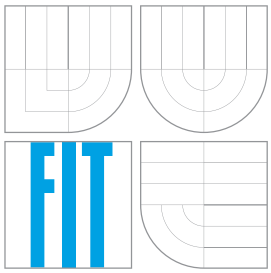
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZPTÝLENÝ KONTEXT VE FORMÁLNÍCH JAZYCÍCH

SCATTERED CONTEXT IN FORMAL LANGUAGES

DISERTAČNÍ PRÁCE
PhD THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ TECHET

VEDOUCÍ PRÁCE
SUPERVISOR

ALEXANDER MEDUNA

BRNO 2008

Licenční smlouva
poskytovaná k výkonu práva užít školní dílo

uzavřená mezi smluvními stranami:

1. Pan

Jméno a příjmení: *Jiří Techet*
Bytem: *Čejkova 27, 61500 Brno*
Narozen: *29.10.1981 v Brně*
(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem *Božetěchova 2, 61266 Brno*
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

. . . . . .
(dále jen „nabyvatel“)

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP): *disertační práce* (dále jen VŠKP nebo dílo)

   Název VŠKP: *Rozptýlený kontext ve formálních jazycích*
   Školitel VŠKP: *Prof. RNDr. Alexander Meduna, CSc.*
   Ústav: *Ústav informačních systémů*
   Datum obhajoby VŠKP: . . . . . .
   VŠKP odevzdal autor nabyvateli v: *tištěné i elektronické formě (3 exempláře)*

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti *ihned po uzavření této smlouvy*.

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne 29. února 2008

. . . . .	. . . . .
Nabyvatel	Autor

Abstrakt

Tato disertační práce studuje teoretické vlastnosti gramatik s rozptýleným kontextem. Výzkum je zaměřen na čtyři hlavní oblasti. Nejprve jsou zkoumány podmínky, za nichž lze odstranit vymazávací pravidla z gramatik s rozptýleným kontextem. Druhou studovanou oblastí jsou modifikace gramatik s rozptýleným kontextem. Konkrétně se jedná o gramatiky s rozptýleným kontextem s jinými než bezkontextovými komponentami, derivace limitované na prvních n nonterminálů, nejlevější derivace a konečně derivace, v nichž je v každém derivačním kroku přepsán maximální, respektive minimální počet nonterminálů. Dále práce studuje generátory vět, ve kterých je každá věta obohacena o sekvenci pravidel použitých během její derivace. Jsou diskutovány kanonické a redukované generátory tohoto druhu. Nakonec je uvedeno několik příkladů na použití gramatik s rozptýleným kontextem při popisu a zpracování přirozeného jazyka.

Klíčová slova

teorie formálních jazyků, gramatiky s rozptýleným kontextem, vyjadřovací síla gramatik, popisná složitost, vymazávací pravidla, kanonické derivace, lingvistika

Abstract

The present thesis studies theoretical properties of scattered context grammars. The research is focused on four main areas. First, it examines the conditions under which erasing productions can be removed form a scattered context grammar. Second, four modifications of scattered context grammars are introduced and studied. Specifically, the considered modifications involve scattered context grammars with non-context-free components, derivations limited to the first n nonterminals, leftmost derivations, and, finally, derivations in which either the maximal or the minimal number of nonterminals is rewritten in every derivation step. Next, the thesis studies generators of sentences in which every sentence is enriched with a sequence of productions used during its generation. Canonical and reduced generators of this kind are discussed. Finally, several applications of scattered context grammars to natural language description and processing are presented.

Keywords

formal language theory, scattered context grammars, generative power, descriptional complexity, erasing productions, canonical derivations, linguistics

Citace

Jiří Techet: Scattered Context in Formal Languages, disertační práce, Brno, FIT VUT v Brně, 2008

Scattered Context in Formal Languages

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně pod vedením pana profesora Alexandera Meduny. Některé části sekcí 2.1 a 2.2 byly inspirovány úvodními kapitolami v [55], výsledek uvedený v sekci 5.3 je společným dílem mne a mého kolegy Tomáše Masopusta. Veškeré ostatní výsledky jsou mé vlastní a vznikly pouze za spolupráce s mým školitelem. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Jiří Techet
29. února 2008

Poděkování

Chtěl bych poděkovat svému školiteli, panu profesoru Alexanderu Medunovi, jak za jeho odborné rady, tak za jeho vždy vstřícný přístup a ochotu kdykoli pomoci. Děkuji Tomáši Masopustovi za vynikající spolupráci, díky níž vznikla část této práce. Dále děkuji kolegům z doktorského studia na Fakultě informačních technologií, kteří mi poskytli cenné podněty, jež se projevily ve výsledné práci.

© Jiří Techet, 2008.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Contents

Chapter 1

Introduction

Making use of highly effective computation performed in parallel, modern computer science technologies are designed to work with information of enormous size. During a single computational step, however, a typical computational process usually needs only some selected elements of the information. As a result, in the information as a whole, it selects a finite number of scattered, but often mutually related elements of information, and simultaneously derives new pieces of information from these elements. By substituting the new pieces of information for the selected elements, it produces new information as a whole and, thereby, completes such a computational step. The newly obtained information is subsequently processed in the same selective way during the next computational step. This computational process successfully ends when some terminating conditions concerning the information are met; as a rule, these conditions simply require that the information consists solely of elements from a prescribed terminating set. The present thesis studies computational processes of this kind.

In general, to discuss some kind of information processing in a rigorous way in computer science, we often formalize the corresponding information processors by formal language models, such as suitable grammars. Then, by studying these grammars and the way they yield their languages in strictly mathematical terms of formal language theory, we actually rigorously investigate the processors that work with the information under discussion. Following this mathematical approach, we formalize the scattered information processors by scattered context grammars, introduced by formal language theory several decades ago (see [16]). We have chosen the theory of formal languages because it straightforwardly and naturally allows us to formalize the scattered information within an information i by *scattered context* of symbols, A_1 through A_n, occurring in i as $i = x_0A_1x_1A_2x_2 \ldots A_nx_n$, where the x's represent i's parts irrelevant to the current computational step. A computational rule according to which the derivation step is made has the form $(A_1, A_2, \ldots, A_n) \rightarrow (y_1, y_2, \ldots, y_n)$, where y_1 through y_n represent the new piece of information derived from A_1 through A_n. In this way, we formalize all the computational rules according to which the computational process performs its steps. As these *scattered context rules* actually define how the words are changed, the set containing these rules forms a *scattered context grammar* as a whole. These rules operate over two sets of symbols. The grammar contains a finite set of *terminal symbols*. Apart from the terminal symbols, it also contains finitely many *nonterminal symbols*, one of which is defined as the *start symbol*. Beginning from the start symbol, the formalized computational process consisting of a sequence of derivation steps successfully ends when the derived words consist solely of terminal symbols. A terminal word derived in this way is included into the language of this grammar, which contains all words derived in this way. As shown later in this thesis, scattered context grammars are equivalent to Turing machines in the sense that both define the family of recursively enumerable languages. As a result, these grammars represent not only an elegant formalization of scattered

information processing but also a powerful generator of languages.

To illustrate the formalization of scattered information processing, consider a file, f, in its binary form. Double f by attaching f to its duplicate as ff, which is a common file operation in practice. Suppose we want to specify the generative process which produces the set of all files doubled in this way. More formally speaking, we want to generate the language $L = \{ff : f \in \{0,1\}^*\}$. Processing this set requires to guarantee that each member of this set is formed by a binary word followed by an equally long binary word that coincides with the preceding word. In greater detail, we need to make sure that both words consist of the same number of binary digits and, in addition, the ith binary digit in the first word coincides with the ith binary digit in the second word. In this case, scattered information processing consists of verifying the identity of the ith binary digit in the first f and the ith binary digit in the second f. The start rule derives FF from the start symbol S by a rule of the form $(S) \rightarrow (FF)$. Then, to guarantee that the ith binary digit in the left word coincides with the ith binary digit in the right word, we simultaneously derive two identical binary digits from the left and from the right F by these scattered context rules $(F,F) \rightarrow (0F,0F)$ and $(F,F) \rightarrow (1F,1F)$. To complete this process, we simultaneously erase both F's by $(F,F) \rightarrow (\varepsilon,\varepsilon)$, where ε is the empty word, which consists of no symbols at all. For instance, 0101 is derived by a sequence of derivation steps S, FF, $0F0F$, $01F01F$, 0101.

To enhance the reason why we have chosen scattered context grammars as a proper grammatical formalization of scattered information processing, we now consider how some other fundamental formal grammars describe the context and perform their derivation steps in order to see why these grammars are less appropriate for this formalization than scattered context grammars.

Regarding the context description, we can divide grammatical rules into context-dependent and context-independent rules in the theory of formal languages. Accordingly, we distinguish between context-dependent grammars, such as phrase-structure grammars or interactive Lindenmayer systems, and context-independent grammars, such as context-free grammars. Context-independent rules are applied quite independently of any other symbols, so they are obviously incapable of formalizing mutually related elements of scattered information. Besides this fundamental drawback, they are significantly less powerful than scattered context grammars; in fact, they cannot even generate the trivial language L above. Context-dependent grammars are more powerful; however, their context-dependent rules depend on the context surrounding the rewritten symbol, so they fail to appropriately formalize widely scattered elements of information. Concerning the description of scattered context, all these grammars are thus less suitable than scattered context grammars.

The performance of derivation steps in grammars should obviously reflect the way scattered information is processed in reality. Therefore, sequential grammars, such as context-free grammars, can hardly serve as a proper formalization of scattered information processing because they rewrite only a single symbol during a derivation step. Although totally parallel grammars, such as L systems (see [31, 59, 61]), reflect scattered information processing more appropriately, this reflection is still not quite adequate from a realistic point of view. Indeed, these grammars work in a completely parallel way since they rewrite all symbols of the sentence during a single derivation step. However, due to hardware limitations, only a finite number of symbols can be processed in one computational step in practice. As scattered context grammars simultaneously rewrite finitely many selected symbols during a single derivation step while leaving the other symbols unchanged, they formalize scattered information processing most appropriately.

Grammars which rewrite finitely many symbols during a single derivation step are referred to as *partially-parallel*. Scattered context grammars are not the only representative of this grammar type; multigrammars (see [24]) and simple matrix grammars (see [21]) belong to this class as well. The reason why we have chosen scattered context grammars for scattered information processing is the simplicity of language description and the flexibility of how their rules are used. In greater de-

tail, scattered context grammars, contrary to multigrammars, do not require the symbols appearing between the scattered elements which are being processed to be described. On the other hand, simple matrix grammars are not economical in situations when no context checks are required as they rewrite a fixed number of symbols during the whole computational process. *Unordered scattered context grammars* (see [9, 15, 36, 56, 82]) are another type of partially-parallel grammars. In order to apply a rule in an ordinary scattered context grammar, the order of the symbols appearing on the left-hand side of the rule has to correspond to the order of these symbols appearing in the sentence; in unordered scattered context grammars, this order is unimportant. Even though their definition resembles the definition of scattered context grammars, the term "unordered scattered context grammar" is slightly misleading—the left and the right context of a symbol cannot be checked because there is no ordering within the rules; it can be only checked whether the considered symbol appears somewhere in the sentence. These grammars are equivalent to programmed grammars (see [6, 82]) and have, therefore, completely different properties than scattered context grammars. As unordered scattered context grammars are not fully capable of capturing scattered-context dependencies, we do not discuss them in this thesis.

Organization

The thesis is divided into three main parts. Part I, *Introduction to Formal Languages*, presents the basic mathematical concepts and the used terminology, formally defines scattered context grammars, and, finally, summarizes the currently known results concerning scattered context grammars. Part II, *Results*, is the key part of the thesis as it brings new results from the area of scattered context grammars. Finally, Part III, *Summary and Conclusion*, summarizes all the obtained results and proposes new directions for future investigation.

In greater detail, we briefly describe the contents of the individual chapters.

Chapter 1, *Introduction*, sets scattered information processing in a broader context and emphasizes the importance of its study. In addition, it outlines the structure of the document.

Chapter 2, *Definitions*, gives information about the needed mathematical background, the basics of formal language theory, and defines various formalisms which are used later in the text. Finally, it introduces the formal definition of a scattered context grammar which is the key concept of this thesis.

Chapter 3, *Related Work*, summarizes all currently known theoretical properties of scattered context grammars. It discusses both scattered context grammars with and without erasing productions, and their modified versions.

Chapter 4, *Conditional Removal of Erasing Productions*, presents a result which demonstrates that erasing production removal is possible in scattered context grammars if their derivations satisfy certain properties. A short version of this result was published in [78, 79], its full proof [54] has been submitted for publication.

Chapter 5, *Restrictions and Extensions*, studies various restricted and extended versions of scattered context grammars. Specifically, Section 5.1, *Non-Context-Free Components of Scattered Context Grammars*, introduces scattered context grammars whose components are not context-free as in ordinary scattered context grammars, and presents the obtained results (see [80]). In Section 5.2, *n-Limited Derivations*, scattered context grammars without erasing productions which use n-limited derivations are studied. These derivations permit only the first n nonterminals to be rewritten. Based on n, we obtain an infinite hierarchy of language families. The result presented in this section has been submitted for publication (see [51]). Section 5.3, *Leftmost Derivations*, mentions a simplified proof of a previously published result which discussed leftmost derivations of scattered context

grammars without erasing productions. This proof is a result of my cooperation with my colleague Tomás Masopust and has been submitted for publication (see [35]). Finally, in Section 5.4, *Maximal and Minimal Rewriting*, maximal and minimal derivations of scattered context grammars without erasing productions are defined and it is proved that grammars with this type of derivations characterize the family of context-sensitive languages (see [52]).

Chapter 6, *Generators of Sentences with Their Parses*, discusses the use of scattered context grammars without erasing productions for generation of their sentences which are enriched with an additional information needed for their possible subsequent analysis. First, Section 6.1, *General Generators*, studies general types of these generators (see [49, 71, 72, 73]), second, their leftmost and rightmost variants are studied in Section 6.2, *Canonical Generators* (see [50, 74, 75, 76, 77]), and, finally, generators with a reduced number of nonterminals and context-sensitive productions are discussed in Section 6.3, *Reduced Generators* (see [53]).

Chapter 7, *Applications in Linguistics*, studies possible applications of scattered context grammars in linguistics. It shows that in natural languages there are many examples of dependencies which are scattered through a sentence. The chapter demonstrates how to transform one kind of sentences with scattered context dependencies to another and gives several examples of these transformations.

Chapter 8, *Conclusion*, summarizes all results obtained in Part II and outlines possible areas for further investigation.

Part I

Introduction to Formal Languages

Chapter 2

Definitions

Even though this thesis aims to be self-contained in the sense that no other sources are needed for understanding all the presented results, the reader should be familiar at least with the basic mathematical terms and principles. However, a deeper knowledge of mathematics and formal language theory is welcomed as this introductory chapter describes the used terminology very briefly which may be insufficient for its full understanding. There is a vast number of excellent publications that introduce the reader to the needed mathematical concepts and the fundamentals of formal language theory (see [3, 8, 13, 17, 18, 23, 27, 29, 32, 60, 65, 66, 67, 70, 85]). In certain chapters, we refer to the relation of our research to compiler construction. For more information about compilers and their construction, see [1, 30, 47, 68, 69].

