ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta jaderná a fyzikálně inženýrská

Katedra matematiky

VÝZKUMNÝ ÚKOL

Konstrukce algoritmů pro paralelní sčítání

Construction of algorithms for parallel addition

Vypracoval: Jan Legerský

Školitel: Ing. Štěpán Starosta, Ph.D.

Akademický rok: 2014/2015

Čestné prohlášení

Prohlašuji na tomto místě, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze dne 2. 9. 2015

Jan Legerský

Poděkování

??

Jan Legerský

Název práce:	**Konstrukce algoritmů pro paralelní sčítání**
Autor:	Jan Legerský
Obor:	Inženýrská informatika
Zaměření:	Matematická informatika
Druh práce:	Výzkumný úkol
Vedoucí práce:	Ing. Štěpán Starosta, Ph.D., KM FIT, ČVUT v Praze
Konzultant:	—
Abstrakt:	ABSTRAKT
Klíčová slova:	Paralelní sčítání, nestandardní numerační systémy.

Title:	**Construction of algorithms for parallel addition**
Author:	Jan Legerský
Abstract:	ABSTRACT
Key words:	Parallel addition, non-standard numeration systems.

Contents

Introduction

DOPSAT PODLE OSTATNIHO

Chapter 1

Preliminaries

CHYBI UVOD, PRECHOZI VYSLEDKY

1.1 Numeration systems

Firstly, we give a general definition of numeration system.

Definition 1. Let $\beta \in \mathbb{C}, |\beta| > 1$ and $\mathcal{A} \subset \mathbb{C}$ be a finite set containing 0. A pair $(\beta, \mathcal{A})$ is called a *positional numeration system* with *base* β and *digit set* $\mathcal{A}$, usually called *alphabet*.

Well-studied are so-called standard numeration systems with an integer base β and an alphabet $\mathcal{A}$ being set of countiguous integers. We restrict ourselves to base β being an algebraic integer and possibly non-integer alphabet $\mathcal{A}$.

Definition 2. Let $(\beta, \mathcal{A})$ be a positional numeration system. We say that a complex number x has a $(\beta, \mathcal{A})$-*representation* if there exist digits $x_n, x_{n-1}, x_{n-2}, \cdots \in \mathcal{A}, n \geq 0$ such that $x = \sum_{j=-\infty}^{n} x_j \beta^j$.

We associate a number x with representation as a bi-infinite string

$$(x)_{\beta,\mathcal{A}} = 0^\omega x_n x_{n-1} \cdots x_1 x_0 \bullet x_{-1} x_{-2} \cdots ,$$

where 0^ω denotes the infinite sequence of zeros. Notice that indices are decreasing from left to right as it is ussual to write the most significant digits first.

Definition 3. Let $(\beta, \mathcal{A})$ be a positional numeration system. The set of all complex numbers with a finite $(\beta, \mathcal{A})$-representation is defined by

$$\mathrm{Fin}_{\mathcal{A}}(\beta) := \left\{ \sum_{j=-m}^{n} x_j \beta^j : n, m \in \mathbb{N}, x_j \in \mathcal{A} \right\} .$$

A representation of $x \in \mathrm{Fin}_{\mathcal{A}}(\beta)$ is

$$(x)_{\beta,\mathcal{A}} = 0^\omega x_n x_{n-1} \cdots x_1 x_0 \bullet x_{-1} x_{-2} \cdots x_{-m} 0^\omega .$$

We omit starting and ending 0^ω when we work with numbers in $\mathrm{Fin}_{\mathcal{A}}(\beta)$ in the following. We remark that existence of an algorithm (standard or parallel) producing a finite $(\beta, \mathcal{A})$-representation of $x + y$ where $x, y \in \mathrm{Fin}_{\mathcal{A}}(\beta)$ implies that the set $\mathrm{Fin}_{\mathcal{A}}(\beta)$ is closed under addition, i.e.,

$$\mathrm{Fin}_{\mathcal{A}}(\beta) + \mathrm{Fin}_{\mathcal{A}}(\beta) \subset \mathrm{Fin}_{\mathcal{A}}(\beta)\,.$$

Designing algorithm for parallel addition requires some redundancy in numeration system.

Definition 4. A numeration system $(\beta, \mathcal{A})$ is called *redundant* if there exists $x \in \mathrm{Fin}_{\mathcal{A}}(\beta)$ which has two different $(\beta, \mathcal{A})$-representations.

Redundant numeration system can enable us to avoid carry propagation in addition. On the other hand, for example comparisons are problematic.

1.2 Parallel addition

A local function which is also often called sliding block code is used to mathematically formalize parallelism.

Definition 5. Let $\mathcal{A}$ and $\mathcal{B}$ be alphabets. A function $\varphi : \mathcal{B}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ is said to be *p-local* if there exist $r, t \in \mathbb{N}_0$ satisfying $p = r + t + 1$ and a function $\phi : \mathcal{B}^p \to \mathcal{A}$ such that, for any $w = (w_j)_{j\in\mathbb{Z}} \in \mathcal{B}^{\mathbb{Z}}$ and its image $z = \varphi(w) = (z_j)_{j\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}}$, we have $z_j = \phi(w_{j+t}, \cdots, w_{j-r})$ for every $j \in \mathbb{Z}$. The parameter t, resp. r, is called *anticipation*, resp. *memory*.

This means that each digit of the image $\varphi(w)$ is computed from p digits of w in a sliding window. Suppose that there is a processor on the each position with access to t input digits on the left and r input digits on the right. Then computation of $\varphi(w)$, where w finite sequence, can be done in constant time independent on the length of w.

Definition 6. Let β be a base and $\mathcal{A}$ and $\mathcal{B}$ two alphabets containing 0. A function $\varphi : \mathcal{B}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ such that

1. for any $w = (w_j)_{j\in\mathbb{Z}} \in \mathcal{B}^{\mathbb{Z}}$ with finitely many non-zero digits, $z = \varphi(w) = (z_j)_{j\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}}$ has only finite number of non-zero digits, and
2. $\sum_{j\in\mathbb{Z}} w_j\beta^j = \sum_{j\in\mathbb{Z}} z_j\beta^j$

is called *digit set conversion* in base β from $\mathcal{B}$ to $\mathcal{A}$. Such a conversion φ is said to be *computable in parallel* if it is p-local function for some $p \in \mathbb{N}$.

In fact, addition on $\mathrm{Fin}_{\mathcal{A}}(\beta)$ can be performed in parallel if there is digit set conversion from $\mathcal{A} + \mathcal{A}$ to $\mathcal{A}$ computable in parallel as we easily output digitwise sum of two $\beta, \mathcal{A})$-representations in parallel.

Now we recall known algorithms for parallel addition in different numeration systems. DODELAT Parallel Addition Introduced by Avizienis in 1961:

ROBERTSON For example:

$$\beta \in \mathbb{N}, \beta \geq 3, \mathcal{A} = \{-a, \ldots, 0, \ldots a\}, b/2 < a \leq b - 1\,.$$

MILENINY A EDITINY ALGORITMY -¿ VETA S VELKOU INTEGEROVOU ABECEDOU

Integer alphabets:

- Base $\beta \in \mathbb{C}, |\beta| > 1$.
- Addition is computable in parallel if and only if β is an algebraic number with no conjugate of modulus 1 [Frougny, Heller, Pelantová, S.].
- Algorithms are known but with large alphabets.

LOWER BOUND VELIKOSTI ABECEDY
ZOBECNENI ABECEDY DO Z[OMEGA]

Definition 7. Let α be a complex number. A set of all polynomials with integer coefficients evaluated in ω is denoted by

$$\mathbb{Z}[\alpha] = \left\{ \sum_{i=0}^{n} a_i \alpha^i : n \in \mathbb{N}, a_i \in \mathbb{Z} \right\}.$$

Notice that $\mathbb{Z}[\alpha]$ is a ring.
Non-integer alphabets:

- Base $\beta \in \mathbb{Z}[\omega] = \left\{ \sum_{j=0}^{d-1} a_j \omega^j \ : a_j \in \mathbb{Z} \right\}$, where $\omega \in \mathbb{C}$ is an algebraic integer of degree d.
- Alphabet $\mathcal{A} \subset \mathbb{Z}[\omega], 0 \in \mathcal{A}$.
- Only few manually found algorithms.

