Sets of integers in different number systems and the Chomsky hierarchy

István Katsányi *

Abstract

The classes of the Chomsky hierarchy are characterized in respect of converting between canonical number systems. We show that the relations of the bases of the original and converted number systems fall into four distinct categories, and we examine the four Chomsky classes in each of the four cases. We also prove that all of the Chomsky classes are closed under constant addition and multiplication. The classes $\mathcal{RE}$ and $\mathcal{CS}$ are closed under every examined operation. The regular languages are closed under addition, but not under multiplication.

Key words: formal languages, Chomsky hierarchy, canonical number systems, number theory, converting.

1 Introduction

It is a thoroughly studied subject within the discipline of formal languages and automata theory, that under which conditions will a set of integers in m–ary notation be regular for a given $m \geq 1$. Cobham has solved the bases of this problem in [1]. His results were extended and generalized by many authors in many ways for example in the papers [2], [3], [4], [5], [6], [7]. Luca and Restivo suggested in their paper ([3]) to study the open problem of the context–free case. In this work, we examine the context–free, the context sensitive and the recursively enumerable classes in addition to the regular languages, hence the examination of the Chomsky–hierarchy in this regard becomes complete.

We also investigate, that under what conditions do certain arithmetical operations alter the Chomsky–class of a language. We prove some closure properties and generalize a theorem related to ranges of polynomials.

2 Preliminaries

When not stated otherwise, we will use the standard notations used in the theory of formal languages (see for example [8]). The set of nonnegative integers will be denoted by $\mathcal{N}$, and the classes of regular, context–free, context sensitive and recursively enumerable languages by $\mathcal{REG}$, $\mathcal{CF}$, $\mathcal{CS}$, $\mathcal{RE}$, respectively. The length of a word u will be denoted by $|u|$. The notation of the empty word is λ.

*Eötvös Loránd University, Department of General Computer Science, 1117 Budapest, Pázmány Péter sétány 1/D, e-mail: `kacsa@ludens.elte.hu`

The mirror image of a word u will be denoted by u^{-1}. We use the same notation for the mirror images of languages: for a language L let $L^{-1} = \{u^{-1} \mid u \in L\}$.

We call a word u a proper base–a integer ($a \geq 1$), if $a = 1$ and u consists of some 1 digits, or $a \geq 2$, u consists of digits $0, \ldots, a-1$, and $u = 0$, or the first digit of u is nonzero. The value of a proper base–a integer u will be denoted by $val_a(u)$. (The value of a proper base–1 integer u is the length of u.) We denote by $L_a(A)$ the language of proper base–a integers, whose values constitutes the set A, where A is a set of nonnegative integers, and a is a positive integer. For example, if $A = \{2^n \mid n \geq 0\}$, then $L_2(A) = \{1, 10, 100, \ldots\}$ can be expressed by the regular expression 10^*, but according to Büchi ([9]) $L_{10}(A) = \{1, 2, 4, 8, 16, \ldots\}$ and $L_1(A) = \{1, 11, 1111, \ldots\}$ are nonregular languages.

We call two integers $a, b \geq 2$ *multiplicatively dependent*, if there exist integers $n, m \geq 1$, such that $a^n = b^m$. Otherwise, they are called *multiplicatively independent*.

We will also use the well–known structure of generalized sequential machines, and a pumping lemma concerning regular languages ([10],[11], [12]):

Definition 1 *A* generalized sequential machine *is a 6–tuple $(Q, \Sigma, \Delta, \sigma, s, F)$, where Q is the finite set of states, Σ is the input alphabet, Δ is the output alphabet, σ is the transition–and–output function from $Q \times \Sigma$ to finite subsets of $Q \times \Delta^*$, $s \in Q$ is the starting state, and $F \subseteq Q$ is the set of final states. For a gsm G, the set of all output words in response to an input word $u \in \Sigma^*$ is denoted by $G(u)$. For a language $L \subseteq \Sigma^*$ we define $G(L) = \bigcup_{u \in L} G(u)$.*

Proposition 1 *Let L be a regular set. Then there is a constant n, such that if z is any word in L, and $|z| \geq n$, we may write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$, and for all $i \geq 0$, uv^iw is in L.*

3 Converting between canonical number systems

One of the main results of our paper is the next theorem:

Theorem 1

	$a = 1, b \geq 2$	$a \geq 2, b = 1$	$a, b \geq 2$, $\exists n, m \geq 1 :$ $a^n = b^m$	$a, b \geq 2$, $\nexists n, m \geq 1 :$ $a^n = b^m$
$\mathcal{REG}$	$\mathcal{REG}$	$\mathcal{CS}$	$\mathcal{REG}$	$\mathcal{CS}^*$
$\mathcal{CF}$	$\mathcal{REG}$	$\mathcal{CS}$	$\mathcal{CF}$	$\mathcal{CS}^*$
$\mathcal{CS}$	$\mathcal{RE}^*$	$\mathcal{CS}$	$\mathcal{CS}$	$\mathcal{CS}$
$\mathcal{RE}$	$\mathcal{RE}$	$\mathcal{RE}$	$\mathcal{RE}$	$\mathcal{RE}$

For each of the 16 inner cells of the table the following holds: If a and b are integers that satisfy the condition shown in the heading of the column of the cell, and A is a set of integers such that $L_a(A)$ belongs to the class shown in *the heading of the row of the cell, then the language $L_b(A)$ belongs to the class shown in the cell. For each cell that is not marked with a *, there exists integers a, b and sets A, such that they satisfy the appropriate conditions, and $L_b(A)$ is not the element of any smaller Chomsky class, than the one shown in the cell.*

The theorem can be proved by a series of lemmas. Let us prove a part of the context–sensitive case first.