This chapter is divided into three sections. Sections 2.1 and 2.2 review the fundamental mathematical notions needed to follow the thesis clearly and accurately. Section 2.3 introduces scattered context grammars and notions related to these grammars which are specific for this thesis. For readers having background in formal language theory, this chapter can be skimmed and treated as a reference for notion and definitions.

2.1 Mathematical Background

This section recalls the basic notions concerning sets, sequences, and relations which are used later in the thesis.

Sets

A *set* A is a collection of elements taken from some prespecified universe. If A contains an element a, then we symbolically write $a \in A$ and refer to a as a member of A. On the other hand, if a is not in A, we write $a \notin A$. The cardinality of A, $|A|$, is the number of A's members. The set that has no member is the *empty set*, denoted $\emptyset$; note that $|\emptyset| = 0$. If A has a finite number of members, then A is a *finite set*; otherwise, A is an *infinite set*. An infinite set A is *countably infinite* if there is a bijection from A to the set of all integers.

A finite set A is customarily specified by listing its members; that is, $A = \{a_1, a_2, \ldots, a_n\}$, where a_1 through a_n are all members of A. An infinite set B is usually specified by a property π, so that B contains all elements satisfying π; in symbols, this specification has the following general format: $B = \{a : \pi(a)\}$. Sets whose members are other sets are usually called *families of sets* rather than sets of sets.

Let A and B be two sets. A is a *subset* of B, symbolically written as $A \subseteq B$, if each member of A also belongs to B. A is a *proper subset* of B, written as $A \subset B$, if $A \subseteq B$ and B contains an element

that is not in A. If $A \subseteq B$ and $B \subseteq A$, A equals B, denoted by $A = B$. The *power set* of A, denoted by 2^A, is the set of all subsets of A. For two sets A and B, their union, intersection, and difference are denoted by $A \cup B$, $A \cap B$, and $A - B$, respectively, and defined as

$$A \cup B = \{a : a \in A \text{ or } a \in B\},$$

$$A \cap B = \{a : a \in A \text{ and } a \in B\},$$

and

$$A - B = \{a : a \in A \text{ and } a \notin B\}.$$

For a property π, the union of elements of A satisfying this property is defined as

$$\bigcup_{\pi} A = \{a : a \in A, \pi(a)\}.$$

For a set A over a universe U, the *complement* of A is denoted by $\overline{A}$ and defined as $\overline{A} = U - A$.

A *sequence* is a list of elements from some universe U. A sequence is *finite* if it represents a finite list of elements; otherwise, it is *infinite*. The *length* of a finite sequence x, denoted by $|x|$, is the number of elements in x. The *empty sequence*, denoted by ε, is the sequence consisting of no element; that is, $|\varepsilon| = 0$. A finite sequence is usually specified by listing its elements. For $V \subseteq U$, $|x|_V$ denotes the number of occurrences of elements from V in a sequence x. For instance, consider a finite sequence x specified as $x = 0,1,0,0$ and observe that $|x| = 4$ and $|x|_{\{0\}} = 3$.

Relations

For two objects a and b, (a,b) denotes the *ordered pair* consisting of a and b in this order. Analogically, the *ordered n-tuple* is denoted by $(a_1,\ldots,a_n)$, for some objects $a_1,\ldots,a_n$. Let A and B be two sets. The *Cartesian product* of A and B, $A \times B$, is defined as

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

A *binary relation* or, briefly, a *relation* ρ from A to B is any subset of $A \times B$; that is, $\rho \subseteq A \times B$. The *domain* of ρ, denoted by $\text{domain}(\rho)$, and the *range* of ρ, denoted by $\text{range}(\rho)$, are defined by

$$\text{domain}(\rho) = \{a : (a,b) \in \rho \text{ for some } b \in B\}$$

and

$$\text{range}(\rho) = \{b : (a,b) \in \rho \text{ for some } a \in A\}.$$

If $A = B$, then ρ is a *relation on* A. A relation σ is a *subrelation* of ρ if σ represents a subset of ρ. The *inverse* of ρ, denoted by ρ^{-1}, is defined as

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

A *function* or, synonymously, a *mapping* from A to B is a relation ϕ from A to B such that for each $a \in A$,

$$|\{b : b \in B, (a,b) \in \phi\}| \leq 1.$$

Let ϕ be a function from A to B. If $\text{domain}(\phi) = A$, ϕ is *total*; otherwise, ϕ is *partial*. If for each $b \in B$,

$$|\{a : a \in A, (a,b) \in \phi\}| \leq 1,$$

ϕ is *injective*. If for each $b \in B$,

$$|\{a : a \in A, (a,b) \in \phi\}| \geq 1,$$

ϕ is *surjective*. If ϕ is both a surjective and an injective, then ϕ is called *bijective*. Injective, surjective and bijective function is briefly called *injection*, *surjection*, and *bijection*, respectively.

Instead of $(a,b) \in \rho$, we often write $a \in \rho(b)$ or $a\rho b$; in other words, $(a,b) \in \rho$, $a\rho b$, and $a \in \rho(b)$ are used interchangeably. If ρ is a function, we usually write $a = \rho(b)$.

We say that a set A is *closed under a binary operation* $\circ$ if for each $a, b \in A$, $a \circ b \in A$. A set that is closed under an operation is said to satisfy a *closure property*. A closure under an n-ary operation is defined analogically. Let ρ be a relation on a set A. For $k \geq 1$, the *k-fold product* of ρ, ρ^k, is recursively defined as (1) $a\rho^1 b$ if and only if $a\rho b$, and (2) $a\rho^k b$ if and only if $a\rho c$ and $c\rho^{k-1} b$ for some c and $k \geq 2$. The *transitive closure* of ρ, ρ^+, is defined as $a\rho^+ b$ if and only if $a\rho^k b$ for some $k \geq 1$, and the *reflexive and transitive closure* of ρ, ρ^*, is defined as $a\rho^* b$ if and only if $a\rho^k b$ for some $k \geq 0$.

2.2 Basics of Formal Language Theory

In this section, we define fundamental mathematical notions and concepts of formal language theory. We focus on languages, language operations, language families, and various types of grammars.

Languages

An *alphabet* T is a finite, nonempty set, whose members are called *symbols*. A finite sequence of symbols from T is a *string* or, synonymously, a *word* over T; specifically, ε is referred to as the *empty string*. By T^*, we denote the set of all strings over T; $T^+ = T^* - \{\varepsilon\}$. Any subset $L \subseteq T^*$ is a *language* over T. If L represents a finite set of strings, L is a *finite language*; otherwise, L is an *infinite language*. For instance, T^*, called the universal language over T, is an infinite language while $\emptyset$ and $\{\varepsilon\}$ are finite; notably, $\emptyset \neq \{\varepsilon\}$ because $|\emptyset| = 0 \neq |\{\varepsilon\}| = 1$. By $\text{alph}(x)$ we denote the set of all symbols occurring in a string x. Set

$$\text{alph}(L) = \bigcup_{x \in L} \text{alph}(x).$$

By analogy with the set theory, sets whose members are languages are called *families of languages*.

By convention, we omit all separating commas in strings. That is, we write $a_1 a_2 \ldots a_n$ rather than $a_1, a_2, \ldots, a_n$.

Let $x, y \in T^*$ be two strings over an alphabet T. The *concatenation* of x with y, denoted by xy, is the string obtained by appending y to x. Observe that for each $w \in T^*$, $w\varepsilon = \varepsilon w = w$. Notice that from an algebraic point of view, T^* and T^+ are the free monoid and the free semigroup, respectively, generated under the operation of concatenation. We say that x is a *prefix* of y if there exists $u \in T^*$ such that $xu = y$; in addition, if $x \notin \{\varepsilon, y\}$, x is a *proper prefix* of y. Similarly, x is a *suffix* of y if there exists $u \in T^*$ such that $ux = y$; in addition, if $x \notin \{\varepsilon, y\}$, x is a *proper suffix* of y. Finally, x is a *substring* or a *subword* of y if there exists $u, v \in T^*$ such that $uxv = y$; in addition, if $x \notin \{\varepsilon, y\}$, x is a *proper substring* or a *proper subword* of y. For all $i \geq 0$ the ith *power of* x, denoted by x^i, is recursively defined as (1) $x^0 = \varepsilon$ and (2) $x^i = xx^{i-1}$ for $i \geq 1$. The *reversal* of x, denoted by $\text{rev}(x)$, is x written in the reverse order.

As languages are defined as sets, all set operations apply to them. Let $L_1, L_2 \subseteq T^*$ be two languages over T. Then, $L_1 \cup L_2$, $L_1 \cap L_2$, and $L_1 - L_2$ denote the union, intersection, and difference

of languages L_1 and L_2, respectively. There are, however, some operations specific to languages. The concatenation of L_1 and L_2, denoted by L_1L_2, is defined as

$$L_1L_2 = \{xy : x \in L_1, y \in L_2\}.$$

The *right quotient* of L_1 with respect to L_2, denoted by L_1/L_2, is defined as

$$L_1/L_2 = \{y : yx \in L_1, \text{ for some } x \in L_2\};$$

similarly, the *left quotient* of L_1 with respect to L_2, denoted by $L_2\backslash L_1$, is defined as

$$L_2\backslash L_1 = \{y : xy \in L_1, \text{ for some } x \in L_2\}.$$

We also use a special type of the right and the left quotient. The *exhaustive right quotient* of L_1 with respect to L_2, denoted by $L_1//L_2$, is defined as

$$L_1//L_2 = \{x : x \in L_1/L_2, \text{ and no word in } L_1/L_2 \text{ is a proper prefix of } x\};$$

similarly, the *exhaustive left quotient* of L_1 with respect to L_2, denoted by $L_2\backslash\backslash L_1$, is defined as

$$L_2\backslash\backslash L_1 = \{x : x \in L_2\backslash L_1, \text{ and no word in } L_2\backslash L_1 \text{ is a proper suffix of } x\}.$$

Apart from binary operations, we also make some unary operations with languages. Let $L \subseteq T^*$. The ith power of L, L^i, is defined as (1) $L^0 = \{\varepsilon\}$ and (2) $L^i = LL^{i-1}$ for $i \geq 1$. The *Kleene star* of L, L^*, is defined as

$$L^* = \bigcup_{i \geq 0} L^i$$

and the *Kleene plus* of L, L^+, is defined as

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

Notice that $L^+ = LL^* = L^*L$ and $L^* = L^+ \cup \{\varepsilon\}$. The *complement* of L is denoted by $\overline{L}$ and defined as $\overline{L} = T^* - L$, and the *reversal* of L, $\mathrm{rev}(L)$, is defined as

$$\mathrm{rev}(L) = \{\mathrm{rev}(x) : x \in L\}.$$

Let T and U be two alphabets. A total function τ from T^* to 2^{U^*} such that $\tau(uv) = \tau(u)\tau(v)$ for each $u, v \in T^*$ is a *substitution* from T^* to U^*. By this definition, $\tau(\varepsilon) = \{\varepsilon\}$ and $\tau(a_1a_2\ldots a_n) = \tau(a_1)\tau(a_2)\ldots\tau(a_n)$, where $n \geq 1$ and $a_i \in T$ for all $1 \leq i \leq n$, so τ is completely specified by defining $\tau(a)$ for each $a \in T$. For $L \subseteq T^*$, we extend the definition of τ to

$$\tau(L) = \bigcup_{w \in L} \tau(w).$$

A total function χ from T^* to U^* such that $\chi(uv) = \chi(u)\chi(v)$ for each $u, v \in T^*$ is a *homomorphism* or, synonymously and briefly, a *morphism* from T^* to U^*. As any homomorphism is obviously a special case of a substitution, we specify χ by analogy with the specification of τ. For $L \subseteq T^*$, we extend the definition of χ to

$$\chi(L) = \{\chi(w) : w \in L\}.$$

A language $L \subseteq T^*$ is called *regular language* if it can be obtained from $\{a\}$, where $a \in T$, and $\emptyset$ using finitely many times the operations of union, concatenation, and Kleene star.

The closure of a family of languages under an operation is defined by analogy with the definition of the closure of a set.

Definition 1. A family of languages $\mathscr{L}$ is said to be *closed under linear erasing* if for each $L \in \mathscr{L}$, homomorphism h and integer k such that $|w| \leq k|h(w)|$ for each $w \in L$, $h(L) \in \mathscr{L}$.

The following definition introduces a family of languages which is closed under several above defined operations.

Definition 2. An *abstract family of languages* (see [14]) is a pair $(T, \mathscr{L})$, or $\mathscr{L}$ when T is understood, where

1. T is a countably infinite set of symbols;
2. For each $L \in \mathscr{L}$ there is a finite set $T_1 \subseteq T$ such that $L \subseteq T_1^*$;
3. $L \neq \emptyset$ for some $L \in \mathscr{L}$;
4. $\mathscr{L}$ is closed under the operations of union, concatenation, Kleene plus, inverse homomorphism, ε-free homomorphism, and intersection with a regular language.

Finally, we introduce some additional definitions used in this thesis. For a finite set of integers I, $\max(I)$ and $\min(I)$ denote the maximal and the minimal element of I, respectively.

Definition 3. Let $\mathrm{perm}(t)$ be the set of all permutations of $\{1,\ldots,t\}$. For some $n,m \geq 0$, define

$$\mathrm{perm}(n,m) = \{(i_1,\ldots,i_{n+m}) \in \mathrm{perm}(n+m) : 1 \leq i_k < i_l \leq n \text{ implies } k < l\}.$$

Let T be an alphabet. For $x_1,\ldots,x_n \in T^*$, $(i_1,\ldots,i_n) \in \mathrm{perm}(n)$, define

$$\mathrm{reorder}((x_1,\ldots,x_n),(i_1,\ldots,i_n)) = (x_{i_1},\ldots,x_{i_n}).$$

Grammars

This section reviews the basics of grammars. Specifically, it provides definitions of regular, right-linear, linear, context-free, context-sensitive, and phrase-structure grammars along with some related notions and basic results. Further, other types of grammars used throughout the thesis are introduced together with their basic properties.

Definition 4. A *phrase-structure* grammar is a quadruple

$$G = (V,T,P,S),$$

where

- V is the *total alphabet*,
- $T \subset V$ is the set of *terminal symbols* or, briefly, *terminals*,
- $P \subseteq V^*(V-T)V^* \times V^*$ is a finite relation,
- $S \in V - T$ is the *start symbol* or, synonymously, *axiom* of G.