1.3 Isomorphism of $\mathbb{Z}[\omega]$ and $\mathbb{Z}^d$

The goal of this section is to show connection between ring $\mathbb{Z}[\omega]$ and $\mathbb{Z}^d$. This enable us to transform division in $\mathbb{Z}[\omega]$ to search for an integer solution of a linear system.

First we recall notion of companion matrix which we use for defining multiplication in $\mathbb{Z}^d$.

Definition 8. Let ω be an algebraic integer of degree $d \geq 1$ with a minimal polynomial $p(x) = x^d + p_{d-1}x^{d-1} + \cdots + p_1 x + p_0 \in \mathbb{Z}[x]$. A matrix

$$S := \begin{pmatrix} 0 & 0 & \cdots & 0 & -p_0 \\ 1 & 0 & \cdots & 0 & -p_1 \\ 0 & 1 & \cdots & 0 & -p_2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & -p_{d-1} \end{pmatrix} \in \mathbb{Z}^d$$

is called a *companion matrix* of the minimal polynomial of ω.

In what follows, we denote basis vectors of $\mathbb{Z}^d$ by

$$e_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, e_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, e_{d-1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

We remark that 1 in e_i is in the $(i+1)$-st row because the index corresponds to the power of a companion matrix in the following definition.

Definition 9. Let ω be an algebraic integer of degree $d \geq 1$ with a minimal polynomial of p and let S be its companion matrix. We define a mapping $\odot_\omega : \mathbb{Z}^d \times \mathbb{Z}^d \to \mathbb{Z}^d$ by

$$u \odot_\omega v := \left(\sum_{i=0}^{d-1} u_i S^i\right) \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-1} \end{pmatrix} \text{ for all } u = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{pmatrix}, v = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-1} \end{pmatrix} \in \mathbb{Z}^d .$$

and we define powers of $u \in \mathbb{Z}^d$ by

$$\begin{aligned} u^0 &= e_0, \\ u^i &= u^{i-1} \odot_\omega u \text{ for } i \in \mathbb{N} . \end{aligned}$$

We will see later that $\mathbb{Z}^d$ equipped with elementwise addition and multiplication $\odot_\omega$ builds a commutative ring. It will follow from the isomorhpism with $\mathbb{Z}[\omega]$. Let us first recall an important property of a companion matrix – it is a root of its own polynomial.

Lemma 1. *Let ω be an algebraic integer with a minimal polynomial p and let S be its companion matrix. Then*

$$p(S) = \mathbb{O} .$$

Proof. Following the proof in [**?**], we have

$$\begin{aligned} e_0 &= S^0 e_0 , \\ Se_0 = e_1 &= S^1 e_0 , \\ Se_1 = e_2 &= S^2 e_0 , \\ Se_2 = e_3 &= S^3 e_0 , \\ &\vdots \\ Se_{d-2} = e_{d-1} &= S^{d-1} e_0 , \\ Se_{d-1} &= S^d e_0 , \end{aligned}$$

where the middle column is obtained by multiplication and the right one by using the previous row. Also by multiplying and substituting

$$\begin{aligned} S^d e_0 = Se_{d-1} &= -p_0 e_0 - p_1 e_1 - \cdots - p_{d-1} e_{d-1} \\ &= -p_0 S^0 e_0 - p_1 S^1 e_0 - \cdots - p_{d-1} S^{d-1} e_0 \\ &= (-p_0 S^0 - p_1 S^1 - \cdots - p_{d-1} S^{d-1}) e_0 \\ &= (S^d - p(S)) e_0 . \end{aligned}$$

Hence

$$p(S) e_0 = \mathbf{0} .$$

Moreover,

$$p(S) e_k = p(S) S^k e_0 = S^k p(S) e_0 = \mathbf{0}$$

for $k = \{0, 1, \ldots, d-1\}$ which implies the statement. □

The following lemma summarises basic properties of a mapping $\odot_\omega$ – factoring out an integer scalar, the unit element, the distributive law and a weaker form of asociativity.

Lemma 2. *Let ω be an algebraic integer of degree d. Then following statements hold for every $u, v, w \in \mathbb{Z}^d$ and $m \in \mathbb{Z}$:*

(i) $(mu) \odot_\omega v = u \odot_\omega (mv) = m(u \odot_\omega v)$,

(ii) $e_0 \odot_\omega v = v \odot_\omega e_0 = v$,

(iii) $(u \odot_\omega e_1^k) \odot_\omega v = u \odot_\omega (e_1^k \odot_\omega v)$ *for* $k \in \mathbb{N}$,

(iv) $(u + v) \odot_\omega w = u \odot_\omega w + v \odot_\omega w$ *and* $u \odot_\omega (v + w) = u \odot_\omega v + u \odot_\omega w$.

Proof. It is easy to see (i) as multiplication of a matrix by a scalar commutes and a scalar can be factored out of a sum.

The first equality of (ii) is trivial from definition and

$$v \odot_\omega e_0 = \sum_{i=0}^{d-1} v_i S^i \cdot e_0 = \sum_{i=0}^{d-1} v_i e_i = v\,.$$

For (iii), we use Lemma **??** and its proof. Assume $k = 1$:

$$\begin{aligned}
(u \odot_\omega e_1) \odot_\omega v &= \left(\sum_{i=0}^{d-1} u_i S^i \cdot e_1\right) \odot_\omega v = \left(\sum_{i=0}^{d-1} u_i S^i \cdot Se_0\right) \odot_\omega v \\
&= \left(\sum_{i=0}^{d-2} u_i e_{i+1} + u_{d-1} S^d e_0\right) \odot_\omega v = \left(\sum_{i=1}^{d-1} u_{i-1} e_i - u_{d-1} \sum_{i=0}^{d-1} p_i e_i\right) \odot_\omega v \\
&= \left(\sum_{i=1}^{d-1} u_{i-1} S^i - u_{d-1} \sum_{i=0}^{d-1} p_i S^i\right) \cdot v \\
&= \left(\sum_{i=1}^{d-1} u_{i-1} S^i + u_{d-1} S^d\right) \cdot v = \sum_{i=0}^{d-1} u_i S^i \cdot S \cdot v \\
&= u \odot_\omega (S \cdot v) = u \odot_\omega (e_1 \odot_\omega v)\,.
\end{aligned}$$

Now we proceed by induction:

$$\begin{aligned}
\left(u \odot_\omega e_1^k\right) \odot_\omega v &= \left(u \odot_\omega (e_1^{k-1} \odot_\omega e_1)\right) \odot_\omega v = \left((u \odot_\omega e_1^{k-1}) \odot_\omega e_1\right) \odot_\omega v \\
&= (u \odot_\omega e_1^{k-1}) \odot_\omega (e_1 \odot_\omega v) = u \odot_\omega \left(e_1^{k-1} \odot_\omega (e_1 \odot_\omega v)\right) \\
&= u \odot_\omega \left((e_1^{k-1} \odot_\omega e_1) \odot_\omega v\right) = u \odot_\omega \left(e_1^k \odot_\omega v\right)\,.
\end{aligned}$$

The statement (iv) follows easily from distributivity of matrix multiplication with respect to addition. □

Now we can prove that there is a correspondence between elements of $\mathbb{Z}[\omega]$ and $\mathbb{Z}^d$.

Theorem 3. *Let ω be an algebraic integer of degree d. Then*

$$\mathbb{Z}[\omega] = \left\{ \sum_{i=0}^{d-1} a_i \omega^i : a_i \in \mathbb{Z} \right\}$$

and $\mathbb{Z}[\omega]$ is isomorphic to $(\mathbb{Z}^d, +, \odot_\omega)$ by a mapping $\pi : \mathbb{Z}[\omega] \to \mathbb{Z}^d$ defined by

$$\pi(u) = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{pmatrix}$$

for every $u = \sum_{i=0}^{d-1} u_i \omega^i \in \mathbb{Z}[\omega]$.