Lemma 1 *Let $a \geq 2$ be an integer, and let A be a set of nonnegative integers, such that the language $L_a(A)$ is context–sensitive. Then for all $b \geq 1$ integers, the language $L_b(A)$ is also context–sensitive.*

Proof. We will construct a grammar $G' = (N', T', S', P')$, such that $L(G') = L_b(A)$ and then we will prove, that there is a constant c such that there exists a derivation in G' of every nonempty word $u \in L(G')$, which uses a workspace of size $c|u|$ or less. For the proof, that this property is sufficient for the generated language to be context sensitive, see e.g. [8], Theorem 10.1.

Let $G = (N, T, S, P)$ be a length–increasing grammar generating the language $L_a(A)$. We may assume, that $T = \{0, 1, \ldots, a-1\}$, and the left–side of each production of G contains only nonterminals, and even the right–sides of the productions contain only terminals in productions of the form $X_i \to i$ ($X_i \in N, 0 \leq i \leq a-1$), and these X_i nonterminals do not appear on the left side of other productions. G' works roughly as follows: it generate words according to the rules of G, enclose it between some markers, and creates a separate block, where the base–b representation of the generated number will evolve. This block first consists of the representation of 0, and then the rules of G' one by one decrease the generated base–a number, and increase the base–b number in the separate block.

Let us suppose first, that $b \geq 2$. Then G' will be the following:

$$\begin{aligned} N' &= N \cup \{S', B, M, E, C, D, F, H, Y_0, Y_1, \ldots, Y_{b-1}\}, \\ T' &= \{0, 1, \ldots, b-1\}, \\ P' &= P \cup P'', \end{aligned}$$

where the newly added nonterminals do not appear in N, and P'' consists of the following rules:

Beginning:

$$S' \to BSCMY_0E \tag{1}$$

Subtracting 1 from the left side:

$$X_iX_jC \to X_iX_{j-1}D \qquad 0 \leq i \leq a-1 \quad 1 \leq j \leq a-1 \tag{2}$$

$$X_iX_0C \to X_iCX_{a-1} \qquad 0 \leq i \leq a-1 \tag{3}$$

$$BX_iC \to BX_{i-1}D \qquad 2 \leq i \leq a-1 \tag{4}$$

$$BX_1C \to BD \tag{5}$$

D goes to the right:

$$DX_i \to X_i D \qquad 0 \le i \le a-1 \tag{6}$$
$$DM \to MD \tag{7}$$
$$DY_i \to Y_i D \qquad 0 \le i \le b-1 \tag{8}$$

Adding 1 to the right side:

$$DE \to FE \tag{9}$$
$$Y_i F \to HY_{i+1} \qquad 0 \le i \le b-2 \tag{10}$$
$$Y_{b-1} F \to FY_0 \tag{11}$$
$$MF \to MHY_1 \tag{12}$$

H goes to the left:

$$Y_i H \to HY_i \qquad 0 \le i \le b-1 \tag{13}$$
$$MH \to CM \tag{14}$$

If the left side is empty, we have finished:

$$BX_0CM \to BCM \tag{15}$$
$$BCM \to \lambda \tag{16}$$
$$E \to \lambda \tag{17}$$
$$Y_i \to i \qquad 0 \le i \le b-1 \tag{18}$$

In order to prove that $L_b(A) = L(G')$ first we prove, that $L_b(A) \subseteq L(G')$. Let us prove two claims first, which shows that the encoded numbers between the symbols B and M (M and E) can be decreased (increased) by one, under certain conditions. Using these two claims the proof of the inclusion will be easy.

Claim 1 *Let u be a word of the form*

$$u = BX_{j_1} \ldots X_{j_k} CMu_0E,$$

where $k \ge 1$, $j_1 \ldots j_k$ is a proper base–a integer of a positive value, and $u_0 \in \{Y_0, \ldots, Y_{b-1}\}^*$. Then

$$u \Longrightarrow^*_{G'} u' = BX_{j'_1} \ldots X_{j'_{k'}} DMu_0E,$$

such that either $k = 1$, $X_1 = 1$ and $k' = 0$, or $j'_1 \ldots j'_{k'}$ is a proper base–a integer, and $val_a(j_1 \ldots j_k) = val_a(j'_1 \ldots j'_{k'}) + 1$.

Proof. Using rules of type (3) the symbol C steps over all X_0 nonterminals, and replace each of them by X_{a-1}. (We cannot decrease the digit zero any more, we must replace it by the maximal digit and try to decrease the preceding digit.)

$$u \Longrightarrow^*_{G'} u_1 = BX_{j_1} \dots X_{j_m} C \overbrace{X_{a-1} \dots X_{a-1}}^{k-m} Mu_0E,$$

where $1 \leq m \leq k$ and $j_m \geq 1$.
If m is at least two, we may „subtract one" form X_{j_m} using a rule of type (2):

$$u_1 \Longrightarrow_{G'} u_2 = BX_{j_1} \dots X_{j_m-1} D \overbrace{X_{a-1} \dots X_{a-1}}^{k-m} Mu_0E,$$

If m is one, but $j_1 \geq 2$, we can still „decrease" X_{j_1} using a rule of type (4):

$$u_1 \Longrightarrow_{G'} u_3 = BX_{j_1-1} D \overbrace{X_{a-1} \dots X_{a-1}}^{k-1} Mu_0E,$$

Finally, if both m and j_1 equal to one, we erase X_1 using rule (5):

$$u_1 \Longrightarrow_{G'} u_4 = BD \overbrace{X_{a-1} \dots X_{a-1}}^{k-1} Mu_0E,$$

From each of u_2, u_3 and u_4 we can derive u' by using rules of type (6). It follows from the construction, that the condition given for u' holds.