The symbols in $V - T$ are referred to as *nonterminal symbols* or, briefly, *nonterminals*. In what follows, each $(x,y) \in P$ is called a *production* or a *rule* and written as

$$x \to y \in P;$$

accordingly, P is called the *set of productions* in G. Given a production $p = x \to y \in P$, we set $\mathrm{lhs}(p) = x$ and $\mathrm{rhs}(p) = y$, which represent the *left-hand side* and the *right-hand side* of the production p, respectively. The relation of a *direct derivation* in G is a binary relation over V^* denoted by $\Rightarrow_G$ and defined in the following way. If $u = z_1xz_2$, $v = z_1yz_2$, and $x \to y \in P$, where $z_1, z_2 \in V^*$, then G makes a *derivation step* from u to v according to $x \to y$, symbolically written as

$$u \Rightarrow_G v\ [x \to y].$$

When no confusion exists, we simplify $u \Rightarrow_G v\ [x \to y]$ to $u \Rightarrow_G v$. By $\Rightarrow_G^k$, we denote the k-fold product of $\Rightarrow_G$. Furthermore, let $\Rightarrow_G^+$ and $\Rightarrow_G^*$ denote the transitive closure of $\Rightarrow_G$ and the reflexive and transitive closure of $\Rightarrow_G$, respectively. If $S \Rightarrow_G^* x$ for some $x \in V^*$, x is called a *sentential form*. If $S \Rightarrow_G^* w$, where $w \in T^*$, $S \Rightarrow_G^* w$ is said to be a *successful derivation* of G. The *language of* G, denoted by $L(G)$, is defined as

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

A *recursively enumerable language* is a language generated by a phrase-structure grammar. In the literature, the phrase-structure grammars are also often defined with productions of the form

$$xAy \to xuy,$$

where $u, x, y \in V^*$ and $A \in V - T$ (see [4]). Both definitions are interchangeable in the sense that the grammars defined in these two ways generate the same family of languages—the family of recursively enumerable languages, denoted by $\mathscr{L}(RE)$.

Definition 5. A *context-sensitive grammar* is a phrase-structure grammar

$$G = (V, T, P, S)$$

such that each production $x \to y \in P$ satisfies

$$|x| \leq |y|.$$

A *context-sensitive language* is a language generated by a context-sensitive grammar. The family of context-sensitive languages is denoted by $\mathscr{L}(CS)$.

Definition 6. A *context-free grammar* is a phrase-structure grammar

$$G = (V, T, P, S)$$

such that each production in P is of the form

$$A \to x,$$

where $A \in V - T$ and $x \in V^*$. A *context-free language* is a language generated by a context-free grammar. The family of context-free languages is denoted by $\mathscr{L}(CF)$.

Definition 7. A *linear grammar* is a phrase-structure grammar

$$G = (V, T, P, S)$$

such that each production in P is of the form

$$A \to xBy, \text{ or } A \to x,$$

where $A, B \in V - T$ and $x, y \in T^*$. A *linear language* is a language generated by a linear grammar. The family of linear languages is denoted by $\mathscr{L}(LIN)$.

Definition 8. A *regular grammar* is a phrase-structure grammar

$$G = (V, T, P, S)$$

such that each production in P is of the form

$$A \to aB, \text{ or } A \to a,$$

where $A, B \in V - T$ and $a \in T$. A *regular language* is a language generated by a regular grammar. The family of regular languages is denoted by $\mathscr{L}(REG)$.

Alternatively, regular languages can be described by right-linear grammars.

Definition 9. A *right-linear grammar* is a phrase-structure grammar

$$G = (V, T, P, S)$$

such that each production in P is of the form

$$A \to xB, \text{ or } A \to x,$$

where $A, B \in V - T$ and $x \in T^*$. The family of languages generated by right-linear grammars is denoted by $\mathscr{L}(RLIN)$.

For the families of languages generated by regular, right-linear, linear, context-free, context-sensitive, and phrase-structure grammars, it holds:

Theorem 1 (see [4])**.**

$$\mathscr{L}(REG) = \mathscr{L}(RLIN) \subset \mathscr{L}(LIN) \subset \mathscr{L}(CF) \subset \mathscr{L}(CS) \subset \mathscr{L}(RE).$$

Lemma 1 (Kuroda Normal Form of Context-Sensitive Grammars, see [28])**.** *Let $L \in \mathscr{L}(CS)$. Then, there exists a context-sensitive grammar $G = (V, T, P, S)$ such that $L = L(G)$ and each production in P is either of the form*

- $AB \to CD$, *or*
- $A \to x$,

where $A, B, C, D \in V - T$, $x \in T \cup (V - T)^2$.

Lemma 2 (Kuroda Normal Form of Phrase-Structure Grammars, see [28])**.** *Let $L \in \mathscr{L}(RE)$. Then, there exists a phrase-structure grammar, $G = (V, T, P, S)$, such that $L = L(G)$ and each production in P is either of the form*

- $AB \to CD$, *or*
- $A \to x$,

where $A, B, C, D \in V - T$, $x \in \{\varepsilon\} \cup T \cup (V - T)^2$.

Besides phrase-structure grammars, we also use simple matrix grammars, state grammars, Extended Post Correspondence, and queue grammars to describe languages in this thesis. We start with the definition of a linear simple matrix grammar.

Definition 10. A *simple matrix grammar of degree n* (see [21]) is an $(n+3)$-tuple

$$G = (V_1, \ldots, V_n, T, P, S),$$

where $n \geq 1$, $V_1, \ldots, V_n$ are alphabets , $T \subset V_i$ for all $1 \leq i \leq n$, $(V_1 - T), \ldots, (V_n - T)$ are pairwise disjoint, $S \notin V_1 \cup \ldots \cup V_n$, and P is a finite set of productions of the following three forms:

1. $(S) \to (x_{11}A_{11}x_{12}A_{12}\ldots x_{1k}A_{1k}\ldots x_{n1}A_{n1}x_{n2}A_{n2}\ldots x_{nk}A_{nk}y)$, where $y \in T^*$, and $A_{ij} \in V_i - T$, $x_{ij} \in T^*$ for all $1 \leq i \leq n$, $1 \leq j \leq k$, for some $k \geq 1$,
2. $(A_1, \ldots, A_n) \to (x_{11}A_{11}x_{12}A_{12}\ldots x_{1k}A_{1k}y_1, \ldots, x_{n1}A_{n1}x_{n2}A_{n2}\ldots x_{nk}A_{nk}y_n)$, where $y_i, x_{ij} \in T^*$, $A_i, A_{ij} \in V_i - T$ for all $1 \leq i \leq n$, $1 \leq j \leq k$, for some $k \geq 1$,
3. $(A_1, \ldots, A_n) \to (x_1, \ldots, x_n)$, where $A_i \in V_i - T$, $x_i \in T^*$ for all $1 \leq i \leq n$.

If

- either $u = S$ and $p = (S) \to (v) \in P$,
- or $u = y_1A_1z_1\ldots y_nA_nz_n$, $v = y_1w_1z_1\ldots y_nw_nz_n$, where $y_i \in T^*$, $z_i \in V_i^*$ for all $1 \leq i \leq n$, and $p = (A_1, \ldots, A_n) \to (w_1, \ldots, w_n) \in P$,

then G makes a derivation step from u to v according to p, symbolically written as $u \Rightarrow_G v\ [p]$. Let $\Rightarrow_G^+$ and $\Rightarrow_G^*$ denote the transitive closure of $\Rightarrow_G$ and the transitive-reflexive closure of $\Rightarrow_G$, respectively. The language of G is denoted by $L(G)$ and defined as $L(G) = \{x \in T^* : S \Rightarrow_G^* x\}$. The family of languages generated by simple matrix grammars of degree n is denoted by $\mathscr{L}(SM, n)$, and

$$\mathscr{L}(SM) = \bigcup_{n=1}^{\infty} \mathscr{L}(SM, n).$$

Definition 11. A *linear simple matrix grammar of degree n* (see [57]) is a simple matrix grammar

$$G = (V_1, \ldots, V_n, T, P, S),$$

where P is a finite set of productions of the following three forms:

1. $(S) \to (x_1A_1\ldots x_nA_nx_{n+1})$, where $A_i \in V_i - T$, $x_j \in T^*$ for all $1 \leq i \leq n$, $1 \leq j \leq n+1$,
2. $(A_1, \ldots, A_n) \to (x_1B_1y_1, \ldots, x_nB_ny_n)$, where $A_i, B_i \in V_i - T$, $x_i, y_i \in T^*$ for all $1 \leq i \leq n$,
3. $(A_1, \ldots, A_n) \to (x_1, \ldots, x_n)$, where $A_i \in V_i - T$, $x_i \in T^*$ for all $1 \leq i \leq n$.

The family of languages generated by linear simple matrix grammars of degree n is denoted by $\mathscr{L}(SM, LIN, n)$, and

$$\mathscr{L}(SM, LIN) = \bigcup_{n=1}^{\infty} \mathscr{L}(SM, LIN, n).$$

Definition 12. A *right-linear simple matrix grammar of degree n* (see [21]) is a linear simple matrix grammar

$$G = (V_1, \ldots, V_n, T, P, S),$$

where P is a finite set of productions of the following three forms:

1. $(S) \to (x_1A_1\ldots x_nA_n)$, where $A_i \in V_i - T$, $x_i \in T^*$ for all $1 \leq i \leq n$,

2. $(A_1,\ldots,A_n) \to (x_1B_1,\ldots,x_nB_n)$, where $A_i,B_i \in V_i - T$, $x_i \in T^*$ for all $1 \le i \le n$,

3. $(A_1,\ldots,A_n) \to (x_1,\ldots,x_n)$, where $A_i \in V_i - T$, $x_i \in T^*$ for all $1 \le i \le n$.

The family of languages generated by right-linear simple matrix grammars of degree n is denoted by $\mathscr{L}(SM,RLIN,n)$, and

$$\mathscr{L}(SM,RLIN) = \bigcup_{n=1}^{\infty} \mathscr{L}(SM,RLIN,n).$$

Theorem 2 (see [21, 57])**.** *For each $n \ge 1$,*

$$\begin{aligned} \mathscr{L}(SM,n) &\subset \mathscr{L}(SM,n+1),\\ \mathscr{L}(SM,LIN,n) &\subset \mathscr{L}(SM,LIN,n+1),\\ \mathscr{L}(SM,RLIN,n) &\subset \mathscr{L}(SM,RLIN,n+1), \end{aligned}$$

$$\mathscr{L}(SM,RLIN,n) \subset \mathscr{L}(SM,LIN,n) \subset \mathscr{L}(SM,n).$$

Theorem 3 (see [21, 57])**.**

$$\mathscr{L}(CF) - \mathscr{L}(SM,LIN) \neq \emptyset,\ \mathscr{L}(CF) - \mathscr{L}(SM,RLIN) \neq \emptyset$$

$$\mathscr{L}(SM,RLIN) \subset \mathscr{L}(SM,LIN) \subset \mathscr{L}(SM) \subset \mathscr{L}(CS).$$

Theorem 4 (Positive closure properties, see [21, 57])**.** *Each of the families $\mathscr{L}(SM,LIN,n)$ and $\mathscr{L}(SM,RLIN,n)$, where $n \ge 1$, is closed under union, reversal, homomorphism, inverse homomorphism, substitution with regular languages, concatenation with regular languages, intersection with regular languages, left and right quotient by regular languages. $\mathscr{L}(SM,LIN)$ and $\mathscr{L}(SM,RLIN)$ are closed under concatenation.*

Theorem 5 (Negative closure properties, see [21, 57])**.** *Each family $\mathscr{L}(SM,LIN,n)$, where $n \ge 1$, is not closed under concatenation with linear languages. Each family $\mathscr{L}(SM,RLIN,n)$, where $n \ge 1$, is not closed under concatenation with $\mathscr{L}(SM,RLIN,2)$. $\mathscr{L}(SM,LIN)$ and $\mathscr{L}(SM,RLIN)$ are not closed under intersection, complement and Kleene star. $\mathscr{L}(SM,LIN)$ is not closed under substitution with linear languages. $\mathscr{L}(SM,RLIN)$ is not closed under substitution with $\mathscr{L}(SM,RLIN,2)$.*

Now, we define another type of a grammar.

Definition 13. A *state grammar* (see [22]) is a sixtuple

$$G = (V,T,K,P,S,p_0),$$

where V is an alphabet, $T \subset V$, K is a finite set of states, $S \in V - T$, $p_0 \in K$, and P is a finite set of productions of the form

$$(A,p) \to (x,q),$$

where $A \in V - T$, $x \in V^+$, and $p,q \in K$. If $u = (rAs,p)$, $v = (rxs,q)$, and $(A,p) \to (x,q) \in P$, where $r,s \in V^*$, and for each $(B,p) \to (y,t) \in P$, $B \notin \mathrm{alph}(r)$, then G makes a derivation step from u to v according to $(A,p) \to (x,q)$, symbolically written as

$$u \Rightarrow_G v\ [(A,p) \to (x,q)]$$

or, simply, $u \Rightarrow_G v$. To emphasize that the jth nonterminal in u is rewritten during a derivation step, we write $u \stackrel{j}{\Rightarrow}_G v$. The *state language* is a language generated by a state grammar G, denoted by $L(G)$, and defined as

$$L(G) = \{x \in T^* : (S, p_0) \Rightarrow_G^* (x, q) \text{ for some } q \in K\}.$$

The family of all state languages is denoted by $\mathscr{L}(ST)$. An *n-limited derivation*, denoted by $x \stackrel{n}{\Rightarrow}_G^* y$, is a derivation in which every derivation step $u \stackrel{j}{\Rightarrow}_G v$ satisfies $j \leq n$. Define

$$L(G,n) = \{x \in T^* : (S, p_0) \stackrel{n}{\Rightarrow}_G^* (x, q) \text{ for some } q \in K\};$$

clearly, $L(G) = \bigcup_{n=1}^{\infty} L(G,n)$. A state grammar G is *of degree n* if and only if $L(G,n) = L(G)$. A state language is said to be of degree n if there is a state grammar G of degree n with $L = L(G)$. If $L(G,n) \neq L(G)$ for all $n \geq 1$, then G and $L(G)$ are said to be *of infinite degree*. The family of state languages of degree n is denoted by $\mathscr{L}(ST,n)$, and $\mathscr{L}(ST,\infty) = \bigcup_{n=1}^{\infty} \mathscr{L}(ST,n)$.

Theorem 6 (see [22])**.** *A state language L is of degree n if and only if $L = L(G,n)$ for some state grammar G.*

Theorem 7 (see [22])**.**

$$\mathscr{L}(CF) = \mathscr{L}(ST,1) \subset \mathscr{L}(ST,2) \subset \ldots \subset \mathscr{L}(ST,\infty) \subset \mathscr{L}(ST) = \mathscr{L}(CS).$$

Theorem 8 (see [22])**.** *Every $\mathscr{L}(ST,n)$, where $n \geq 1$, is an abstract family of languages.*

Next, we continue by defining an Extended Post Correspondence.