Proof. Obviously, $\mathbb{Z}[\omega] \supset \left\{ \sum_{i=0}^{d-1} a_i \omega^i : a_i \in \mathbb{Z} \right\}$. Let $p(x) = x^d + p_{d-1}x^{d-1} + \dots p_1 x + p_0$ be the minimal polynomial of ω. Assume $u \in \mathbb{Z}[\omega]$, $x = \sum_{i=0}^{n} u_i \omega^i$ for some $n \geq d$. By $p(\omega) = 0$, we have an equation $\omega^d = -p_{d-1}\omega^{d-1} - \dots - p_1\omega - p_0$ which enables us to write

$$u = u_n \omega^n + \sum_{i=0}^{n-1} u_i \omega^i = u_n \omega^{n-d}(-b_{d-1}\omega^{d-1} - \dots - b_1\omega - b_0) + \sum_{i=0}^{n-1} u_i \omega^i \,.$$

Thus $x \in \left\{ \sum_{i=0}^{d-1} a_i \omega^i : a_i \in \mathbb{Z} \right\}$ by induction with respect to n.

Let us check now that the mapping π is well-define. Assume on contrary that there exists $v \in \mathbb{Z}[\omega]$ and $i_0 \in \{0, 1, \dots, d-1\}$ such that $v = \sum_{i=0}^{d-1} v_i \omega^i = \sum_{i=0}^{d-1} v'_i \omega^i$ and $v_{i_0} \neq v'_{i_0}$. Then

$$\sum_{i=0}^{d-1} (v'_i - v_i)\omega^i = 0$$

and $\sum_{i=0}^{d-1}(v'_i - v_i)x^i \in \mathbb{Z}[x]$ is non-zero polynomial of degree smaller than degree of minimal polynomial p, a contradiction.

Clearly, π is bijection. Let $u = \sum_{i=0}^{d-1} u_i \omega^i$ and $v = \sum_{i=0}^{d-1} v_i \omega^i$ be elements of $\mathbb{Z}[\omega]$. Consider

$$\begin{aligned} \omega v = \omega \sum_{i=0}^{d-1} v_i \omega^i &= \sum_{i=0}^{d-2} v_i \omega^{i+1} + v_{d-1}(\underbrace{-p_{d-1}\omega^{d-1} - \dots - p_1\omega - p_0}_{=\omega^d}) \\ &= -p_0 v_{d-1} + \sum_{i=1}^{d-1} (v_{i-1} - v_{d-1}p_i)\omega^i \,. \end{aligned}$$

Hence

$$\begin{aligned} \pi(\omega v) = -p_0 v_{d-1} e_0 + \sum_{i=1}^{d-1} (v_{i-1} - v_{d-1}p_i)e_i &= S \cdot \pi(v) \\ = e_1 \odot_\omega \pi(v) = \pi(\omega) \odot_\omega \pi(v) \,. \end{aligned}$$

Suppose for induction with respect to i that

$$\pi(\omega^{i-1}v) = (\pi(\omega))^{i-1} \odot_\omega \pi(v)\,.$$

Then

$$\pi(\omega^i v) = \pi(\omega(\omega^{i-1}v)) = \pi(\omega) \odot_\omega \pi(\omega^{i-1}v) = \pi(\omega) \odot_\omega \left((\pi(\omega))^{i-1} \odot_\omega \pi(v)\right) = (\pi(\omega))^i \odot_\omega \pi(v)\,,$$

where we use (iii) in Lemma **??** for the last equality.

Now we multiply v by $m \in \mathbb{Z} \subset \mathbb{Z}[\omega]$:

$$\pi(mv) = \pi\left(m\sum_{i=0}^{d-1} v_i\omega^i\right) = \pi\left(\sum_{i=0}^{d-1} mv_i\omega^i\right) = m\pi(v) = (me_0) \odot_\omega \pi(v) = \pi(m) \odot_\omega \pi(v)\,.$$

As π is obviously aditive, we conclude:

$$\begin{aligned}\pi(uv) = \pi\left(\sum_{i=0}^{d-1} u_i\omega^i v\right) &= \sum_{i=0}^{d-1} \pi(\omega^i u_i v) = \sum_{i=0}^{d-1} \pi(\omega)^i \odot_\omega (\pi(u_i) \odot_\omega \pi(v)) \\ &= \sum_{i=0}^{d-1} \pi(\omega^i u_i) \odot_\omega \pi(v) = \pi\left(\sum_{i=0}^{d-1} u_i\omega^i\right) \odot_\omega \pi(v) = \pi(u) \odot_\omega \pi(v)\,.\end{aligned}$$

□

Due to this theorem we may work with integer vectors instead of elements of $\mathbb{Z}[\omega]$ and multiplication in $\mathbb{Z}[\omega]$ is replaced by multiplying by an appropriate matrix. We also obtained associativity and commutativity of $\odot_\omega$ as $\mathbb{Z}[\omega]$ is an associative and commutative ring.

The last theorem of this section is a practical tool for divisibility in $\mathbb{Z}[\omega]$. To check whether an element of $\mathbb{Z}[\omega]$ is divisible by another element, we look for an integer solution of a linear system. Moreover, this solution provides a result of the division in the positive case.

Theorem 4. *Let ω be an algebraic integer of degree d and let S be a companion matrix of its minimal polynomial. Let $\beta = \sum_{i=0}^{d-1} b_i\omega^i$ be a nonzero element of $\mathbb{Z}[\omega]$. Then for every $u \in \mathbb{Z}[\omega]$*

$$u \in \beta\mathbb{Z}[\omega] \iff S_\beta^{-1} \cdot \pi(u) \in \mathbb{Z}^d\,,$$

where $S_\beta = \sum_{i=0}^{d-1} b_i S^i$.

Proof. Observe first that S_β is nonsingular. Otherwise, there exists $y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{d-1} \end{pmatrix} \in \mathbb{Z}^d, y \neq \mathbf{0}$ such that $S_\beta \cdot y = 0$. Thus

$$\pi(\beta) \odot_\omega y = \mathbf{0} \iff \beta\pi^{-1}(y) = 0\,.$$

Since $\beta \neq 0$, we have

$$0 = \pi^{-1}(y) = \sum_{i=0}^{d-1} y_i\omega^i\,,$$

which contradict that degree of ω is d.

Now

$$\begin{aligned} u \in \beta\mathbb{Z}[\omega] &\iff (\exists v \in \mathbb{Z}[\omega])(u = \beta v) \\ &\iff (\exists v \in \mathbb{Z}[\omega])(\pi(u) = \pi(\beta) \odot_\omega \pi(v) = S_\beta \cdot \pi(v)) \\ &\iff \pi(v) = S_\beta^{-1} \cdot \pi(u) \in \mathbb{Z}^d . \end{aligned}$$

Clearly, if u is divisible by β, then $v = u/\beta = \pi^{-1}(S_\beta^{-1} \cdot \pi(u)) \in \mathbb{Z}[\omega]$. □

Chapter 2

Design of method

NUTNOST REDUNDANCE, PRO VSECHNY DEFINICE A VETY NENI-LI UVEDENO JINAK Let ω be an algebraic integer and $(\beta, \mathcal{A})$ be a numeration system such that a base $\beta \in \mathbb{Z}[\omega]$ and an alphabet $\mathcal{A}$ is a finite subset of $\mathbb{Z}[\omega]$.

The general concept of addition (standard or parallel) in any numeration system $(\beta, \mathcal{A})$, such that $\mathrm{Fin}_{\mathcal{A}}(\beta)$ is closed under addition, is following: we add numbers digitwise and then we convert the result into the alphabet $\mathcal{A}$. Obviously, digitwise addition is computable in parallel, thus the crucial point is the conversion of the obtained result. It can be easily done in a standard way but a parallel digit set conversion is nontrivial. However, formulas are basically same but the choice of coefficients differs.

Now we go step by step more precisely. Let $x = \sum_{-m'}^{n'} x_i\beta^i, y = \sum_{-m'}^{n'} y_i\beta^i \in \mathrm{Fin}_{\mathcal{A}}(\beta)$ with $(\beta, \mathcal{A})$-representantions padded by zeros to have the same length. We set

$$\begin{aligned} w = x + y &= \sum_{-m'}^{n'} x_i\beta^i + \sum_{-m'}^{n'} y_i\beta^i = \sum_{-m'}^{n'} (x_i + y_i)\beta^i \\ &= \sum_{-m'}^{n'} w_i\beta^i\,, \end{aligned}$$

where $w_j = x_j + y_j \in \mathcal{A} + \mathcal{A}$. Thus, $w_{n'}w_{n'-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'}$ is a $(\beta, \mathcal{A} + \mathcal{A})$-representation of $w \in \mathrm{Fin}_{\mathcal{A}+\mathcal{A}}(\beta)$.

