsou ignorovány. Pořadí pravidel v souboru je rozhodující pro jejich použití, proto je potřeba v souboru tabulky násobení uvést speciální pravidla (t -exponent nebo exponent v hranatých závorkách je fixní) před těmi obecnými. Formát zápisu jednotlivých pravidel je následující:

```
PB DELIM_ELEMENT_MUL PB OP_EQUAL OB ... OB DELIM_COMMA,
```

kde PB je prvek báze (tedy prvek popsaný gramatikou v 2.2.3 s koeficientem 1), OB je násobek prvku báze (jakýkoliv prvek podle gramatiky 2.2.3), tokeny OP_EQUAL a DELIM_COMMA jsou popsány v tabulce tokenů 2.1. Důležité je ukončit každé pravidlo (i to úplně poslední) oddělovačem čárka (token DELIM_COMMA). Uvedeme příklad zápisu pravidla z tabulky násobení:

$$(t^n(x^{[k]}x)) \cdot t^m = t^{m+n}(x^{[k]}x) - mt^{m+n-1}(\frac{1}{2}u^{[k]} - z^{[k]}x - \frac{1}{2}z^{[k+1]}),$$

které by se v tomto formátu zapsalo následovně (na konci pravidla je uvedena ukončovací čárka):

```
t^(n) (x^[k]x)**t^(m) = t^(m+n) (x^[k]x) - (1)/(2)m*t^(m+n-1)u^[k]
+ m*t^(m + n - 1)z^[k] x + (1)/(2)m*t^(m + n - 1)z^[k + 1],.
```

Vytýkání před závorky není implementováno a pro správné načtení tabulky násobení je potřeba všechny závorky v pravidlech roznásobit.

Syntaxe zápisu obecného aritmetického výrazu

Obecný aritmetický výraz v této superalgebře se skládá z lineární kombinace prvků báze, otevíracích a uzavíracích závorek a operace vektorového součinu. Operace vektorového součinu je reprezentována tokenem DELIM_ELEMENT_MUL (popsaným v tabulce tokenů 2.1). Každý aritmetický výraz musí být ukončen čárkou (tokenem DELIM_COMMA). Uvedeme zde dva příklady zápisu nějakého aritmetického výrazu ve formátu specifikovaném v sekci 1.1 a také v námi navrhovaném formátu. Prvním příkladem aritmetického výrazu je:

$$2t^5x \cdot u^{[k]}x + 17t^7 - 15t^7,$$

který by se zapsal následovně (na konci tohoto výrazu je ukončovací čárka):

```
2*t^(5)x**u^[k]x + 17*t^(7) - 15*t^(7),.
```

Druhým příkladem aritmetického výrazu je:

$$((x \cdot t^6) \cdot x^{[3]}) \cdot (x + x + t^5x),$$

který by se zapsal jako (na konci tohoto výrazu je ukončovací čárka):

$$((x**t^{(6)})**x^{[3]})**(x + x + t^{(5)}x),.$$

Superalgebra s kterou pracujeme je neasociativní, a tudíž pokud není nějaký výraz jednoznačně uzávorkován, použije se přirozené levé uzávorkování, pokud tento výraz není uvnitř nějakých závorek. V opačném případě je použito přirozené pravé uzávorkování, podrobněji popsáno v 2.2.1.

Formát souboru nil_n superidentity pro pevné n

V této podsekci popíšeme formát souboru nil_n superidentity pro konkrétní n . Tato superidentita byla definována v sekci 1.5. V souboru nil_n superidentity jsou ignorovány bílé znaky a komentáře (zbytek řádku za symbolem $\{\#\}$). V souboru je očekávána pouze jedna superidentita následujícího tvaru:

$$W(a_1, a_2, \dots, a_n) = AR,$$

kde AR je aritmetický výraz skládající se pouze z tokenů z množiny $\{\text{FUNC_SGN}, \text{SYM_LBRACKET}, \text{OP_POW}, \text{SYM_IDENTITY}, \text{DELIM_MUL}, \text{SYM_RBRACKET}, \text{EXPR_PART}, \text{DELIM_ELEMENT_MUL}\}$, tudíž oproti obecnému aritmetickému výrazu zde používáme proměnné superidentity (které byly uvedeny na levé straně rovnice) a symbolu parity těchto proměnných, který je zastoupen tokenem FUNC_SGN . Definice superidentity musí být jako obvykle ukončena oddělovací čárkou (tokenem DELIM_COMMA). Tato definice superidentity musí být syntakticky správně, tj. po dosazení za proměnné a jejich znaménka to musí být validní aritmetický výraz podle pravidel uvedených v 2.2.3. Na závěr této podsekcce uvedeme příklad nil_2 superidentity:

$$W(a_1, a_2) = a_1 \cdot a_2 + (-1)^{\overline{a_1 a_2}} a_2 \cdot a_1,$$

která by se v námi navrhovaném formátu zapsala následovně (zápis superidentity je ukončen oddělovací čárkou):

$$W(a_1, a_2) = a_1**a_2 + (-1)^{((\text{sgn}(a_1) \text{sgn}(a_2)))}*a_2**a_1,.$$

Program samozřejmě zvládá vyhodnotit i jakékoliv jiné identity či superidentity, které lze zapsat v tomto formátu, ale tím se v této práci nebudeme zabývat.

V následující kapitole popíšeme, jak byl tento návrh implementován a poté ve zkratce popíšeme postup instalace a spuštění tohoto programu.

Implementace vlastního návrhu

V této kapitole popíšeme implementaci programu pro práci s prvky superalgebry popsané v sekci 1.6. V implementaci jsme se drželi návrhu popsaného v sekci 2.2 a implementace byla tedy přímočará. Jediná obtížnější část implementace byla spojení vyvíjeného programu s kernelem *Wolfram Mathematica*, kernel je *Wolfram Mathematica* notebook bez grafického rozhraní přijímající příkazy v jazyce *Wolfram Language*. Postup vytvoření tohoto spojení podrobně popíšeme v následující sekci 3.1 a bude moci sloužit jako inspirace pro ostatní, jež potřebují využívat funkcionalitu *Wolfram Mathematica* v nějakém svém externím programu. V dalších sekcích této kapitoly popíšeme konfigurační soubor, použité třídy a jejich účel, a nakonec uvedeme stručný návod k instalaci a použití tohoto programu. Podrobnější uživatelská příručka je k dispozici v příloze A této práce.

