VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

A RESTRICTION OF SENTETIAL FORMS OF SCATTERED CONTEXT GRAMMARS

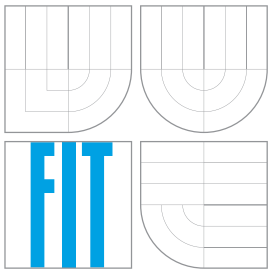
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ ŠIMÁČEK

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

OMEZENÍ VĚTNÝCH FOREM GRAMATIK S ROZPTÝLENÝM KONTEXTEM

A RESTRICTION OF SENTETIAL FORMS OF SCATTERED CONTEXT GRAMMARS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — JIŘÍ ŠIMÁČEK
AUTHOR

VEDOUCÍ PRÁCE — prof. RNDr. ALEXANDER MEDUNA, CSc.
SUPERVISOR

BRNO 2008

Abstrakt

Tato práce zavádí pojem zobecněných gramatik s rozptýleným kontextem, které se od tradičních liší tím, že levé strany pravidel mohou obecně obsahovat řetězec neterminálních symbolů namísto jediného neterminálu. Dále jsou studovány dva typy omezení při větné derivaci v těchto gramatikách. Nechť k je konstanta. První omezení požaduje, aby přepsání všech symbolů nastalo mezi prvními k symboly v prvním souvislém bloku neterminálů ve větné formě v každém derivačním kroku. Druhé omezení definuje derivaci pouze nad větnými formami, které obsahují nejvýše k výskytů neterminálů. Jako hlavní výsledek práce demonstruje, že oba typy omezení snižují vyjadřovací sílu na úroveň bezkontextových gramatik.

Klíčová slova

gramatiky s rozptýleným kontextem, zobecnění gramatik, omezení větné derivace, vyjadřovací síla

Abstract

This work introduces and discusses generalized scattered context grammars that are based upon sequences of productions whose left-hand sides are formed by nonterminal strings, not just single nonterminals. It places two restrictions on the derivations in these grammars. More specifically, let k be a constant. The first restriction requires that rewriting all symbols occurs within the first k symbols of the first continuous block of nonterminals in the sentential form during every derivation step. The other restriction defines the derivations over sentential forms containing no more than k occurrences of nonterminals. As its main result, the thesis demonstrates that both restrictions decrease the generative power of these grammars to the power of context-free grammars.

Keywords

scattered context grammars, grammatical generalization, derivation restriction, generative power

Citace

Jiří Šimáček: A Restriction of Sentetial Forms of Scattered Context Grammars, diplomová práce, Brno, FIT VUT v Brně, 2008

A Restriction of Sentetial Forms of Scattered Context Grammars

Declaration

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged. The thesis has been created under the supervision of prof. RNDr. Alexander Meduna, CSc.

.....
Jiří Šimáček
May 7, 2008

Acknowlegement

Author wishes to thank prof. RNDr. Alexander Meduna, CSc. and Mgr. Tomáš Masopust, Ph.D. for their contributions to this work.

Contents

Chapter 1

Introduction

Scattered context grammars are based upon finite sets of sequences of context-free productions having a single nonterminal on the left-hand side of every production (see [5]). According to a sequence of n context-free productions, these grammars simultaneously rewrites n nonterminals in the current sentential form according to the n productions in the order corresponding to the appearance of these productions in the sequence. It is well-known that they characterize the family of recursively enumerable languages (see [9]).

In this work, we generalize these grammars so that the left-hand side of every production may consist of a string of several nonterminals rather than a single nonterminal. Specifically, we discuss two derivation restrictions in scattered context grammars generalized in this way. To explain these restrictions, let k be a constant. The first restriction requires that all simultaneously rewritten symbols occur within the first k symbols of the first continuous block of nonterminals in the current sentential form during every derivation step. The other restriction defines the grammatical derivations over sentential forms containing no more than k occurrences of nonterminals.

As the main result, we demonstrate that both restrictions decrease the generative power of generalized scattered context grammars to the generative power of context-free grammars. As ordinary scattered context grammars represent special cases of their generalized versions, they also characterize only the family of context-free languages if they are restricted in this way.

This result concerning the derivation restrictions is of some interest when compared to analogical restrictions in terms of other grammars working in a context-sensitive way. Over its history, formal language theory has studied many restrictions placed on the way grammars derive sentential forms and on the forms of productions. In [7], Matthews studied derivations of grammars in the strictly leftmost (rightmost) way—that is, rewritten symbols are preceded (succeeded) only by terminals in the sentential form during the derivation. Later, in [8], he combined both approaches—leftmost and rightmost derivations—so that any sentential form during the derivation is of the form xWy, where x and y are terminal strings, W is a nonterminal string, and a production is applicable only to a leftmost or rightmost substring of W. In both cases, these restrictions result into decreasing the generative power of type-0 grammars to the power of context-free grammars.

Whereas Matthews studied restrictions placed on the forms of derivations, other authors studied the forms of productions. In [2], Book proved that if the left-hand side of any non-context-free production contains besides exactly one nonterminal only terminals, then the generative power of type-0 grammars decreases to the power of context-free grammars. He also proved that if the left-hand side of any non-context-free production has as its left

context a terminal string and the left context is at least as long as the right context, then the generative power of type-0 grammars decreases to the power of context-free grammars, too. In [4], Ginsburg and Greibach proved that if the left-hand side of any production is a nonterminal string and the right-hand side contains at least one terminal, then the generated language is context-free. Finally, in [1], Baker proved a stronger result. This result says that if any left-hand side of a production either has, besides terminals, only one nonterminal, or there is a terminal substring, β, on the right-hand side of the production such that the length of β is greater than the length of any terminal substring of the left-hand side of the production, then the generative power of type-0 grammars decreases to the power of context-free grammars. For more details, see page 198 in [11] and the literature cited there.

This work is based on the paper *Two Power-Decreasing Derivation Restrictions in Generalized Scattered Context Grammars* [6], which was written in cooperation with Mgr. Tomáš Masopust, Ph.D. and prof. RNDr. Alexander Meduna, CSc.

Chapter 2

Definitions

This chapter presents definitions of the terms which are used in this work. Although it is brief in some cases, it aims to be a complete reference. Depending on the reader, it can be necessary to consult additional sources.

2.1 Basic Definitions

This section provides the mathematical background which serves as the basis for the advanced formalisms.

2.1.1 Sets

A *set* is one of the most fundamental mathematical concept, which was developed by a German mathematician Georg Cantor in 19th century. Originally, he gave the following definition:

> By a *set* we mean any collection M into a whole of definite, distinct objects m (which are called the *elements* of M) of our perception (Anschauung in the original) or of our thought.

Unfortunately, informal definitions of this kind suffers from certain issues, such as *Russell's paradox*. Consider a set

$$R = \{x : x \text{ is a set and } x \text{ doesn't contain } x\},$$

which is the set of all sets which do not contain themselves as members. By the definition of R, one can rule out these consequences:

1. R contains R implies R doesn't contain R,
2. R doesn't contain R implies R contains R.

Although precisely stated, R doesn't appear to be well defined set.

Since then, there was an effort to solve these problems such as the formulation of *axiomatic set theory*; however, when we deal with issues concerning theoretical computer science, these problems usually don't arise and we will use the following definition, which is very similar to Cantor's one.

Definition 1. A set is a group of distinct objects considered as a whole. These objects are called *elements* of the set.

The set can be described in two ways. Either by specifying members of the set explicitly ($S = \{3, 5, 7, 11\}$) or by defining properties of the elements in the set ($S = \{x : x \text{ is even }\}$). The latter way is usual in cases, in which the set contains infinitely many elements.

To express that a set S contains an element a, we write

$$a \in S.$$

Subsets

If B contains all elements contained in A, then A is said to be a *subset* of B, symbolically

$$A \subseteq B.$$

Also B is said to be a *superset* of A ($B \supseteq A$). Additionally A is a *proper subset* of B ($A \subset B$) if $A \subseteq B$ and there exists an element x, such that $x \notin A$ and $x \in B$. We define $B \supset A$ analogously. In cases, where $A \subseteq B$ and $B \subseteq A$ at the same time, A and B contains exactly the same elements and we write

$$A = B.$$

Cardinality

The *cardinality* of a set S is the number of elements in S, written as

$$|S|.$$

If S contains no elements ($|S| = 0$), then S is called *empty set* and denoted by the symbol $\emptyset$. Moreover,

$$A = B \text{ implies } |A| = |B|,$$

which means that

$$|A| \neq |B| \text{ implies } A \neq B.$$

Note that there are infinite sets which have infinite cardinalities, but these cardinalities can differ (for example cardinality of the set of all natural numbers is different from the cardinality of the set of all *real* numbers).

Power Sets

Often we want to refer to a set of all subsets of A. This is denoted as *power set*. To express that B is power set of A, we write $B = 2^A$. If $|A|$ is finite, then $|B| = 2^{|A|}$. If $B = 2^A$, then $\emptyset \in B$ and $A \in B$. Note that

$$2^\emptyset = \{\emptyset\} \neq \emptyset.$$

Fundamental Set Operations

Given a group of sets, one can construct a new set using certain operations over sets. We can take two sets (A and B) and "add" them together, effectively creating a new set which contains all elements of the original sets. This operation is called *union*, written as

$$A \cup B,$$

and the resulting set is defined as

$$A \cup B = \{x : x \in A \text{ or } x \in B\}.$$

Unions have the following properties:

- $A \cup B = B \cup A$,
- $A \subseteq (A \cup B)$,
- $A \cup A = A$,
- $A \cup \emptyset = A$,
- $A \subseteq B$ if and only if $A \cup B = B$.

Also we can introduce *intersection* which creates the set that contains only those elements contained in both sets. To denote this fact, we write

$$A \cap B.$$

The resulting set is defined as

$$A \cap B = \{x : x \in A \text{ and } x \in B\}.$$

Intersections have the following properties:

- $A \cap B = B \cap A$,
- $(A \cap B) \subseteq A$,
- $A \cap A = A$,
- $A \cap \emptyset = \emptyset$,
- $A \subseteq B$ if and only if $A \cap B = A$.

The two sets can be "subtracted". This operation is called *set difference* (or *relative complement*) and creates a new set that contains only those elements which are in the first set, but not in the second one. The set difference is denoted by

$$A \setminus B \text{ or by } A - B$$

and defined as

$$A \setminus B = \{x : x \in A \text{ and } x \notin B\}.$$

If there is some set U which represents the universe that contains all elements we are dealing with, we can introduce *complement* (or *absolute complement*) of a set, written as

$$A'.$$

The complement is defined as

$$A' = U \setminus A.$$

Complement operations have the following properties:

- $A \cup A' = U$,
- $A \cap A' = \emptyset$,
- $(A')' = A$,
- $A \setminus A = \emptyset$,
- $A \setminus B = A \cap B'$.