We also use column notation of addition in the following, e.g.,

$$\begin{array}{l} x_{n'}\ x_{n'-1}\cdots x_1\ x_0 \bullet x_{-1}\ x_{-2}\ \cdots x_{-m'} \\ y_{n'}\ y_{n'-1}\cdots y_1\ \ y_0 \bullet y_{-1}\ y_{-2}\ \ \cdots y_{-m'} \\ \hline w_{n'}w_{n'-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'}\,. \end{array}$$

As we want to obtain a $(\beta, \mathcal{A})$-representation of w, we search a sequence

$$z_nz_{n-1}\cdots z_1z_0z_{-1}z_{-2}\cdots z_{-m}$$

such that $z_j \in \mathcal{A}$ and

$$z_nz_{n-1}\cdots z_1z_0 \bullet z_{-1}z_{-2}\cdots z_{-m} = (w)_{\beta,\mathcal{A}}\,.$$

In the next, we consider without lost of generality only β-integers since modification for representations with rational part is obvious:

$$\beta^m \cdot z_n z_{n-1} \cdots z_1 z_0 \bullet z_{-1} z_{-2} \cdots z_{-m} = z_n z_{n-1} \cdots z_1 z_0 z_{-1} z_{-2} \cdots z_{-m} \bullet$$

Particulary, let $(w)_{\beta,\mathcal{A}+\mathcal{A}} = w_{n'} w_{n'-1} \cdots w_1 w_0 \bullet$. We search $n \in \mathbb{N}$ and $z_n, z_{n-1}, \ldots, z_1, z_0 \in \mathcal{A}$ such that $(w)_{\beta,\mathcal{A}} = z_n z_{n-1} \cdots z_1 z_0 \bullet$.

We use suitable representation of zero to convert digits w_j into the alphabet $\mathcal{A}$.

Definition 10. For a base $\beta \in \mathbb{Z}[\omega]$, a polynomial $R(x) = r_s x^s + r_{s-1} x^{s-1} + \cdots + r_1 x + r_0$ with coefficients $r_i \in \mathbb{Z}[\omega]$, such that $R(\beta) = 0$, is called a *rewriting rule*. MA TAM BYT Z[OMEGA] NEBO Z[BETA]???

An arbitrary rewriting rule may be used for conversion, then so called *core coefficient*, i.e., one of the coefficients which is greatest in modulus, is applied to convert a digit w_j. For our purpose, we use the simplest possible rewriting rule

$$R(x) = x - \beta \in \mathbb{Z}[\omega][x] \,.$$

As $0 = \beta^j \cdot R(\beta) = 1 \cdot \beta^{j+1} - \beta \cdot \beta^j$, we have a representation of zero

$$1(-\beta) \underbrace{0 \cdots 0}_{j} \bullet = (0)_\beta \,.$$

for all $j \in \mathbb{N}$. We multiply this representation by $q_j \in \mathbb{Z}[\omega]$ which is called a *weight coefficient* to obtain representation of zero

$$q_j(-q_j\beta) \underbrace{0 \cdots 0}_{j} \bullet = (0)_\beta \,.$$

This is digitwise added to $w_n w_{n-1} \cdots w_1 w_0 \bullet$ to convert the digit w_j into the alphabet $\mathcal{A}$. It causes a *carry* q_j on the $(j+1)$-th position from the conversion of j-th digit. The conversion runs from right ($j = 0$) to left until all digits and carries are converted into the alphabet $\mathcal{A}$:

$$\begin{array}{ccccccc}
w_n w_{n-1} & \cdots & w_{j+1} & w_j & w_{j-1} & \cdots & w_1 w_0 \bullet \\
 & & & & q_{j-2} & \iddots & \\
 & & & q_{j-1} & -\beta q_{j-1} & & \\
 & & q_j & -\beta q_j & & & \\
 & \iddots & -\beta q_{j+1} & & & & \\
\hline
z_{n+s} \cdots z_n z_{n-1} & \cdots & z_{j+1} & z_j & z_{j-1} & \cdots & z_1\ z_0 \bullet
\end{array}$$

Hence, the desired formula for conversion on the j-th position is

$$z_j = w_j + q_{j-1} - q_j \beta$$

for $j \in \mathbb{N}_0$. We set $q_{-1} = 0$ as there is no carry from the right on the 0-th position.

Clearly, the value of w is preserved:

$$\begin{aligned}\sum_{j\geq 0} z_j\beta^j &= w_0 - \beta q_0 + \sum_{j>0}(w_j + q_{j-1} - q_j\beta)\beta^j \\ &= \sum_{j\geq 0} w_j\beta^j + \sum_{j>0} q_{j-1}\beta^j - \sum_{j\geq 0} q_j\cdot\beta^{j+1} \\ &= \sum_{j\geq 0} w_j\beta^j + \sum_{j>0} q_{j-1}\beta^j - \sum_{j>0} q_{j-1}\cdot\beta^j \\ &= \sum_{j\geq 0} w_j\beta^j = w\,.\end{aligned}$$

The weight coefficient q_j must be chosen such that the converted digit is in the alphabet $\mathcal{A}$, i.e.,

$$z_j = w_j + q_{j-1} - q_j\beta \in \mathcal{A}\,. \tag{2.1}$$

Choice of weight coefficients is the crucial part in the case of construction of parallel addition algorithms. The method determining weight coefficients for a given input is decribed in Section **??**.

On the other hand, it is trivial for standard numeration systems. Notice that

$$z_j \equiv w_j + q_{j-1} \mod \beta\,.$$

Assume now a standard numeration system $(\beta, \mathcal{A})$, where

$$\beta \in \mathbb{N}\,, \beta \geq 2\,, \mathcal{A} = \{0, 1, 2, \ldots, \beta - 1\}\,.$$

There is only one representative of each class modulo β. Therefore, the digit z_j is uniquely determined for a given digit $w_j \in \mathcal{A} + \mathcal{A}$ and carry q_{j-1} and thus so is the weight coefficient q_j. This means that $q_j = q_j(w_j, q_{j-1})$ for all $j \geq 0$. Generally,

$$q_j = q_j(w_j, q_{j-1}(w_{j-1}, q_{j-2})) = \cdots = q_j(w_j, \ldots, w_1, w_0)$$

and

$$z_j = z_j(w_j, \ldots, w_1, w_0)\,,$$

which implies that addition runs in linear time.

We want to the digit set conversion from $\mathcal{A} + \mathcal{A}$ into $\mathcal{A}$ be computable in parallel, i.e., there exist constants $r, t \in \mathbb{N}_0$ such that for all $j \geq 0$ is $z_j = z_j(w_{j+r}, \ldots, w_{j-t})$. To avoid the dependency on all less, respectivelly more, significant digits, we need variety in the choice of weight coefficient q_j. This implies that the used numeration system must be redundant.

NEJAKE LEMMA O TOM, ZE NEREDUNDANTNI SYSTEM VEDE NA LINEARNI CAS???

2.1 Method

In order to construct a parallel digit set conversion in numeration system $(\beta, \mathcal{A})$ we consider more general case of conversion from an *input alphabet* $\mathcal{B}$ such that $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A} + \mathcal{A}$ instead

of the alphabet $\mathcal{A}+\mathcal{A}$. As menshioned above, the key problem is to find for every $j \geq 0$ a weight coefficient q_j such that

$$z_j = \underbrace{w_j}_{\in\mathcal{B}} + q_{j-1} - q_j\beta \in \mathcal{A}$$

for any input $w \in \mathrm{Fin}_{\mathcal{B}}(\beta), (w)_{\beta,\mathcal{B}} = w_{n'}w_{n'-1}\dots w_1w_0\bullet$. We remark that the weight coefficient q_{j-1} is determined by the input w. For digit set conversion to be computable in parallel we demand to digit $z_j = z_j(w_{j+r},\dots,w_{j-t})$ for a fixed anticipation r and memory t in $\mathbb{N}_0$.