3.1 Spojení programu s kernelem *Wolfram Mathematica* pomocí *WSTP*

V této sekci uvedeme postup, jakým jsme postupovali při propojování našeho programu s kernelem *Wolfram Mathematica*. Tento postup je platný pro propojení programu napsaného v jazyce *C/C++* s kernelem *Wolfram Mathematica* ve verzi 10.0 či novější, může tedy sloužit jako návod pro ostatní vývojáře, kteří potřebují v nějakém svém programu využít funkcionalitu *Wolfram Mathematica*. Kód pro vytvoření spojení s kernelem *Wolfram Mathematica* je pro všechny operační systémy stejný, odlišnosti nastávají až při kompilaci programu, protože na různých operačních systémech jsou potřeba různé podpůrné knihovny. Tato sekce byla napsána na základě oficiální vývojářské příručky [23] a vlastních zkušeností s vývojem tohoto programu. Tento návod byl otestován v operačním systému *Mac OS X El Capitan*.

3. IMPLEMENTACE VLASTNÍHO NÁVRHU

Funkce potřebné pro vytvoření připojení ke kernelu jsou definovány v hlavičkovém souboru `wstp.h` a implementovány v knihovnách `libWSTPi4.a` a `wstp.framework`. Po instalaci *Wolframu Mathematica* verze 10.0 a novější do vašeho *Mac OS X* systému, se budou všechny tyto soubory nacházet v adresáři `/Applications/Mathematica.app/Contents/SystemFiles/Links/WSTP/DeveloperKit/MacOSX-x86-64`. Tento adresář však v běžném nastavení kompilátor jazyka *C/C++* neprohledává, pro zkompilování vašeho programu je tedy nezbytné tyto soubory zkopírovat do nějakého adresáře, který je prohledáván kompilátorem, a nebo změnit nastavení kompilátoru. U kompilátorů *GCC* nebo *clang* to lze vyřešit použitím přepínače `-I` následovaným cestou k adresáři obsahující hlavičkový soubor `wstp.h` a přepínačem `-L` následovaným cestou ke knihovnam `libWSTPi4.a` a `wstp.framework`, ve standardní instalaci programu *Wolframu Mathematica* jsou tyto dvě cesty totožné.

Vývoj aplikace využívající spojení s kernelem *Wolframu Mathematica* začneme vložením (`#include`) hlavičkového souboru `wstp.h` do našeho programu. Pro využívání funkcí *Wolframu Mathematica* v externím programu je nejprve potřeba vytvořit spojení s kernelem, to lze provést například způsobem uvedeným v ukázce 3.1, kde proměnná `pathToKernel` obsahuje cestu ke kernelu, která je po standardní instalaci `/Applications/Mathematica.app/Contents/MacOS/WolframKernel`. A proměnná `pathToExec` obsahuje cestu k programu, který chce toto spojení vytvořit (ve funkci `main` lze použít vstupní parametr `argv[0]`). V ukázce je také nastíněno ověřování, že vytvoření spojení proběhlo v pořádku. Pokud se spojení podaří vytvořit, lze tímto protokolem posílat data přímo do kernelu *Wolframu Mathematica* a nechávat si vyhodnotit libovolné příkazy, což je znázorněno v druhé ukázce 3.2, kde si necháme kernelem vyhodnotit výraz $\sin(x)^2 + \cos(x)$ v bodě $x = \frac{\pi}{4}$, zpět se nám vrátí odpověď ve formě *C-stringu* (`1/2 + 1/Sqrt[2]`). Po dokončení práce s *Wolframem Mathematica* je potřeba spojení s kernelem uzavřít. Pro správné ukončení spojení je potřeba zavolat funkce `WSClose(link)` a `WSDeinitialize(env)` v tomto pořadí.

Zdrojový kód 3.1: Ukázka inicializace spojení s kernelem *Wolframu Mathematica* v jazyce *C++*

```
const char * pathToExec = argv[0]; // cesta ke spouštěnému programu
const char * pathToKernel = "/Applications/Mathematica.app/Contents/MacOS/WolframKernel";
const int NumberOfKernelParams = 5;
const char ** KernelParams = {pathToExec, "-linkmode", "
```

3.1. Spojení programu s kernelem *Wolfram Mathematica* pomocí *WSTP*

```
    launch", "-linkname", pathToKernel };
int  errorNumber = 0;
WSENV env = WSInitialize(0);
WSLINK link = WSOpenArgcArgv(env, NumberOfKernelParams,
    KernelParams, &errorNumber);

if ( link == (WSLINK)0 || errorNumber != WSEOK ||
    WSActivate(link) == 0)
{
    // nastala chyba, její kód je v proměnné
    errorNumber
    exit(0);
}
```

Zdrojový kód 3.2: Ukázka volání funkcí *Wolfram Mathematica* přes *WSTP* v jazyce *C++*

```
WSLINK link; // vytvořené spojení v předchozí ukázce
const char * exprToEval = "Sin[x]^2+Cos[x]/.x->Pi/4";
WSPutFunction(link, "ToString", 2);
WSPutFunction(link, "FullSimplify", 1);
WSPutFunction(link, "ToExpression", 1);
WSPutString(link, exprToEval);
WSPutSymbol(link, "InputForm");
WSEndPacket(link);
// u všech těchto volání funkcí bychom měli kontrolovat
// jejich návratový kód, pokud je 0, tak nastala chyba a
// její číslo můžeme získat z funkce WSError(link).
while ((packet = WSNextPacket(link)) && packet !=
    RETURNPKT)
{
    WSNewPacket(link);
}
if (WSError(link) != WSEOK)
{
    // nastala chyba
```

```
}  
char * tmpResult;  
WSGetString(link, &tmpResult);  
//po dokončení práce s výsledkem je nutné uvolnit paměť  
// pro něj alokovanou  
WSReleaseString(link, tmpResult);  
// ukončení spojení  
WSClose(link);  
WSDeinitialize(env);
```

Třída realizující připojení a komunikaci s kernelem *Wolframu Mathematica* v našem programu je definována v souboru `KernelConnection.hpp`. Při inicializaci se tato třída připojí ke kernelu způsobem popsáným výše. Třída se stará o testování správnosti syntaxe načtených koeficientů a exponentů prvků (testuje syntaxi jazyka *Wolfram Language*), provádí jejich zjednodušování a dále umožňuje převádět tyto koeficienty a exponenty do formátu *LaTeX*.

3.2 Popis tříd programu a konfiguračního souboru

V této sekci popíšeme třídy programu a jejich význam, tento popis může sloužit lidem, kteří by chtěli tento program nějak rozšířit, či se jím inspirovat při tvorbě nějakého jiného programu. Na konci této sekce uvedeme *UML* diagram tříd programu, tento diagram je k dispozici také v *PDF* na příloženém *CD*.