An *ordered pair* is a collection of two elements a and b denoted by (a, b) such that (a, b) is different from (b, a), unless $a = b$. In other words, let (a, b) and (c, d) be two ordered pairs. Then,

$$(a, b) = (c, d) \text{ if and only if } a = c \text{ and } b = d.$$

An ordered pair can contain other ordered pairs as the components, so it is possible to define *n-tuple* in the following way:

$$(x_1, x_2, \ldots, x_n) = (x_1, (x_2, \ldots, x_n)).$$

$A \times B$ denotes the *Cartesian product* which is set of all ordered pairs, such that the first component is the element of A and the second one is the element of B, formally

$$A \times B = \{(a, b) : a \in A \text{ and } b \in B\}.$$

The Cartesian product has these properties:

- $A \times \emptyset = \emptyset$,
- $A \times (B \cup C) = (A \times B) \cup (A \times C)$,
- $|A \times B| = |A| \cdot |B|$.

2.1.2 Relations

Another very important mathematical concept, which is also crucial for the theory of formal languages, is *relation*.

Definition 2. A *n-ary relation* ρ over the sets $A_1, \ldots, A_n$ is a subset of the Cartesian product of these sets:

$$\rho \subseteq A_1 \times \cdots \times A_n.$$

If $A_1 = A_2 = \cdots = A_n = A$, we can write

$$\rho \subseteq A^n$$

instead of $\rho \subseteq A_1 \times \cdots \times A_n$.

Example 1. Consider a binary operation $\wedge$ (logical *and*) defined over $A = \{true, false\}$. $\wedge \subseteq A^3$ can be represented by the ternary relation:

$$\wedge = \{(false, false, false), (false, true, false), (true, false, false), (true, true, true)\}.$$

Binary Relations

The case, where $n = 2$, is very common and in such situations we use the term *binary relation*. Let $\rho \subseteq A \times B$ be a binary relation. A is called *domain* of ρ, B is called *range*. In cases, where $A = B$, ρ is said to be *relation on* A (B). The *inverse* of ρ is defined as

$$\rho^{-1} = \{(b, a) : (a, b) \in \rho\}.$$

To denote that $(a, b) \in \rho$, we can write $a\rho b$, and $\neg(a\rho b)$ or, sometimes, $a\not\rho b$ to denote that $(a, b) \notin \rho$.

Example 2. Let $A = \{1, 2, \ldots, 5\}$. We can define property "being a divisor of" (which is denoted by $|$) over the set A in the following way:

$$| = \{(1,1), (1,2), (1,3), (1,4), (1,5), (2,2), (2,4), (3,3), (4,4), (5,5)\}.$$

Note that such relations are often defined over infinite sets, where listing each member of the set becomes impractical, so these relations are better specified by their properties. The general definition of relation $| \subseteq \mathbb{N}^2$ ($\mathbb{N}$ refers to the set of all natural numbers) can be following:

$$| = \{(a, b) \in \mathbb{N}^2 : b = ka \text{ for some } k \in \mathbb{N}\}.$$

When dealing with the binary relations, we can investigate various properties. Assume $\rho \in A \times A$, then:

- ρ is *reflexive* if $a\rho a$ for each $a \in A$,
- ρ is *irreflexive* if $\neg(a\rho a)$ for each $a \in A$,
- ρ is *symmetric* if $a\rho b$ implies $b\rho a$ for each $a, b \in A$,

- ρ is *asymmetric* if $a\rho b$ implies $\neg(b\rho a)$ for each $a, b \in A$,
- ρ is *antisymmetric* if $a\rho b$ and $b\rho a$ implies $a = b$ for each $a, b \in A$,
- ρ is *transitive* if $a\rho b$ and $b\rho c$ implies $a\rho c$ for each $a, b, c \in A$.

There are two important classes of relations:

- *equivalence* relations—these are reflexive, symmetric and transitive. A typical representative of this class is *equality relation* (=) defined over natural numbers. Note that there also exists *identity relation*, which is somewhat stronger than general equality. a and b are said to be equal, if they share the same properties, but they are identical if they refer to the same entity; however, the identity and the equality actually refer to the same relation in most cases.
- *order* relations—these are reflexive, antisymmetric and transitive. A typical representative of this class is order $\leq$ of natural numbers.

Fundamental Relation Operations

Given a group of relations, one can construct new relations using operations over these relations. Besides the set operations such as the union, there are certain operations which are unique to relations. One of these is the operation of a composition.

Definition 3. Let $\phi \in A \times B$, $\psi \in B \times C$ be two binary relations. Then, composition of ϕ and ψ, denoted

$$\psi \circ \phi,$$

is defined as

$$\psi \circ \phi = \{(a, c) \in A \times C : (a, b) \in \phi \text{ and } (b, c) \in \psi \text{ for some } b \in B\}.$$

The composition has following properties:

- $T \circ (S \circ R) = (T \circ S) \circ R$,
- $(S \circ R)^{-1} = R^{-1} \circ S^{-1}$.

Closures of Relations

Other interesting operations are *closures* of relations. Closure is a new relation that contains the original relation and possesses some additional property, for example transitivity, at the same time.

Definition 4. Let ρ be a relation on a set A and k be a natural number. The *k-fold product* of ρ, written as

$$\rho^k,$$

is defined as

$$\rho^k = \overbrace{\rho \circ \rho \circ \cdots \circ \rho}^{k}.$$

Definition 5. The *transitive closure* of ρ, denoted as

$$\rho^+,$$

is defined in the following way:

$$a\rho^+b \text{ if and only if } a\rho^i b$$

for some $i \geq 1$. This means that there exists $c_1, \ldots, c_n \in A$ for some $n \geq 0$, such that

$$a\rho c_1, c_1\rho c_2, \ldots, c_n\rho b.$$

Alternatively, transitive closure can be defined as

$$\rho^+ = \bigcup_{i\geq 1} \rho^i.$$

Definition 6. The *reflexive and transitive closure* of ρ, denoted as

$$\rho^*,$$

is defined as

$$a\rho^*b \text{ if and only if } a\rho^i b$$

for some $i \geq 0$, where $a\rho^0 b$ is identity relation over A. Alternatively,

$$\rho^* = \bigcup_{i\geq 0} \rho^i.$$

2.2 Basic Concepts of the Theory of Formal Languages

This section introduces fundamental terms which are used throughout the rest of this work.

2.2.1 Alphabet and Strings

The following part is adapted from the chapter 1 of [10].

Definition 7. An *alphabet* is a finite, nonempty set of elements, which are called *symbols*.

Example 3. Let

$$\Sigma_{Eng} = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}.$$

Then, Σ_{Eng} is an alphabet; moreover, Σ_{Eng} contains all lowercase letters of English alphabet.

A *string* is a sequence of symbols, sometimes denoted also as *word*. The string that contains no symbols, denoted by ε, is called *empty string*. By using recursive definition, one can define string in the following way:

Definition 8. Let Σ be an alphabet:

1. ε is a string over Σ,
2. if x is a string over Σ and $a \in \Sigma$, then xa is a string over Σ.

Example 4. Consider the alphabet Σ_{Eng}. Then,

$$article, author$$

are strings over Σ_{Eng}.

The length of x is the number of all symbols in x.

Definition 9. Let x be a string over Σ. The *length* of x, denoted as $|x|$, is defined as follows:

1. if $x = \varepsilon$, then $|x| = 0$,
2. if $x = ay$, where $a \in \Sigma$ and y is a string over Σ, then $|x| = 1 + |y|$.

Note that length of a string has similar properties to those of cardinality of a set, particularly

$$(x = y \text{ implies } |x| = |y|) \Leftrightarrow (|x| \neq |y| \text{ implies } x \neq y).$$

We use $\text{occur}(W, x)$ to denote the number of occurrences of symbols from some predefined set W.

Definition 10. Let x be a string over Σ and $W \subseteq \Sigma$. $\text{occur}(W, x)$ is defined in the following way:

1. $\text{occur}(W, \varepsilon) = 0$,
2. $\text{occur}(W, ax) = 1 + \text{occur}(W, x)$, if $a \in W$,
3. $\text{occur}(W, ax) = \text{occur}(W, x)$, if $a \notin W$.

Then, $\#_a x$ denotes the number of occurrences of the symbol a in x:

$$\#_a x = \text{occur}(\{a\}, x).$$

Additionally, $\text{alph}(x)$ denotes the set of all symbols appearing in x:

$$a \in \text{alph}(x) \Leftrightarrow \#_a x \geq 1$$

Operations on Strings

Definition 11. Let x and y be two strings over Σ. Then, xy denotes the *concatenation* of x and y.

The concatenation has the following properties:

- $x\varepsilon = \varepsilon x = x$,
- $x(yz) = (xy)z$,
- $|xy| = |x| + |y|$.

Definition 12. Let x and y be two strings over Σ. For $i \geq 0$, the ith *power* of x, denoted as x^i, is defined in following way:

1. $x^0 = \varepsilon$,
2. $x^i = xx^{i-1}$, for $i \geq 1$.

The power of a string has these properties:

- $x^i x^j = x^j x^i$,
- $x^i x^j = x^{i+j}$,
- $(x^i)^j = x^{ij}$,

where $i, j \geq 0$.

The reversal of a string is the string, where the symbols of the original string appear in the reverse order.

Definition 13. Let x be a string over Σ. The *reversal* of x, denoted as $(x)^R$, is defined in this way:

1. if $x = \varepsilon$, then $(x)^R = \varepsilon$,
2. if $x = ay$, where $a \in \Sigma$ and y is a string over Σ, then $(x)^R = (y)^R a$.

The reversal of a string has the following properties:

- $x = y$ if and only if $(x)^R = (y)^R$,
- $((x)^R)^R = x$,
- $(xy)^R = (y)^R(x)^R$.

Definition 14. Let x and y be two words over Σ. Then, x is a *prefix* of y if there exists a string, z, over Σ, such that $xz = y$; moreover, if $x \notin \{\varepsilon, y\}$, then x is a *proper prefix* of y.

For a string y, prefix(y) denotes the set of all prefixes of y:

$$\text{prefix}(y) = \{x : x \text{ is a prefix of } y\}.$$

Definition 15. Let x and y be two words over Σ. Then, x is a *suffix* of y if there exists a string, z, over Σ, such that $zx = y$; moreover, if $x \notin \{\varepsilon, y\}$, then x is a *proper suffix* of y.

For a string y, suffix(y) denotes the set of all suffixes of y:

$$\text{suffix}(y) = \{x : x \text{ is a suffix of } y\}.$$

Definition 16. Let x and y be two words over Σ. Then, x is a *substring* of y if there exists two strings, z and z', over Σ, such that $zxz' = y$; moreover, if $x \notin \{\varepsilon, y\}$, then x is a *proper substring* of y.

For a string y, substring(y) denotes the set of all substrings of y:

$$\text{substring}(y) = \{x : x \text{ is a substring of } y\}.$$

The prefix, suffix and substring have the following properties:

- $\text{prefix}(x) \subseteq \text{substring}(x)$,
- $\text{suffix}(x) \subseteq \text{substring}(x)$,
- $\{\varepsilon, x\} \subseteq \text{prefix}(x) \cap \text{suffix}(x) \cap \text{substring}(x)$.