Claim 2 *Let u be a word of the form*

$$u = Bu_0DMY_{l_1} \dots Y_{l_s}E,$$

where $u_0 \in \{X_0, \dots, X_{a-1}\}^*$, $s \geq 1$ and $l_1 \dots l_s$ is a proper base–b integer. Then

$$u \Longrightarrow^*_{G'} u' = Bu_0CMY_{l'_1} \dots Y_{l'_{s'}}E,$$

where $s' \geq 1$, $l'_1 \dots l'_{s'}$ is a proper base–b integer and

$$val_b(l_1 \dots l_s) + 1 = val_b(l'_1 \dots l'_{s'}).$$

Proof. By rules of type (7), (8) and (9) we derive u_1:

$$u \Longrightarrow^*_{G'} u_1 = Bu_0MY_{l_1} \dots Y_{l_s}FE.$$

Then by the rule (11) F steps over all Y_{b-1}, which cannot be increased any more, so these are replaced by Y_0-s:

$$u_1 \Longrightarrow^*_{G'} u_2 = Bu_0MY_{l_1} \dots Y_{l_m} F \overbrace{Y_0 \dots Y_0}^{s-m} E,$$

where either $m = 0$ or $1 \le m \le s$ and $0 \le l_m \le b-2$. If m is at least 1, then we may apply a rule of type (10), and increase the rightmost non–maximal digit:

$$u_2 \Longrightarrow_{G'} u_3 = Bu_0MY_{l_1}\dots Y_{l_{m-1}}HY_{l_m+1}\overbrace{Y_0\dots Y_0}^{s-m}E.$$

If m is zero, we use rule (12) and introduce a new digit:

$$u_2 \Longrightarrow_{G'} u_4 = Bu_0MHY_1\overbrace{Y_0\dots Y_0}^{s}E.$$

From both u_3 and u_4 we can derive u' by using rules of type (13) and (14). It follows from the construction, that the condition given for u' holds.

Now let us prove, that for every $v \in L_b(A)$ there exists a derivation

$$S' \Longrightarrow^*_{G'} v.$$

By definition, there must be a word $u \in L(G)$, such that $val_a(u) = val_b(v)$. Let us denote the digits of u by $i_1, i_2, \dots, i_{|u|}$. Hence $u = i_1 i_2 \dots i_{|u|}$ $(0 \le i_1, i_2, \dots, i_{|u|} \le a-1)$, and the following derivations are valid:

$$\begin{aligned} S &\Longrightarrow^*_{G} X_{i_1}X_{i_2}\dots X_{i_{|u|}},\\ S' &\Longrightarrow^*_{G'} BX_{i_1}X_{i_2}\dots X_{i_{|u|}}CMY_0E. \end{aligned}$$

If $u = 0$, then we continue the derivation by the rules (15), (16), (17), and $Y_0 \to 0$. We derived 0, which is the base–b representation of u. If $u \ne 0$, then we may use Claim 1 and Claim 2 consecutively $val_a(u)$ times. By that

$$S' \Longrightarrow^*_{G'} BCMY_{j_1}\dots Y_{j_s}E,$$

such that $val_b(j_1 \dots j_s) = val_a(u)$. Using rules of type (16), (17), and (18) we obtain, that

$$S' \Longrightarrow^*_{G'} j_1 \dots j_s = v,$$

which is exactly what we wanted to show.

Now let us prove, that $L(G') \subseteq L_b(A)$. Let u be an arbitrary word in $L(G')$, and let its derivation be the following:

$$S' \Longrightarrow_{G'} BSCMY_0E \Longrightarrow^*_{G'} u \in {T'}^*$$

We must not use rules of type $X_i \to i$ $(0 \le i \le a-1)$ during the derivation, or else the nonterminals B and M cannot be erased because of the terminal i between them. As no X_i $(0 \le i \le a-1)$ nonterminals exist on the left side of a production of another type in P', the steps of this derivation can be interchanged in such a way, that we get the following derivation:

$$\begin{aligned}
S' &\Longrightarrow^*_{G'} BX_{i_1}\dots X_{i_n}CMY_0E \Longrightarrow^*_{G'} && (19)\\
&\Longrightarrow^*_{G'} BCMY_{j_1}\dots Y_{i_k}E \Longrightarrow_{G'} && (20)\\
&\Longrightarrow_{G'} Y_{j_1}\dots Y_{i_k}E \Longrightarrow_{G'} && (21)\\
&\Longrightarrow_{G'} Y_{j_1}\dots Y_{i_k} \Longrightarrow^*_{G'} && (22)\\
&\Longrightarrow^*_{G'} j_1\dots j_k = u && (23)
\end{aligned}$$

At each step of the derivation (20), there can be used only one rule. For that, and for the arguments mentioned before $val_a(i_1\dots i_n) = val_b(j_1\dots j_k)$. On the other hand obviously $S \Longrightarrow^*_G i_1\dots i_n$, so $val_a(i_1\dots i_n) \in A$, which means, that $u \in L_b(A)$.

Finally, let us prove, that $L(G') \in CS$. We will show, that for an appropriately chosen constant c every derivation of every word $u \in L(G')$ uses a workspace of size $c|u|$ or less. This condition is obviously stronger than required for $L(G')$ being context–sensitive.

Let us consider an arbitrary derivation $\mathcal{D} : S' \Longrightarrow^*_{G'} u$ of an arbitrary word $u \in L(G')$. It is obvious, that every derivation of the word $u = 0$ uses a workspace of size $6 = 6\,|u|$. Otherwise, we can split the derivation into two parts:

$$\mathcal{D} : S' \Longrightarrow^*_{G'} BCMu' \Longrightarrow^*_{G'} u.$$

We call the *first part* of the derivation $\mathcal{D}$ the steps, where the derived word contains the nonterminals B and M, and there are at least one symbol between them, not counting the symbols C and D. We will refer the rest of the derivation as the *second part*.