3.2.1 Konfigurační soubor

Konfigurační soubor tohoto programu je soubor `Constants.h`, v tomto souboru je možné změnit parametry programu. Dvě nejdůležitější proměnné jsou `KERNEL_PATH`, která udává cestu ke kernelu *Wolframu Mathematica*, pokud překládáte program manuálně, je potřeba tuto proměnnou nastavit na adekvátní hodnotu podle Vašeho systému. Druhá proměnná o které by měl vědět běžný uživatel je proměnná `MultiTableFile`, která udává cestu k tabulce násobení prvků na bázi. Předpřipravená tabulka násobení prvků báze superalgebry, definované v 1.6, je v souboru `config/MultiplicationTable.txt` (relativní cesta vzhledem ke kořenovému adresáři tohoto programu). Pokud chcete použít jinou tabulku násobení, musíte do proměnné `MultiTableFile` přiřadit cestu k nové tabulce násobení a program poté znovu zkompileovat.

3.2.2 Syntaktický analyzátor

Třída syntaktického analyzátoru je definovaná v souboru `Parser.hpp` a obsahuje v sobě lexikální analyzátor popsáný v sekci 2.2.3 a definovaný pra-

vidly v souboru `Tokenizer.lex`, soubor definující rozpoznávané tokeny, viz tabulka 2.1, je `TokenTypes.hpp`. Tato třída se stará o práci se vstupem, při spuštění programu o načtení tabulky násobení, syntaktickou analýzu jednotlivých prvků a celých aritmetických výrazů zadaných uživatelem, načítání superidentit a zobrazování případných syntaktických chyb, kterých se uživatel dopustil. Na starost má také řízení vyhodnocování aritmetických výrazů, obsahuje zásobník (oboustranně zřetězený spojový seznam) do kterého vkládá již zanalyzované části výrazu a poté podle postupu uvedeného v sekci 2.2.1 vyhodnocuje tento výraz.

3.2.3 Prvky báze

Třída reprezentující násobky jednotlivých prvků báze superalgebry je definovaná v souboru `Element.hpp` a má na starost uchování těchto prvků v paměti. Násobkem prvku báze myslíme libovolný prvek báze s libovolným koeficientem. V této superalgebře existují 4 typy různých prvků, viz 1.6. U každého prvku si pamatujeme jeho koeficient, t -exponent, exponent v hranatých závorkách, jeho typ a exponent posledního x (v definici báze superalgebry je tento exponent zastupován symbolem σ). Tato třída také obsahuje metodu vypočítávající paritu konkrétního prvku. Dále tato třída implementuje zjednodušující pravidla uvedená v 1.6. Posledním účelem této třídy je výpis konkrétního prvku ve formátu *Wolfram Mathematica* nebo *LaTeX*.

3.2.4 Tabulka násobení

Třída reprezentující tabulku násobení prvků bázi, která byla uvedena v 1.6, je definována v souboru `Multiplication.hpp`. Tabulka je reprezentována jako vektor pravidel, kde se každé pravidlo skládá z uspořádané dvojice prvků báze a pravé strany pravidla, která je vektorem násobků prvků báze superalgebry (pravá strana je lineární kombinací prvků báze).

3.2.5 Kontrola nad programem

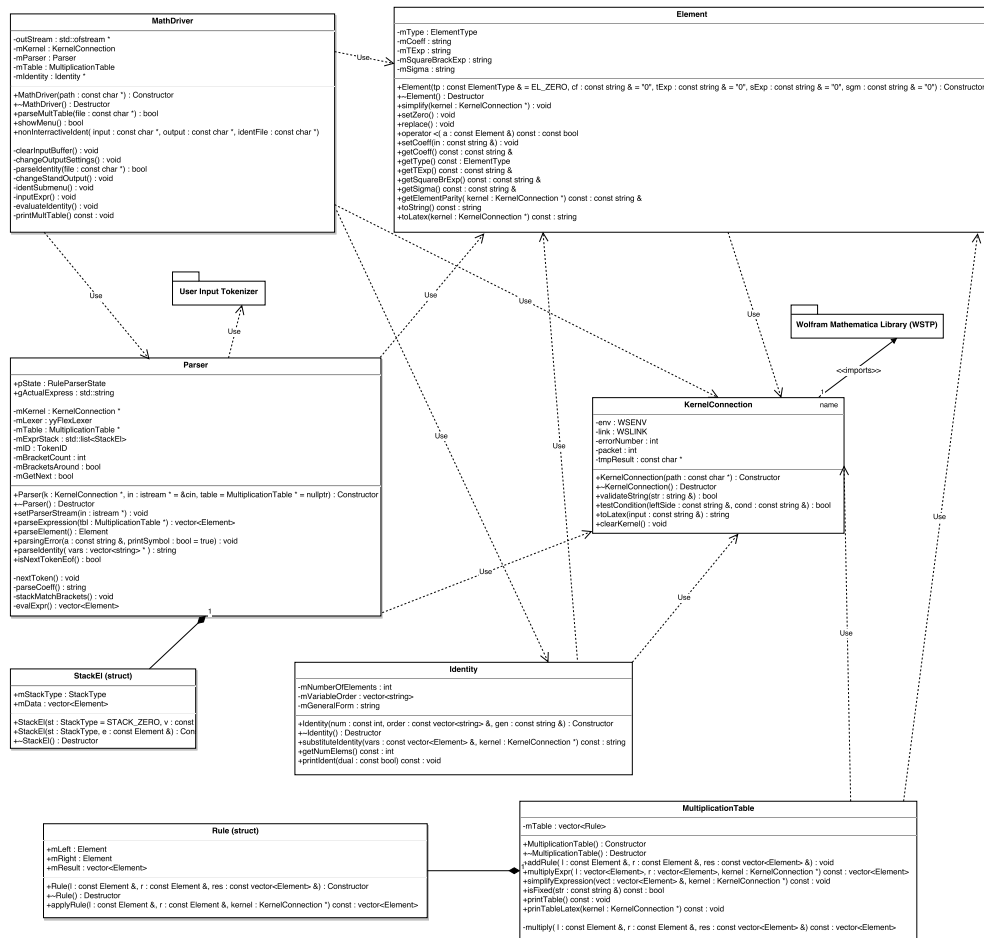
Řízení celého programu má na starost třída definovaná v souboru `MathDriver.hpp`, která inicializuje všechny komponenty programu po jeho spuštění a po úspěšné inicializaci zobrazí uživateli menu. V tomto menu mu umožní nastavit program podle jeho představ (formát výstupu, výstup do souboru, debugovací režim) a dále mu umožní vypsát tabulku násobení prvků na bázi superalgebry, zadat libovolný aritmetický výraz podle syntaxe uvedené v 2.2.3, který bude poté vyhodnocen a výsledek bude zobrazen uživateli. Dále umožňuje uživateli načíst nějakou superidentitu ze souboru formátu 2.2.3 a poté tuto superidentitu vyhodnotit pro uživatelem zadané prvky superalgebry. V programu může být najednou načtena maximálně jedna superidentita, uložení této superidentity má na starost třída definovaná v souboru `Identity.hpp`.