2.2.2 Language

Let Σ be an alphabet. Then,

$$\Sigma^*$$

denotes the set of all strings over Σ. Set

$$\Sigma^+ = \Sigma^* - \{\varepsilon\}.$$

In other words, Σ^+ denotes the set of all nonempty strings over Σ. Now, we can define a language as a set of strings over Σ.

Definition 17. Let Σ be an alphabet, and let $L \subseteq \Sigma^*$. Then, L is a *language* over Σ.

For every alphabet, Σ, Σ^* represents a language over Σ. Σ^* contains all strings over Σ and it is called *universal language*. By the above definition $\emptyset$ and $\{\varepsilon\}$ are languages over any alphabet. Note that

$$\emptyset \neq \{\varepsilon\}.$$

Example 5. Let

$$L_{title} = \{a, context, forms, grammars, of, restriction, scattered, sentential\}.$$

Then, L is a language over Σ_{Eng} which contains all words (strings) that appear in the title of this work.

Definition 18. Let L be a language. L is *finite* if $|L| = n$, for some $n \geq 0$; otherwise, L is *infinite*.

Example 6. Consider a language of all sentences which appear in this thesis. This language is finite; however, the language of all meaningful English sentences is infinite (with assumption that there is no limit on the length of the sentences).

While a finite language can be fully specified by enumerating all strings the language contains, this is not possible for infinite languages. In such cases, we require models which can describe an infinite language by the finite way. Some of these models are described in the next section.

2.3 Standard Language Models and Language Classification

This section describes some of the basic models of languages.

2.3.1 Phrase-structure Grammars

A *phrase-structure grammar* is one of the fundamental models of formal languages. It comprises of a set of working symbols called *total alphabet*, a set of symbols that appear in the strings of a given language, called *terminals*, a set of rules, which controls, how the words of that language are generated, and a starting symbol.

Definition 19. A *phrase-structure grammar* or, simply, a *grammar* is a quadruple $G = (V, T, P, S)$, where

1. V is a *total alphabet*,
2. $T \subseteq V$ is an *alphabet of terminals*,
3. $S \in (V - T)$ is the *starting symbol*,
4. $P \in (V^*(V - T)V^* \times V^*)$ is a finite relation.

$N = V - T$ is the alphabet of *nonterminals*. Elements of P are called *productions*, so P is referred to as the *set of productions*. Instead of $(u, v) \in P$, we can write

$$u \to v \in P$$

or

$$u \to v,$$

if P is grounded. u is called *left-hand side* of the production, v is called *right-hand side*. Note that according to the definition, each production contains at least one nonterminal on its left-hand side.

Definition 20. Let $G = (V, T, P, S)$ be a phrase-structure grammar, $\alpha \to \beta \in P$, $x, y \in V^*$. Then, $x\alpha y$ *directly derives* $x\beta y$ in G, denoted as

$$x\alpha y \Rightarrow x\beta y \ [\alpha \to \beta],$$

or

$$x\alpha y \Rightarrow x\beta y.$$

A relation $\Rightarrow \in V^* \times V^*$ is called a *direct derivation*. In other words, G makes a *derivation step* from $x\alpha y$ to $x\beta y$ according to $\alpha \to \beta$.

To denote that the derivation step was performed by G, we write

$$x\alpha y \Rightarrow_G x\beta y.$$

To generate certain string, we start with the starting symbol, S, of the grammar, G, as the current *sentential form*. Then, we attempt to rewrite it according to some production from P. This succeeds, if there exists some production with S on its left-hand side. By rewriting S by the right-hand side of the selected production, we obtain new sentential

form. The corresponding string can be further processed in the same way, until there is no rule with the left-hand side that appears somewhere within the sentential form. Note that the choice of the productions as well as the selection of the substring in the sentential form, which is being rewritten, is strictly nondeterministic. In fact, such choice affects, what (if any) terminal string is being generated.

Definition 21. Let $G = (V, T, P, S)$ be a phrase-structure grammar. Then,

$$\Rightarrow^n$$

denotes n-fold product (recall previous definitions) of $\Rightarrow$. Moreover,

$$\Rightarrow^+ \text{ and } \Rightarrow^*$$

denote the transitive closure of $\Rightarrow$ and transitive and reflexive closure of $\Rightarrow$, respectively.

Now, we can specify the language generated by G (the language that G models) by the following definition.

Definition 22. Let $G = (V, T, P, S)$ be a phrase-structure grammar. The language of G, symbolically

$$\mathcal{L}(G),$$

is defined as

$$\mathcal{L}(G) = \{w \in T^* : S \Rightarrow^* w\}.$$

In other words, whenever G finishes derivation of the sentential form with a string that comprises of the symbols from T exclusively, then that string belongs to the language described by G.

The following example shows the possible process of generating of the single string in detail.

Example 7. Let $G = (\{S, a, b\}, \{a, b\}, P, S)$, where

$$P = \{S \to aSb, S \to \varepsilon\}.$$

Grammar G generates the string *aaaaabbbbb* by the following sequence of productions:

$$\begin{array}{rcll} S & \Rightarrow & aSb & [\ S \to aSb\ ] \\ & \Rightarrow & aaSbb & [\ S \to aSb\ ] \\ & \Rightarrow & aaaSbbb & [\ S \to aSb\ ] \\ & \Rightarrow & aaaaSbbbb & [\ S \to aSb\ ] \\ & \Rightarrow & aaaaaSbbbbb & [\ S \to aSb\ ] \\ & \Rightarrow & aaaaabbbbb & [\ S \to \varepsilon\ ] \end{array}$$

It can be easily observed that $\mathcal{L}(G) = \{a^n b^n : n \geq 0\}$.

Note that this language is typical representative for the class of *context-free languages*, which is discussed later.

2.3.2 Chomsky Hierarchy

A phrase-structure grammar, as defined above, describes a broad range of languages. According to different restrictions, which can be placed on such grammar, one can obtain system of several classes of languages called *Chomsky hierarchy* (see [11]). There are 4 language families altogether:

1. *recursively enumerable* or *type-0* languages
2. *context-sensitive* or *type-1* languages
3. *context-free* or *type-2* languages
4. *regular* or *type-3* languages

A language is of type-0, if it can be generated by an unrestricted phrase-structure grammar. This class contains all languages that can be described by some finite model. Another formalism, which can describe all recursively enumerable languages, is *Turing machine* (see [10]). Because the knowledge of this concept is not related to the results, which are discussed in this work, the details of Turing machines are omitted. Note that there are also languages beyond the class of type-0 languages. An example of such language is a language of all programs that halts for a given input. The family of recursively enumerable languages is denoted as **RE**.

The class of context-sensitive languages contains any language which can be generated by a grammar $G = (V, T, P, S)$, where all productions of P are of the form

$$\alpha A \beta \rightarrow \alpha \gamma \beta,$$

where $\alpha, \beta \in V^*$, $A \in (V - T)$, $\gamma \in V^+$ or, alternatively, all productions

$$\alpha \rightarrow \beta$$

satisfies

$$|\alpha| \leq |\beta|.$$

This type of grammar is called a *context-sensitive grammar*.

Example 8. Let $G = (\{B, C, S\}, \{a, b, c, d\}, P, S)$, where

$$P = \{S \rightarrow aBSCd, Ba \rightarrow aB, dC \rightarrow Cd, S \rightarrow abcd, Bb \rightarrow bb, cC \rightarrow cc\}.$$

The following lines demonstrate derivation of the string $aabbccdd$ in G:

$$\begin{array}{rcll} S & \Rightarrow & aBSCd & [\ S \rightarrow aBSCd\ ] \\ & \Rightarrow & aBabcdCd & [\ S \rightarrow abcd\ ] \\ & \Rightarrow & aaBbcdCd & [\ Ba \rightarrow aB\ ] \\ & \Rightarrow & aabbcdCd & [\ Bb \rightarrow bb\ ] \\ & \Rightarrow & aabbcCdd & [\ dC \rightarrow Cd\ ] \\ & \Rightarrow & aabbccdd & [\ cC \rightarrow cc\ ] \end{array}$$

By analysing G, one can rule out that $\mathcal{L}(G) = \{a^n b^n c^n d^n : n \geq 1\}$.

The abovementioned example shows, what the *context* means in the terms of phrase-structure grammar. In this case, the single derivation step is affected by more than one

symbol in the sentential form at a time. By using context-sensitive productions, we can, for example, shift the given symbol to the left or to the right within the sentential form, which would be otherwise impossible. The family of context-sensitive languages is denoted as **CS**.

By restricting productions in P of a grammar $G = (V, T, P, S)$ to the form:

$$A \rightarrow \alpha,$$

where $A \in (V - T)$, $\alpha \in V^*$, we obtain a *context-free grammar*, which is a model of any language that belongs to the family of context-free languages. A context-free language can also be described by a *pushdown automaton*, which is discussed later. This family is very interesting from the practical point of view, because certain subclass of type-2 languages (deterministic context-free languages—see [10]) can be analysed very effectively. This is the reason, why the most programming languages are described by a context-free grammar at the syntax level. The family of context-free languages is denoted as **CF**.

Any type-3 language can be generated by a phrase-structure grammar $G = (V, T, P, S)$, where P contains only productions of the form

$$A \rightarrow aB$$

or

$$A \rightarrow a,$$

where $A, B \in (V - T)$, $a \in T$. Such grammar is sometimes denoted as *right-linear grammar*. There also exists *left-linear grammar* which is defined in similar way and it has the same generative power. A regular languages can be described by a *finite state automaton* or by a *regular expression* (see [10]). Both models are omitted here, because of they are not related to this work as well. The family of regular languages is denoted as **REG**.

Besides the 4 classes mentioned above, there is a family of *recursive* languages. These are languages that can be described by a Turing machine which always halts for any given input (see [10]). Although the Chomsky hierarchy is very often used to classify languages and their models, it is not always the most suitable one. For example *L-systems* (see [11]), which always rewrites all symbols in the sentential form in the single derivation step (in parallel), create the completely different hierarchy of languages and such hierarchy is not comparable to the Chomsky one.

2.3.3 Pushdown Automata

Another possible model for context-free languages is a *pushdown automaton* (see [10]). A pushdown automaton comprises of a finite state control, additional memory called a *stack* or a *pushdown* and a set of rules which defines, how the state of the automaton and its stack are modified during the computation over an input string.

Definition 23. A *pushdown automaton*, M, is a septuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $q_0 \in Q$ is the initial state,
4. Γ is a finite pushdown alphabet,

5. $\delta \in (\Gamma Q(\Sigma \cup \{\varepsilon\}) \times (\Gamma^* Q)$ is a finite relation,

6. $Z_0 \in \Gamma$ is the initial pushdown symbol, and

7. $F \subseteq Q$ is a set of final states.

Elements of δ are called *rules* and δ is referred to as the *set of rules*. Instead of $(u, v) \in \delta$, we can write

$$u \to v \in \delta$$

or

$$u \to v,$$

if δ is grounded. u is called *left-hand side* of the rule, v is called *right-hand side*.

Definition 24. A *configuration* of a pushdown automaton M is any string of the form

$$\gamma q s,$$

where $\gamma \in \Gamma^*$ holds the content of the stack, $q \in Q$ describes the current state and $s \in \Sigma^*$ is the remaining (unread) part of the input string.