We introduce following definitions to obtain the desired digit set conversion.

Definition 11. Let $\mathcal{B}$ be a set such that $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A}+\mathcal{A}$. Then any finite set $\mathcal{Q} \subset \mathbb{Z}[\omega]$ containing 0 such that

$$\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}$$

is called a *set of weight coefficients*.

We see that

$$(\forall w_j \in \mathcal{B})(\forall q_{j-1} \in \mathcal{Q})(\exists q_j \in \mathcal{Q})(w_j + q_{j-1} - q_j\beta \in \mathcal{A})\,.$$

Accordingly, there is a weight coefficient $q_j \in \mathcal{Q}$ for any carry from the right $q_{j-1} \in \mathcal{Q}$ and any digit w_j in the input alphabet $\mathcal{B}$. I.e., we satisfy basic conversion formula (**??**). Notice that $q_{-1} = 0$ is in $\mathcal{Q}$ by definition. Thus, all weight coeficients may be chosen from $\mathcal{Q}$ inductively.

Definition 12. Let M be an integer and $q : \mathcal{B}^M \to \mathcal{Q}$ be a mapping such that

$$w_j + q(w_{j-1},\dots,w_{j-M}) - \beta q(w_j,\dots,w_{j-M+1}) \in \mathcal{A}$$

for all $w_j, w_{j-1},\dots,w_{j-M} \in \mathcal{B}$. Then q is called a *weight function* and M is called a *length of window*.

JE TREBA POZADOVAT q(0,...,0)=0 NEBO TO Z NECEHO PLYNE?

Having a weight function q, we define a function $\phi : \mathcal{B}^{M+1} \to \mathcal{A}$ by

$$\phi(w_j,\dots,w_{j-M}) = w_j + \underbrace{q(w_{j-1},\dots,w_{j-M})}_{=q_{j-1}} - \beta\,\underbrace{q(w_j,\dots,w_{j-M+1})}_{=q_j} =: z_j\,,$$

which verifies that the conversion is indeed $(M+1)$-local function with anticipation $r = 0$ and memory $t = M$.

The construction of a parallel conversion algorithm by so-called *extending window method* consists of two phases. In the first one, we find a minimal possible weight coefficients set $\mathcal{Q}$. It serves as the starting point for the second phase in which e increment the expected length of the window M until the weight function q is uniquely defined for each $(w_j, w_{j-1},\dots,w_{j-M+1}) \in \mathcal{B}^M$.

2.1.1 Phase 1 – Weight coefficients set

The goal of the first phase is to compute a weight coefficients set $\mathcal{Q}$, i.e., to find a set $\mathcal{Q} \ni 0$ such that

$$\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}\,.$$

We build $\mathcal{Q}$ iteratively so that we extend $\mathcal{Q}$ in a way to cover all elements on the left-hand side with original $\mathcal{Q}$ by elements on the right-hand side with extended $\mathcal{Q}$. This procedure is repeated until the extended weight coefficients set is the same as original one.

In other words, we start with $\mathcal{Q} = \{0\}$ meaning that we search all weight coefficients q_j necessary for conversion for the case where there is no carry from the right, i.e., $q_{j-1} = 0$. We add them to weight coeffcient set $\mathcal{Q}$. These weight coefficients now may appear as a carry q_{j-1}. If there are no suitable weight coefficients q_j in weight coefficients set $\mathcal{Q}$ to cover all sums of added coefficients and digits of input alphabet $\mathcal{B}$, we extend $\mathcal{Q}$ by appropriate ones. And so on until there is no need to add more elements. Here we mean by expression "a weight coefficient q covers an element x" that there is $a \in \mathcal{A}$ such that $x = a + \beta q$.

The precise description of the semi-algorithm in a pseudocode in Algorithm **??**.

Algorithm 1 Search for weight coefficients set (Phase 1)

1: $k := 0$
2: $Q_0 := \{0\}$
3: **repeat**
4: By Algorithm **??**, extend $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ in a minimal possible way so that

$$\mathcal{B} + \mathcal{Q}_k \subset \mathcal{A} + \beta\mathcal{Q}_{k+1}$$

5: $k := k + 1$
6: **until** $\mathcal{Q}_k = \mathcal{Q}_{k+1}$
7: $\mathcal{Q} := \mathcal{Q}_k$
8: **return** $\mathcal{Q}$

To extend $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$, we first find all possible candidates to cover each element of $\mathcal{B}+\mathcal{Q}_k$ (Algorithm **??**). We can improve the performance of Algorithm **??** by substituting the set $\mathcal{B} + \mathcal{Q}_k$ for $(\mathcal{B} + \mathcal{Q}_k) \setminus (\mathcal{B} + \mathcal{Q}_{k-1})$ on the line **??** because

$$\mathcal{B} + \mathcal{Q}_{k-1} \subset \mathcal{A} + \beta\mathcal{Q}_k \subset \mathcal{A} + \beta\mathcal{Q}_{k+1}$$

for any $\mathcal{Q}_{k+1} \supset \mathcal{Q}_k$.

An added element from each list of `candidates` is chosen as the smallest one unless there is already a covering element contained in $\mathcal{Q}_k$ (Algorithm **??**).

We may slightly improve this procedure: for example we may first extend $\mathcal{Q}_k$ by all single-element lists of `candidates`. These elements may be enough to cover also other elements of $\mathcal{B}+\mathcal{Q}_k$. It implies that the resulting $\mathcal{Q}$ is dependent on the way of selection from `candidates`.

2.1.2 Phase 2 – Weight function

We want to find a length of the window M and a weight function $q : \mathcal{B}^M \to \mathcal{Q}$. The idea of the extending window method is to reduce necessary weight coefficients from weight coefficients set $\mathcal{Q}$ obtained in the previous section up to single value by enlarging number of considered input digits. The following notation is used:

Algorithm 2 Search for candidates

Input: previous weight coefficients set $\mathcal{Q}_k$, alternatively pre-previous weight coefficients set $\mathcal{Q}_{k-1}$

```
candidates:= empty list of lists
for all x ∈ B + Q_k do {Alternatively, x ∈ (B + Q_k) \ (B + Q_{k-1})}
    cand_for_x:= empty list
    for all a ∈ A do
        if (x − a) is divisible by β in Z[ω] NEBO ZBETA??? then
            Append (x−a)/β to cand_for_x
        end if
    end for
    Append cand_for_x to candidates
end for
return candidates
```

Algorithm 3 Extending intermediate weight coefficients set

Input: `candidates` from Algorithm **??**, previous weight coefficients set $\mathcal{Q}_k$

```
Q_{k+1} := Q_k
for all cand_for_x in candidates do
    if no element of cand_for_x in Q_{k-1} then
        Add the smallest element (in absolute value) of cand_for_x to Q_k
    end if
end for
return Q_{k+1}
```

Definition 13. Let $\mathcal{Q}$ be a weight coefficients set and $w_j \in \mathcal{B}$. Denote by $\mathcal{Q}_{[w_j]}$ any set such that

$$(\forall q_{j-1} \in \mathcal{Q})(\exists q_j \in \mathcal{Q}_{[w_j]})(w_j + q_{j-1} - q_j\beta \in \mathcal{A})\,.$$

By induction with respect to $m \in \mathbb{N}, m > 1$, denote by $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ any subset of $\mathcal{Q}_{[w_j,\dots,w_{j-m+2}]}$ such that

$$(\forall q_{j-1} \in \mathcal{Q}_{[w_{j-1},\dots,w_{j-m+1}]})(\exists q_j \in \mathcal{Q}_{[w_j,\dots,w_{j-m+1}]})(w_j + q_{j-1} - q_j\beta \in \mathcal{A})$$

for all $(w_j, \dots, w_{j-m+1}) \in \mathcal{B}^m$.