3. IMPLEMENTACE VLASTNÍHO NÁVRHU

Na konec této sekce přidáváme UML diagram těchto tříd, diagram je poněkud velký, pokud byste ho chtěli podrobně studovat, je k dispozici také jako samostatný *PDF* dokument na přiloženém *CD*.

3.2. Popis tříd programu a konfiguračního souboru

3.2.6 UML diagram tříd programu



3.3 Instalace programu

Program pro svůj úspěšný překlad a běh potřebuje *Wolfram Mathematica* ve verzi 10.0 a novější, kompilátor jazyka *C++* (funkčnost ověřena pro *GCC* a *clang*) s podporou standardu *C++11*. V systému *OS X* knihovny (*libc++*, *framework Foundation*), v linuxových systémech knihovny (*libm*, *libpthread*, *librt*, *libdl*, *libstdc++*, *libuuid*), některé knihovny mohou mít v různých distribucích *Linuxu* rozdílná jména. Pokud splňujete všechny výše uvedené požadavky, můžete pro vytvoření Makefile spustit *bashovský* skript `install.sh`, který se pokusí najít správné cesty ke knihovnám a kernelu programu *Wolfram Mathematica* a vytvoří `Makefile` pomocí kterého lze program snadno přeložit. Skript v rámci testu zkusí také spustit kernel *Wolfram Mathematica*, tento test slouží k ověření správnosti cesty ke kernelu a k zajištění, že máte program *Wolfram Mathematica* aktivovaný. Pokud se Vám na příkazové řídce objeví žádost „*In[1]:=*“, tak je vše v pořádku a můžete kernel ukončit příkazem „*Quit[]*“ a stisknutím klávesy `ENTER`.

Po úspěšném dokončení skriptu `install.sh` můžete program nainstalovat příkazem `make all`, který vygeneruje pomocí nástroje *lex* lexikální analyzátor a poté zkompileje celý program. Na systémech *Linux* navíc vytvoří spouštěcí skript, protože v některých linuxových systémech se při instalaci programu *Wolfram Mathematica* správně neexportují sdílené knihovny, toto lze řešit úpravou proměnné prostředí `LD_LIBRARY_PATH` a nebo spouštěním programu skriptem `run.sh`, který toto vyřeší za Vás. Pokud z nějakého důvodu nemůžete zkompileovat program pomocí vytvořeného *makefile* a nebo se Vám *makefile* vůbec nevytvořil, můžete zkompileovat program manuálně. Podívejte se do *makefile* a zjistěte, s jakým příkazem je problém, a podle chybové hlášky tento příkaz adekvátně modifikujte. V případě, že se Vám *makefile* vůbec nevytvořil, můžete využít soubor `Makefile_default`, který obsahuje okomentované příkazy nutné pro přeložení našeho programu na systému *Mac OS X*, na systému *Linux* je potřeba změnit názvy knihoven dle úvodního odstavce této sekce.

Program jsem bez problémů nainstaloval na systémech *OS X El Capitan* a *Ubuntu 16.04.2*, ale neměl by být problém tento program nainstalovat i na systému *Windows*. Pro případné zájemce přikládám odkaz na oficiální vývojářskou příručku pro systém *Windows*: <https://reference.wolfram.com/language/tutorial/WSTPDeveloperGuide-Windows.html#14115>, která obsahuje návod v anglickém jazyce pro zkompileování a spuštění programu využívajícího *WSTP* na systému *Windows*.

3.4 Spuštění a ovládání programu

Program lze spustit přímo příkazem `./AlgebraTool`, pokud nemáte v proměnné prostředí `LD_LIBRARY_PATH` přidanou cestu ke knihovně `libWSTP`, využijte ke spuštění skript `run.sh`, který správně nastaví cestu k požadované knihovně a spustí program. Po spuštění programu se načte tabulka násobení prvků báze a naváže se spojení s kernelem *Wolframu Mathematica* a pokud vše proběhne v pořádku, zobrazí se Vám v terminálu hlavní menu programu. Pro zvolení nějaké nabízené možnosti zadejte číslo Vaší volby a stiskněte klávesu `ENTER`. Například volba číslo 1 Vám umožní zadat nějaký aritmetický výraz v syntaxi uvedené v 2.2.3, který program poté vyhodnotí a zobrazí Vám jeho výsledek. Volba 2 zobrazí menu pro práci se superidentitami, kde můžete načíst superidentitu ze souboru formátu uvedeného v 2.2.3 a poté tuto superidentitu nechat vyhodnotit pro konkrétní prvky superalgebry. Ovládání programu je vcelku intuitivní, a proto se mu zde nebudu více věnovat, podrobnější informace jsou k nalezení v uživatelské příručce v příloze A této práce.

V následující kapitole zmíníme, jak byl program testován a jaké chyby byly naším programem odhaleny ve výpočtech publikovaných v článcích [2], [3] a disertační práci [1].

Testování programu a získané výsledky

V této kapitole nejdříve zmíníme, jak byl náš program testován. Největší důraz byl při testování kladen na správnost vyhodnocování aritmetických výrazů a nil_n superidentit v superalgebře popsané v 1.6. V druhé části této kapitoly zveřejníme naším programem nalezené chyby v článcích [2], [3] a disertační práci [1].