Definition 25. Let $xAqay$ be a configuration of M, where $x \in \Gamma^*$, $A \in \Gamma$, $q \in Q$, $a \in \Sigma$ and $y \in \Sigma^*$, and $Aqa \to \gamma p \in \delta$ be a rule. Then, M makes a *move* from $xAqay$ to $x\gamma py$ according to $Aqa \to \gamma p$, written as

$$xAqay \Rightarrow x\gamma py \ [Aqa \to \gamma p]$$

or

$$xAqay \Rightarrow x\gamma py.$$

A relation $\Rightarrow \in (\Gamma^* Q \Sigma^*)^2$ is called a *transition relation* and it is sometimes denoted also by the symbol $\vdash$.

To denote that the move was performed by M, we write

$$xAqay \Rightarrow_M x\gamma py.$$

For a given input string, s, M starts the computation in the configuration $Z_0 q_0 s$. Then, M subsequently selects rules from δ (one at a time) according to the topmost pushdown symbol, current state and the first symbol in the remaining part of the input string. Alternatively, it can make a move by modifying the current state or the stack without reading a symbol from the input. To apply the rule $Aqa \to \gamma p \in \delta$, the topmost pushdown symbol, A, is replaced by γ and the current state, q, is changed to p, while a is removed from the input. M continues processing until it reaches the configuration, in which no next move is possible.

Definition 26. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown automaton. Then,

$$\Rightarrow^n$$

denotes n-fold product (recall previous definitions) of $\Rightarrow$. Moreover,

$$\Rightarrow^+ \text{ and } \Rightarrow^*$$

denote the transitive closure of $\Rightarrow$ and transitive and reflexive closure of $\Rightarrow$, respectively.

Definition 27. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown automaton. The language of M, symbolically

$$\mathcal{L}(M),$$

is defined as

$$\mathcal{L}(M) = \{w \in \Sigma^* : Z_0 q_0 w \Rightarrow^* f,\ f \in F\}.$$

In other words, whenever M finishes computation for a given string in one of the final states, its stack is empty and the whole input is processed, then that string belongs to the language described by M. Note that there is no need to require final state and empty stack at the same time, because all variants have the same generative power (see [10]).

The following example shows the possible computation over the single string in detail.

Example 9. Consider an automaton $M = (\{q, r, f\}, \{a, b\}, \{\bot, A\}, \delta, q, \bot, \{f\})$, where

$$\begin{array}{rcl}
\delta & = & \{ \\
 & & \begin{array}{rcl} \bot qa & \to & \bot Aq, \\ Aqa & \to & AAq, \\ \bot q & \to & \bot r, \\ Aq & \to & Ar, \\ Arb & \to & r, \\ \bot r & \to & f \end{array} \\
 & & \} .
\end{array}$$

The following part shows the computation of M over the string $aaaaabbbbb$:

$$\begin{array}{rcll}
(q, aaaaabbbbb, \bot) & \Rightarrow & (q, aaaaabbbbb, A\bot) & [\ \bot qa \to \bot Aq\ ] \\
 & \Rightarrow & (q, aaaabbbbb, AA\bot) & [\ Aqa \to AAq\ ] \\
 & \Rightarrow & (q, aaabbbbb, AAA\bot) & [\ Aqa \to AAq\ ] \\
 & \Rightarrow & (q, aabbbbb, AAAA\bot) & [\ Aqa \to AAq\ ] \\
 & \Rightarrow & (q, abbbbb, AAAA\bot) & [\ Aqa \to AAq\ ] \\
 & \Rightarrow & (q, bbbbb, AAAAA\bot) & [\ Aqa \to AAq\ ] \\
 & \Rightarrow & (r, bbbbb, AAAAA\bot) & [\ Aq \to Ar\ ] \\
 & \Rightarrow & (r, bbbb, AAAA\bot) & [\ Arb \to r\ ] \\
 & \Rightarrow & (r, bbb, AAA\bot) & [\ Arb \to r\ ] \\
 & \Rightarrow & (r, bb, AA\bot) & [\ Arb \to r\ ] \\
 & \Rightarrow & (r, b, A\bot) & [\ Arb \to r\ ] \\
 & \Rightarrow & (r, \varepsilon, \bot) & [\ Arb \to r\ ] \\
 & \Rightarrow & (f, \varepsilon, \varepsilon) & [\ \bot r \to f\ ]
\end{array}$$

It can be observed that $\mathcal{L}(M) = \{a^n b^n : n \geq 0\}$.

Extended Pushdown Automata

An *extended pushdown automaton* is a variant of pushdown automaton, where the rules in δ are generalized to the form

$$(\Gamma^* Q(\Sigma \cup \{\varepsilon\}) \times (\Gamma^* Q).$$

In other words, instead of reading just the topmost symbol on the pushdown in a single move, extended pushdown automaton can read any finite string which is on the top of the stack. This modification has the same generative power as ordinary pushdown automata (see [10]), but the usage of this model is more comfortable in many cases.

2.4 Advanced Definitions

This section defines the new notion of *generalized scattered context grammars*. Then, it introduces *Post correspondence problem* and its variants. Finally, it formalizes the two derivation restrictions studied in this paper.

2.4.1 Generalized Scattered Context Grammar

In 1969, Greibach and Hopcroft proposed *scattered context grammars* (see [5]). This type of grammar can rewrite several nonterminals appearing in the sentential form at the same time. In fact, this is achieved by applying the sequence of context-free productions. We generalize this by extending the left-hand sides of the productions to any nonterminal string. Such grammar is called the *generalized scattered context grammar*.

Definition 28. A *generalized scattered context grammar*, a **SCG** for short, is a quadruple, $G = (V, T, P, S)$, where

1. V is a total alphabet,
2. $T \subseteq V$ is an alphabet of terminals,
3. $S \in (V - T)$ is the starting symbol, and
4. $P \in ((V - T)^+)^i \times (V^*)^i$, where $1 \leq i \leq k$ for some $k \geq 1$, is a finite relation.

$N = V - T$ is the alphabet of *nonterminals*. Elements of P are called *productions*, so P is referred to as the *set of productions*. Instead of $((u_1, \ldots, u_n), (v_1, \ldots, v_n)) \in P$, we can write

$$(u_1, \ldots, u_n) \rightarrow (v_1, \ldots, v_n) \in P$$

or

$$(u_1, \ldots, u_n) \rightarrow (v_1, \ldots, v_n),$$

if P is grounded. If each production $p = (u_1, \ldots, u_n) \rightarrow (v_1, \ldots, v_n) \in P$ satisfies $|u_i| = 1$, for all $1 \leq i \leq n$, then G is an ordinary *scattered context grammar*. Set $\pi(p) = n$. If $\pi(p) \geq 2$, then p is said to be a *context-sensitive* production. If $\pi(p) = 1$, then p is said to be *context-free*. $(u_1, \ldots, u_n)$ is called *left-hand side* of the production, $(v_1, \ldots, v_n)$ is called *right-hand side*.

Definition 29. Let $G = (V, T, P, S)$ be a generalized scattered context grammar and $(\alpha_1, \ldots, \alpha_n) \rightarrow (\beta_1, \ldots, \beta_n) \in P$, $x_i \in V^*$, for all $1 \leq i \leq n$. Then, $x_0\alpha_1x_1 \ldots \alpha_nx_n$ *directly derives* $x_0\beta_1x_1 \ldots \beta_nx_n$ in G, denoted as

$$x_0\alpha_1x_1 \ldots \alpha_nx_n \Rightarrow x_0\beta_1x_1 \ldots \beta_nx_n \ [(\alpha_1, \ldots, \alpha_n) \rightarrow (\beta_1, \ldots, \beta_n)],$$

or

$$x_0\alpha_1x_1 \ldots \alpha_nx_n \Rightarrow x_0\beta_1x_1 \ldots \beta_nx_n.$$

A relation $\Rightarrow \in V^* \times V^*$ is called a *direct derivation*. In other words, G makes a derivation step from $x_0\alpha_1x_1 \ldots \alpha_nx_n$ to $x_0\beta_1x_1 \ldots \beta_nx_n$ according to $(\alpha_1, \ldots, \alpha_n) \rightarrow (\beta_1, \ldots, \beta_n)$.

To denote that the derivation step was performed by G, we write

$$x_0\alpha_1 x_1 \dots \alpha_n x_n \Rightarrow_G x_0\beta_1 x_1 \dots \beta_n x_n.$$

A generalized scattered context grammar works in a similar way as a phrase-structure grammar does. It takes its starting symbol as the current sentential form. Then, it subsequently selects productions which match the string being rewritten. This time, all strings on the left-hand side of the production have to appear (non-overlapped and in the specified order) in the current sentential form. The process finishes, when there is no production available for application.

Definition 30. Let $G = (V, T, P, S)$ be a generalized scattered context grammar. Then,

$$\Rightarrow^n$$

denotes n-fold product (recall previous definitions) of $\Rightarrow$. Moreover,

$$\Rightarrow^+ \text{ and } \Rightarrow^*$$

denote the transitive closure of $\Rightarrow$ and transitive and reflexive closure of $\Rightarrow$, respectively.

We define the language generated by a generalized scattered context grammar in the same way as in case of a phrase-structure grammar.

Definition 31. Let $G = (V, T, P, S)$ be a generalized scattered context grammar. The language of G, symbolically

$$\mathcal{L}(G),$$

is defined as

$$\mathcal{L}(G) = \{w \in T^* : S \Rightarrow^* w\}.$$

The following example demonstrates possible derivation of a single string in the generalized scattered context grammar.

Example 10. Consider a generalized scattered context grammar, $G = (V, \{a, b, c\}, P, S)$, where

$$P = \{S \to ABC, (A, B, C) \to (aA, bB, cC), (A, B, C) \to (\varepsilon, \varepsilon, \varepsilon)\}.$$

G generates the string $aaaaabbbbbccccc$ by the following sequence of productions:

$$\begin{array}{lcll}
S & \Rightarrow & ABC & [\ S \to ABC\ ] \\
 & \Rightarrow & aAbBcC & [\ (A,B,C) \to (aA,bB,cC)\ ] \\
 & \Rightarrow & aaAbbBccC & [\ (A,B,C) \to (aA,bB,cC)\ ] \\
 & \Rightarrow & aaaAbbbBcccC & [\ (A,B,C) \to (aA,bB,cC)\ ] \\
 & \Rightarrow & aaaaAbbbbBccccC & [\ (A,B,C) \to (aA,bB,cC)\ ] \\
 & \Rightarrow & aaaaaAbbbbbBcccccC & [\ (A,B,C) \to (aA,bB,cC)\ ] \\
 & \Rightarrow & aaaaabbbbbccccc & [\ (A,B,C) \to (\varepsilon,\varepsilon,\varepsilon)\ ]
\end{array}$$

If we examine G further, then $\mathcal{L}(G) = \{a^n b^n c^n : n \geq 0\}$.

Generalized scattered context grammars, as well as ordinary scattered context grammars, characterize the family of recursively enumerable languages, unless some additional restriction is placed on the derivation of the sentential form.