If we denote by v the base–a representation of $val_b(u)$, then the number of the letters between the nonterminals B and M is at most $|v| + 1$, because the grammar G is length–increasing. The one extra symbol could be C or D.

During the first part of the derivation there cannot be more than $|u| + 1$ symbols between the letters M and E, and during the second part, each derived word is not longer than $|u| + 5$. Now the extra symbols are B, C, M, E, and one of the letters D, F and H. We obtained, that during the derivation $\mathcal{D}$ the length of the longest derived word is at most $|u| + |v| + 6$.

Since it can be shown, that $|v| \le (1 + \frac{\log b}{\log a})\,|u|$, there must be a constant c, such that every derived word during the derivation D is shorter, than $c\,|u|$. This proves, that the language $L(G')$ is context–sensitive.

The case of $b = 1$ is similar, we only have to make some minor modification in the definition of the grammar and in the proofs. The main difference is in the rule sets (9)–(12), but the modification should cause the reader no trouble. □

The following lemma is a consequence of Church's thesis, but we will give a direct proof, too.

Lemma 2 *Let $a \geq 1$ be an integer, and let A be a set of nonnegative integers, such that the language $L_a(A)$ is recursively enumerable. Then for all $b \geq 1$ integers, the language $L_b(A)$ is also recursively enumerable.*

Proof. For the case of $a \geq 2$ we use the very same construction as the one used in Lemma 1, but this time we can not suppose the original grammar to be length–increasing, hence the resulting grammar may generate languages, which are not context–sensitive. For the unary case we also use the same ideas, but we change the rules (2)–(5) to the following one: $X_1C \to D$. Using the same thoughts, we may prove, that the language generated by the constructed grammar equals to $L_b(A)$. □

The following lemma is an immediate consequence of Lemma 2.

Lemma 3 *Let A be a set of nonnegative integers, such that the language $L_1(A)$ is context–sensitive. Then for all $b \geq 2$ integers, the language $L_b(A)$ is recursively enumerable.*

There are no known examples of sets, that have context–sensitive representation in the unary number system, and a non–context–sensitive representation in an other number system, so we cannot be sure, that the recursively enumerable class is the smallest one, which contains this type of languages. In contrast to this, we have nice results concerning multiplicatively dependent bases.

Lemma 4 *Let $a, b \geq 2$ be two multiplicatively dependent integers, and let A be a set of nonnegative integers. Then $L_a(A) \in \mathcal{F}$ holds iff $L_b(A) \in \mathcal{F}$ is true, where $\mathcal{F}$ is one of the classes $\mathcal{RE}$, $\mathcal{CF}$, $\mathcal{REG}$.*

Proof. We will construct a generalized sequential machine G, such that when the input of G is the mirror image of an integer expressed in the base–a number system, then the output of G is the mirror image of the same integer expressed in the base–b number system. This completes the proof, because the classes in question are all closed under gsm–mappings and mirror images.

Let n and m be two positive integers, such that $a^n = b^m$. (These numbers must exists, because a and b are multiplicatively dependent integers.) The constructed gsm is the following one:

$$G = (\{f\} \cup \{s_{i,j} \mid 0 \leq i \leq a^n - 1, 0 \leq j \leq n - 1\}, T_a, T_b, \sigma, s_{0,0}, \{f\}),$$

where $T_a = \{0, 1, \ldots, a-1\}$, $T_b = \{0, 1, \ldots, b-1\}$, and σ is defined by the following. If n is at least two, then for all integers i, j and letter $x \in T_a$, such that $0 \leq i \leq a^{n-2} - 1$, and $0 \leq j \leq n-2$ we have

$$\sigma(s_{i,j}, x) = \{(s_{i+xa^j, j+1}, \lambda), (f, u)\},$$

where u is a word, such that u^{-1} is a proper base–b integer, and $val_b(u^{-1}) = i + xa^j$. Moreover (for any values of n), we have

$$\sigma(s_{i,n-1}, x) = \{(s_{0,0}, u0^{m-|u|}), (f, u)\},$$

where $0 \leq i \leq a^{n-1} - 1$, $x \in T_a$, and u is a word, such that u^{-1} is a proper base–b integer, and $val_b(u^{-1}) = i + xa^{n-1}$.

The work of the gsm G is based upon the fact, that we can divide the number to be converted to distinct n–digit blocks, and we can make the conversion in each block independently of the other blocks. Each n–digit block will be converted to an m–digit block in the base–b number system. The gsm reads n consecutive digits from its input, keeping the value of the read (reversed) number in its internal state. The read number must be less, then a^n, so the number of internal states will not be infinite. After reading n digits, G outputs m digits: the reverse of the base–b representation of the read number, using leading zeroes, if necessary, and then it resets its counter, and begins a new cycle. At any moment, G may stop its operation by writing the mirror image of the base–b representation of the stored number to the output. This time we omit leading zeroes. After that step, G goes to the unique final state, which has no following state, so this step can only be used with success at the end of the input. The details of the proof, that for every proper base–a integer u, $G(u^{-1})$ consists of exactly one word, and that $v = G(u^{-1})^{-1}$ is a proper base–b integer, such that $val_a(u) = val_b(v)$ are left to the reader. $\square$

The following lemma is an immediate consequence of Lemma 1:

Lemma 5 *Let $a, b \geq 2$ be two multiplicatively independent integers, and let $A \subseteq \mathcal{N}$ be a set, such that $L_a(A) \in \mathcal{REG}$, or $L_a(A) \in \mathcal{CF}$. Then $L_b(A) \in \mathcal{CS}$.*

By the work of Büchi ([9]) we know, that if $a, b \geq 2$ are two multiplicatively independent integers and $A \subseteq \mathcal{N}$ is a set, such that $L_a(A) \in \mathcal{REG}$, then $L_b(A)$ may be nonregular. However, it is not known, that the language $L_b(A)$ can be non–context–free, too. Our conjunction is, that it can.