4.1 Testování programu

Nejdříve byl otestován návrh programu, zda-li podporuje všechny požadované funkcionality. Poté jsme provedli unit testy jednotlivých komponent programu. Spolu s naší vedoucí jsme ověřili správnost načítání tabulky násobení prvků na bázi. Vytvořili jsme soubor ve formátu 2.2.3, který obsahoval celou tabulku násobení uvedenou v sekci 1.6, poté jsme tento soubor nechali programem načíst a vypsat ve formátu *LaTeX* a několikrát jsme manuálně ověřili jeho korektnost. Tím bylo zároveň otestováno i načítání jednotlivých prvků a výpis ve formátu *LaTeX*. Stejným způsobem bylo ověřeno načítání superidentit ze souboru. Nejnáročnější test spočíval v ověření správnosti vyhodnocování aritmetických výrazů, jelikož jsme nemohli použít výpočty uvedené ve výše zmíněných článcích, nejdříve jsme ověřili správnost násobení na prvcích báze, čímž bylo otestováno použití správného pravidla z tabulky násobení pro jakoukoliv dvojici prvků báze superalgebry, a poté jsme ještě manuálně vypočítali přibližně 25 výrazů a ověřili, že se shodují s výsledky, které vypočítal náš program. V závěru jsme otestovali vyhodnocování superidentit a správné dosazování do nich a pomocí nástroje *Apple Instruments* byla otestována práce s pamětí a její správné uvolňování. Program všemi výše uvedenými testy pro-

šel bez problémů.

Jediná odchylka od očekávaného chování programu byla objevena při kontrole tabulky násobení. V sekci popisující tabulku násobení superalgebry je definována funkce $c(j)$ jako:

$$c(j) = \frac{j(j-1)}{2}, \quad j \in \mathbb{Z}.$$

Při načítání tabulky násobení náš program funkci $c(j)$ nahrazuje jejím předpisem, v některých pravidlech se vyskytují koeficienty tohoto tvaru: $(-1)^{c(j)}$, které náš program interpretuje jako: $i^{j(j-1)}$. Při výpisu tabulky násobení to může působit jako chyba, protože ve vstupním souboru tabulky násobení žádná imaginární jednotka i nikde není použita. Chyba to ale není, jelikož platí:

$$(-1)^{c(j)} = (-1)^{\frac{1}{2}j(j-1)} = \left(\sqrt{(-1)}\right)^{j(j-1)} = i^{j(j-1)}.$$

4.2 Nalezené chyby v publikovaných člancích

Naším programem jsme ověřili výpočty prováděné při konstrukci báze alternativní nil_n superalgebry s jedním lichým generátorem, tato báze byla zmíněna na konci sekce 1.6.2. V této části bude $\mathcal{B}_n = \text{Alt-}\mathcal{N}il[\emptyset; x]$ značit alternativní nil_n superalgebru s jedním lichým generátorem x . \mathcal{I}_n bude značit ideál \mathcal{A} generovaný prvky $W_n(u_1, u_2, \dots, u_n)$, $u_1, u_2, \dots, u_n \in \mathcal{A}_0 \cup \mathcal{A}_1$.

V této sekci poukážeme na chyby nalezené v člancích [2], [3] a disertační práci [1]. Výpočty uvedené v člancích [2] a [3] jsou také uvedeny v odpovídajících kapitolách disertační práce, proto všechny chyby v těchto člancích jsou i v disertační práci. Dále v této sekci budeme zmiňovat pouze chyby v disertační práci. Neupravené soubory s výsledky vypočítanými naším programem jsou k dispozici na *CD* v adresáři **results**.

V části zabývající se konstrukcí báze \mathcal{B}_2 jsme neobjevili žádnou chybu a všechny výsledky uvedené v této části disertační práce se shodovaly s výsledky, kterých jsme dosáhli použitím našeho programu.

V části zabývající se konstrukcí báze \mathcal{B}_3 jsme objevili následující chyby. Na straně 43 jsou uvedeny dvě rovnice:

$$\begin{aligned} (t^2x - tx^{[3]}) \cdot x &= \frac{1}{2}t^3 - t(x^{[3]}x) + \frac{1}{3}tx^{[4]} - \frac{1}{6}z^{[4]}, \\ x \cdot (t^2x - tx^{[3]}) &= \frac{1}{2}t^3 - t(x^{[3]}x) + \frac{2}{3}tx^{[4]}. \end{aligned}$$

Pravé strany těchto rovnic jsou prohozené, správně by to mělo být následovně:

$$\begin{aligned}(t^2x - tx^{[3]}) \cdot x &= \frac{1}{2}t^3 - t(x^{[3]}x) + \frac{2}{3}tx^{[4]}, \\ x \cdot (t^2x - tx^{[3]}) &= \frac{1}{2}t^3 - t(x^{[3]}x) + \frac{1}{3}tx^{[4]} - \frac{1}{6}z^{[4]},\end{aligned}$$

prohození pravých stran těchto dvou rovnic se nám v tomto případě nezmění, jaké prvky budou náležet ideálu \mathcal{I}_3 . A tudíž tyto chyby nemají vliv na nalezenou bázi \mathcal{B}_3 .

Na straně 44 je uveden výraz:

$$W_3(x, t, u^{[k]}) = 6(-1)^{k+1}tu^{[k]}x + 2(-1)^k t^2 z^{[k]},$$

podle výsledků našeho programu je zde špatný koeficient u druhého členu, správně by to mělo být následovně:

$$W_3(x, t, u^{[k]}) = 6(-1)^{k+1}tu^{[k]}x + 6(-1)^k t^2 z^{[k]},$$

chyba je pouze v koeficientu, a tudíž tato chyba také neovlivní, jaké prvky budou náležet do ideálu \mathcal{I}_3 .

Žádná z nalezených chyb nemá vliv na bázi \mathcal{B}_3 , a tudíž konstatujeme, že báze popsaná v disertační práci je skutečně bázi \mathcal{B}_3 .

Náš program bohužel neumí vyhodnotit nil_n superidentitu pro obecné n , a proto jsme při ověřování báze \mathcal{B}_n museli vždy zafixovat n a porovnat výsledky výpočtů v disertační práci pro dané n s výsledky, které vypočítal náš program. Z časových důvodů jsme to ověřili pouze pro $n = \{4, 5, 6, 7\}$ a zde uvádíme nalezené nesrovnalosti ve výsledcích.

Na straně 52 jsou uvedeny dva výrazy:

$$\begin{aligned}W_n(x^{[k]}, x, t, \dots, t) \cdot x &= \frac{2}{3}t^{n-2}x^{[k+2]} - t^{n-2}(x^{[k+1]}x) \\ &\quad + (n-2)t^{n-3}\left(\frac{1}{3}u^{[k+1]} - \frac{1}{2}u^{[k]}x - \frac{1}{2}z^{[k+1]}x\right), \\ x \cdot W_n(x^{[k]}, x, t, \dots, t) &= (-1)^k \left(t^{n-2}\left(\frac{1}{3}x^{[k+2]} - x^{[k+1]}x\right) \right. \\ &\quad \left. - (n-2)t^{n-3}\left(\frac{1}{2}u^{[k]}x - \frac{1}{6}u^{[k+1]} + \frac{1}{2}z^{[k+1]}x + \frac{1}{6}z^{[k+2]}\right) \right).\end{aligned}$$

Oba výrazy mají jen špatné koeficienty, takže tato chyba neovlivní nalezenou bázi \mathcal{B}_n . Správně by obě pravé strany těchto výrazů měly být ještě vynásobeny koeficientem $\frac{n!}{2}$.