2.4.2 Post Correspondence Problem

In 1946, Emil Post formulated an interesting undecidable problem in the field of formal languages known as *Post correspondence problem* (see [11]).

Definition 32. Let $E = \{(u_1, v_1), \ldots, (u_n, v_n)\}$, where $u_i, v_i \in T^*$, for all $1 \leq i \leq n$. Then E is an instance of *Post correspondence problem*. The solution to this problem is a sequence of indices $i_1 \ldots i_k \in \{1, \ldots, n\}^*$, where $1 \leq k \leq K$ $(K \geq 1)$, such that

$$u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}.$$

The next example shows the solution of a simple instance of this problem.

Example 11. Let $E = \{(aba, a), (bbb, aaa), (aab, abab), (bb, babba)\}$. Then, the solution of E is

$$x = 1431,$$

because

$$aba\ bb\ aab\ aba = ababbaababa = a\ babba\ abab\ a.$$

On the other side, if the instance didn't contain the pair $(bb, babba)$, then there would be no solution at all.

Extended Post Correspondence Problem

Definition 33. For an alphabet $T = \{a_1, \ldots, a_n\}$, there is an *extended Post correspondence problem*, E, defined as

$$E = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_n})),$$

where $u_i, v_i, z_{a_j} \in \{0, 1\}^*$, for each $1 \leq i \leq r$, $1 \leq j \leq n$. The solution of the extended Post correspondence problem is a sequence $s_1 \ldots s_l b_1 \ldots b_k$, where $s_1, \ldots, s_l \in \{1, \ldots, r\}$, $b_1, \ldots, b_k \in T$ and $k, l \geq 1$, such that

$$v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{b_1} \ldots z_{b_k}.$$

Now we define language of the extended Post correspondence problem.

Definition 34. Let $E = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_n}))$ be an instance of extended Post correspondence problem. The language represented by E is the set

$$\mathcal{L}(E) = \{b_1 \ldots b_k \in T^* : \text{ exists } s_1, \ldots, s_l \in \{1, \ldots, r\}, l \geq 1, \\ v_{s_1} \ldots v_{s_l} = u_{s_1} \ldots u_{s_l} z_{b_1} \ldots z_{b_k} \text{ for some } k \geq 0\}.$$

It is well known that for each recursively enumerable language, L, there is an extended Post correspondence problem, E, such that $\mathcal{L}(E) = L$ (see Theorem 1 in [3]).

Example 12. Let $E = (\{(01, 0), (1, 1)\}, (0_a, 1_b))$ for an alphabet $\{a, b\}$. Then, the solution of E is

$$x = 12b,$$

because

$$01\ 1 = 011 = 0\ 1\ 1_b.$$

The language defined by E is

$$\mathcal{L}(E) = \{\varepsilon, b\}.$$

2.4.3 Derivation Restrictions

The previous parts consider only the basic variants of a derivation, where it is sufficient to match certain substrings in the sentential form against the left-hand side of the production. Additionally, we can refine the relation (of the direct derivation) by proposing new criteria which the sentential form has to satisfy.

Restriction on the Number of Occurrences (Context-free Grammars)

A *restriction on the number of occurrences* requires that all sentential forms contain at most k nonterminals (for some $k \geq 1$).

Definition 35. Let $G = (V, T, P, S)$ be a context-free grammar, $A \to \alpha \in P$, $x, y \in V^*$, $\text{occur}(V - T, xAy) \leq m$ and $\text{occur}(V - T, x\alpha y) \leq m$, where $m \geq 1$. Then, xAy *directly derives* $x\alpha y$ with respect to the *number of occurrences* in G, denoted as

$$xAy \ {}_m\Rightarrow\ x\alpha y\ [A \to \alpha]$$

or

$$xAy \ {}_m\Rightarrow\ x\alpha y.$$

In other words, G makes a derivation step from xAy to $x\alpha y$ according to $A \to \alpha$ with respect to ${}_m\Rightarrow$.

Definition 36. Let $G = (V, T, P, S)$ be a context-free grammar. Then,

$${}_m\Rightarrow^k$$

denotes k-fold product (recall previous definitions) of ${}_m\Rightarrow$. Moreover,

$${}_m\Rightarrow^+ \text{ and } {}_m\Rightarrow^*$$

denote the transitive closure of ${}_m\Rightarrow$ and transitive and reflexive closure of ${}_m\Rightarrow$, respectively.

Definition 37. Let $G = (V, T, P, S)$ be a context-free grammar. The language generated by G in a m-limited way, symbolically

$$\mathcal{L}(G, m),$$

is defined as

$$\mathcal{L}(G, m) = \{w \in T^* : S \ {}_m\Rightarrow^*\ w\}.$$

Next, we define the two derivation restrictions discussed in this thesis.

Restriction on the Depth of Rewriting (SCG)

A *restriction on the depth of rewriting* requires that all productions are applied within the finite prefix of the first continuous sequence of nonterminals in the sentential form.

Definition 38. Let $G = (V, T, P, S)$ be a generalized scattered context grammar and $k \geq 1$. If there is $(\alpha_1, \ldots, \alpha_n) \to (\beta_1, \ldots, \beta_n) \in P$, $u = x_0\alpha_1x_1 \ldots \alpha_nx_n$, and $v = x_0\beta_1x_1 \ldots \beta_nx_n$, where

1. $x_0 \in T^*N^*$,
2. $x_i \in N^*$, for all $0 < i < n$,
3. $x_n \in V^*$ and
4. $\text{occur}(x_0\alpha_1x_1 \ldots \alpha_n, N) \leq k$.

Then, u *directly derives* v with respect to the *depth of rewriting* in G, denoted as

$$u \; {}_k\Diamond\Rightarrow \; v \; [(\alpha_1, \ldots, \alpha_n) \to (\beta_1, \ldots, \beta_n)]$$

or

$$u \; {}_k\Diamond\Rightarrow \; v.$$

In other words, G makes a derivation step from u to v according to $(\alpha_1, \ldots, \alpha_n) \to (\beta_1, \ldots, \beta_n)$ with respect to ${}_k\Diamond\Rightarrow$.

Definition 39. Let $G = (V, T, P, S)$ be a generalized scattered context grammar. Then,

$${}_k\Diamond\Rightarrow^n$$

denotes n-fold product (recall previous definitions) of ${}_k\Diamond\Rightarrow$. Moreover,

$${}_k\Diamond\Rightarrow^+ \text{ and } {}_k\Diamond\Rightarrow^*$$

denote the transitive closure of ${}_k\Diamond\Rightarrow$ and transitive and reflexive closure of ${}_k\Diamond\Rightarrow$, respectively.

Definition 40. Let $G = (V, T, P, S)$ be a generalized scattered context grammar. The language generated by G with respect to the depth of rewriting (${}_k\Diamond\Rightarrow$), symbolically

$${}_{k-left}\mathcal{L}(G, m),$$

is defined as

$${}_{k-left}\mathcal{L}(G, m) = \{w \in T^* : S \; {}_k\Diamond\Rightarrow^* \; w\}.$$

Restriction on the Number of Occurrences (SCG)

A *restriction on the number of occurrences* requires that all sentential forms produced during the derivation of a string contain no more than $k \geq 1$ blocks of nonterminals. Optionally, the length of all blocks can be limited at the same time.

Definition 41. Let $G = (V, T, P, S)$ be a generalized scattered context grammar and $m, h \geq 1$. $W(m)$ denotes the set of all strings $x \in V^*$ satisfying 1 given next. $W(m, h)$ denotes the set of all strings $x \in V^*$ satisfying 1 and 2 given next.

1. $x \in (T^*(V - T)^*)^m T^*$;
2. ($y \in \text{substring}(x)$ and $|y| > h$) implies $\text{alph}(y) \cap T \neq \emptyset$.

Definition 42. Consider a generalized scattered context grammar $G = (V, T, P, S)$. Let $(\alpha_1, \ldots, \alpha_n) \to (\beta_1, \ldots, \beta_n) \in P$, $u = x_0\alpha_1 x_1 \ldots \alpha_n x_n$, and $v = x_0\beta_1 x_1 \ldots \beta_n x_n$, where

1. $x_0 \in V^*$,
2. $x_i \in N^*$, for all $0 < i < n$, and
3. $x_n \in V^*$.

Then, u *directly derives* v with respect to the *block of nonterminals* in G, denoted as

$$u \circ\!\Rightarrow v \ [(\alpha_1, \ldots, \alpha_n) \to (\beta_1, \ldots, \beta_n)]$$

or

$$u \circ\!\Rightarrow v.$$

In other words, G makes a derivation step from u to v according to $(\alpha_1, \ldots, \alpha_n) \to (\beta_1, \ldots, \beta_n)$ with respect to $\circ\!\Rightarrow$.

Definition 43. Let $G = (V, T, P, S)$ be a generalized scattered context grammar, $u, v \in V^*$, and $u \circ\!\Rightarrow v$ in G. Then,

$$u \ {}_m\!\circ\!\Rightarrow v \text{ if and only if } u, v \in W(m)$$

and

$$u \ {}^h_m\!\circ\!\Rightarrow v \text{ if and only if } u, v \in W(m, h).$$

In other words, G makes a derivation step from u to v with respect to ${}_m\!\circ\!\Rightarrow$ (${}^h_m\!\circ\!\Rightarrow$).

Definition 44. Let $G = (V, T, P, S)$ be a generalized scattered context grammar. Then,

$$\circ\!\Rightarrow^n ({}_m\!\circ\!\Rightarrow^n, {}^h_m\!\circ\!\Rightarrow^n)$$

denotes n-fold product (recall previous definitions) of $\circ\!\Rightarrow$ (${}_m\!\circ\!\Rightarrow$, ${}^h_m\!\circ\!\Rightarrow$). Moreover,

$$\circ\!\Rightarrow^+ ({}_m\!\circ\!\Rightarrow^+, {}^h_m\!\circ\!\Rightarrow^+) \text{ and } \circ\!\Rightarrow^* ({}_m\!\circ\!\Rightarrow^*, {}^h_m\!\circ\!\Rightarrow^*)$$

denote the transitive closure of $\circ\Rightarrow$ ($_m\circ\Rightarrow$, $^h_m\circ\Rightarrow$) and transitive and reflexive closure of $\circ\Rightarrow$ ($_m\circ\Rightarrow$, $^h_m\circ\Rightarrow$), respectively.

Definition 45. Let $G = (V, T, P, S)$ be a generalized scattered context grammar. The languages generated by G with respect to the number of occurrences ($_m\circ\Rightarrow$, $^h_m\circ\Rightarrow$), symbolically

$$_{nonter}\mathcal{L}(G, m) \text{ and } _{nonter}\mathcal{L}(G, m, h),$$

are defined as

$$\begin{aligned} _{nonter}\mathcal{L}(G, m) &= \{w \in T^* : S \ _m\circ\Rightarrow^* w\} \\ _{nonter}\mathcal{L}(G, m, h) &= \{w \in T^* : S \ ^h_m\circ\Rightarrow^* w\} \end{aligned}$$

Example Derivations

It is important to understand the derivation restrictions well in order to verify the results presented in this work. That is the reason, why we present several examples of how the derivation restrictions work "in practice".