Definition 14. For an alphabet $T = \{a_1, \ldots, a_n\}$, an *Extended Post Correspondence* (see [12]) E is 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 $1 \leq i \leq r$ and $1 \leq j \leq n$. The language represented by an Extended Post Correspondence E, denoted by $L(E)$, is defined as

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

Theorem 9 (see [12])**.** *For each $L \in \mathscr{L}(RE)$ there exists an Extended Post Correspondence E such that $L(E) = L$.*

Finally, a queue grammar is defined, a normal form of its productions, which is used throughout the thesis, is introduced, and it is proved that for every queue grammar there exists a queue grammar in this normal form.

Definition 15. A *queue grammar* (see [25]) is a sixtuple

$$Q = (V, T, W, F, R, g),$$

where V and W are alphabets satisfying $V \cap W = \emptyset$, $T \subset V$, $F \subset W$, $g \in (V - T)(W - F)$, and

$$R \subseteq (V \times (W - F)) \times (V^* \times W)$$

is a finite relation such that for each $a \in V$, there exists an element $(a,b,x,c) \in R$. If $u = arb$, $v = rxc$, and $(a,b,x,c) \in R$, $r,x \in V^*$, where $a \in V$ and $b,c \in W$, then Q makes a derivation step from u to v according to (a,b,x,c), symbolically written as

$$u \Rightarrow_Q v\ [(a,b,x,c)],$$

or, simply, $u \Rightarrow_Q v$. The language generated by a queue grammar Q, denoted by $L(Q)$, is defined as

$$L(Q) = \{x \in T^* : g \Rightarrow_Q^+ xf, f \in F\}.$$

Theorem 10 (see [25]). *For each $L \in \mathscr{L}(RE)$ there exists a queue grammar Q such that $L(Q) = L$.*

Lemma 3. *Let Q' be a queue grammar. Then, there exists a queue grammar*

$$Q = (V,T,W' \cup \{1,f\},\{f\},R,g)$$

such that $L(Q') = L(Q)$, where $W' \cap \{1,f\} = \emptyset$, each $(a,b,x,c) \in R$ satisfies $a \in V - T$ and either

- *$b \in W'$, $x \in (V-T)^*$, $c \in W' \cup \{1,f\}$ or*
- *$b = 1$, $x \in T$, $c \in \{1,f\}$.*

Proof. Let $Q' = (V',T,W',F',R',g')$ be any queue grammar. Set $\Phi = \{\bar{a} : a \in T\}$. Define the homomorphism α from $(V')^*$ to $((V'-T) \cup \Phi)^*$ as $\alpha(a) = \bar{a}$ for each $a \in T$ and $\alpha(A) = A$ for each $A \in V' - T$. Set $V = V' \cup \Phi$, $W = W' \cup \{1,f\}$, $F = \{f\}$, and $g = \alpha(a_0)q_0$ for $g' = a_0q_0$. Define the queue grammar $Q = (V,T,W,F,R,g)$, with R constructed in the following way:

1. For each $(a,b,x,c) \in R'$, where $c \in W' - F'$, add $(\alpha(a),b,\alpha(x),c)$ to R;
2. (a) For each $(a,b,x,c) \in R'$, where $c \in F'$, add $(\alpha(a),b,\alpha(x),1)$ to R;
   (b) For each $(a,b,\varepsilon,c) \in R'$, where $c \in F'$, add $(\alpha(a),b,\varepsilon,f)$ to R;
3. For each $a \in T$, add
   (a) $(\bar{a},1,a,1)$ and
   (b) $(\bar{a},1,a,f)$ to R.

Clearly, each $(a,b,x,c) \in R$ satisfies $a \in V - T$ and either $b \in W'$, $x \in (V-T)^*$, $c \in W' \cup \{1,f\}$ or $b = 1$, $x \in T$, $c \in \{1,f\}$.

To see that $L(Q') \subseteq L(Q)$, consider any $v \in L(Q')$. As $v \in L(Q')$, $g' \Rightarrow_{Q'}^* vt$, where $v \in T^*$ and $t \in F'$. Express $g' \Rightarrow_{Q'}^* vt$ as

$$g' \Rightarrow_{Q'}^* axc \Rightarrow_{Q'} vt\ [(a,c,y,t)],$$

where $a \in V'$, $x,y \in T^*$, $xy = v$, and $c \in W' - F'$. This derivation is simulated by Q as follows. First, Q uses productions from (1) to simulate $g' \Rightarrow_{Q'}^* axc$. Then, it uses a production from (2) to simulate $axc \Rightarrow_{Q'} vt$. For $x = \varepsilon$, a production from (2b) can be used to generate $\varepsilon \in L(Q)$ in the case of $\varepsilon \in L(Q')$; otherwise a production from (2a) is used. This part of simulation can be expressed as

$$g \Rightarrow_Q^* \alpha(ax)c \Rightarrow_Q \alpha(v)1.$$

At this point, $\alpha(v)$ satisfies $\alpha(v) = \bar{a}_1 \dots \bar{a}_n$, where $a_i \in T$ for all $1 \leq i \leq n$, for some $n \geq 1$. The productions from (3) of the form $(\bar{a}, 1, a, 1)$, where $a \in T$, replace every $\bar{a}_j$ with a_j, where $1 \leq j \leq n-1$, and, finally, $(\bar{a}, 1, a, f)$, where $a \in T$, replaces $\alpha(a_n)$ with a_n. As a result, we obtain the sentence vf, so $L(Q') \subseteq L(Q)$.

To establish $L(Q) \subseteq L(Q')$, observe that the use of a production from (2b) in Q before the sentential form is of the form $\alpha(ax)c$, where $a \in V'$, $x \in T^*$, $c \in W' - F'$, leads to an unsuccessful derivation. Similarly, the use of (2b) if $x \neq \varepsilon$ leads to an unsuccessful derivation as well. The details are left to the reader. As a result, $L(Q) \subseteq L(Q')$.

As $L(Q') \subseteq L(Q)$ and $L(Q) \subseteq L(Q')$, we obtain $L(Q) = L(Q')$. ■

Consider the queue grammar $Q = (V, T, W, F, R, g)$ from Lemma 3. Its properties imply that Q generates every word in $L(Q) - \{\varepsilon\}$ so it passes through 1. Before it enters 1, it generates only words from $(V-T)^*$; after entering 1, it generates only words from T. The following two corollaries express this property formally.

Corollary 1. *Let Q be a queue grammar that satisfies the properties given in Lemma 3. Then, Q generates every $y \in L(Q) - \{\varepsilon\}$ in this way:*

$$\begin{aligned}
a_0 q_0 &\Rightarrow_Q x_0 q_1 && [(a_0, q_0, z_0, q_1)] \\
&\;\;\vdots \\
&\Rightarrow_Q x_{k-1} q_k && [(a_{k-1}, q_{k-1}, z_{k-1}, q_k)] \\
&\Rightarrow_Q x_k 1 && [(a_k, q_k, z_k, 1)] \\
&\Rightarrow_Q x_{k+1} b_1 1 && [(a_{k+1}, 1, b_1, 1)] \\
&\;\;\vdots \\
&\Rightarrow_Q x_{k+m-1} b_1 \dots b_{m-1} 1 && [(a_{k+m-1}, 1, b_{m-1}, 1)] \\
&\Rightarrow_Q b_1 \dots b_m f && [(a_{k+m}, 1, b_m, f)],
\end{aligned}$$

where $k, m \geq 1$, $g = a_0 q_0$, $a_1, \dots, a_{k+m} \in V - T$, $b_1, \dots, b_m \in T$, $z_0, \dots, z_k \in (V-T)^*$, $q_0, \dots, q_k, 1 \in W - F$, $f \in F$, $x_0, \dots, x_{k+m-1} \in (V-T)^+$, and $y = b_1 \dots b_m$. ■

Corollary 2. *Let Q be a queue grammar that satisfies the properties given in Lemma 3. Then, Q generates a non-empty string of terminals during the last step of every successful derivation of a sentence from $L(Q) - \{\varepsilon\}$.* ■

If some grammars define the same language, they are referred to as *equivalent grammars*. This equivalence is central to this thesis because we often discuss how to transform some grammars to some other grammars so that both the original grammars and the transformed grammars are equivalent.

2.3 Scattered Context Grammars

In this section, we define scattered context grammars without erasing productions, originally introduced in [16], and scattered context grammars with erasing productions studied in [82]. In addition, we illustrate these definitions on several examples.

Sometimes, the notions used in this theses differ from the notions originally used in the research papers discussing this topic. The reason for this change is the fact that each paper used a slightly different terminology so some terms had to be modified to to present all the results in a unified way.

Definition 16. A *scattered context grammar* is a quadruple

$$G = (V, T, P, S),$$

where

- V is the total alphabet,
- $T \subset V$ is the set of terminals,
- P is a finite set of productions of the form

  $$(A_1, \ldots, A_n) \to (x_1, \ldots, x_n),$$

  where $n \geq 1$, $A_i \in V - T$ and $x_i \in V^*$ for all $1 \leq i \leq n$,
- $S \in V - T$ is the start symbol of G.

If

$$u = u_1 A_1 u_2 A_2 \ldots u_n A_n u_{n+1},$$
$$v = u_1 x_1 u_2 x_2 \ldots u_n x_n u_{n+1},$$

and

$$p = (A_1, \ldots, A_n) \to (x_1, \ldots, x_n) \in P,$$

where $u_i \in V^*$ for all $1 \leq i \leq n+1$, then G makes a derivation step from u to v according to p, symbolically written as

$$u \Rightarrow_G v\ [p],$$

or, simply, $u \Rightarrow_G v$. In addition, if $A_i \notin \text{alph}(u_i)$ for all $1 \leq i \leq n$, then the direct derivation is *leftmost*, and we write

$$u \ {}_{\text{lm}}\Rightarrow_G v\ [p];$$

if $A_i \notin \text{alph}(u_{i+1})$ for all $1 \leq i \leq n$, then the direct derivation is *rightmost*, and we write

$$u \ {}_{\text{rm}}\Rightarrow_G v\ [p].$$

Set $\text{lhs}(p) = A_1 A_2 \ldots A_n$, $\text{rhs}(p) = x_1 x_2 \ldots x_n$, and $\text{len}(p) = |A_1 \ldots A_n| = n$. If $\text{len}(p) \geq 2$, p is said to be a *context-sensitive production*, while for $\text{len}(p) = 1$, p is said to be *context-free*. The *scattered context language*, is a language generated by a scattered context grammar $G = (V, T, P, S)$, denoted by $L(G)$, and defined as

$$L(G) = \{x \in T^* : S \Rightarrow_G^* x\}.$$

The family of languages generated by scattered context grammars is denoted by $\mathscr{L}(SC)$.

Definition 17. A *propagating scattered context grammar* is a scattered context grammar

$$G = (V, T, P, S)$$

in which each

$$(A_1, \ldots, A_n) \to (x_1, \ldots, x_n) \in P$$

satisfies $x_i \in V^+$ for all $1 \leq i \leq n$. The *propagating scattered context language* is a language generated by a propagating scattered context grammar. The family of languages generated by propagating scattered context grammars is denoted by $\mathscr{L}(PSC)$.

Example 1. Consider the non-context-free language $L = \{a^n b^n c^n : n \geq 0\}$. This language can be generated by the scattered context grammar

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

where

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

For example, the sentence *aabbcc* is generated by G as follows:

$$S \Rightarrow_G ABC \Rightarrow_G aAbBcC \Rightarrow_G aaAbbBccC \Rightarrow_G aabbcc.$$

The generated language, however, belongs to the family of propagating scattered context languages as there exists the grammar

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

where

$$\begin{aligned} P' = \{&(S) \to (\varepsilon),\\ &(S) \to (ABC),\\ &(A,B,C) \to (aA,bB,cC),\\ &(A,B,C) \to (a,b,c)\}. \end{aligned}$$

In order to generate $\varepsilon \in L$ by G', P' contains the production $(S) \to (\varepsilon)$, and S does not appear on any production's right-hand side. As it is always possible to add the production $(S) \to (\varepsilon)$, and not allow S to appear on a right-hand side of any production, to simplify the notation, $\varepsilon \in L$, where L is a propagating scattered context language, is not discussed in the rest of the thesis.

Example 2 (see [5, 33])**.** Consider the non-context-free language

$$L = \{a^{2^n} : n \geq 0\}.$$

This language is generated by the propagating scattered context grammar

$$G = (\{S,W,X,Y,Z,A,a\},\{a\},P,S),$$

where

$$\begin{aligned} P = \{&1 : (S) \to (a),\\ &2 : (S) \to (aa),\\ &3 : (S) \to (WAXY),\\ &4 : (W,A,X,Y) \to (a,W,X,AAY),\\ &5 : (W,X,Y) \to (a,W,AXY),\\ &6 : (W,X,Y) \to (Z,Z,a),\\ &7 : (Z,A,Z) \to (Z,a,Z),\\ &8 : (Z,Z) \to (a,a)\}. \end{aligned}$$

In what follows, we demonstrate that $L(G) = L$. The productions (1) and (2) generate the strings of $\{a^{2^n} : n = 0,1\}$ while (3) starts off the derivation of other strings in L. Consider the derivation of

$a^{16} \in L(G)$:

$$\begin{aligned}
S &\Rightarrow_G WAXY && [3]\\
&\Rightarrow_G aWXA^2Y && [4]\\
&\Rightarrow_G a^2WA^3XY && [5]\\
&\Rightarrow_G a^3WA^2XA^2Y && [4]\\
&\Rightarrow_G a^4WAXA^4Y && [4]\\
&\Rightarrow_G a^5WXA^6Y && [4]\\
&\Rightarrow_G a^6WA^7XY && [5]\\
&\Rightarrow_G a^6ZA^7Za && [6]\\
&\Rightarrow_G^7 a^{13}ZZa && [7^7]\\
&\Rightarrow_G a^{16} && [8].
\end{aligned}$$

Observe that in a successful derivation, productions (4) and (5) are applied in a cycle, and after the required number of A's is obtained, the derivation is finished by productions (6), (7), and (8). In greater detail, observe that the production $(W,A,X,Y) \rightarrow (a,W,X,AAY)$ removes one A between W and X, and inserts two A's between X and Y. In a successful derivation, this production has to rewrite the leftmost nonterminal A. After all A's between W and X are removed, the production $(W,X,Y) \rightarrow (a,W,AXY)$ can be used to bring all A's between X and Y back between W and X, and the cycle can repeat again. Alternatively, the production (6) can be used, which initializes the final phase of the derivation in which all A's are replaced with a's by productions (7) and (8).

Finally, we introduce several measures of the descriptional complexity of scattered context grammars which are used in Chapters 3 and 6.