Ostatní výpočty uvedené v této části disertační práce se pro $n = \{4, 5, 6, 7\}$ shodují s výsledky, které vypočítal náš program. A tudíž potvrzujeme výsledky uvedené v disertační práci.

Závěr

Jedním z cílů této práce bylo seznámit se s matematickým pozadím neasociativních matematických struktur (algeber a superalgeber) a aplikacemi, které s těmito neasociativními strukturami pracují. Tento cíl byl splněn a jeho splnění je popsáno v kapitolách 1 a 2.1. Hlavním cílem této práce bylo navrhnout a vytvořit aplikaci pracující s prvky alternativní nil_n superalgebry s jedním lichým generátorem, což bylo splněno v kapitolách 2.2 a 3. Aplikace byla skutečně vytvořena a otestována, že počítá správně. Tato aplikace byla poté použita k ověření výsledků z článků [2], [3] a disertační práce [1], výsledky tohoto ověření byly uveřejněny v kapitole 4.

Výstupem této práce je tedy nejen vyvinutá aplikace pro práci s prvky alternativní nil_n -superalgebry, ale také ověření již dříve publikovaných výsledků. V neposlední řadě pak také vytvoření návodu v českém jazyce (společně s ukázkami kódu) pro připojení externího programu ke kernelu *Wolfram Mathematica*, zveřejněného v sekci 3.1.

Vyvinutý program je specializován pouze na práci s prvky jedné konkrétní alternativní superalgebry, avšak v budoucnu by mohl být rozšířen na obecný nástroj pro práci s neasociativními strukturami do převeden do podoby modulu *Wolfram Mathematica*. Jedná se však o netriviální a časově náročnou úpravu, jež je nad rámec této práce, ale její realizace by mohla být cílem nějaké jiné bakalářské či diplomové práce.

Uživatelská příručka

A.1 Instalace

Program pro svůj úspěšný překlad a běh potřebuje *Wolfram Mathematicu* ve verzi 10.0 a novější, kompilátor jazyka *C++* (funkčnost ověřena pro *GCC* a *clang*) s podporou standardu *C++11*. V systému *OS X* knihovny (*libc++*, *framework Foundation*), v linuxových systémech knihovny (*libm*, *libpthread*, *librt*, *libdl*, *libstdc++*, *libuuid*), některé knihovny mohou mít v různých distribucích *Linuxu* rozdílná jména. Pokud splňujete všechny výše uvedené požadavky, můžete pro vytvoření Makefile spustit *bashovský* skript `install.sh`, který se pokusí najít správné cesty ke knihovnám a kernelu programu *Wolfram Mathematica* a vytvoří `Makefile` pomocí kterého lze program snadno přeložit. Skript v rámci testu zkusí také spustit kernel *Wolfram Mathematica*, tento test slouží k ověření správnosti cesty ke kernelu a k zajištění, že máte program *Wolfram Mathematica* aktivovaný. Pokud se Vám na příkazové řídce objeví žádost „*In[1]:=*“, tak je vše v pořádku a můžete kernel ukončit příkazem „*Quit[]*“ a stisknutím klávesy ENTER.

Po úspěšném dokončení skriptu `install.sh` můžete program nainstalovat příkazem `make all`, který vygeneruje pomocí nástroje *lex* lexikální analyzátor a poté zkompileje celý program. Na systémech *Linux* navíc vytvoří spouštěcí skript, protože v některých linuxových systémech se při instalaci programu *Wolfram Mathematica* správně neexportují sdílené knihovny, toto lze řešit úpravou proměnné prostředí `LD_LIBRARY_PATH` a nebo spouštěním programu skriptem `run.sh`, který toto vyřeší za Vás. Pokud z nějakého důvodu nemůžete zkompileovat program pomocí vytvořeného *makefile* a nebo se Vám *makefile* vůbec nevytvořil, můžete zkompileovat program manuálně. Podívejte se do *makefile* a zjistěte, s jakým příkazem je problém, a podle chybové hlášky tento příkaz adekvátně modifikujte. V případě, že se Vám *makefile* vůbec nevytvořil, můžete využít soubor `Makefile_default`, který obsahuje okomentované příkazy nutné pro přeložení našeho programu na systému *Mac OS X*, na systému

Linux je potřeba změnit názvy knihoven dle úvodního odstavce této sekce.

A.2 Ovládání programu

Program lze spustit buď skriptem (pouze na systémech Linux) `run.sh` a nebo přímo příkazem `./AlgebraTool`. Program se standardně spouští v interaktivním módu s vypnutými debugovacími výpisy. Debugovací mód lze zapnout v menu aplikace, ale pokud potřebujete debuggovat načítání tabulky násobení prvků báze, lze tento mód zapnout přepínačem `-v`, tedy spustit program příkazem `./AlgebraTool -v`.

Aplikace také obsahuje neinteraktivní mód pro hromadné výpočty superidentit. Pro spuštění programu v neinteraktivním módu použijte přepínač `-ni`, tedy použijte příkaz: `./AlgebraTool -ni input output identity`, kde `identity` je cesta k souboru definujícímu superidentitu, kterou chcete vyhodnotit, `input` je soubor udávající pro jaké prvky chcete tuto identitu vyhodnotit. Prvky ve vstupním souboru oddělujte čárkou, program vždy vezme n prvků, kde n je počet argumentů zvolené identity, a vyhodnotí tuto identitu pro tyto prvky. Toto dělá dokud jsou ve vstupním souboru prvky, a nebo nastane nějaká chyba (např. špatná syntaxe apod.). Výsledky výpočtů se zapisují ve formátu *LaTeX* do souboru `output`. Tento mód lze tedy snadno využít pro výpočet velkého množství superidentit bez nutnosti zadávat je manuálně po jedné do programu.