Example 13. Consider the generalized scattered context grammar,

$$G = (\{S, A, B, C\}, \{a, b, c\}, P, S),$$

where

$$P = \{S \to ABC, (A, B, C) \to (AA, BB, CC), A \to a, B \to b, C \to c\},$$

which generates language $\mathcal{L}(G) = \{a^n b^n c^n : n \geq 1\}$.

An unrestricted derivation can produce

$$ABC \Rightarrow AABBCC,$$

while

$$ABC \ _2\circ\not\Rightarrow AABBCC,$$

because nonterminal C appears beyond the specified prefix length (2). Note that we can't improve the generative power by increasing the limit, because we always reach this limit by applying the production $(A, B, C) \to (AA, BB, CC)$ by a finite number of times.

An unrestricted derivation can also produce

$$AABBCC \Rightarrow AAbBCC,$$

but in the case of the restriction on the number of occurrences

$$AABBCC \ _1\circ\not\Rightarrow AAbBCC,$$

because the maximum number of nonterminal blocks, which are allowed in the sentential form at the same time, is 1, while the modified sentential form would contain 2 such blocks. Note that this restriction doesn't actually change the language generated by G.

Additionally,

$$ABC \ ^3_1\circ\not\Rightarrow AABBCC,$$

because the new sentential form contain a block of nonterminals which exceeds the specified length (3).

Finally,

$$aAbBcC \not\Rightarrow aAAbBBcCC,$$

because the symbols, which are being rewritten, do not appear in the same block of nonterminals.

2.4.4 Language Families

The following definition introduces new language families which are related to the results studied in this work.

Definition 46. Let **CFG**s denote the set of all context-free grammars and **SCG**s denote the family of generalized scattered context grammars. We define these language families:

$$\begin{aligned}
{}_{m}CF &= \{L : L = \mathcal{L}(G, m), G \in \textbf{CFG}\text{s}\} \text{ for all } m \geq 0 \\
{}_{k-left}SC &= \{L : L = {}_{k-left}\mathcal{L}(G), G \in \textbf{SCG}\text{s}\} \text{ for all } k \geq 0 \\
{}_{nonter}SC(m) &= \{L : L = {}_{nonter}\mathcal{L}(G, m), G \in \textbf{SCG}\text{s}\} \text{ for all } m \geq 1 \\
{}_{nonter}SC(m, h) &= \{L : L = {}_{nonter}\mathcal{L}(G, m, h), G \in \textbf{SCG}\text{s}\} \text{ for all } m, h \geq 1
\end{aligned}$$

Chapter 3

Results

This chapter presents the main results of this work. First, we prove that restricting the derivation in a generalized scattered context grammar to the finite prefix decreases the generative power of this model to the family of context-free languages.

Then, we investigate the case, in which a generalized scattered context grammar is restricted on the finite number of blocks of nonterminals. We show that this restriction leads to the family of recursively enumerable languages and thus it doesn't reduce the generative power at all.

Finally, we demonstrate that restricting a scattered context grammar to the sentential forms with only the finite number of finite blocks of nonterminals reduces the generative power to the family of context-free languages again.

3.1 Restriction on the Finite Prefix

This section proves that restricting all possible derivations to those, which occurs within the first k ($k \geq 1$) symbols of the first continuous block of nonterminals, decreases the generative power of the generalized scattered context grammars to the family of context-free languages.

3.1.1 Short Description

For a given k ($k \geq 1$), we can see each sentential form as the string of the form

$$x\alpha y,$$

where $x \in T^*$, $\alpha \in (V - T)^*$, $y \in V^*$ and $|\alpha| \leq k$. Moreover, y starts with nonterminal in those cases, where $|\alpha| = k$, only. If k is a constant, which limits the length of the prefix, then all possible productions can occur within α. This means that

$$x\alpha y \Rightarrow x\beta y,$$

where $\beta \in V^*$. Now, we can rewrite $x\beta y$ as $x'\alpha' y'$, where $x' \in T^*$ ($x \in \text{prefix}(x')$), $\alpha' \in (V - T)^*$, $y' \in V^*$ and $|\alpha'| \leq k$. Hence, we can rewrite α' in the next step, and so on.

This process can be effectively simulated by an extended pushdown automaton, M, which works in the following way:

1. M pushes the starting symbol to its pushdown.
2. M matches all terminal symbols, which appear on the top of the stack, against the input string until it reaches the first nonterminal.
3. M accepts, if the stack contains only the initial symbol.
4. M subsequently reads (at most k) nonterminals from the stack and records them in its state.
5. M simulates the derivation in the generalized scattered context grammar on the limited prefix using information in its state
6. M continues with 2.

The next part contains the formal proof of the statement that uses the idea mentioned above.

3.1.2 Formal Proof

Theorem 1 *Let k be a positive integer. Then, $CF = {}_{k-left}SC$.*

Proof. Let $G = (V, T, P, S)$ be a generalized scattered context grammar. Consider the following pushdown automaton

$$M = (\{q, r, f\} \cup \{[\gamma, s] : \gamma \in N^*, |\gamma| \leq k, s \in \{q, r\}\}, T, V \cup \{Z\}, \delta, [S, q], Z, \{f\}),$$

where $Z \notin V$, and δ contains rules of the following forms:

1. $[\beta_0 A_1 \beta_1 \dots A_n \beta_n, q] \to (\beta_0 \alpha_1 \beta_1 \dots \alpha_n \beta_n)^R [\varepsilon, r]$
   if $(A_1, \dots, A_n) \to (\alpha_1, \dots, \alpha_n) \in P$; $\beta_i \in N^*$, $0 \le i \le n$;
2. $A[A_1 \dots A_n, r] \to [A_1 \dots A_n A, r]$ if $n < k$, $A \in N$;
3. $[A_1 \dots A_k, r] \to [A_1 \dots A_k, q]$;
4. $a[A_1 \dots A_n, r] \to a[A_1 \dots A_n, q]$ if $n < k$, $a \in T$;
5. $Z[A_1 \dots A_n, r] \to Z[A_1 \dots A_n, q]$ if $n < k$;
6. $a[\varepsilon, r]a \to [\varepsilon, r]$ if $a \in T$;
7. $Z[\varepsilon, r] \to f$.

We prove that $\mathcal{L}(M) = {}_{k-left}\mathcal{L}(G)$.

($\subseteq$:) By induction on the number of rules constructed in 1 used in a sequence of moves, we prove the following claim.

Claim 1.1 If $Z\alpha^R[\beta_0 A_1 \beta_1 \dots A_n \beta_n, q]w \Rightarrow^* f$, then $\beta_0 A_1 \beta_1 \dots A_n \beta_n \alpha \ {}_k\Leftrightarrow^* w$.

Proof. Basis: Only one rule constructed in 1 is used. Then,

$$Z\alpha^R[\beta_0 A_1 \beta_1 \dots A_n \beta_n, q]uw \Rightarrow Z(\beta_0 \alpha_1 \beta_1 \dots \alpha_n \beta_n \alpha)^R [\varepsilon, r]uw \Rightarrow^* f,$$

where $(A_1, \dots, A_n) \to (\alpha_1, \dots, \alpha_n) \in P$, $n \le k$, and $\beta_0 \alpha_1 \beta_1 \dots \alpha_n \beta_n \alpha \in T^*$. Therefore, $\beta_0 = \dots = \beta_n = \varepsilon$, and $\alpha_1 \dots \alpha_n \alpha = uw$. Then,

$$A_1 \dots A_n w \ {}_k\Leftrightarrow uw.$$

Induction hypothesis: Suppose that the claim holds for all sequences of moves containing no more than i rules constructed in 1.

Induction step: Consider a sequence of moves containing $i+1$ rules constructed in 1. Then,

$$\begin{array}{lll}
 & Z\alpha^R[\beta_0 A_1 \beta_1 \dots A_l \beta_l, q]w & \\
\Rightarrow & Z\alpha^R(\beta_0 \alpha_1 \beta_1 \dots \alpha_l \beta_l)^R[\varepsilon, r]w & \text{(by a rule constructed in 1)} \\
\Rightarrow^* & Z\alpha'[\varepsilon, r]w' & \text{(by rule constructed in 6)} \\
\Rightarrow^* & Z\alpha''[\beta'_0 B_1 \beta'_1 \dots B_m \beta'_m, r]w' & \text{(by rule constructed in 2)} \\
\Rightarrow & Z\alpha''[\beta'_0 B_1 \beta'_1 \dots B_m \beta'_m, q]w' & \text{(by a rule constructed in 3, 4, or 5)} \\
\Rightarrow^* & f &
\end{array}$$

where $\alpha' \in V^*N \cup \{\varepsilon\}$, $v \in T^*$, $\alpha' v^R = \alpha^R (\beta_0 \alpha_1 \beta_1 \dots \alpha_l \beta_l)^R$, and $vw' = w$. Then, by the production $(A_1, \dots, A_l) \to (\alpha_1, \dots, \alpha_l)$,

$$\beta_0 A_1 \beta_1 \dots A_l \beta_l \alpha \ {}_k\Leftrightarrow \beta_0 \alpha_1 \beta_1 \dots \alpha_l \beta_l \alpha,$$

where $|\beta_0 A_1 \beta_1 \dots A_l \beta_l| \le k$,

$$\beta_0 \alpha_1 \beta_1 \dots \alpha_l \beta_l \alpha = v(\alpha')^R = v\beta'_0 B_1 \beta'_1 \dots B_m \beta'_m (\alpha'')^R,$$

and, by the induction hypothesis,

$$v\beta'_0 B_1 \beta'_1 \dots B_m \beta'_m (\alpha'')^R \ {}_k\Leftrightarrow^* vw'.$$

Hence, the inclusion holds. △

($\supseteq$:) Next, we prove the following claim.

Claim 1.2 If $\beta \ {}_k\Leftrightarrow^* w$, where $\beta \in NV^*$, then $Z\beta^R[\varepsilon, r]w \Rightarrow^* f$.

Proof. By induction on the length of derivations.

Basis: Let $A_1 \dots A_n w \ {}_k\Leftrightarrow \alpha_1 \dots \alpha_n w$ $(\alpha_1 \dots \alpha_n = \alpha)$, where $\alpha w \in {}_{k-left}\mathcal{L}(G)$, and $(A_1, \dots, A_n) \to (\alpha_1, \dots, \alpha_n) \in P$, $1 \leq n \leq k$. M simulates this derivation step as follows.

	$Zw^R A_n \dots A_1[\varepsilon, r]\alpha w$	
$\Rightarrow^n$	$Zw^R[A_1 \dots A_n, r]\alpha w$	(by rule constructed in 2)
$\Rightarrow$	$Zw^R[A_1 \dots A_n, q]\alpha w$	(by a rule constructed in 4 or 5)
$\Rightarrow$	$Zw^R\alpha^R[\varepsilon, r]\alpha w$	(by a rule constructed in 1)
$\Rightarrow^{\lvert\alpha w\rvert}$	$Z[\varepsilon, r]$	(by rule constructed in 6)
$\Rightarrow$	f	(by the rule constructed in 7)

Induction hypothesis: Suppose that the claim holds for all derivations of length i or less.