The following lemma is another consequence of Lemma 1. It follows from Büchi's Theorem, that this lemma cannot be further strengthened.

Lemma 6 *Let $a \geq 2$ be an integer and let $A \subseteq \mathcal{N}$ be a set, such that $L_a(A) \in \mathcal{REG}$, or $L_a(A) \in \mathcal{CF}$. Then $L_1(A) \in \mathcal{CS}$.*

Using the fact, that every context–free language over a one–letter alphabet is also regular (see for example [8], Theorem 7.3), we get the following consequence of Cobham's Theorem ([1]):

Lemma 7 *Let $b \geq 2$ be an integer and let $A \subseteq \mathcal{N}$ be a set, such that $L_1(A) \in \mathcal{REG}$, or $L_1(A) \in \mathcal{CF}$. Then $L_b(A) \in \mathcal{REG}$.*

At the end of this section we repeat the table of Theorem 1, indicating in each cell the number of the lemma, which proves the part of the theorem that belongs to the cell.

	$a=1, b\geq 2$	$a\geq 2, b=1$	$a,b\geq 2$, $\exists n,m\geq 1:$ $a^n=b^m$	$a,b\geq 2$, $\nexists n,m\geq 1:$ $a^n=b^m$
$\mathcal{REG}$	$\mathcal{REG}$ (7.)	$\mathcal{CS}$ (6.)	$\mathcal{REG}$ (4.)	$\mathcal{CS}^*$ (5.)
$\mathcal{CF}$	$\mathcal{REG}$ (7.)	$\mathcal{CS}$ (6.)	$\mathcal{CF}$ (4.)	$\mathcal{CS}^*$ (5.)
$\mathcal{CS}$	$\mathcal{RE}^*$ (3.)	$\mathcal{CS}$ (1.)	$\mathcal{CS}$ (1.)	$\mathcal{CS}$ (1.)
$\mathcal{RE}$	$\mathcal{RE}$ (2.)	$\mathcal{RE}$ (2.)	$\mathcal{RE}$ (2.)	$\mathcal{RE}$ (2.)

4 Arithmetic operations on languages

This section deals with the second main part of the paper: language properties of arithmetic operations on sets.

First we define some operations over sets of integers:

Definition 2 *Let $A, B \subseteq \mathcal{N}$ be two sets of integers, and let $c \geq 0$ be an integer. Let us define*

$$A + B = \{a + b \mid a \in A, b \in B\}, \qquad c + A = A + c = \{c\} + A,$$
$$A \cdot B = \{ab \mid a \in A, b \in B\}, \qquad c \cdot A = A \cdot c = \{c\} \cdot A,$$
$$A^B = \{a^b \mid a \in A, b \in B\}, \qquad A^c = A^{\{c\}}.$$

Theorem 2

	$c+A$	$c\cdot A$	$A+B$	$A\cdot B$	A^B
$\mathcal{REG}$	$\mathcal{REG}$	$\mathcal{REG}$	$\mathcal{REG}$	$\mathcal{CS}^*$	$\mathcal{CS}^*$
$\mathcal{CF}$	$\mathcal{CF}$	$\mathcal{CF}$	$\mathcal{CS}^*$	$\mathcal{CS}^*$	$\mathcal{CS}^*$
$\mathcal{CS}$	$\mathcal{CS}$	$\mathcal{CS}$	$\mathcal{CS}$	$\mathcal{CS}$	$\mathcal{CS}$
$\mathcal{RE}$	$\mathcal{RE}$	$\mathcal{RE}$	$\mathcal{RE}$	$\mathcal{RE}$	$\mathcal{RE}$

Let $a \geq 1$ be an integer. Then each inner cell of the table shows the Chomsky–class of the language $L_a(C)$, where $c \geq 0$, $A, B \subseteq \mathcal{N}$ such that $L_a(A)$ (and $L_a(B)$, if appropriate) belongs to the class shown in the heading of the row of the cell, and C is the result of the operation written in the heading of the column of the cell. With the exception of the elements marked with a $*$, the presented classes are the smallest ones in the Chomsky–hierarchy with this property.

Again, we prove this theorem by a series of lemmas. One of the key lemmas is the following:

Lemma 8 *Let $a \geq 1$ be an integer, and let $A, B \subseteq \mathcal{N}$ be two sets, such that the languages $L_a(A)$ and $L_a(B)$ are context–sensitive. Then the languages $L_a(A + B)$, $L_a(A \cdot B)$ and $L_a(A^B)$ are also context–sensitive.*

Proof sketch. We can use the same technique as the one used in the proof of Lemma 1. We start with two length–increasing grammar, that generate $L_a(A)$

and $L_a(B)$, respectively, and construct a grammar, that generates the resulting language.