A.2.1 Menu

Struktura menu programu je následující:

1.....	Vyhodnocení libovolného výrazu v superalgebře
2.....	Práce se superidentitami (podmenu)
1.....	Vyhodnocení aktuální superidentity pro zadané prvky
2.....	Vypsání aktuální superidentity
3.....	Načtení nové superidentity ze souboru
4.....	Návrat do hlavního menu
3.....	Nastavení chování programu (podmenu)
1.....	Zapnutí/Vypnutí detailního výpisu prováděných operací
2.....	Zapnutí/Vypnutí debugovacího módu
3.....	Zapnutí/Vypnutí ukládání výsledků výpočtů do souboru
4.....	Změna formátu výstupu(<i>Wolfram Mathematica/LaTeX</i>)
5.....	Návrat do hlavního menu
4.....	Výpis tabulky násobení ve zvoleném formátu
5.....	Ukončení programu

A.3 Příklady použití

Tato sekce slouží spíše pro ilustraci, jak může vypadat práce s tímto programem.

Vyhodnocování aritmetických výrazů

Chcete vyhodnotit nějaký aritmetický výraz. Návod: spusťte program a v hlavním menu vyberte možnost číslo 1. Zadejte aritmetický výraz, který chcete vyhodnotit. Například chcete vyhodnotit výraz:

$$25t^8x \cdot x^{[9]} + 17t^2u^{[3]} \cdot (t \cdot x + x \cdot t).$$

Zadejte do terminálu tento výraz v určeném formátu (každý příkaz je ukončen čárkou):

`25*t^(8)x**x^[9] + 17*t^(2)u^[3]**(t**x + x**t),`

a odešlete příkaz klávesou *ENTER*. Program Vám v okamžiku vrátí odpověď:

$$\begin{aligned} (25) * t^{(8)}x^{**x}^{[9]} + (17) * t^{(2)}u^{[3]**(t**x + x**t)} \\ = (25) * t^{(8)}x^{[10]} + (-25) * t^{(8)}(x^{[9]}x). \end{aligned}$$

Chcete vyhodnotit jiný výraz:

$$(((x \cdot x) \cdot x) \cdot x) \cdot x.$$

Do terminálu Vám ale stačí zadat: `x**x**x**x**x`, a program tento výraz vyhodnotí v přirozeném levém uzávorkování a v okamžiku Vám vrátí odpověď:

$$x**x**x**x**x = (1/4) * t^{(2)}x + (1/6) * (x^{[4]}x).$$

Chcete vyhodnotit stejný výraz, ale v přirozeném pravém uzávorkování:

$$x \cdot (x \cdot (x \cdot (x \cdot x))).$$

Můžete buď vypsát všechny závorky a nebo si usnadnit práci a použít jenom jeden pár závorek. Zadejte výraz takto `(x**x**x**x**x)` a program ho vyhodnotí v přirozeném pravém uzávorkování a v okamžiku Vám vrátí odpověď:

$$\begin{aligned} (x**x**x**x**x) &= (1/4) * t^{(2)}x + (-1/6) * (x^{[4]}x) \\ &+ (1/6) * x^{[5]} + (-1/2) * tx^{[3]}. \end{aligned}$$

Vypsání tabulky násobení v *LaTeXu*

Chcete vypsát tabulku násobení ve formátu *LaTeX*. Návod: spusťte program, v hlavním menu zvolte možnost číslo 3 a v podmenu nastavení výstupu zvolte možnost číslo 4. Nyní je veškerý výstup programu ve formátu *LaTeX*. Poté se vraťte do hlavního menu zvolením možnosti číslo 5. V hlavním menu zvolte možnost číslo 4 a program vypíše tabulku násobení jako *LaTeX* dokument, který lze hned přeložit.

Výpis výsledků do souboru

Chcete si ukládat vypočítané výsledky do souboru. Návod: spusťte program a přejděte do podmenu nastavení výstupu (volba číslo 3 v hlavním menu), zde zvolte možnost číslo 3 a zadejte jméno souboru, kam chcete ukládat výsledky. Od této chvíle až do vypnutí přesměrování do souboru budou všechny zobrazené výsledky zapsány do souboru, který jste si zvolili. Výsledky budou ale také stále zobrazovány v terminálu.

Vyhodnocení nil_n superidentitu

Chcete vyhodnotit nil_n superidentitu pro nějaké konkrétní n . Návod: spusťte program a jděte do podmenu pro práci se superidentitami (možnost číslo 2). Zde zvolte možnost číslo 3 a zadejte jméno souboru obsahující předpis konkrétní superidentitu. Například `config/nil4.txt`, program načte předpis pro nil_4 superidentitu. Nyní můžete zvolit možnost číslo 1 a nechat si vyhodnotit tuto superidentitu pro nějaké prvky. Například zadáte 4 prvky: t, t, tx, t , a odešlete klávesou *ENTER*. Program Vám po chvíli zobrazí odpověď:

$$W(t, t, tx, t) = (24) * t^{(4)}x + (-36) * t^{(3)}x^{[3]}.$$

Můžete si také aktuálně načtenou superidentitu nechat vypsát zvolením možnosti číslo 2. Pokud chcete změnit superidentitu s kterou pracujete, zvolte možnost číslo 3 a program načte novou superidentitu. Program pracuje s maximálně jednou superidentitou současně.

Zapnutí debugovacího módu

Máte problém se syntaxí nějakého výrazu a chcete zjistit, kde je chyba. Řešením je zapnout debugovací mód a zkusit zadat problematický výraz. Návod: spusťte program a jděte do podmenu nastavení programu (volba číslo 3 v hlavním menu). V podmenu zvolte možnost číslo 3 a program bude od této doby zobrazovat informace z parseru o tom, jaké tokeny zpracovává a v jakých stavech se při tom nachází. Vraťte se do hlavního menu (volba číslo 5) a zadejte problematický výraz. Parser bude zobrazovat, jak zpracovává váš vstup a až narazí na chybu, tak přestane zpracovávat vstup a vrátí se do hlavního menu. Vy pak můžete zjistit jaký symbol parser neočekával a podle toho adekvátně opravit Váš výraz.