Recall the scheme of conversion for better understanding of the definition and method:

$$\begin{array}{cccc}
\cdots w_{j+1} & w_j & w_{j-1} & \cdots w_{j-M+1} w_{j-M} \cdots \\
 & q_{j-1} & -\beta q_{j-1} & \\
q_j & -\beta q_j & & \\
\hline
\cdots z_{j+1} & z_j & z_{j-1} & \cdots z_{j-M+1}\ z_{j-M} \cdots
\end{array}$$

The idea is to check all possible right carries $q_{j-1} \in \mathcal{Q}$ and determine values $q_j \in \mathcal{Q}$ such that

$$z_j = w_j + q_{j-1} - q_j\beta \in \mathcal{A}\,.$$

So we obtain a set $\mathcal{Q}_{[w_j]} \subset \mathcal{Q}$ of weight coefficients which are necessary to convert digit w_j with any carry $q_{j-1} \in \mathcal{Q}$. Assuming that we know input digit w_{j-1}, the set of possible carries from the right is also reduced to $\mathcal{Q}_{[w_{j-1}]}$. Thus we may reduce the set $\mathcal{Q}_{[w_j]}$ to a set $\mathcal{Q}_{[w_j,w_{j-1}]} \subset \mathcal{Q}_{[w_j]}$ which is necessary to cover elements of $w_j + \mathcal{Q}_{[w_{j-1}]}$. Prolonging length of window in this manner may lead to a unique weight coefficient q_j for enough given input digits.

Accordingly, the weight function q is found if there is $M \in \mathbb{N}$ such that

$$\#\mathcal{Q}_{[w_j,\dots,w_{j-M+1}]} = 1$$

for all $w_j, \dots, w_{j-M+1} \in \mathcal{B}^M$.

The precise desription of the construction of the weight function is in Algorithm **??**.

For construction of the set $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ we first choose elements which are the only ones covering some value in $x \in w_0 + \mathcal{Q}_{[w_{j-1},\dots,w_{j-m+1}]}$. Then we add to $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ one by one elements from $\mathcal{Q}_{[w_j,\dots,w_{j-m+2}]}$ covering an uncovered value until each desired value equals $a + \beta q_j$ for some q_j in $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ and $a \in \mathcal{A}$. The pseudocode is in Algorithm **??**. We remark that this algorithm does not provide the trully minimal set $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ in the sense of size, but it is sufficiently small. Phase 2 can be modified by replacing Algorithm **??** for different one. It is possible that the effort to reduce size of $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ as much as possible is not the best for convergence of Phase 2.

Notice that for given length of the window M, number of calls of Algorithm **??** within Algorithm **??** is

$$\sum_{m=1}^{M} \#\mathcal{B}^m = \#\mathcal{B} \sum_{m=0}^{M-1} \#\mathcal{B}^m = \#\mathcal{B}\frac{\#\mathcal{B}^M - 1}{\#\mathcal{B} - 1}\,.$$

It implies that the time complexity grows exponentially as about $\#\mathcal{B}^M$. The required memory is also exponentional because we have to store at least for $m \in \{M-1, M\}$ sets $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ for all $w_j, \dots, w_{j-m+1} \in \mathcal{B}$.

Algorithm 4 Search for weight function (Phase 2)

Input: weight coefficients set $\mathcal{Q}$

$m := 1$
for all $w_j \in \mathcal{B}$ **do**
    Find set $\mathcal{Q}_{[w_j]} \subset \mathcal{Q}$ such that

$$w_j + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_j]}$$

end for
while $\max\{\#\mathcal{Q}_{[w_j,\ldots,w_{j-m+1}]} : (w_j, \ldots, w_{j-m+1}) \in \mathcal{B}^m\} > 1$ **do**
    $m := m + 1$
    for all $(w_j, \ldots, w_{j-m+1}) \in \mathcal{B}^m$ **do**
        Find set $\mathcal{Q}_{[w_j,\ldots,w_{j-m+1}]} \subset \mathcal{Q}_{[w_j,\ldots,w_{j-m+2}]}$ such that

$$w_j + \mathcal{Q}_{[w_{j-1},\ldots,w_{j-m+1}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_j,\ldots,w_{j-m+1}]}\,,$$

    end for
end while
$M := m$
for all $(w_j, \ldots, w_{j-M+1}) \in \mathcal{B}^M$ **do**
    $q(w_j, \ldots, w_{j-M+1}) :=$ only element of $\mathcal{Q}_{[w_j,\ldots,w_{j-M+1}]}$
end for
return q

Algorithm 5 Search for set $\mathcal{Q}_{[w_j,\ldots,w_{j-m+1}]}$

Input: Input digit w_j, set of carries $\mathcal{Q}_{[w_{j-1},\ldots,w_{j-m+1}]}$, previous set of possible weight coefficients $\mathcal{Q}_{[w_j,\ldots,w_{j-m+2}]}$

`list_of_coverings`:=empty list of lists
for all $x \in w_j + \mathcal{Q}_{[w_{j-1},\ldots,w_{j-m+1}]}$ **do**
    Build a list `x_covered_by` of weight coefficients $q_j \in \mathcal{Q}_{[w_j,\ldots,w_{j-m+2}]}$ such that

$$x = a + \beta q_j \qquad \text{for some } a \in \mathcal{A}\,.$$

    Append `x_covered_by` to `list_of_coverings`
end for
$\mathcal{Q}_{[w_j,\ldots,w_{j-m+1}]}$:= empty set
while `list_of_coverings` is nonempty **do**
    Pick any element q of one of the shortest lists `x_covered_by`
    Add the element q to $\mathcal{Q}_{[w_j,\ldots,w_{j-m+1}]}$
    Remove lists `x_covered_by` containing the element q from `list_of_coverings`
end while
return $\mathcal{Q}_{[w_j,\ldots,w_{j-m+1}]}$

Chapter 3

Convergence

UVOD, ZAVISLOST NA HUSTYCH MNOZINACH Unfortunately, the extending window method is not always finite. The algorithm may lead to an infinite loop in both phases. However, the following lemmma gives a sufficient condition for convergence of the Phase 1.

Lemma 5. *Let $\mathcal{A}$ and $\mathcal{B}$ be finite subsets of $\mathbb{Z}[\omega]$ such that $\mathcal{A}$ contains at least one representative of each congruence class modulo β in $\mathbb{Z}[\omega]$. NEBO ZBeta???? Then, there exists a subset $\mathcal{Q} \subset \mathbb{Z}[\omega]$ such that $\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}$ and all elements of $\mathcal{Q}$ are limited by constant $R \in \mathbb{R}^+$ in modulus.*

Proof. Denote $A := \max\{|a| : a \in \mathcal{A}\}$ and $B := \max\{|b| : b \in \mathcal{B}\}$. Consequently, set $R := \frac{A+B}{|\beta|-1}$ and $\mathcal{Q} := \{q \in \mathbb{Z}[\omega] : |q| \leq R\}$. The set $\mathcal{Q}$ is nonempty because obviously $A > 0$. Any element $x = b + q$ with $b \in \mathcal{B}$ and $q \in \mathcal{Q}$ can be written as $x = a + \beta q'$ for some $a \in \mathcal{A}$ due to existence of representative of each congruence class in $\mathcal{A}$. We prove that $|q'| \leq R$:

$$|q'| = \frac{|b + q - a|}{|\beta|} \leq \frac{B + R + A}{|\beta|} \leq \frac{1}{|\beta|}\left(A + B + \frac{A+B}{|\beta|-1}\right) = \frac{A+B}{|\beta|}\left(\frac{\beta}{|\beta|-1}\right) = R\,.$$

Hence $q' \in \mathcal{Q}$ and thus $x = b + q \in \mathcal{A} + \beta\mathcal{Q}$. □

We plug in the alphabet $\mathcal{A}$ and input alphabet $\mathcal{B}$. Because of the choice of the smallest elements in Algorithm **??**, we know by the lemma that the weight coefficient set $\mathcal{Q}$ constructed in Phase 1 has elements bounded by constant R.

Theorem 6. *Let ω be an algebraic integer such that any complex circle centered at 0 contains only finitely many elements of $\mathbb{Z}[\omega]$. Let $\beta \in \mathbb{Z}[\omega], |\beta| > 1$. Let $\mathcal{A}$ and $\mathcal{B}$ be finite subsets of $\mathbb{Z}[\omega]$ such that $\mathcal{A}$ contains at least one representative of each congruence class modulo β in $\mathbb{Z}[\omega]$. NEBO ZBeta???? Then, there exists a finite subset $\mathcal{Q} \subset \mathbb{Z}[\omega]$ such that $\mathcal{B}+\mathcal{Q} \subset \mathcal{A}+\beta\mathcal{Q}$.*

Proof. Directly from Lemma **??** and assumption on $\mathbb{Z}[\omega]$. □

Therefore, Phase 1 succesfully ends if there can be only finitely many elements in $\mathbb{Z}[\omega]$ NEBO Zbeta??? bounded by constant R as intermediate weight coefficient sets $\mathcal{Q}_k$ have elements bounded by R for all k. We categorize an algebraic integer ω which generates $\mathbb{Z}[\omega] \ni \beta$ as follows:

- $\omega \in \mathbb{Z}$ implies $\mathbb{Z}[\omega] = \mathbb{Z}$ which has required property and thus Phase 1 converges.