For the language $L_a(A+B)$ we construct a grammar, that first generates words, which contain two encoded representation of words from $L_a(A)$ and $L_a(B)$, separated by some markers. Then the grammar decreases the value of the first word and increases the value of the second word one by one. Finally, the grammar erases the markers, and generate a terminal word. More formally, every terminal derivation fits to the following derivation scheme:

$$\begin{aligned}
S &\Longrightarrow BS_1CMS_2E \Longrightarrow^* \\
&\Longrightarrow^* BX_{i_1}\dots X_{i_k}CMX_{j_1}\dots X_{j_l}E \Longrightarrow^* \\
&\Longrightarrow^* BX_{i'_1}\dots X_{i'_{k'}}CMX_{j'_1}\dots X_{j'_{l'}}E \Longrightarrow^* \\
&\Longrightarrow^* BCMX_{j''_1}\dots X_{j''_{l''}}E \Longrightarrow^* \\
&\Longrightarrow^* j''_1\dots j''_{l''},
\end{aligned}$$

where

$$val_a(i_1\dots i_k)+val_a(j_1\dots j_l) = val_a(i'_1\dots i'_{k'})+val_a(j'_1\dots j'_{l'}) = val_a(j''_1\dots j''_{l''}).$$

The case of $L_a(A\cdot B)$ is a little bit more complex. Again, we construct a grammar, that first generates words, which contain two encoded representation of words from $L_a(A)$ and $L_a(B)$, separated by some marker. The multiplication can be done by repeated addition. We repeatedly decrease the value of the first word by one, and add the value of the second word to a third block of the generated word, which represents the result. When the value of the first word becomes zero, we erase the special markers, the second operand, the temporary storage and generate a terminal word. The operation of the grammar can be illustrated by the following derivation scheme:

$$\begin{aligned}
S &\Longrightarrow BS_1CM_1S_2M_2X_0M_3X_0E \Longrightarrow^* \\
&\Longrightarrow^* BX_{i_1}\dots X_{i_k}CM_1X_{j_1}\dots X_{j_l}M_2X_0M_3X_0E \Longrightarrow^* \\
&\Longrightarrow^* BX_{i'_1}\dots X_{i'_{k'}}M_1X_{j'_1}\dots X_{j'_{l'}}DM_2X_{e_1}\dots X_{e_o}M_3BX_{f_1}\dots X_{f_p}E \Longrightarrow^* \\
&\Longrightarrow^* BCM_1X_{j_1}\dots X_{j_l}M_2X_0M_3BX_{f'_1}\dots X_{f'_{p'}}E \Longrightarrow^* \\
&\Longrightarrow^* f'_1\dots f'_{p'},
\end{aligned}$$

where

$$\begin{aligned}
val_a(i_1\dots i_k)\cdot val_a(j_1\dots j_l) &= val_a(i'_1\dots i'_{k'})\cdot val_a(j_1\dots j_l)+ \\
&+ val_a(j'_1\dots j'_{l'}) + val_a(f_1\dots f_p) = \\
&= val_a(f'_1\dots f'_{p'}), \text{ and} \\
val_a(j_1\dots j_l) &= val_a(j'_1\dots j'_{l'}) + val_a(e_1\dots e_0).
\end{aligned}$$

The addition itself is more complex than it was in the grammar for the language $L_a(A+B)$, because we must somehow preserve the original operand. For that after each decrementation we increment the value of two blocks: one block will denote the final result, the other one is used to maintain information on the original operand. When the block of the operand becomes empty, we have the original value of it in another block. Then we do some tricks with the markers and zero out the temporary storage.

The case of $L_a(A^B)$ requires an additional step: we do the raising to the power by repeated multiplication, the multiplication by repeated addition and the addition by repeated incrementation. The solution uses 6 blocks. In the first block there is the first operand. In the second, there is the second operand in the beginning, but it gets decremented one by one during the derivation. In the remaining 4 blocks the grammar performs a multiplication, as described before. The details are rather long and needs no new techniques, therefore it is omitted here.

For all three of the constructed grammars, because the original grammars are length–increasing, the generated word is at least as long as any of the operands, and we always use a bounded number of special symbols, the constructed grammars have a workspace which size is at most a linear function of the length of the generated word, hence the generated languages are context–sensitive. $\square$

The same constructions also work, when the representations of the operands are recursively enumerable. Now the grammars of the operands may not be length–increasing, and for that the constructed grammars may generate non–context–sensitive languages. This can be formulated as the next lemma:

Lemma 9 *Let $a \geq 1$ be an integer, and let $A, B \subseteq \mathcal{N}$ be two sets, such that the languages $L_a(A)$ and $L_a(B)$ are recursively enumerable. Then the languages $L_a(A+B)$, $L_a(A \cdot B)$ and $L_a(A^B)$ are also recursively enumerable.*

The following lemma states, that all of the Chomsky classes are closed under constant addition and multiplication.

Lemma 10 *Let $a \geq 1$ be an integer, and let $A \subseteq \mathcal{N}$ be a set, such that $L_a(A) \in \mathcal{F}$, where $\mathcal{F}$ is one of the classes $\mathcal{RE}$, $\mathcal{CS}$, $\mathcal{CF}$, $\mathcal{REG}$. Then for all $c \geq 0$ integer $L_a(c+A), L_a(c \cdot A) \in \mathcal{F}$.*

Proof sketch. The cases of $\mathcal{F} = \mathcal{CS}$ and $\mathcal{F} = \mathcal{RE}$ hold, because they are consequences of Lemmas 8 and 9, since $L_a(\{c\})$ is obviously regular. We can prove the remaining cases as follows:

It is known, that the classes $\mathcal{CF}$ and $\mathcal{REG}$ are closed under gsm–mappings and mirror images. We can easily construct two generalized sequential machines: G_1 and G_2, such that they depend on only a and c, and for all $u \in L_a(A)^{-1}$

$$val_a(G_1(u)^{-1}) = c + val_a(u^{-1}),$$

and

$$val_a(G_2(u)^{-1}) = c \cdot val_a(u^{-1}).$$

G_1 works exactly like humans do when add two numbers using paper and pencil. The constant c is coded in the internal structure of G_1, and the (mirror image) of the second operand is read from the input tape. The machine operates from right to left, digit by digit, handling the carry when necessary. The operation of G_2 differs a little from the way humans multiply: it also operates from right to left, but it multiplies one digit of the input by c and handles the carry. In this style the carry may be more than the base of the number system, but it is always less than c.