Seznam použitých zkratek

WSTP Wolfram Symbolic Transfer Protocol

GCC The GNU Compiler Collection

clang C language family frontend for LLVM

Flex The Fast Lexical Analyzer

UML The Unified Modeling Language

ACM Association for Computing Machinery

SIGSAM Special Interest Group in Symbolic and Algebraic Manipulation

JIT just-in-time (compilation)

PDF Portable Document Format

Obsah přiloženého CD

readme.txt.....	stručný popis obsahu CD
exe	adresář se spustitelnou formou implementace
└─ AlgebraTool.....	přeložený program pro systém Mac OS X
src	
└─ impl.....	zdrojové kódy implementace
└─ config..	adresář s tabulkou násobení a definicemi n superidentit
└─ install.sh..	bashovský skript pro automatické vytvoření Makefile
└─ Makefile_default...	vzorový Makefile pro Mac OS X s komentáři
└─ thesis	zdrojová forma práce ve formátu \LaTeX
text	text práce
└─ thesis.pdf	text práce v PDF
results.....	adresář s vypočítanými výsledky naším programem
other	
└─ class_diag.pdf	UML diagram tříd programu v PDF
└─ user_guide.pdf	Uživatelská příručka v PDF

Literatura

- [1] SCHOLTZOVÁ, J. THE SUPERALGEBRA APPROACH TO STUDY OF SUBSPACES OF SKEW-SYMMETRIC ELEMENTS IN FREE ALGEBRAS [online]. Praha, 2013 [cit. 2017-01-22]. Dostupné z: <https://dspace.cvut.cz/handle/10467/15441>. Disertační práce. České vysoké učení technické v Praze. Fakulta elektrotechnická. Katedra matematiky. Vedoucí práce Natalia Žukovec.
- [2] SCHOLTZOVÁ, J. a N. ZHUKAVETS. The free alternative nil-superalgebra of index 3 on one odd generator. Journal of Algebra and Its Applications. 2010, 2(Vol. 9), 209-222. DOI: <http://dx.doi.org/10.1142/S0219498810003847>
- [3] ZHUKAVETS, N. The free alternative nil-superalgebra of index n on one odd generator. Communications in Algebra. 2014, 1(Vol. 42), 96-107. DOI: <http://dx.doi.org/10.1080/00927872.2012.706844>
- [4] DOMBEK, D. a K. KLOUDA. Lineární algebra: Studijní text [online]. 2016 [cit. 2017-01-22]. Dostupné z: <https://edux.fit.cvut.cz/courses/BI-LIN/lectures>
- [5] OLŠÁK, P. Úvod do algebry, zejména lineární. 2., přeprac. vyd. V Praze: České vysoké učení technické, 2013. ISBN 978-80-01-05291-4.
- [6] KLOUDA, K. Handouty k přednáškám předmětu MI-MPI [online]. [cit. 2017-01-22]. Dostupné z: <https://edux.fit.cvut.cz/courses/MI-MPI/lectures>
- [7] WEISSTEIN, E. W. Field Characteristic: From MathWorld-A Wolfram Web Resource. [online]. [cit. 2017-01-22]. Dostupné z: <http://mathworld.wolfram.com/FieldCharacteristic.html>
- [8] HALAS, Z. Okruhy [online]. [cit. 2017-03-20]. Dostupné z: <http://www.karlin.mff.cuni.cz/~halas/BcSemin/Faktorokruhy.pdf>

- [9] BUCHMANN, A. A Brief History of Quaternions and the Theory of Holomorphic Functions of Quaternionic Variables [online]. Orange, CA 92866 USA: Department of Mathematics and Computer Sciences Schmid College of Science Chapman University [cit. 2017-01-23]. Dostupné z: http://www.maa.org/sites/default/files/images/upload_library/46/HOMSIGMAA/Buchmann.pdf
- [10] BAEZ, J. Octonions: Introduction [online]. 2001 [cit. 2017-01-23]. Dostupné z: <http://math.ucr.edu/home/baez/octonions/node1.html>
- [11] WEISSTEIN, E. W. Octonion: From MathWorld-A Wolfram Web Resource. [online]. [cit. 2017-03-07]. Dostupné z: <http://mathworld.wolfram.com/Octonion.html>
- [12] ZHUKAVETS, N. M. a I. P. SHESTAKOV. A base of the free alternative superalgebra on one odd generator. Doklady Mathematics. 2008, 78(2), 693-695. DOI: 10.1134/S106456240805013X. ISSN 1064-5624. Dostupné také z: <http://link.springer.com/10.1134/S106456240805013X>
- [13] ACM SPECIAL INTEREST GROUP IN SYMBOLIC AND ALGEBRAIC MANIPULATION. Computer Algebra Software [online]. 2015 [cit. 2017-05-04]. Dostupné z: <https://www.sigsam.org/Resources/Software.html>
- [14] JACOBS, D. P., S. V. MUDDANA a A. J. OFFUTT. A Computer Algebra System for Nonassociative Identities [online]. 1990 [cit. 2017-05-04]. Dostupné z: <https://people.cs.clemson.edu/~dpj/albertstuff/albertpapers/5thH.pdf>
- [15] O'CONNOR, J.J. a E.F. ROBERTSON. Abraham Adrian Albert [online]. 2001 [cit. 2017-05-04]. Dostupné z: http://www-groups.dcs.st-and.ac.uk/history/Biographies/Albert_Abraham.html
- [16] WOLFRAM RESEARCH. Wolfram Mathematica: Modern Technical Computing [online]. [cit. 2017-05-06]. Dostupné z: <https://www.wolfram.com/mathematica/>
- [17] WOLFRAM RESEARCH. NonCommutativeMultiply. Wolfram Language Documentation [online]. [cit. 2016-12-11]. Dostupné z: <http://reference.wolfram.com/language/ref/NonCommutativeMultiply.html>
- [18] WOLFRAM RESEARCH. TeXForm [online]. [cit. 2017-05-10]. Dostupné z: <https://reference.wolfram.com/language/ref/TeXForm.html>
- [19] PARTOW, A. C++ Mathematical Expression Library [online]. [cit. 2017-05-08]. Dostupné z: <http://partow.net/programming/exprtk/index.html>

- [20] WOLFRAM RESEARCH. Wolfram LibraryLink [online]. [cit. 2017-05-08]. Dostupné z: <https://reference.wolfram.com/language/guide/LibraryLink.html>
- [21] WOLFRAM RESEARCH. WSTP (Wolfram Symbolic Transfer Protocol) [online]. [cit. 2017-05-08]. Dostupné z: <https://www.wolfram.com/wstp/>
- [22] THE FLEX PROJECT. Lexical Analysis With Flex, for Flex 2.6.2 [online]. [cit. 2017-05-11]. Dostupné z: <http://westes.github.io/flex/manual/>
- [23] WOLFRAM RESEARCH. WSTP Development in C (Mac OS X) [online]. [cit. 2017-05-13]. Dostupné z: <https://reference.wolfram.com/language/tutorial/WSTPDeveloperGuide-Macintosh.html>