Induction step: Consider a derivation of length $i + 1$. Let

$$\beta_0 B_1 \beta_1 \dots B_l \beta_l \gamma \ {}_k\Leftrightarrow \beta_0 \alpha_1 \beta_1 \dots \alpha_l \beta_l \gamma \ {}_k\Leftrightarrow^i \varphi w,$$

where $\varphi w \in {}_{k-left}\mathcal{L}(G)$, $\beta_0 B_1 \beta_1 \dots B_l \beta_l \in N^+$, and either $|\beta_0 B_1 \beta_1 \dots B_l \beta_l| = k$, or $|\beta_0 B_1 \dots B_l \beta_l| < k$, $\beta_0 \alpha_1 \beta_1 \dots \alpha_l \beta_l \gamma = \varphi\psi$, where $\varphi \in T^*$, $\psi \in NV^* \cup \{\varepsilon\}$, and $\gamma \in TV^* \cup \{\varepsilon\}$. Then,

	$Z(\beta_0 B_1 \beta_1 \dots B_l \beta_l \gamma)^R[\varepsilon, r]\varphi w$	
$\Rightarrow^*$	$Z\gamma^R[\beta_0 B_1 \beta_1 \dots B_l \beta_l, r]\varphi w$	(by rule constructed in 2)
$\Rightarrow$	$Z\gamma^R[\beta_0 B_1 \beta_1 \dots B_l \beta_l, q]\varphi w$	(by a rule constructed in 3 or 4)
$\Rightarrow$	$Z(\varphi\psi\gamma)^R[\varepsilon, r]\varphi w$	(by a rule constructed in 1)
$\Rightarrow^*$	$Z(\psi\gamma)^R[\varepsilon, r]w$	(by a rule constructed in 6)
$\Rightarrow^*$	f	(by the induction hypothesis)

Hence, the claim holds. △

Now, if

$$S \Rightarrow u\alpha \Rightarrow^* uw,$$

where $u \in T^*$ and $\alpha \in NV^*$, then

$$Z[S, q]uw \Rightarrow Z(u\alpha)^R[\varepsilon, r]uw \Rightarrow^* Z\alpha^R[\varepsilon, r]w \Rightarrow^* f,$$

by rules constructed in 1 and 6 and the previous claim. For $\alpha = \varepsilon$,

$$Z[S, q]u \Rightarrow Zu^R[\varepsilon, r]u \Rightarrow^* f.$$

Hence, the other inclusion holds. □

3.2 Restriction on the Finite Number of Blocks

This section demonstrates that by restricting the sentential forms (which appear during the derivation in the generalized scattered context grammars) on the strings that contain only the finite number ob nonterminal blocks, the generative power remains unchanged.

Additionally, we show that it is enough to consider only the derivations which contain no more than one such block to generate the family of recursively enumerable languages as well.

3.2.1 Short Description

We can prove that $RE = {}_{nonter}SC(1)$ by using results achieved in [3]. First, the problem is formulated by means of extended Post correspondence problem, which also characterize type-0 languages. Now, consider the instance E (of the problem) over the alphabet Σ:

$$E = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_n})),$$

which has the solution of the form

$$s_1 \ldots s_l b_1 \ldots b_k.$$

We create a scattered context grammar, G, which, in the first phase, generates the string

$$x = \alpha\$\$\beta\gamma,$$

where $\alpha, \beta \in \{0,1\}^*$, $\gamma \in \Sigma^*$. Then, it verifies that x is of the form

$$z_{b_k} \ldots z_{b_1} u_{s_l} \ldots u_{s_1} \$\$ v_{s_1} \ldots v_{s_l} b_1 \ldots b_k$$

and that

$$u_{s_1} \ldots u_{s_l} z_{b_1} \ldots z_{b_k} = v_{s_1} \ldots v_{s_l}.$$

During the verification of the generated string, G erases all nonterminals and if the verification succeeds, it generates the string

$$b_1 \ldots b_k.$$

All sentential forms in such derivation contain only a single continuous block of nonterminals which confirms the validity of the statement in question.

3.2.2 Formal Proof

Theorem 2 $RE = {}_{nonter}SC(1)$.

Proof. Let $L \subseteq \{a_1, \ldots, a_n\}^*$ be a recursively enumerable language. There is an extended Post correspondence problem,

$$E = (\{(u_1, v_1), \ldots, (u_r, v_r)\}, (z_{a_1}, \ldots, z_{a_n})),$$

where $u_i, v_i, z_{a_j} \in \{0,1\}^*$, for each $1 \leq i \leq r$, $1 \leq j \leq n$, such that $\mathcal{L}(E) = L$; that is, $w = b_1 \ldots b_k \in L$ if and only if $w \in \mathcal{L}(E)$. Set $V = \{S, A, 0, 1, \$\} \cup T$. Define the **SCG** $G = (V, T, P, S)$ with P constructed as follows:

1. For every $a \in T$, add
   (a) $(S) \to ((z_a)^R Sa)$, and
   (b) $(S) \to ((z_a)^R Aa)$ to P;
2. (a) For every $(u_i, v_i) \in E$, $1 \leq i \leq r$, add $(A) \to ((u_i)^R A v_i)$ to P;
   (b) Add $(A) \to (\$\$)$ to P;
3. Add
   (a) $(0, \$, \$, 0) \to (\$, \varepsilon, \varepsilon, \$)$,
   (b) $(1, \$, \$, 1) \to (\$, \varepsilon, \varepsilon, \$)$, and
   (c) $(\$) \to (\varepsilon)$ to P.

Claim 2.1 Let $w_1, w_2 \in \{0,1\}^*$. Then, $w_1\$\$w_2 \Rightarrow_G^* \varepsilon$ if and only if $w_1 = (w_2)^R$.

Proof. If: Let $w_1 = (w_2)^R = b_1 \dots b_k$, for some $k \geq 0$. By productions (3a) and (3b) followed by two applications of (3c), we obtain

$$\begin{array}{lll} b_k \dots b_2 b_1 \$\$ b_1 b_2 \dots b_k & \Rightarrow & b_k \dots b_2 \$\$ b_2 \dots b_k \\ & \Rightarrow^* & b_k \$\$ b_k \\ & \Rightarrow & \$\$ \Rightarrow \$ \Rightarrow \varepsilon. \end{array}$$

Therefore the if-part of the claim holds.

Only if: Suppose that $|w_1| \leq |w_2|$. We demonstrate that

$$w_1\$\$w_2 \Rightarrow_G^* \varepsilon \text{ implies } w_1 = (w_2)^R$$

by induction on $k = |w_1|$.

Basis: Let $k = 0$. Then, $w_1 = \varepsilon$ and the only possible derivation is

$$\$\$w_2 \Rightarrow \$w_2 \ [(3c)] \Rightarrow w_2 \ [(3c)].$$

Hence, we can derive ε only if $w_1 = (w_2)^R = \varepsilon$.

Induction Hypothesis: Suppose that the claim holds for all w_1 satisfying $|w_1| < k$ for some $k \geq 0$.

Induction Step: Consider $w_1 a\$\$b w_2$ with $a \neq b$, $a, b \in \{0,1\}$. If $w_1 = w_{11} b w_{12}$, $w_{11}, w_{12} \in \{0,1\}^*$, then either (3a) or (3b) can be used. In either case, we obtain

$$w_1 a\$\$b w_2 \Rightarrow w_{11}\$w_{12} a w_{21}\$w_{22},$$

where $b w_2 = w_{21} b w_{22}$, $w_{21}, w_{22} \in \{0,1\}^*$, and $w_{12} a w_{21} \in N^+$ cannot be removed by any production from the sentential form. The same is true when $w_2 = w'_{21} a w'_{22}$, $w'_{21}, w'_{22} \in \{0,1\}^*$. Therefore, the derivation proceeds successfully only if $a = b$. Thus,

$$w_1 a\$\$b w_2 \Rightarrow w_1\$\$w_2 \Rightarrow^* \varepsilon,$$

and from the induction hypothesis,

$$w_1 = (w_2)^R.$$

Analogously, the same result can be proved for $|w_1| \geq |w_2|$ which implies that the only-if part of the claim holds.

Therefore, the claim holds. $\triangle$

Examine the introduced productions to see that G always generates $b_1 \dots b_k \in \mathcal{L}(E)$ by a derivation of this form:

$$
\begin{array}{lll}
S & \Rightarrow & (z_{b_k})^R S b_k \\
 & \Rightarrow & (z_{b_k})^R (z_{b_{k-1}})^R S b_{k-1} b_k \\
 & \Rightarrow^* & (z_{b_k})^R \dots (z_{b_2})^R S b_2 \dots b_k \\
 & \Rightarrow & (z_{b_k})^R \dots (z_{b_2})^R (z_{b_1})^R A b_1 b_2 \dots b_k \\
 & \Rightarrow & (z_{b_k})^R \dots (z_{b_1})^R (u_{s_l})^R A v_{s_l} b_1 \dots b_k \\
 & \Rightarrow^* & (z_{b_k})^R \dots (z_{b_1})^R (u_{s_l})^R \dots (u_{s_1})^R A v_{s_1} \dots v_{s_l} b_1 \dots b_k \\
 & \Rightarrow & (z_{b_k})^R \dots (z_{b_1})^R (u_{s_l})^R \dots (u_{s_1})^R \$\$ v_{s_1} \dots v_{s_l} b_1 \dots b_k \\
 & & = (u_{s_1} \dots u_{s_l} z_{b_1} \dots z_{b_k})^R \$\$ v_{s_1} \dots v_{s_l} b_1 \dots b_k \\
 & \Rightarrow^* & b_1 \dots b_k.
\end{array}
$$

Productions introduced in steps 1 and 2 of the construction find nondeterministically the solution of the extended Post correspondence problem which is subsequently verified by productions from step 3. Therefore, $w \in L$ if and only if $w \in \mathcal{L}(G)$ and the theorem holds. $\square$

3.3 Restriction on the Blocks of the Finite Length

At last, we prove that by permitting only the sentential forms, which contain only nonterminal blocks of finite length, the generative power of the generalized scattered context grammars decreases (unlike previous result) to the family of context-free languages.

Moreover, if we limit the number of these blocks at the same time, we obtain a hierarchy on the class of context-free languages.

3.3.1 Short Description

To show that ${}_mCF = {}_{nonter}SC(m,h)$, we can encode each block of nonterminals into the single nonterminal:

$$x\alpha y \rightsquigarrow x\langle\alpha\rangle y,$$

where $x, y \in V^*$, $\alpha \in (V-T)^+$. Then, we can simulate the derivation of the generalized scattered context grammar by the productions performed using a context-free grammar.

Note that the number of nonterminal blocks in the sentential forms generated by the generalized scattered context grammar is equal to the number of nonterminals appearing in the corresponding sentential forms generated by the equivalent context-free grammar.

3.3.2 Formal Proof

Theorem 3 *Let m and h be positive integers. Then, ${}_mCF = {}_{nonter}SC(m,h)$.*

Proof. ${}_mCF \subseteq {}_{nonter}SC(m,h)$, because each context-free grammar is the special case of the generalized scattered context grammar. This is true even in the case, where such restriction applies.