We get, that

$$\begin{aligned} L_a(c+A) &= G_1(L_a(A)^{-1})^{-1}, \\ L_a(c \cdot A) &= G_2(L_a(A)^{-1})^{-1}, \end{aligned}$$

from which the theorem follows, because of the closure properties mentioned before. □

The following lemma deals with addition and multiplication of regularly representable sets:

Lemma 11 *Let $a \geq 1$ be an integer, and $A, B \subseteq \mathcal{N}$ be two sets, such that the languages $L_a(A)$ and $L_a(B)$ are regular. Then the language $L_a(A+B)$ is also regular. However, for all $a \geq 2$ bases there exist sets $A, B \subseteq \mathcal{N}$, such that the languages $L_a(A)$ and $L_a(B)$ are regular, but the language $L_a(A \cdot B)$ is nonregular.*

Proof. We prove the first part of the lemma first. The case of $a = 1$ follows from the fact, that the class of regular languages are closed under concatenation. To prove the case of $a \geq 2$ we construct a nondeterministic finite automaton, which accepts the language $(L_a(A+B))^{-1}$. From this, the case follows, because the class of regular languages are closed under mirror images.

Let $A_1 = (T, Q_1, \delta_1, s_1, F_1)$ be a deterministic finite automaton accepting $L_a(A)^{-1}$, and let $A_2 = (T, Q_2, \delta_2, s_2, F_2)$ be a DFA accepting $L_a(B)^{-1}$, where $T = \{0, 1, \ldots, n-1\}$. The constructed NFA will be the following (f and e are new symbols):

$$A_3 = (T, Q_3, \delta_3, (s_1, s_2, 0), F_3),$$

where

$$\begin{aligned} Q_3 &= \{f\} \cup (Q_1 \cup \{e\}) \times (Q_2 \cup \{e\}) \times \{0, 1\}, \\ F_3 &= \{f\} \cup (F_1 \cup \{e\}) \times (F_2 \cup \{e\}) \times \{0\}, \end{aligned}$$

and δ_3 is determined by the following formulas ($q_1, q_1' \in Q_1$, $q_2, q_2' \in Q_2$, $t \in T$, $c \in \{0, 1\}$):

$$\begin{aligned} &\delta_3((q_1, q_2, c), t) \ni (q_1', q_2', c') \qquad \Leftrightarrow \quad \exists n, m \geq 0 \ : \\ &\qquad \delta_1(q_1, n) = q_1' \ \wedge \ \delta_2(q_2, m) = q_2' \ \wedge \\ &\qquad \wedge \ ((t = n + m + c \wedge c' = 0) \vee (a + t = n + m + c \wedge \ c' = 1)) \\ &\delta_3((q_1, q_2, c), \lambda) \ni (\{e\}, q_2, c) \qquad \Leftrightarrow \quad q_1 \in F_1 \end{aligned}$$

$$\begin{aligned}
&\delta_3((q_1,q_2,c),\lambda) \ni (q_1,\{e\},c) && \Leftrightarrow \quad q_2 \in F_2 \\
&\delta_3((q_1,\{e\},c),t) \ni (q_1',\{e\},c') && \Leftrightarrow \quad \exists n \geq 0 \ : \delta_1(q_1,n) = q_1' \wedge \\
&\qquad\qquad \wedge \ ((t = n + c \wedge c' = 0) \vee (a + t = n + c \wedge c' = 1)) \\
&\delta_3((\{e\},q_2,c),t) \ni (\{e\},q_2',c') && \Leftrightarrow \quad \exists m \geq 0 \ : \delta_2(q_2,m) = q_2' \wedge \\
&\qquad\qquad \wedge \ ((t = m + c \wedge c' = 0) \vee (a + t = m + c \wedge \ c' = 1)) \\
&\delta_3((q_1,q_2,1),1) \ni f && \Leftrightarrow \quad q_1 \in F_1 \wedge \ q_2 \in F_2
\end{aligned}$$

The constructed automaton simulate both of the original ones. It tries to guess step by step the two operands of the addition, that result in the read number. Apart from state f, its states have three components: they contain the states of the original automata, and that whether there arose a carry bit in the last addition. When one of the enclosed automata stops, then the symbol e will be included in the state, instead of the internal state of the simulated automaton.

When the automaton is in a state (q_1, q_2, c), it has the following options:

- It reads a symbol t, and guess two numbers, n and m, such that the started addition can be continued by these numbers. It feeds the two simulated automaton with n and m, respectively. The new state will consist of the new states of the included automata, and the new carry bit.

- If one of the included automata is in a final state, it can decide, that the operand belonging to that automaton has ended. Afterwards, only the other internal automaton takes part in the addition.

- If there is a carry ($c = 1$), the read symbol is 1, and both of the included automata are in a final state, A_3 can decide, that both of the operands have ended. The new state will be f. This state has no following states. With this transition we can handle the case, when there is a carry in the final addition.

The automaton accepts a word, if after reading the whole number one of the following conditions is satisfied:

- the automaton is in state f, or

- both of the included automata are in a final state, and the carry bit is 0, or

- one of the included automata is in a final state, the other one has stopped before, and the carry bit is 0.

The details of the proof, that A_3 accepts exactly $(L_a(A + B))^{-1}$ are left to the reader.