Definition 18. Let $G = (V,T,P,S)$ be a scattered context grammar. Then, its *nonterminal complexity* is the number of nonterminals in G. If G is a scattered context grammar, then its *degree of context sensitivity*, symbolically written as $\mathrm{dcs}(G)$, is defined as the number of context-sensitive productions in G. The *maximum context sensitivity* of G is the greatest number in

$$\{\mathrm{len}(p_i) - 1 \,:\, 1 \leq i \leq |P|\},$$

symbolically denoted by $\mathrm{mcs}(G)$. The *overall context sensitivity* of G, denoted by $\mathrm{ocs}(G)$, is the sum of all members in

$$\{\mathrm{len}(p_i) - 1 \,:\, 1 \leq i \leq |P|\}.$$

Chapter 3

Related Work

This chapter summarizes the known results of the research into scattered context grammars with and without erasing productions. Since their introduction in [16], nearly forty years ago, there have been published many papers dealing with this subject. In short, the studied topics can be divided into three most important areas: (1) general study of propagating scattered context grammars and the generated language families; (2) study of modified versions of propagating scattered context grammars; (3) study of scattered context grammars with erasing productions. In what follows, we look at these areas more deeply.

Properties of Propagating Scattered Context Grammars

In this section, we describe the properties of propagating scattered context grammars. We start with a normal form that was first mentioned in the original article [16] by S. Greibach and J. Hopcroft. Scattered context productions in this normal form contain at most two context-free components while in general, their number is unlimited.

Definition 19. A *2-limited propagating scattered context grammar* is a propagating scattered context grammar $G = (V, T, P, S)$ such that

1. $(A_1, \ldots, A_n) \to (w_1, \ldots, w_n) \in P$ implies $n \leq 2$, and for each $1 \leq i \leq n$, $1 \leq |w_i| \leq 2$ and $w_i \in (V - \{S\})^*$,
2. $(A) \to (w) \in P$ implies $A = S$.

The paper shows that every propagating scattered context language can be described by the above normal form.

Theorem 11. *If G is a propagating scattered context grammar, then there exists a 2-limited grammar $\bar{G}$ with $L(\bar{G}) = L(G)$.*

Next, the same paper studies the closure properties of the family of propagating scattered context languages under various operations and demonstrates that this family properly contains the family of context-free languages. The closure properties are also studied for a special type of homomorphism and erasing which are referred to as k-restricted and k-limited, respectively.

Definition 20. A family of languages $\mathscr{L}$ is said to be *closed under k-limited erasing*, where k is a positive integer, if whenever $c \notin T$ and $L \in \mathscr{L}$ with

$$L \subseteq (T(\{\varepsilon\} \cup \{c\} \cup \ldots \cup \{c\}^k))^*,$$

$h(c) = \varepsilon$ and $h(a) = a$ for each $a \in T$, then $h(L) \in \mathscr{L}$.

Definition 21. A homomorphism h from T_1^* to T_2^* is *k-restricted* on $L \subseteq T_1^*$ if $h(w) = \varepsilon$ for $w \in L$ implies $w = \varepsilon$, and $h(w) \neq \varepsilon$ for every subword w, $|w| \geq k$ of each word in L. A family of languages $\mathscr{L}$ is said to be *closed under restricted homomorphism* if $h(L) \in \mathscr{L}$ whenever $L \subseteq T_1^*$ is in $\mathscr{L}$ and h is a homomorphism on T_1^* which is $k-1$ restricted on L for some positive integer k.

The following theorem summarizes the main closure results.

Theorem 12. *The family of propagating scattered context languages is closed under substitution by an ε-free context free language, intersection with a regular language, permutations, ε-free homomorphism, reversal, restricted homomorphism, k-limited erasing, inverse homomorphism, concatenation, intersection, ε-free substitution, and linear erasing.*

The following corollaries follow from the results given in Theorem 12.

Corollary 3. $\mathscr{L}(CF) \subset \mathscr{L}(PSC) \subseteq \mathscr{L}(CS)$.

Corollary 4. *The family of propagating scattered context languages is an abstract family of languages.*

Corollary 5. *If L is a recursively enumerable language, then there exists a propagating scattered context language L' and a homomorphism h such that $h(L') = L$.*

Corollary 6. *The family of propagating scattered context languages is not closed under arbitrary homomorphism and quotient by a regular language.*

Corollary 7. *The emptiness problem is recursively unsolvable for propagating scattered context grammars.*

The paper introduces one important open problem which has remained unsolved since then.

Open Problem 1. $\mathscr{L}(CS) = \mathscr{L}(PSC)$?

The generative power of propagating scattered context grammars was further studied by [58]. This paper compares their generative power to simple matrix grammars and proves the following result.

Theorem 13. $\mathscr{L}(SM) \subset \mathscr{L}(PSC)$.

Special attention was payed to Corollary 5 by various papers. First, [7] demonstrated that any recursively enumerable language can be characterized by a propagating scattered context grammar so that every string x of this language is represented by $\{\$\}^* x$ in the propagating scattered context grammar, where \$ is an arbitrary symbol not present in the language. The interesting part of this result is the fact that any recursively enumerable language can be described by a grammar whose power is at most context-sensitive by adding some fill-in symbols into every sentence. Formally, the result is stated as follows.

Theorem 14. *For every recursively enumerable language L, there exists a propagating scattered context grammar G such that $L = \{\$\}^+ \backslash\backslash L(G)$.*

In Chapter 6, we perform a generation of this kind as well. However, instead of adding useless symbols \$ to every sentence, we try to increase the amount of information every string contains by adding supplementary data which may be useful for subsequent parsing of this string. Theorem 14 was later improved by [37]. It proved that such a language can be generated by a propagating scattered context grammar which performs only leftmost or rightmost derivations. In fact, it demonstrated that there is no need to explicitly restrict the grammar—if the grammar performs a

non-leftmost derivation step, the generation is unsuccessful. Accordingly, we modify our generators in Chapter 6 to satisfy this property as well. This topic was further studied by [38], where the total number of the nonterminals needed for this generation was reduced to four. In relation to this result, we discuss reduced generators in Chapter 6 which, in addition, perform only leftmost derivations and use a reduced number of context-sensitive productions. A slightly different approach was presented in [46]. Instead of creating the sequence of the fill-in symbols, \$, at the beginning or at the end of the generated sentence, they are spread over the sentence so that they appear between every two neighboring terminal symbols. The paper further demonstrates that the grammar can be constructed in such a way that there is the same number of these symbols between every two neighboring terminals in every string generated by the grammar.

Modifications of Propagating Scattered Context Grammars

Open Problem 1 mentioned in the previous section led to various modifications of propagating scattered context grammars which made it possible to describe all context-sensitive languages.

One of the first modifications was discussed in [56]. The paper uses the notion of *negative-context grammars*. In essence, negative-context grammars are context-free grammars in which each context-free production has two sets of nonterminals, L and R, appended to it. A production is applicable to a sentential form if and only if there is no symbol from L on the left of the nonterminal to be rewritten and no symbol from R on the right of this nonterminal. The paper demonstrates that if scattered context productions are added to such a grammar, the resulting grammar is powerful enough to describe all context-sensitive languages.

A different approach was presented in [5]. This paper defines three kinds of context-free productions: *leftmost*, *checking*, and *global*. Productions in leftmost mode are applicable if and only if in front of the nonterminal to be rewritten there does not occur the same nonterminal. In checking mode, the application of a production depends on whether the sentential form contains the nonterminal from its left-hand side or not: if it does, the production is applied; otherwise, the sentential form remains unchanged. (This mode resembles appearance checking in matrix grammars, see [6] for details.) Finally, a production in global mode rewrites all nonterminals in the sentential form which coincide with its left-hand side in one derivation step. By allowing these kinds of productions to be components of propagating scattered context productions and combining them with ordinary context-free productions, the authors characterize the family of context-sensitive languages.

Propagating scattered context grammars with regulation, namely *propagating scattered context grammars with appearance checking* and *propagating scattered context grammars with unconditional transfer* were introduced in [9]. For its relative complexity, we do not give the formal definition of this modification in this paper. If interested, the reader can find more information about these grammars in [9]. Again, the paper shows that propagating scattered context grammars with appearance checking and unconditional transfer are able to generate all context-sensitive languages.

The following two modifications are more interesting from our point of view because they change the basic definition of a propagating scattered context grammar only slightly, preserving its uniformity and simplicity. *Propagating scattered context grammars which use leftmost derivations* were studied by [82]. It was proved that propagating scattered context grammars which use only leftmost derivations characterize the family of context-sensitive languages. In Section 5.3, we present a much simplified proof of the same result. Finally, [15] studied scattered context grammars with erasing productions in which the length of the concatenation of nonterminals appearing on the left-hand side of a production has to be less or equal to the concatenation of the strings appearing on its right-hand side. We give a formal definition of this grammar next.

Definition 22. An *extended propagating scattered context grammar* is a scattered context grammar $G = (V,T,P,S)$ in which every $(A_1,\ldots,A_n) \to (x_1,\ldots,x_n) \in P$ satisfies

$$|x_1 \ldots x_n| \geq n.$$

The family of languages generated by extended propagating scattered context grammars is denoted by $\mathscr{L}(PSC,\text{ext})$.

The paper proves the following theorem.

Theorem 15. $\mathscr{L}(PSC,\text{ext}) = \mathscr{L}(CS)$.

In conclusion of this section, we point out to an interesting fact. It seems that the presented results speak in favor of the hypothesis that propagating scattered context grammars are not powerful enough to describe all context-sensitive languages. Indeed, each of the presented grammars requires some additional modification of the basic concept to be able to generate all context-sensitive languages. On the other hand, no one has been able to prove this hypothesis yet.

Properties of Scattered Context Grammars

Scattered context grammars with erasing productions were first mentioned in [26]. In [82] it was proved that scattered context grammars with erasing productions are powerful enough to describe all recursively enumerable languages. Later, [39] demonstrated a trivial proof of this result.

Theorem 16. $\mathscr{L}(RE) = \mathscr{L}(SC)$.

Proof. Let $L \in \mathscr{L}(RE)$. By Corollary 5, there exists a propagating scattered context grammar $G' = (V,T',P',S)$ and a homomorphism h such that $L = h(L(G'))$. Without loss of generality assume $\text{alph}(L) \cap T' = \emptyset$. Define the scattered context grammar

$$G = (V \cup \text{alph}(L), \text{alph}(L), P \cup P', S),$$

where

$$P = \{(a) \to (h(a)) : a \in T'\}.$$

Clearly, $L(G) = L$. Therefore, $\mathscr{L}(RE) \subseteq \mathscr{L}(SC)$. Obviously, $\mathscr{L}(SC) \subseteq \mathscr{L}(RE)$, so $\mathscr{L}(RE) = \mathscr{L}(SC)$. ■

Further, [82] proved that if only leftmost derivation steps are used in scattered context grammars, it can describe every recursively enumerable language as well.

Other theoretical properties of scattered context grammars were studied by [44, 45]. We do not describe the results of these papers in greater detail; however, we present an interesting corollary of the main result in [44]. It states that scattered context grammars which contain only one nonterminal cannot characterize all recursively enumerable languages.

Particular attention was paid to the descriptional complexity of scattered context grammars. There have been many papers dealing with this topic, often improving previously published results. For example, all results proved in [10, 11, 40] were improved by the subsequent papers. In [43] it was demonstrated that every recursively enumerable language can be generated by a scattered context grammar with at most three nonterminals.

Theorem 17. *For every recursively enumerable language L, there is a scattered context grammar* $G = (V,T,P,S)$ *satisfying* $|V - T| = 3$, $\text{dcs}(G) = \infty$, $\text{mcs}(G) = \infty$, $\text{ocs}(G) = \infty$, *such that* $L(G) = L$.

The same paper also stated the following open problem.

Open Problem 2. Are two-nonterminal scattered context grammars powerful enough to characterize the family of recursively enumerable languages?

There have been several attempts to reduce both the number of nonterminals and the number of context-sensitive productions. Two results of this kind are from [48, 81].

Theorem 18. *For every recursively enumerable language L, there is a scattered context grammar $G = (V,T,P,S)$ satisfying $|V-T| = 5$, $\mathrm{dcs}(G) = 2$, $\mathrm{mcs}(G) = 3$, $\mathrm{ocs}(G) = 6$, such that $L(G) = L$.*

Theorem 19. *For every recursively enumerable language L, there is a scattered context grammar $G = (V,T,P,S)$ satisfying $|V-T| = 8$, $\mathrm{dcs}(G) = 6$, $\mathrm{mcs}(G) = 1$, $\mathrm{ocs}(G) = 6$, such that $L(G) = L$.*

Another, not yet published result was presented in [34].

Theorem 20. *For every recursively enumerable language L, there is a scattered context grammar $G = (V,T,P,S)$ satisfying $|V-T| = 4$, $\mathrm{dcs}(G) = 4$, $\mathrm{mcs}(G) = 5$, $\mathrm{ocs}(G) = 20$, such that $L(G) = L$.*

Apart form reducing the total number of nonterminals and context-sensitive productions, [41] studied the way how to transform phrase-structure grammars to scattered context grammars as economically as possible. It demonstrated that when converting a phrase-structure grammar in Kuroda normal form to a scattered context grammar, only a small number of extra nonterminals and context-sensitive productions has to be added. To describe this result precisely, we first define the needed notation.

Definition 23. Let G be a grammar, and let P be G's set of productions. We separate P into two disjoint subsets—the *set of context-free productions*, $\mathrm{cfree}(P)$, and the *set of context-dependent productions*, $\mathrm{cdep}(P)$. A production, $p \in P$, belongs to $\mathrm{cfree}(P)$ if and only if the left-hand side of p consists of one nonterminal; otherwise, p belongs to $\mathrm{cdep}(P)$. Specifically, if $G = (V,T,P,S)$ is a phrase-structure grammar, then

$$\begin{aligned}\mathrm{cfree}(P) &= \{A \to x : A \to x \in P \text{ and } A \in V-T\},\\ \mathrm{cdep}(P) &= P - \mathrm{cfree}(P).\end{aligned}$$

If $G = (V,T,P,S)$ is a scattered context grammar, then

$$\begin{aligned}\mathrm{cfree}(P) &= \{(A) \to (x) : (A) \to (x) \in P\},\\ \mathrm{cdep}(P) &= P - \mathrm{cfree}(P).\end{aligned}$$

The main result of [41] follows next.

Theorem 21. *Let $H = (M,T,R,S)$ be a phrase-structure grammar in Kuroda normal form. Then, there exists a scattered context grammar, $G = (V,T,P,E)$, that satisfies*

1. $L(G) = L(H)$,
2. $|M| = |V| + 5$,
3. $|\mathrm{cdep}(P)| = |\mathrm{cdep}(R)| + 4$,
4. $|\mathrm{cfree}(P)| = |\mathrm{cfree}(R)| + 1$.

As we have seen from the number of articles written about scattered context grammars, this topic represents a vivid research area in formal language theory. In the following part, we strive to improve our knowledge about the way how scattered context grammars work by further examining their theoretical properties.