We prove that ${}_{nonter}SC(m,h) \subseteq {}_mCF$. For the purpose of this part, we propose an additional definition.

Definition 47. Let $\alpha = x_0y_1x_1 \ldots y_nx_n$, where $x_i \in T^*$, $y_i \in N^+$, for $0 \leq i \leq n$, and for all $0 < i < n$, $x_i \neq \varepsilon$. Then,

$$f(\alpha) = x_0\langle y_1\rangle x_1 \ldots \langle y_n\rangle x_n,$$

where $\langle y_i\rangle$ is a new nonterminal, for all $0 \leq i \leq n$.

Let $G_{SC} = (V, T, P, S)$ be a generalized scattered context grammar. Introduce a context-free grammar $G_{CF} = (V', T, P', \langle S\rangle)$, where $V' = \{\langle\gamma\rangle : \gamma \in N^*, 1 \leq |\gamma| \leq h\} \cup T$ and P' is constructed as follows:

1. for each $\gamma = x_0\alpha_1x_1 \ldots \alpha_nx_n$, where $x_i \in N^*$, $\alpha_i \in N^+$, $1 \leq |\gamma| \leq h$, and

   $$(\alpha_1, \ldots, \alpha_n) \rightarrow (\beta_1, \ldots, \beta_n) \in P,$$

   add

   $$\langle\gamma\rangle \rightarrow f(x_0\beta_1x_1 \ldots \beta_nx_n)$$

   to P'.

Claim 3.1 Let $S \overset{h}{}_{m}\!\!\circ\!\!\Rightarrow^{k} \omega$ in G_{SC}, where $\omega \in V^*$, $k \geq 0$. Then, $\langle S\rangle \ {}_{m}\!\!\Rightarrow^{k} f(\omega)$ in G_{CF}.

Proof. By induction on $k = 0, 1, \ldots$.

Basis: Let $k = 0$, thus $S \overset{h}{}_{m}\!\!\circ\!\!\Rightarrow^{0} S$ in G_{SC}. Then, $\langle S\rangle \ {}_{m}\!\!\Rightarrow^{0} \langle S\rangle$ in G_{CF}. As $f(S) = \langle S\rangle$, the basis holds.

Induction hypothesis: Suppose that the claim holds for all $0 \leq m \leq k$, where k is a non-negative integer.

Induction step: Let

$$S \overset{h}{}_{m}\!\!\circ\!\!\Rightarrow^{k} \phi\gamma\psi \overset{h}{}_{m}\!\!\circ\!\!\Rightarrow \phi\gamma'\psi$$

in G_{SC}, and the last production applied during the derivation is

$$(\alpha_1, \ldots, \alpha_n) \to (\beta_1, \ldots, \beta_n),$$

where $\phi \in V^*T \cup \{\varepsilon\}$, $\gamma = x_0\alpha_1x_1 \ldots \alpha_nx_n$, $\psi \in TV^* \cup \{\varepsilon\}$, $\gamma' = x_0\beta_1x_1 \ldots \beta_nx_n$, $\alpha_i, x_i \in N^*$, and $\beta_i \in V^*$. By the induction hypothesis,

$$\langle S\rangle \ {}_{m}\!\!\Rightarrow^{k} f(\phi\gamma\psi).$$

By the definition of f, ϕ, and ψ, $f(\phi\gamma\psi) = f(\phi)\langle\gamma\rangle f(\psi)$. Hence, we can use the production

$$\langle\gamma\rangle \to f(\gamma') \in P'$$

introduced in 1 in the construction to obtain

$$f(\phi)\langle\gamma\rangle f(\psi) \ {}_{m}\!\!\Rightarrow f(\phi)f(\gamma')f(\psi).$$

By the definition of f, ϕ, and ψ, $f(\phi)f(\gamma')f(\psi) = f(\phi\gamma'\psi)$. As a result,

$$\langle S\rangle \ {}_{m}\!\!\Rightarrow^{k} f(\phi)\langle\gamma\rangle f(\psi) \ {}_{m}\!\!\Rightarrow f(\phi\gamma'\psi)$$

and, therefore, $\langle S\rangle \ {}_{m}\!\!\Rightarrow^{k+1} f(\phi\gamma'\psi)$ and the claim holds for $k + 1$. △

Claim 3.2 Let $\langle S\rangle \ {}_{m}\!\!\Rightarrow^{k} \omega$ in G_{CF}, where $\omega \in V'^*$, $k \geq 0$. Then, $S \overset{h}{}_{m}\!\!\circ\!\!\Rightarrow^{k} f^{-1}(\omega)$ in G_{SC}.

Proof. By induction on $k = 0, 1, \ldots$.

Basis: Let $k = 0$, thus $\langle S\rangle \ {}_{m}\!\!\Rightarrow^{0} \langle S\rangle$ in G_{CF}. Then $S \overset{h}{}_{m}\!\!\circ\!\!\Rightarrow^{0} S$ in G_{SC}. As $f^{-1}(\langle S\rangle) = S$, the basis holds.

Induction hypothesis: Suppose that the claim holds for all $0 \leq m \leq k$, where k is a non-negative integer.

Induction step: Let

$$\langle S\rangle \ {}_{m}\!\!\Rightarrow^{k} \phi\langle\gamma\rangle\psi \ {}_{m}\!\!\Rightarrow \phi\gamma'\psi$$

in G_{CF}, and the last production applied during the derivation is

$$\langle\gamma\rangle \to \gamma',$$

where $\phi \in V^*T \cup \{\varepsilon\}$, $\gamma = x_0\alpha_1x_1 \dots \alpha_nx_n$, $\psi \in TV^* \cup \{\varepsilon\}$, $\gamma' = f(x_0\beta_1x_1 \dots \beta_nx_n)$, $\alpha_i, x_i \in N^*$, and $\beta_i \in V^*$. By the induction hypothesis,

$$S \underset{m}{\overset{h}{\circ\Rightarrow}}{}^{k} f^{-1}(\phi\langle\gamma\rangle\psi).$$

By the definition of f, ϕ, and ψ, $f^{-1}(\phi\langle\gamma\rangle\psi) = f^{-1}(\phi)\gamma f^{-1}(\psi)$. There exists

$$(\alpha_1, \dots, \alpha_n) \to (\beta_1, \dots, \beta_n) \in P$$

by 1, thus

$$f^{-1}(\phi)\gamma f^{-1}(\psi) \underset{m}{\overset{h}{\circ\Rightarrow}} f^{-1}(\phi)f^{-1}(\gamma')f^{-1}(\psi).$$

By the definition of f, ϕ, and ψ, $f^{-1}(\phi)f^{-1}(\gamma')f^{-1}(\psi) = f^{-1}(\phi\gamma'\psi)$. As a result

$$S \underset{m}{\overset{h}{\circ\Rightarrow}}{}^{k} f^{-1}(\phi)\gamma f^{-1}(\psi) \underset{m}{\overset{h}{\circ\Rightarrow}} f^{-1}(\phi\gamma'\psi)$$

and, therefore, $S \underset{m}{\overset{h}{\circ\Rightarrow}}{}^{k+1} f^{-1}(\phi\gamma'\psi)$ and the claim holds for $k+1$. △

Hence, the theorem holds. □

Chapter 4

Conclusion

This work extends the notion of scattered context grammars by the productions containing the string of several nonterminals on their left-hand sides instead of a single nonterminal. Next, we study two types of derivation restrictions, which are placed on the sentential forms during the derivation.

By restricting a generalized scattered context grammar to the productions, which are applied in the finite prefix of the first block of nonterminals appearing in the sentential form, we decrease the generative power of this model to the family of context-free languages. This fact is demonstrated using a constructive proof (section 3.1), which shows that it is possible to simulate a restricted derivation in a scattered context grammar by an equivalent pushdown automaton.

On the other hand, the proof in the section 3.2 shows that any recursively enumerable language can be generated by a scattered context grammar which is restricted only to the sentential forms which contain no more than one continuous block of nonterminals. The previous result, presented by Geffert (see [3]), places an equivalence between the class of recursively enumerable languages and the class of languages that can be generated by an extended Post correspondence problem. Additionally, we show that it is possible to create a scattered context grammar, which can find any solution to a given instance of such problem (in a restricted way), thus it generates the same language. Hence, this restriction doesn't modify the generative power at all.

The last result (section 3.3) proves that by permitting the sentential forms containing nonterminal blocks of a finite length only, the generative power of the generalized scattered context grammars decreases to the family of context-free languages. This fact is demonstrated by a construction of an equivalent context-free grammar, which simulates the restricted derivation in a scattered context grammar. Moreover, if we allow only the finite number of these blocks, we obtain an infinite hierarchy on the family of context free languages.

The thesis, as a whole, presents some of the possible restrictions of the sentential forms and shows certain techniques which are useful for establishing the generative power of the restricted models. We show that by using these restrictions the generative power of scattered context grammars decreases to the family of context-free languages or it remains the same in some cases.

4.1 Open problems

This work shows that the restricted versions of scattered context grammars characterize languages, which are strictly context-free, or they fall into the family of recursively enumerable languages. Are there other restrictions, which lead to the class of languages that lies between the families of recursively enumerable and context-free languages?

On the other hand, these restrictions can be placed on the different models. What is the generative power of the *grammar systems* (see [11]) restricted in this way? Is it the same as in the case of generalized scattered context grammars?

Bibliography

[1] B. S. Baker. Context-sensitive grammars generating context-free languages. In M. Nivat, editor, *Automata, Languages and Programming*, pages 501–506. North-Holland, Amsterdam, 1972.

[2] R. V. Book. Terminal context in context-sensitive grammars. *SIAM Journal of Computing*, 1:20–30, 1972.

[3] V. Geffert. Context-free-like forms for the phrase-structure grammars. In M. Chytil, L. Janiga, and V. Koubek, editors, *Mathematical Foundations of Computer Science*, volume 324 of *Lecture Notes in Computer Science*, pages 309–317. Springer-Verlag, 1988.

[4] S. Ginsburg and S. Greibach. Mappings which preserve context-sensitive languages. *Information and Control*, 9:563–582, 1966.

[5] S. Greibach and J. Hopcroft. Scattered context grammars. *Journal of Computer and System Sciences*, 3:233–247, 1969.

[6] T. Masopust, A. Meduna, and J. Šimáček. Two power-decreasing derivation restrictions in generalized scattered context grammars (to appear). *Acta Cybernetica*, 2008(1):1–11, 2008.

[7] G. Matthews. A note on symmetry in phrase structure grammars. *Information and Control*, 7:360–365, 1964.

[8] G. Matthews. Two-way languages. *Information and Control*, 10:111–119, 1967.

[9] A. Meduna. A trivial method of characterizing the family of recursively enumerable languages by scattered context grammars. *EATCS Bulletin*, pages 104–106, 1995.

[10] A. Meduna. *Automata and Languages: Theory and Applications*. Springer-Verlag, London, 2000.

[11] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 1. Springer-Verlag, Berlin, 1997.