To prove the second part of the lemma, let us consider the following example:

$$A := \left\{ \frac{a^k - 1}{a - 1} \;\middle|\; k \geq 1 \right\},$$
$$B := \{a^k + 1 | k \geq 1\}.$$

$L_a(A)$ and $L_a(B)$ can be expressed by the regular expressions 1^* and 10^*1, respectively, hence they are obviously regular. Let us suppose, that the language $L = L_a(A \cdot B)$ is regular. Then we can apply to L the pumping lemma concerning regular languages (Proposition 1). Let n be the constant appearing in the lemma. Let us consider the word $u = 1^n 0 1^n$. (The word u is in L, because $1^n 0 1^n = 1^n \cdot 10^n 1$.) By the pumping lemma, we can iterate somewhere in the subword of the first n symbol of u, and for some $l \geq 1$, $u' = 1^{n+l} 0 1^n \in L$. But this is a contradiction, because u' cannot be a base–a representation of a product of proper operands. □

By this lemma, we have finished the proof of Theorem 2. Like we did for Theorem 1, we repeat the table of Theorem 2, showing the lemmas, which belongs to the cells. The assertions that belongs to cells marked with a * are trivial consequences of lemma 8.

	$c + A$	$c \cdot A$	$A + B$	$A \cdot B$	A^B
$\mathcal{REG}$	$\mathcal{REG}$ (10.)	$\mathcal{REG}$ (10.)	$\mathcal{REG}$ (11.)	$\mathcal{CS}^*$	$\mathcal{CS}^*$
$\mathcal{CF}$	$\mathcal{CF}$ (10.)	$\mathcal{CF}$ (10.)	$\mathcal{CS}^*$	$\mathcal{CS}^*$	$\mathcal{CS}^*$
$\mathcal{CS}$	$\mathcal{CS}$ (10.)	$\mathcal{CS}$ (10.)	$\mathcal{CS}$ (8.)	$\mathcal{CS}$ (8.)	$\mathcal{CS}$ (8.)
$\mathcal{RE}$	$\mathcal{RE}$ (10.)	$\mathcal{RE}$ (10.)	$\mathcal{RE}$ (9.)	$\mathcal{RE}$ (9.)	$\mathcal{RE}$ (9.)

As a corollary of Theorem 2 we get results, which in some sense extend the Theorem of Horváth about the ranges of polynomials published in [13]:

Theorem 3 *Let $a \geq 1$ be a positive integer, $A_0, A_1 \subseteq \mathcal{N}$ finite sets, $X \subseteq \mathcal{N}$ a set for which $L_a(X) \in \mathcal{F}$, where $\mathcal{F}$ is one of the classes $\mathcal{RE}, \mathcal{CS}, \mathcal{CF}, \mathcal{REG}$. Then $L_a(A_1 \cdot X + A_0) \in \mathcal{F}$.*

Proof. It follows from the definitions, that

$$A_1 \cdot X + A_0 = \bigcup_{a_1 \in A_1, a_0 \in A_0} a_1 \cdot X + a_0.$$

By Theorem 10, we know that $a_1 \cdot X + a_0 \in \mathcal{F}$ for all $a_1, a_0 \geq 0$. From that the theorem follows, because all of the mentioned classes are closed under (finite) union. □

For higher „dimensions” we get weaker results, again as a direct consequence of Theorem 10.

Theorem 4 *Let $a \geq 1$ be a positive integer, $A_0, A_1, \ldots A_n \subseteq \mathcal{N}$ finite sets, $X \subseteq \mathcal{N}$ a set for which $L_a(X) \in \mathcal{F}$, where $\mathcal{F}$ is one of the classes $\mathcal{RE}, \mathcal{CS}$. Then $L_a(A_n X^n + \cdots + A_1 \cdot X + A_0) \in \mathcal{F}$.*

5 Conclusions

In this paper we have examined two aspects of the connection of formal languages and number theory. We got results for the problem of converting between canonical number systems and performing arithmetic operations on sets, but there are many other open problems in this domain, we mentioned some in the text. We believe, that further research of this area can be fruitful for both fields.

Acknowledgement
The author wishes to thank László Hunyadvári for his useful suggestions.

References

[1] Alan Cobham. On the base-dependance of sets of numbers recognizable by finite automata. *Mathematical Systems Theory*, 3(2):186–192, 1969.

[2] Karel Culik II and Arto Salomaa. Ambiguity and decision problems concerning number systems. *Information and Control*, 56(3):139–153, 1983.

[3] Aldo de Luca and Antonio Restivo. Star-free sets of integers. *Theoretical Computer Science*, 43(2-3):265–275, 1986.

[4] Juha Honkala. Bases and ambiguity of number systems. *Theoretical Computer Science*, 31(1-2):61–71, May 1984.

[5] Juha Honkala. A decision method for the recognizability of sets defined by number systems. *RAIRO Inform. Théor. Appl.*, 20(4):395–403, 1986.

[6] Juha Honkala. A decision method for the unambiguity of sets defined by number systems. *The Journal of Universal Computer Science*, 1(9):648–653, September 1995.

[7] Christian Michaux and Roger Villemaire. Presburger arithmetic and recognizability of sets of natural numbers by automata: New proofs of Cobham's and Semenov's theorems. *Annals of Pure and Applied Logic*, 77(3):251–277, 19 February 1996.

[8] Arto Salomaa. *Formal Languages*. Academic Press, New York, 1973.

[9] J. Büchi. Weak second order logic and finite automata. *Z. Math. Logik, Grundlag. Math.*, 5:66–62, 1960.

[10] Grzegorz Rozenberg and Arto Salomaa. *Handbook of Formal Languages*. Springer Verlag, Berlin, Heidelberg, New York., 1997.

[11] Yehoshua Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonologie, Sprachwissenschaft und Kommunikationsforschung*, 14:113–124, 1961.

[12] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, N. Reading, MA, 1980.

[13] Horváth Sándor. Ranges of polynomials are non–context–free in any number system. To appear in: Pure Mathematics and Applications.