Part II

Results

Chapter 4

Conditional Removal of Erasing Productions

This section concentrates its investigation on the role of erasing productions and the way they are applied in scattered context grammars. While scattered context grammars with erasing productions characterize the family of recursively enumerable languages, the same grammars without erasing productions cannot generate any non-context-sensitive language (see Corollary 3 and Theorem 16). As a result, in general, we cannot convert any scattered context grammar with erasing productions to an equivalent scattered context grammar without these productions. In what follows, we demonstrate that this is always possible if the original grammar *erases nonterminals in a generalized k-limited way*, where $k \geq 0$, in a successful derivation; in every sentential form of the derivation, between any two neighboring symbols from which the grammar derives non-empty strings, there is a string of no more than k nonterminals from which the grammar derives empty strings later in the derivation. Consequently, the scattered context grammars that have erasing productions but apply them in a generalized k-limited way are equivalent to the grammars that do not have erasing productions at all.

In Theorem 12 it was demonstrated that the family of propagating scattered context languages is closed under k-limited erasing. Note that our definition of generalized k-limited erasing differs significantly from the way how symbols are erased by k-limited erasing. In the case of k-limited erasing a language can be generated by a propagating scattered context grammar if at most k symbols are deleted between every two neighboring terminals in a sentence. On the contrary, in the case of generalized k-limited erasing virtually unlimited number of symbols can be deleted between every two neighboring terminals in a sentence if during the derivation process between two neighboring non-erasable symbols there is a string of at most k erasable symbols. Therefore, the present result represents a generalization of the previously proved theorem.

Before we define generalized k-limited erasing formally, we give several auxiliary definitions.

Definition 24. The *core grammar underlying a scattered context grammar* $G = (V,T,P,S)$ is denoted by $\mathrm{core}(G)$ and defined as the context-free grammar $\mathrm{core}(G) = (V,T,P',S)$ with

$$P' = \{A_i \to x_i : (A_1,\ldots,A_i,\ldots,A_n) \to (x_1,\ldots,x_i,\ldots,x_n) \in P\}.$$

Definition 25. Let $G = (V,T,P,S)$ be a scattered context grammar and let $\mathrm{core}(G)$ be a core grammar underlying G. Let

$$\begin{aligned} & u_1A_1u_2A_2\ldots u_nA_nu_{n+1} = v \\ \Rightarrow_G\ & u_1x_1u_2x_2\ldots u_nx_nu_{n+1} = w\ [(A_1,\ldots,A_n) \to (x_1,\ldots,x_n)], \end{aligned}$$

where $u_i \in V^*$ for all $1 \leq i \leq n+1$. The *partial m-step context-free simulation* of this step by $\text{core}(G)$ is denoted by

$$\text{cf}_m(v \Rightarrow_G w)$$

and defined as $\text{core}(G)$'s m-step derivation of the form

$$\begin{aligned} & u_1A_1u_2A_2 \dots u_nA_nu_{n+1} \\ \Rightarrow_{\text{core}(G)} & u_1x_1u_2A_2 \dots u_nA_nu_{n+1} \\ \Rightarrow^{m-1}_{\text{core}(G)} & u_1x_1u_2x_2 \dots u_mx_mu_{m+1}A_{m+1} \dots u_nA_nu_{n+1}, \end{aligned}$$

where $m \leq n$. The *context-free simulation* of G's derivation step, denoted by

$$\text{cf}(v \Rightarrow_G w),$$

is the partial n-step context-free simulation of this step. Let $v = v_1 \Rightarrow^*_G v_n = w$ be of the form

$$v_1 \Rightarrow_G v_2 \Rightarrow_G v_3 \Rightarrow_G \dots \Rightarrow_G v_n.$$

The context-free simulation of $v \Rightarrow^*_G w$ by $\text{core}(G)$ is denoted by $\text{cf}(v \Rightarrow^*_G w)$ and defined as

$$v_1 \Rightarrow^*_{\text{core}(G)} v_2 \Rightarrow^*_{\text{core}(G)} v_3 \Rightarrow^*_{\text{core}(G)} \dots \Rightarrow^*_{\text{core}(G)} v_n$$

such that for all $1 \leq i \leq n-1$, $v_i \Rightarrow^*_{\text{core}(G)} v_{i+1}$ is the context-free simulation of $v_i \Rightarrow_G v_{i+1}$.

Definition 26. Let $G = (V,T,P,S)$ be a scattered context grammar and let $\text{core}(G)$ be a core grammar underlying G. Let $S \Rightarrow^*_G x$ be of the form $S \Rightarrow^*_G uAv \Rightarrow^*_G x$. Let $\text{cf}(S \Rightarrow^*_G x)$ be the context-free simulation of $S \Rightarrow^*_G x$. Let t be the derivation tree corresponding to $S \Rightarrow^*_{\text{core}(G)} x$ (regarding derivation trees and related notions, we use the terminology of [42]). Consider a subtree rooted at A in t. If the frontier of this subtree is ε, then G *erases* A in $S \Rightarrow^*_G uAv \Rightarrow^*_G x$, symbolically written as $\grave{A}$, and if this frontier differs from ε, then G *does not erase* A during this derivation, symbolically written as $\acute{A}$. If $w = \acute{A}_1 \dots \acute{A}_n$ or $w = \grave{A}_1 \dots \grave{A}_n$, for some $n \geq 1$, we write $\acute{w}$ or $\grave{w}$, respectively.

Generalized k-limited erasing is defined next.

Definition 27. Let $G = (V,T,P,S)$ be a scattered context grammar and let $k \geq 0$. *G erases nonterminals in a generalized k-limited way in $S \Rightarrow^*_G y$*, where $y \in L(G)$, if every sentential form x occurring in $S \Rightarrow^*_G y$ satisfies the following two properties:

1. Every $x = uAvBw$, $\acute{A}$, $\acute{B}$, $\grave{v}$, satisfies $|v| \leq k$;
2. Every $x = uAw$, $\acute{A}$, satisfies: if $\grave{u}$ or $\grave{w}$, then $|u| \leq k$ or $|w| \leq k$, respectively.

Set

$$\begin{aligned} L(G,\varepsilon,k) = \{x \in T^* : {} & S \Rightarrow^*_G x, \text{ and } G \text{ erases nonterminals} \\ & \text{in a generalized } k\text{-limited way in } S \Rightarrow^*_G x\}. \end{aligned}$$

A scattered context grammar G *erases its nonterminals in a generalized k-limited way* if $L(G) = L(G,\varepsilon,k)$. The family of languages generated by scattered context grammars which erase their nonterminals in a generalized k-limited way is denoted by $\mathscr{L}(SC,\varepsilon,k)$.

We illustrate these definitions on the following example.

Example 3. Let $G_1 = (V_1, T, P_1, S_1)$ and $G_2 = (V_2, T, P_2, S_2)$ be right-linear grammars which satisfy $(V_1 - T) \cap (V_2 - T) = \emptyset$. The following example demonstrates the use of generalized k-limited erasing to construct the grammar G satisfying

$$L(G) = \{ww : w \in L(G_1) \backslash L(G_2)\}.$$

Suppose that $S, X, Y \notin V_1 \cup V_2$. Define the homomorphism α from T^* to $\{\bar{a} : a \in T\}^*$ as $\alpha(a) = \bar{a}$ for all $a \in T$. We construct the grammar

$$G = (V_1 \cup V_2 \cup \{\bar{a} : a \in T\} \cup \{S, X, Y\}, T, P, S)$$

with P defined as follows:

1. Add $(S) \rightarrow (XS_1XS_2XS_1XS_2)$ to P;
2. For each $A \rightarrow xB \in P_1 \cup P_2$, add $(A, A) \rightarrow (\alpha(x)B, \alpha(x)B)$ to P;
3. For each $A \rightarrow x \in P_1 \cup P_2$, add $(A, A) \rightarrow (\alpha(x), \alpha(x))$ to P;
4. For each $a \in T$, add $(X, \bar{a}, X, \bar{a}, X, \bar{a}, X, \bar{a}) \rightarrow (\varepsilon, X, \varepsilon, X, \varepsilon, X, \varepsilon, X)$ to P;
5. Add $(X, X, X, X) \rightarrow (\varepsilon, Y, \varepsilon, Y)$ to P;
6. For each $a \in T$, add $(Y, \bar{a}, Y, \bar{a}) \rightarrow (a, Y, a, Y)$ to P;
7. Add $(Y, Y) \rightarrow (\varepsilon, \varepsilon)$ to P.

Observe that $L(G) = L$. Let $y \in T^*$ be the longest string such that $A \rightarrow yB \in P_1$ or $A \rightarrow y \in P_1$. Similarly, let $z \in T^*$ be the longest string such that $A \rightarrow zB \in P_2$ or $A \rightarrow z \in P_2$. Then, observe that G erases its nonterminals in a generalized $(|y| + |z| + 3)$-limited way for all $w \in L$, $w \neq \varepsilon$.

For instance, consider

$$G_1 = (\{S_1, a\}, \{a, b\}, \{S_1 \rightarrow aaS_1, S_1 \rightarrow \varepsilon\}, S_1)$$

with $L(G_1) = \{a^{2n} : n \geq 0\}$ and

$$G_2 = (\{S_2, a, b\}, \{a, b\}, \{S_2 \rightarrow aaaaS_2, S_2 \rightarrow b\}, S_2).$$

with $L(G_2) = \{a^{4n}b : n \geq 0\}$. We construct the grammar

$$G = (\{S_1, S_2, S, X, Y, \bar{a}, \bar{b}, a, b\}, \{a, b\}, P, S),$$

where

$$\begin{aligned}
P = \{1 :\ & (S) \rightarrow (XS_1XS_2XS_1XS_2),\\
& 2_a : (S_1, S_1) \rightarrow (\bar{a}\bar{a}S_1, \bar{a}\bar{a}S_1),\\
& 2_b : (S_2, S_2) \rightarrow (\bar{a}\bar{a}\bar{a}\bar{a}S_2, \bar{a}\bar{a}\bar{a}\bar{a}S_2),\\
& 3_a : (S_1, S_1) \rightarrow (\varepsilon, \varepsilon),\\
& 3_b : (S_2, S_2) \rightarrow (\bar{b}, \bar{b}),\\
& 4_a : (X, \bar{a}, X, \bar{a}, X, \bar{a}, X, \bar{a}) \rightarrow (\varepsilon, X, \varepsilon, X, \varepsilon, X, \varepsilon, X),\\
& 4_b : (X, \bar{b}, X, \bar{b}, X, \bar{b}, X, \bar{b}) \rightarrow (\varepsilon, X, \varepsilon, X, \varepsilon, X, \varepsilon, X),\\
& 5 :\ (X, X, X, X) \rightarrow (\varepsilon, Y, \varepsilon, Y),\\
& 6_a : (Y, \bar{a}, Y, \bar{a}) \rightarrow (a, Y, a, Y),\\
& 6_b : (Y, \bar{b}, Y, \bar{b}) \rightarrow (b, Y, b, Y),\\
& 7 :\ (Y, Y) \rightarrow (\varepsilon, \varepsilon)\}.
\end{aligned}$$

This grammar generates the language

$$L(G) = L = \{ww : w \in L(G_1) \backslash L(G_2)\} = \{a^{2n}ba^{2n}b : n \geq 0\}$$

and, in addition, erases its nonterminals in a generalized 9-limited way in every derivation, which can be easily seen on the following derivation:

$$\begin{aligned}
S &\Rightarrow_G XS_1XS_2XS_1XS_2\ [1] \Rightarrow_G X\bar{a}\bar{a}S_1XS_2X\bar{a}\bar{a}S_1XS_2\ [2_a] \\
&\Rightarrow_G X\bar{a}\bar{a}S_1X\bar{a}\bar{a}\bar{a}\bar{a}S_2X\bar{a}\bar{a}S_1X\bar{a}\bar{a}\bar{a}\bar{a}S_2\ [2_b] \\
&\Rightarrow_G X\bar{a}S_1X\bar{a}\bar{a}\bar{a}S_2X\bar{a}S_1X\bar{a}\bar{a}\bar{a}S_2\ [4_a] \Rightarrow_G XS_1X\bar{a}\bar{a}S_2XS_1X\bar{a}\bar{a}S_2\ [4_a] \\
&\Rightarrow_G XX\bar{a}\bar{a}S_2XX\bar{a}\bar{a}S_2\ [3_a] \Rightarrow_G XX\bar{a}\bar{a}\bar{b}XX\bar{a}\bar{a}\bar{b}\ [3_b] \Rightarrow_G Y\bar{a}\bar{a}\bar{b}Y\bar{a}\bar{a}\bar{b}\ [5] \\
&\Rightarrow_G aY\bar{a}\bar{b}aY\bar{a}\bar{b}\ [6_a] \Rightarrow_G aaY\bar{b}aaY\bar{b}\ [6_a] \Rightarrow_G aabYaabY\ [6_b] \\
&\Rightarrow_G aabaab\ [7].
\end{aligned}$$

Notice that S_2 is not deleted in a successful derivation of $w \in L$. As the derivation can always proceed so that there are at most four $\bar{a}$'s in front of S_2 and two $\bar{a}$'s in front of S_1, in the case when all $\bar{a}$'s, S_1 and both X's are deleted in front of S_2, $k = 9$ so the erasing is performed in a generalized 9-limited way.

The previous example illustrates that erasing productions are often used to verify some relations between individual parts of a sentential form. The introduction of these productions, however, does not have to mean that the generated language belongs to a more powerful family of languages. For instance, the existence of erasing productions in the grammar in Example 3 suggests that the language cannot be generated by any propagating scattered context grammar. However, the generated language, $\{a^{2n}ba^{2n}b : n \geq 0\}$, is clearly context-sensitive and can be easily generated by a propagating scattered context grammar.

The following theorem shows that if generalized k-limited erasing is used during a generation of a string, its derivation can be performed by a scattered context grammar which does not have erasing productions at all.