- $\omega \in \mathbb{R} \setminus \mathbb{Z}$ implies $\mathbb{Z}[\omega]$ being dense in $\mathbb{R}$. CITACE NEBO DUKAZ??? Thus the convergence of Phase 1 is not guaranteed and we have an example for which it does not converge.
- $\omega \in \mathbb{C} \setminus \mathbb{R}$, ω being quadratic algebraic integer implies that $\mathbb{Z}[\omega]$ is not dense in $\mathbb{C}$ and thus Phase 1 can converge. CO NEJAKY DUKAZ NEBO LEPSI ZDUVODNENI?????? NOT DENSE PORAD JESTE NEZNAMENA, ZE JICH NEMUZE BYT NEKONECNE...
- $\omega \in \mathbb{C} \setminus \mathbb{R}$, ω being algebraic integer of degree ≥ 3 implies $\mathbb{Z}[\omega]$ is dense in $\mathbb{C}$ and therefore the convergence of Phase 1 is not guaranteed.

We focus on Phase 2 now. For shorter notation, set

$$\mathcal{Q}_b^m := \mathcal{Q}_{[\underbrace{b,\ldots,b}_{m}]}$$

for $m \in \mathbb{N}$ and $b \in \mathcal{B}$.

Obviously, finitness of Phase 2 implies that there exists a length of window M such that the set $\mathcal{Q}_b^M$ contains only one element for all $b \in \mathcal{B}$.

Algorithm 6 Check input $bb\ldots b$

Input: Weight coefficient set $\mathcal{Q}$, digit $b \in \mathcal{B}$

1: $m := 1$
2: Find minimal set $\mathcal{Q}_b^1 \subset \mathcal{Q}$ such that

$$b + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}_b^1 \,.$$

3: **while** $\#\mathcal{Q}_b^m > 1$ **do**
4: $\quad m := m + 1$
5: $\quad$ By Algorithm **??**, find minimal set $\mathcal{Q}_b^m \subset \mathcal{Q}_b^{m-1}$ such that

$$b + \mathcal{Q}_b^{m-1} \subset \mathcal{A} + \beta\mathcal{Q}_b^m \,.$$

6: $\quad$ **if** $\#\mathcal{Q}_b^m = \#\mathcal{Q}_b^{m-1}$ **then**
7: $\quad\quad$ **return** Phase 2 does not converge for input $bb\ldots b$.
8: $\quad$ **end if**
9: **end while**
10: **return** Weight coefficient for input $bb\ldots b$ is the only element of $\mathcal{Q}_b^m$.

For arbitrary m, sets $\mathcal{Q}_b^m$ can be easily constructed separately for each $b \in \mathcal{B}$. Using Lemma **??**, Algorithm **??** checks whether Phase 2 stops processing input digits $bb\ldots b$. Thus, non-finitness of Phase 2 can be revealed by running it for each input digit $b \in \mathcal{B}$.

Lemma 7. *Let $m_0 \in \mathbb{N}$ be a length of window and $b \in \mathcal{B}$ be a letter of an input alphabet such that sets $\mathcal{Q}_b^{m_0}$ and $\mathcal{Q}_b^{m_0-1}$ produced by Algorithm **??** within Phase 2 have the same size. Then*

$$\#\mathcal{Q}_b^m = \#\mathcal{Q}_b^{m_0} \qquad \forall m \geq m_0 - 1 \,.$$

Particulary, if $\#\mathcal{Q}_b^{m_0} \geq 2$, Phase 2 does not converge.

Proof. It is enough to prove the base case of an induction as the inductive step is analogic. The set $\mathcal{Q}_b^{m_0+1}$ is find by Algorithm **??** such that

$$b + \mathcal{Q}_b^{m_0} \subset \mathcal{A} + \beta\mathcal{Q}_b^{m_0+1}$$

and set $\mathcal{Q}_b^{m_0}$ was found by the same algorithm such that

$$b + \mathcal{Q}_b^{m_0-1} \subset \mathcal{A} + \beta\mathcal{Q}_b^{m_0} .$$

As $\mathcal{Q}_b^{m_0} \subset \mathcal{Q}_b^{m_0-1}$, the assumption of the same size implies

$$\mathcal{Q}_b^{m_0} = \mathcal{Q}_b^{m_0-1} .$$

It means that Algorithm **??** runs with the same input and hence

$$\mathcal{Q}_b^{m_0+1} = \mathcal{Q}_b^{m_0} .$$

Phase 2 ends when there is only one element in $\mathcal{Q}_{[w_j,\dots,w_{j-m+1}]}$ for all $(w_j, \dots, w_{j-m+1}) \in \mathcal{B}^m$ for some fixed length of window m. But if $\#\mathcal{Q}_b^{m_0} \geq 2$, size of $\mathcal{Q}_{[b,\dots,b]}$ does not decrease despite of extending the length of window. □

Chapter 4

Implementation

DOPSAT POPISY TRID, ODKAZY NA ALGORITMY, POPSAT INTERFACE The designed method requires the arithmetics in $\mathbb{Z}[\omega]$. Therefore, we have chosen Python-based programming language Sage for the implementation of method as it contains many ready-to-use mathematical structure. Specifically, we use the provided class *PolynomialQuotientRing* to represent elements of $\mathbb{Z}[\omega]$ and *NumberField* for obtaining numerical complex value of them. Using Sage is very convenient as it also offers easily usable datastructures or tools for plotting. Thus the code is more readible and we may focus on the algorithmic part of problem. On the other hand, Sage is considerably slower than for C++ or other low-level languages. Nevertheless, it is sufficient for our purpose.

The implementation is object-oriented. It consists of four classes. Class *AlgorithmForParallelAddition* contains structures for computations in $\mathbb{Z}[\omega]$ and links neccessary instances and functions to construct algorithm for parallel addition by the extending window method for an algebraic integer ω given by its minimal polynomial p and approximate complex value, a base $\beta \in \mathbb{Z}[\omega]$, an alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$ and an input alphabet $\mathcal{B}$. Phase 1 of the extending window method is implemented in class *WeightCoefficientsSetSearch* and Phase 2 in *WeightFunctionSearch*. Class *WeightFunction* holds the weight function q computed in Phase 2. All classes are described in the following sections including lists of all methods with short desription. MAJI TADY BYT POPSANE I TAKOVETO KRATKE METODY U KTERYCH JE Z NAZVY ZREJME CO DELAJI? Sometimes, notation from Chapter **??** is used for better understanding.

Basically, weight function can be found just by creating an instance of *AlgorithmForParallelAddition* and calling **findWeightFunction()**. For more comfortable usage, our implementation includes two interfaces – command line version and graphic user interface using interact in SageMath Cloud. The whole implementation is on the attached CD or it can be downloaded from `https://github.com/Legersky/ParallelAddition`.

4.1 Class AlgorithmForParallelAddition

This class constructs neccessary structures for computation in $\mathbb{Z}[\omega]$. It is *PolynomialQuotientRing* obtained as a *PolynomialRing* over integers factored by polynomial p. This is used for representation of elements of $\mathbb{Z}[\omega]$ and arithmetics. We remark that it is independent on the choice of root of the minimal polynomial p. But as we need also comparisons of numbers in $\mathbb{Z}[\omega]$ in modulus, we specify ω by its approximate complex value and we form a factor ring

of rational polynomials by using class *NumberField*. This enables us to get absolute values of elements of $\mathbb{Z}[\omega]$ which can be then compared.

Method **findWeightFunction()** links together both phases of the extending window method to find the weight function q. That is used in the methods for addition and digit set conversion to process them as local functions. There are also verification methods.