Theorem 22. *For each $k \geq 0$ and every scattered context grammar G, there is a propagating scattered context grammar $\bar{G}$ such that $L(G, \varepsilon, k) = L(\bar{G})$.*

Proof. Let $G = (V, T, P, S)$ be a scattered context grammar. For each

$$p = (A_1, \ldots, A_i, \ldots, A_n) \rightarrow (x_1, \ldots, x_i, \ldots, x_n) \in P,$$

let $\lfloor p, i \rfloor$ denote $A_i \rightarrow x_i$ for all $1 \leq i \leq n$. Let

$$\Psi = \{\lfloor p, i \rfloor : p \in P, 1 \leq i \leq \text{len}(p)\} \text{ and } \Psi' = \{\lfloor p, i \rfloor' : \lfloor p, i \rfloor \in \Psi\}.$$

Set

$$\bar{N}_1 = \{\langle x \rangle : x \in (V - T)^* \cup (V - T)^* T (V - T)^*, |x| \leq 2k + 1\}.$$

For each $\langle x \rangle \in \bar{N}_1$ and $\lfloor p, i \rfloor \in \Psi$, define

$$\text{lhs-replace}(\langle x \rangle, \lfloor p, i \rfloor) = \{\langle x_1 \lfloor p, i \rfloor x_2 \rangle : x_1 \,\text{lhs}(\lfloor p, i \rfloor) x_2 = x\}.$$

Set

$$\bar{N}_2 = \{\langle x \rangle : \langle x \rangle = \text{lhs-replace}(\langle y \rangle, \lfloor p, i \rfloor), \langle y \rangle \in \bar{N}_1, \lfloor p, i \rfloor \in \Psi\}.$$

For each $\langle x \rangle \in \bar{N}_1$ and $\lfloor p, i \rfloor' \in \Psi'$, define

$$\text{insert}(\langle x \rangle, \lfloor p, i \rfloor') = \{\langle x_1 \lfloor p, i \rfloor' x_2 \rangle : x_1 x_2 = x\}.$$

Set

$$\bar{N}_2' = \{\langle x\rangle : \langle x\rangle = \text{insert}(\langle y\rangle, \lfloor p,i\rfloor'), \langle y\rangle \in \bar{N}_1, \lfloor p,i\rfloor' \in \Psi'\}.$$

For each $x = \langle x_1\rangle \dots \langle x_n\rangle \in (\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_2')^*$ for some $n \geq 1$, define

$$\text{join}(x) = x_1 \dots x_n.$$

For each $x \in \bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_2'$, define

$$\text{split}(x) = \{y : x = \text{join}(y)\}.$$

Set $\bar{V} = T \cup \bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_2' \cup \{\bar{S}\}$. Define the propagating scattered context grammar

$$\bar{G} = (\bar{V}, T, \bar{P}, \bar{S})$$

with $\bar{P}$ constructed as follows:

1. For each $p = (S) \to (x) \in P$, add
   $(\bar{S}) \to (\langle\lfloor p,1\rfloor\rangle)$ to $\bar{P}$;
2. For each $\langle x\rangle \in \bar{N}_1$, each $X \in \text{insert}(\langle x\rangle, \lfloor p,n\rfloor')$, where $p \in P$, $\text{len}(p) = n$, each $\langle y\rangle \in \bar{N}_1$, and each $Y \in \text{lhs-replace}(\langle y\rangle, \lfloor q,1\rfloor)$, where $q \in P$, add
   (a) $(X, \langle y\rangle) \to (\langle x\rangle, Y)$, and
   (b) $(\langle y\rangle, X) \to (Y, \langle x\rangle)$ to $\bar{P}$;
   (c) if $\langle x\rangle = \langle y\rangle$, add
   $(X) \to (Y)$ to $\bar{P}$;
   (d) Add $(X) \to (\langle x\rangle)$ to $\bar{P}$;
3. For each $\langle x\rangle \in \bar{N}_1$, each $X \in \text{insert}(\langle x\rangle, \lfloor p,i\rfloor')$, where $p \in P$, $i < \text{len}(p)$, each $\langle y\rangle \in \bar{N}_1$, and each $Y \in \text{lhs-replace}(\langle y\rangle, \lfloor p,i+1\rfloor)$, where $q \in P$, add
   (a) $(X, \langle y\rangle) \to (\langle x\rangle, Y)$ to $\bar{P}$;
   (b) if $\langle x\rangle = \langle y\rangle$ and $X = x_1\lfloor p,i\rfloor' x_2$, $Y = y_1\lfloor p,i+1\rfloor y_2$ satisfy $|x_1\lfloor p,i\rfloor'| < |y_1\lfloor p,i+1\rfloor|$, add
   $(X) \to (Y)$ to $\bar{P}$;
4. For each
   $$\langle x_1\lfloor p,i\rfloor x_2\rangle \in \text{lhs-replace}(\langle x\rangle, \lfloor p,i\rfloor),$$
   $\langle x\rangle \in \bar{N}_1$, $\lfloor p,i\rfloor \in \Psi$, and each $Y \in \text{split}(x_1\,\text{rhs}(\lfloor p,i\rfloor)\lfloor p,i\rfloor' x_2)$, add
   $(\langle x_1\lfloor p,i\rfloor x_2\rangle) \to (Y)$ to $\bar{P}$;
5. For each $a \in T$, add
   $(\langle a\rangle) \to (a)$ to $\bar{P}$.

Denote the set of productions introduced in step i of the construction by $\bar{P}_i$, for $1 \leq i \leq 5$.

Let $S \Rightarrow_{\bar{G}}^* y \Rightarrow_{\bar{G}}^* w$, $w \in L(\bar{G})$, $y \in (\bar{N}_1 \cup \bar{N}_2 \cup \bar{N}_2')^*$, and let for each $\langle z\rangle \in \text{alph}(y)$ there exist

1. $A \in \text{alph}(z)$ such that $\acute{A}$ or
2. $\lfloor p,i\rfloor \in \text{alph}(z)$, $\lfloor p,i\rfloor \in \Psi$, $A = \text{lhs}(\lfloor p,i\rfloor)$ such that $\acute{A}$.

Then, we write $\check{y}$.

Basic Idea. The propagating scattered context grammar $\bar{G}$ simulates G by using nonterminals of the form $\langle \ldots \rangle$. In each nonterminal of this form, during every simulated derivation step, $\bar{G}$ records a substring corresponding to the current sentential form of G. The grammar $\bar{G}$ performs its derivation so that each of the nonterminals $\langle \ldots \rangle$ contains at least one symbol which is not erased later in the derivation.

The production constructed in (1) only initializes the simulation process. By productions introduced in (2) through (4), $\bar{G}$ simulates the application of a scattered context production $p \in P$ in a left-to-right way. In greater detail, by using a production of (2), $\bar{G}$ nondeterministically selects a scattered context production $p \in P$. Suppose that p consists of context-free productions $r_1, \ldots, r_{i-1}, r_i, \ldots, r_n$. By using productions of (3) and (4), $\bar{G}$ simulates the application of r_1 through r_n one by one. To explain this in greater detail, suppose that $\bar{G}$ has just completed the simulation of r_{i-1}. Then, to the right of this simulation, $\bar{G}$ selects $\mathrm{lhs}(r_i)$ by using a production of (3). That is, this selection is made inside of $\bar{G}$'s nonterminal in which the simulation of r_{i-1} has been performed or in one of the nonterminals appearing to the right of this nonterminal. After this selection, by using a production of (4), $\bar{G}$ performs the replacement of the selected symbol $\mathrm{lhs}(r_i)$ with $\mathrm{rhs}(r_i)$. If a terminal occurs inside of a nonterminal of $\bar{G}$, then a production of (5) allows $\bar{G}$ to change this nonterminal to the terminal contained in it.

Formal Proof.

Claim 1. Every successful derivation in $\bar{G}$ can be expressed in the following way:

$$\begin{aligned} \bar{S} &\Rightarrow_{\bar{G}} \langle \lfloor p, 1 \rfloor \rangle \; [p_1] \\ &\Rightarrow^{*}_{\bar{G}} u \qquad [\Phi] \\ &\Rightarrow^{+}_{\bar{G}} v \qquad [\Theta], \end{aligned}$$

where $p_1 = (\bar{S}) \to (\langle \lfloor p, 1 \rfloor \rangle) \in \bar{P}_1$, $u = \langle a_1 \rangle \ldots \langle a_n \rangle$, $n \geq 1$, $a_1, \ldots, a_n \in T$, $v = a_1 a_2 \ldots a_n$, Φ and Θ are sequences of productions from $(\bar{P}_2 \cup \bar{P}_3 \cup \bar{P}_4)$, and $\bar{P}_5$, respectively.

Proof. Productions from $\bar{P}_1$ are the only productions with $\bar{S}$ on their left-hand sides. Therefore, the derivation starts with a step made by one of these productions, so

$$\begin{aligned} \bar{S} &\Rightarrow_{\bar{G}} \langle \lfloor p, 1 \rfloor \rangle \; [p_1] \\ &\Rightarrow^{*}_{\bar{G}} v. \end{aligned}$$

Observe that $\bar{S}$ does not appear on the right-hand side of any of the introduced productions. Therefore, productions from $\bar{P}_1$ are not used during the rest of the derivation.

The productions from $\bar{P}_5$ are the only productions with their right-hand sides over T; therefore, they must be used to finish the derivation. Observe that none of the productions from $\bar{P}_1 \cup \bar{P}_2 \cup \bar{P}_3 \cup \bar{P}_4$ rewrites nonterminals of the form $\langle a \rangle$, where $a \in T$. As all productions from $\bar{P}_5$ are context-free, they can be applied whenever $\langle a \rangle$ appears in the sentential form. Without loss of generality we can assume that they are applied at the end of the derivation process; therefore,

$$\begin{aligned} \bar{S} &\Rightarrow_{\bar{G}} \langle \lfloor p, 1 \rfloor \rangle \; [p_1] \\ &\Rightarrow^{*}_{\bar{G}} u \qquad [\Phi] \\ &\Rightarrow^{+}_{\bar{G}} v \qquad [\Theta], \end{aligned}$$

so the claim holds. □

Claim 2. Let

$$w_1 \in \mathrm{split}(u_1 \lfloor p,1 \rfloor u_2 A_2 \dots u_n A_n u_{n+1}) \text{ and } \lambda = u_1 A_1 u_2 A_2 \dots u_n A_n u_{n+1},$$

$u_1,\dots,u_{n+1} \in V^*$, $A_1,\dots,A_n \in V-T$, $A_1 = \mathrm{lhs}(\lfloor p,1 \rfloor)$, $p = (A_1,\dots,A_n) \to (x_1,\dots,x_n) \in P$, and $\breve{w}_1$; then, every partial h-step context-free simulation

$$\mathrm{cf}_h(\lambda = u_1 A_1 u_2 A_2 \dots u_n A_n u_{n+1} \Rightarrow_G u_1 x_1 u_2 x_2 \dots u_n x_n u_{n+1}\ [p])$$

of the form

$$\begin{aligned}
& u_1 A_1 u_2 A_2 u_3 A_3 \dots u_n A_n u_{n+1} = \lambda \\
\Rightarrow_{\mathrm{core}(G)}\ & u_1 x_1 u_2 A_2 u_3 A_3 \dots u_n A_n u_{n+1} \\
\Rightarrow_{\mathrm{core}(G)}\ & u_1 x_1 u_2 x_2 u_3 A_3 \dots u_n A_n u_{n+1} \\
\Rightarrow^{h-2}_{\mathrm{core}(G)}\ & u_1 x_1 u_2 x_2 u_3 x_3 \dots u_h x_h u_{h+1} A_{h+1} \dots u_n A_n u_{n+1}
\end{aligned}$$

is performed in $\mathrm{core}(G)$ if and only if

$$\begin{aligned}
& w_1 \\
\Rightarrow_{\bar{G}}\ & w'_1\ [p^4_1] \\
\Rightarrow_{\bar{G}}\ & w_2\ [p^3_2] \\
\Rightarrow_{\bar{G}}\ & w'_2\ [p^4_2] \\
& \vdots \\
\Rightarrow^{2h-5}_{\bar{G}}\ & w_h\ [p^3_h] \\
\Rightarrow_{\bar{G}}\ & w'_h\ [p^4_h]
\end{aligned}$$

is performed in $\bar{G}$, where $p^3_2,\dots,p^3_h \in \bar{P}_3$, $p^4_1,\dots,p^4_h \in \bar{P}_4$, and

$$\begin{aligned}
& w'_1 \in \mathrm{split}(u_1 x_1 \lfloor p,1 \rfloor' u_2 A_2 \dots u_n A_n u_{n+1}), \\
& w_2 \in \mathrm{split}(u_1 x_1 u_2 \lfloor p,2 \rfloor u_3 \dots u_n A_n u_{n+1}), \\
& w'_2 \in \mathrm{split}(u_1 x_1 u_2 x_2 \lfloor p,2 \rfloor' u_3 \dots u_n A_n u_{n+1}), \\
& \quad \vdots \\
& w_h \in \mathrm{split}(u_1 x_1 u_2 x_2 \dots u_h \lfloor p,h \rfloor u_{h+1} A_{h+1} \dots u_n A_n u_{n+1}), \\
& w'_h \in \mathrm{split}(u_1 x_1 u_2 x_2 \dots u_h x_h \lfloor p,h \rfloor' u_{h+1} A_{h+1} \dots u_n A_n u_{n+1});
\end{aligned}$$

in addition, each $w \in \{w_2,\dots,w_h,w'_1,\dots,w'_h\}$ satisfies $\breve{w}$.

Proof.

Only If. The only-if part is proved by the induction on g for all partial g-step context-free simulations, for $g \geq 1$.

Basis. Let $g = 1$. Then,

$$u_1 A_1 u_2 A_2 \dots u_n A_n u_{n+1} \Rightarrow_{\mathrm{core}(G)} u_1 x_1 u_2 A_2 \dots u_n A_n u_{n+1},$$

and

$$w_1 \in \mathrm{split}(u_1 \lfloor p,1 \rfloor u_2 A_2 \dots u_n A_n u_{n+1}).$$

Notice that each production from $\bar{P}_2 \cup \bar{P}_3 \cup \bar{P}_4$ contains one symbol from

$$\mathrm{insert}(\langle x \rangle, \lfloor p,i \rfloor') \cup \mathrm{lhs\text{-}replace}(\langle x \rangle, \lfloor p,i \rfloor),$$

where $\langle x\rangle \in \bar{N}_1$, $\lfloor p,i\rfloor' \in \Psi'$, $\lfloor p,i\rfloor \in \Psi$, on its left- and right-hand side. Therefore, only one of these symbols occurs in every sentential form of $\bar{G}$. As

$$w_1 \in \mathrm{split}(u_1\lfloor p,1\rfloor u_2A_2\dots u_nA_nu_{n+1}),$$

only a production from $\bar{P}_4$ can be used in $\bar{G}$. Sentential form w_1 can be expressed as

$$w_1 \in \mathrm{split}(u_{11})\langle u_{12}\lfloor p,1\rfloor u_{21}\rangle\,\mathrm{split}(u_{22}),$$

where $u_{11}u_{12}=u_1$, $u_{21}u_{22}=u_2A_2\dots u_nA_nu_{n+1}$. Consider the following two forms of $\langle u_{12}\lfloor p,1\rfloor u_{21}\rangle$.