Moreover, many methods for printing, plotting and saving outputs are implemented.

Constructor:

__init__(*minPol_str, embd, alphabet, base, name='NumerationSystem', inputAlphabet='', printLog=True, printLogLatex=False, verbose=0*)

Take *minPol_str* which is symbolic expression in the variable x of minimal polynomial p. The closest root of *minPol_str* to *embd* is used as the ring generator ω. The structures for $\mathbb{Z}[\omega]$ are constructed as described above. Setters **setAlphabet**(*alphabet*), **setInputAlphabet**(A) and **setBase**(*base*) are called. Messages saved to logfile during existence of an instance are printed (using LaTeX) on standard output depending on *printLog* and *printLogLatex*. The level of messages for a development is set by *verbose*.

Setters and getters:

setAlphabet(A)

Check if A is subset of $\mathbb{Z}[\omega]$. Set alphabet $\mathcal{A}$:=A

setInputAlphabet(B)

If B is empty, $\mathcal{A}+\mathcal{A}$ is used. Set the input alphabet $\mathcal{B}$:=B. Check if $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A}+\mathcal{A}$.

setBase(*base*)

Set β:=*base*.

getRingGenerator()

Return ring generator ω.

getBase()

Return the base β.

getBaseCC()

Return the base β as a complex number.

getAlphabet()

Return the alphabet $\mathcal{A}$.

getInputAlphabet()

Return the input alphabet $\mathcal{B}$.

getMinPolynomial()

Return the minimal polynomial p of ring generator ω.

getMinPolynomialOfBase()

Return the minimal polynomial p of base β.

getWeightCoefSet()

Return the weight coefficients set $\mathcal{Q}$. If it is not computed, return *None*.

getWeightFunction()

Return the weight function q. If it is not computed, return *None*.

getName()

Return the name of the numeration system.

getDictOfSetting()

Return the dictionary of the attributes of the numeration system.

——————————EXTENDING WINDOW METHOD——————————

_findWeightCoefSet(*max_iterations, method_number***)**

Create an instance of *WeightCoefficientsSetSearch(method_number)* and call its method **findWeightCoefficientsSet**(*max_iterations*) to obtain a weight coefficients set $\mathcal{Q}$.

addWeightCoefSetIncrement(*increment***)**

Save *increment* from extending intermediate weight coefficients set $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$.

_findWeightFunction(*max_input_length,method_number***)**

Create an instance of *WeightFunctionSearch(method_number)* and call its methods **check_one_letter_inputs**(*max_input_length*) and **findWeightFunction**(*max_input_length*) to obtain a weight function q.

findWeightFunction(*max_iterations, max_input_length, method_weightCoefSet=2, method_weightFunSearch*

Return the weight function q obtaind by calling **_findWeightCoefSet**(*max_iterations,method_weightC* and **_findWeightFunction**(*max_input_length, method_weightFunSearch*).

——————————PARALLEL ADDITION AND CONVERSION——————————

addParallel(a,b**)**

Sum up numbers represented by lists of digits a and b digitwise and convert the result by **parallelConversion**().

parallelConversion($\_w$**)**

Return $(\beta, \mathcal{A})$-representation of number represented by list $\_w$ of digits in input alphabet $\mathcal{B}$. It is computed locally according to the equation **??** and using weight function q.

localConversion(w**)**

parallelConversion_using_localConversion(w**)**

——————————SANITY CHECK——————————

sanityCheck_addition(*num_digits***)**

sanityCheck_conversion(*num_digits*)

————————RING CONVERSIONS, AUXILIARY RING FUNCTIONS————————

list2BaseRing(*_digits*)

list2Ring(*_digits*)

ring2NumberField(*num_from_ring*)

ring2CC(*num_from_ring*)

getCoordinates(*num*)

sumOfSets(*A,B*)

_computeInverseBaseCompanionMatrix()

divideByBase(*divided_number*)

————————PRINT FUNCTIONS————————

addLog(*_log, latex=False*)

printWeightFunction()

printWeightFunctionInfo()

printWeightCoefSet()

printLatexInfo()

———————PLOT FUNCTIONS———————————————————————————————

plot(*nums_from_ring, labeled=True, color='red', size=20, fontsize=10*)

plotAlphabet()

plotWeightCoefSet(*estimation=False*)

plotLattice()

plotPhase1(*legend_xshift=8,*)

legend(*def legend(k,covered, new, alphabet*)

plotPhase2(*digits,*)

———————SAVE FUNCTIONS———————————————————————————————

saveInfoToTexFile(*filename, header=True*)

saveLog(*filename*)

saveWeightFunctionToTexFile(*filename*)

saveWeightFunctionToCsvFile(*filename*)

saveLocalConversionToCsvFile(*filename*)

saveUnsolvedInputsToCsv(*filename*)

inputSettingToSageFile(*filename*)

saveImages(*images, folder,name, img_size=10*)

4.2 Class WeightCoefficientsSetSearch

——————————-CONSTRUCTOR————————————————————————-

__init__(*algForParallelAdd, method*)

————————————SEARCH FOR WEIGHT COEFFICIENT SET————————————————————————————-

_findCandidates(*C*)

_chooseQk_FromCandidates(*cand_for_all*)

_getQk(*C*)

findWeightCoefficientsSet(*maxIterations*)

————————————AUXILIARY FUNCTIONS————————————————————————————————-

_findSmallest(*list_from_Ring*)

4.3 Class WeightFunctionSearch

————————————CONSTRUCTOR, SETTERS————————————————————————————————-

__init__(*algForParallelAdd, weightCoefSet, method*)

————————————SEARCH FOR WEIGHT FUNCTION————————————————————————————-

_find_weightCoef_for_comb_B(*combinations, max_len*)

_findQw(*w_tuple*)

findWeightFunction(*max_input_length*)

check_one_letter_inputs(*max_input_length*)

————————————PRINT FUNCTION————————————————————————————-

printCsvQww()

4.4 Class WeightFunction

——————————CONSTRUCTOR, GETTERS——

__init__(*B*)

getMaxLength()

getMapping()

——————————ADDING INPUTS, CALL FUNCTION——————————————————————————————————

getWeightCoef(*w*)

addWeightCoefToInput(*_input, coef*)

——————————PRINT FUNCTIONS——

printInfo()

printLatexInfo()

printMapping()

printLatexMapping()

printCsvMapping()

4.5 User interfaces

POPSANY VZOROVY VSTUPNI SOUBOR PRO CMD A INTERACT

Chapter 5

Testing

USPESNE A NEUSPESNE PRIKLADY - otestovat ty z Mileniných poznámek? Pridat nejake kubické?

5.0.1 Eisenstein base $\beta = -1 + \omega$ with $\omega = \exp \frac{2\pi\imath}{3} = -\frac{1}{2} + \frac{\imath\sqrt{3}}{2}$

KAZDEMU NEJAKY TAKOVYTO VYSTUP:

Numeration System: eisenstein
Minimal polynomial of ω: $t^2 + t + 1$
Base $\beta = \omega - 1$
Minimal polynomial of base: $x^2 + 3x + 3$
Alphabet $\mathcal{A} = \{0, 1, -1, \omega, -\omega, -\omega - 1, \omega + 1\}$
Input alphabet $\mathcal{B} = \mathcal{A} + \mathcal{A}$
Phase 1 was succesfull.
Weight Coefficient Set:

$\mathcal{Q} = \{0, 1, 2, -\omega, 2\omega, 2\omega + 1, 2\omega + 2, \omega - 1, \omega, \omega + 1, \omega + 2, -\omega - 1, -\omega - 2, -2\omega, -\omega + 1, -1, -2\omega - 2, -2\omega -$

Number of elements in the weight coefficient set $\mathcal{Q}$ is: 19
Weight function Info:
Phase 2 was succesfull.
Maximal input length: 3
Number of inputs: 6085

5.0.2 Penney base $\beta = -1 + \imath$ with $\imath = \exp \frac{2\pi\imath}{4}$

5.0.3 Base $\beta = 1 + \imath$

5.0.4 Base $\beta = -\frac{3}{2} + \frac{\imath\sqrt{11}}{2}$

5.0.5 Base $\beta = -2 + \imath$

Summary

DOPSAT PODLE OSTATNIHO

Bibliography

[1] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.