1. $$\langle E^1_1\dots E^1_{l^1}B^1\dots E^\alpha_1\dots \underline{E}^\alpha_m\dots E^\alpha_{l^\alpha}B^\alpha\dots E^\beta_1\dots E^\beta_{l^\beta}B^\beta E^{\beta+1}_1\dots E^{\beta+1}_{l^{\beta+1}}\rangle$$

   with $\underline{E}^\alpha_m=\lfloor p,1\rfloor$, $E^\alpha_m=\mathrm{lhs}(\lfloor p,1\rfloor)$, $\acute{B}^i$ for all $1\le i\le\beta$, $\grave{E}^i_1,\dots,\grave{E}^i_{l^i}$ for all $1\le i\le\beta+1$. If G erases nonterminals in a generalized k-limited way in the simulated derivation, then $\bar{G}$ satisfies

   $$|\mathrm{rhs}(\lfloor p,1\rfloor)|+l^\alpha-1\le k.$$

   Then, the corresponding simulating production $p^4_1\in\bar{P}_4$ is applicable and by its use, the nonterminal $\langle u_{12}\lfloor p,1\rfloor u_{21}\rangle$ can be rewritten and splitted, so we obtain $\langle y_1\rangle\dots\langle y_o\rangle$ for some $o\ge 1$, satisfying

   $$\begin{aligned} y_1\dots y_o = {} & E^1_1\dots E^1_{l^1}B^1\dots E^\alpha_1\dots\mathrm{rhs}(\lfloor p,1\rfloor)\lfloor p,1\rfloor'\dots\\ & \dots E^\alpha_{l^\alpha}B^\alpha\dots E^\beta_1\dots E^\beta_{l^\beta}B^\beta E^{\beta+1}_1\dots E^{\beta+1}_{l^{\beta+1}}, \end{aligned}$$

   and $B^i\in\mathrm{alph}(y_j)$ for all $1\le j\le o$ and some i, $1\le i\le\beta$. If G does not erase nonterminals in a generalized k-limited way in the simulated derivation step, no production in $\bar{G}$ can be applied and the derivation is blocked.

2. $$\langle E^1_1\dots E^1_{l^1}B^1\dots E^\alpha_1\dots E^\alpha_{l^\alpha}\underline{B}^\alpha\dots E^\beta_1\dots E^\beta_{l^\beta}B^\beta E^{\beta+1}_1\dots E^{\beta+1}_{l^{\beta+1}}\rangle$$

   with $\underline{B}^\alpha=\lfloor p,1\rfloor$, $B^\alpha=\mathrm{lhs}(\lfloor p,1\rfloor)$, $\acute{B}^i$ for all $1\le i\le\beta$ and $\grave{E}^i_1,\dots,\grave{E}^i_{l^i}$ for all $1\le i\le\beta+1$. Consider the simulated production denoted by $\lfloor p,1\rfloor$ of the form

   $$B^\alpha\to C^1_1\dots C^1_{p^1}D^1\dots C^\gamma_1\dots C^\gamma_{p^\gamma}D^\gamma C^{\gamma+1}_1\dots C^{\gamma+1}_{p^{\gamma+1}}$$

   with $\acute{D}^i$ for all $1\le i\le\gamma$, $\grave{C}^i_1,\dots,\grave{C}^i_{p^i}$ for all $1\le i\le\gamma+1$. If G erases nonterminals in a generalized k-limited way in the simulated derivation, the production satisfies $p^2,\dots,p^\gamma\le k$, and $p^1+l^\alpha\le k$, $p^{\gamma+1}+l^{\alpha+1}\le k$. Then, the corresponding simulating production $p^4_1\in\bar{P}_4$ is applicable, and by its use $\langle u_{12}\lfloor p,1\rfloor u_{21}\rangle$ can be rewritten and splitted, so we obtain $\langle y_1\rangle\dots\langle y_o\rangle$ for some $o\ge 1$, satisfying

   $$\begin{aligned} y_1\dots y_o = {} & E^1_1\dots E^1_{l^1}B^1\dots E^\alpha_1\dots E^\alpha_{l^\alpha}\\ & C^1_1\dots C^1_{p^1}D^1\dots C^\gamma_1\dots C^\gamma_{p^\gamma}D^\gamma C^{\gamma+1}_1\dots C^{\gamma+1}_{p^{\gamma+1}}\lfloor p,1\rfloor'\dots\\ & \dots E^\beta_1\dots E^\beta_{l^\beta}B^\beta E^{\beta+1}_1\dots E^{\beta+1}_{l^{\beta+1}}, \end{aligned}$$

   and $B^i\in\mathrm{alph}(y_j)$ or $D^q\in\mathrm{alph}(y_j)$ for all $1\le j\le o$ and some i,q, $1\le i\le\beta$, $1\le q\le\gamma$. If G does not erase nonterminals in a generalized k-limited way in the simulated derivation step, then no production in $\bar{G}$ can be applied and the derivation is blocked.

Therefore, some production $p_1^4 \in \bar{P}_4$ is applicable so we obtain

$$w_1 \Rightarrow_{\bar{G}} w_1' \ [p_1^4]$$

with

$$w_1' \in \mathrm{split}(u_{11})\,\mathrm{split}(u_{12}x_1\lfloor p,1\rfloor' u_{21})\,\mathrm{split}(u_{22}),$$

so $w_1' \in \mathrm{split}(u_{11}u_{12}x_1\lfloor p,1\rfloor' u_{21}u_{22})$, so

$$w_1' \in \mathrm{split}(u_1x_1\lfloor p,1\rfloor' u_2A_2\ldots u_nA_nu_{n+1}).$$

Induction Hypothesis. Suppose that the claim holds for all partial k-step context-free simulations, where $k \leq g$, for some $g \geq 1$.

Induction Step. Consider a partial $(g+1)$-step context-free simulation

$$x \Rightarrow^{g+1}_{\mathrm{core}(G)} z,$$

where $z \in V^*$. As $g+1 \geq 2$, there exists $y \in V^*$ such that

$$x \Rightarrow^{g}_{\mathrm{core}(G)} y \Rightarrow_{\mathrm{core}(G)} z\ [A_{g+1} \to x_{g+1}],$$

where $y = u_1x_1u_2x_2\ldots u_gx_gu_{g+1}A_{g+1}\ldots u_nA_nu_{n+1}$, and by the induction hypothesis

$$w_1 \Rightarrow^{2g-1}_{\bar{G}} w_g'.$$

We perform the partial $(g+1)$-step context-free simulation of a production p, so $g < \mathrm{len}(p)$ and

$$w_g' \in \mathrm{split}(u_1x_1u_2x_2\ldots u_gx_g\lfloor p,g\rfloor' u_{g+1}A_{g+1}\ldots u_nA_nu_{n+1});$$

therefore, only productions from $\bar{P}_3$ can be used. The sentential form w_g' has either of the following two forms:

1. $\mathrm{split}(r_1)\langle r_2\lfloor p,g\rfloor' r_3\rangle\,\mathrm{split}(r_4A_{g+1}r_5)$ such that $y = r_1r_2r_3r_4A_{g+1}r_5$. In this case, a production from step (3a) can be used and after its application, we obtain

$$w_{g+1} = \mathrm{split}(r_1)\langle r_2r_3\rangle\,\mathrm{split}(r_4\lfloor p,g+1\rfloor r_5).$$

2. $\mathrm{split}(r_1)\langle r_2\lfloor p,g\rfloor' u_{g+1}A_{g+1}r_3\rangle\,\mathrm{split}(r_4)$ such that $y = r_1r_2u_{g+1}A_{g+1}r_3r_4$. In this case, a production from step (3b) can be used and after its application, we obtain

$$w_{g+1} = \mathrm{split}(r_1)\langle r_2u_{g+1}\lfloor p,g+1\rfloor r_3\rangle\,\mathrm{split}(r_4).$$

Thus, this derivation step can be expressed as $w_g' \Rightarrow_{\bar{G}} w_{g+1}\ [p_{g+1}^3]$, where $p_{g+1}^3 \in \bar{P}_3$ and

$$w_{g+1} \in \mathrm{split}(u_1x_1u_2x_2\ldots u_gx_gu_{g+1}\lfloor p,g+1\rfloor\ldots u_nA_nu_{n+1}).$$

If $\check{w}_g'$, then $\check{w}_{g+1}$ as well. At this point, only productions from $p_{g+1}^4 \in \bar{P}_4$ can be used. The proof of

$$w_{g+1} \Rightarrow_{\bar{G}} w_{g+1}'\ [p_{g+1}^4]$$

is analogous to the proof of

$$w_1 \Rightarrow_{\bar{G}} w_1' \ [p_1^4]$$

described in the Basis and is left to the reader. As a result,

$$w_{g+1}' \in \text{split}(u_1x_1u_2x_2 \ldots u_{g+1}x_{g+1} \lfloor p, g+1 \rfloor' u_{g+2}A_{g+2} \ldots u_nA_nu_{n+1})$$

satisfying $\breve{w}_{g+1}'$.

If. The if part is proved by the induction on g for all g-step derivations in $\bar{G}$, for $g \geq 1$.

Basis. Let $g = 1$. Then, $w_1 \Rightarrow_{\bar{G}} w_1' \ [p_1^4]$ and

$$\lambda = u_1A_1u_2A_2 \ldots u_nA_nu_{n+1}.$$

Clearly,

$$u_1A_1u_2A_2 \ldots u_nA_nu_{n+1} \Rightarrow_{\text{core}(G)} u_1x_1u_2A_2 \ldots u_nA_nu_{n+1} \ [A_1 \rightarrow x_1].$$

Induction Hypothesis. Suppose that the claim holds for all k-step derivations, where $k \leq g$, for some $g \geq 1$.

Induction Step. Consider a derivation

$$w_1 \Rightarrow_{\bar{G}}^{2(g+1)-1} w_{g+1}',$$

where

$$w_{g+1}' \in \text{split}(u_1x_1u_2x_2 \ldots u_{g+1}x_{g+1} \lfloor p, g+1 \rfloor' u_{g+2}A_{g+2} \ldots u_nA_nu_{n+1}).$$

Since $2(g+1)-1 \geq 3$, there exists a derivation

$$w_1 \Rightarrow_{\bar{G}}^{2g-1} w_g' \Rightarrow_{\bar{G}} w_{g+1} \Rightarrow_{\bar{G}} w_{g+1}',$$

where

$$w_{g+1} \in \text{split}(u_1x_1u_2x_2 \ldots u_gx_gu_{g+1} \lfloor p, g+1 \rfloor u_{g+2}A_{g+2} \ldots u_nA_nu_{n+1}).$$

By the induction hypothesis, there is a derivation

$$u_1A_1 \ldots u_nA_nu_{n+1} \Rightarrow_{\text{core}(G)}^{g} u_1x_1 \ldots u_gx_gu_{g+1}A_{g+1} \ldots u_nA_nu_{n+1}.$$

Clearly,

$$\begin{aligned} &u_1x_1 \ldots u_gx_gu_{g+1}A_{g+1}u_{g+2}A_{g+2} \ldots u_nA_nu_{n+1} \\ &\Rightarrow_{\text{core}(G)} u_1x_1 \ldots u_gx_gu_{g+1}x_{g+1}u_{g+2}A_{g+2} \ldots u_nA_nu_{n+1} \ [A_{g+1} \rightarrow x_{g+1}]. \end{aligned}$$

□

Claim 3. The result from Claim 2 holds for a context-free simulation.

Proof. As the context-free simulation of a scattered context production $(A_1, \ldots, A_n) \rightarrow (x_1, \ldots, x_n)$ is a partial context-free simulation of the length n, Claim 2 holds for a context-free simulation as well. □

Let

$$\begin{aligned} & u_1A_1u_2A_2\dots u_nA_nu_{n+1} \\ \Rightarrow_G\ & u_1x_1u_2x_2\dots u_nx_nu_{n+1} \quad [p=(A_1,\dots,A_n)\to(x_1,\dots,x_n)]) \end{aligned}$$

for some $p \in P$. Then, $\Rrightarrow$ denotes the simulation of this derivation step in $\bar{G}$ as shown in Claim 2 and 3. We write

$$w_1 \Rrightarrow w'_n\ [p],$$

or, shortly, $w_1 \Rrightarrow w'_n$. Therefore, $w_1 \Rightarrow_{\bar{G}}^{2n-1} w'_n$ from Claim 2 is equivalent to $w_1 \Rrightarrow w'_n$.

Claim 4. Let $x_1 \in V^*$ and $\bar{x}'_1 \in \mathrm{split}(x'_{11}\lfloor p_1,1\rfloor x'_{12})$, where $x'_{11}\,\mathrm{lhs}(\lfloor p_1,1\rfloor)x'_{12} = x_1$, $\lfloor p,1\rfloor \in \Psi$, and $\breve{\bar{x}}'_1$; then, every derivation

$$\begin{aligned} & x_1 \\ \Rightarrow_G\ & x_2 \quad [p_1] \\ & \vdots \\ \Rightarrow_G\ & x_{m+1}\ [p_m] \end{aligned}$$

is performed in G if and only if

$$\begin{aligned} & \bar{x}'_1 \\ \Rrightarrow\ & \bar{x}_1 \quad [p_1] \\ \Rightarrow_{\bar{G}}\ & \bar{x}'_2 \quad [p'_2] \\ \Rrightarrow\ & \bar{x}_2 \quad [p_2] \\ \Rightarrow_{\bar{G}}\ & \bar{x}'_3 \quad [p'_3] \\ & \vdots \\ \Rrightarrow\ & \bar{x}_m \quad [p_m] \\ \Rightarrow_{\bar{G}}\ & \bar{x}'_{m+1}\ [p'_{m+1}] \end{aligned}$$

is performed in $\bar{G}$, where $x_2,\dots,x_{m+1} \in V^*$, $p_1,\dots,p_m \in P$, $p'_2,\dots,p'_{m+1} \in \bar{P}_2$,

$$\begin{aligned} & \bar{x}_i \in \mathrm{split}(x_{i1}\lfloor p_i,\mathrm{len}(p_i)\rfloor' x_{i2}), \\ & \bar{x}'_j \in \mathrm{split}(x'_{j1}\lfloor p_j,1\rfloor x'_{j2}), \end{aligned}$$

for all $1 \le i \le m$, $2 \le j \le m$, and

$$\bar{x}'_{m+1} \in \mathrm{split}(x'_{(m+1)1}\lfloor p_{m+1},1\rfloor x'_{(m+1)2}) \text{ for } x_{m+1} \notin T^*,$$

or

$$\bar{x}'_{m+1} \in \mathrm{split}(x_{m+1}) \text{ for } x_{m+1} \in T^*,$$

where $x_{i1}x_{i2} = x_i$ for all $1 \le i \le m$, $x'_{j1}\,\mathrm{lhs}(\lfloor p_j,1\rfloor)x'_{j2} = x_j$ for all $2 \le j \le m+1$, and each $\bar{x} \in \{\bar{x}_1,\dots,\bar{x}_m,\bar{x}'_2,\dots,\bar{x}'_{m+1}\}$ satisfies $\breve{\bar{x}}$.

Proof.

Only If. The only-if part is proved by the induction on g for all g-step derivations in G, for $g \ge 0$.

Basis. Let $g = 0$. Then, $x_1 \Rightarrow_G^0 x_1$ and $\bar{x}'_1 \Rightarrow_{\bar{G}}^0 \bar{x}'_1$.

Induction Hypothesis. Suppose that the claim holds for all k-step derivations, where $k \le g$, for some $g \ge 0$.

Induction Step. Consider a derivation

$$x_1 \Rightarrow_G^{g+1} x_{g+2}.$$

Since $g+1 \geq 1$, there exists a derivation

$$x_1 \Rightarrow_G^g x_{g+1} \Rightarrow_G x_{g+2}.$$

By the induction hypothesis, there is a derivation

$$\bar{x}'_1 \Rrightarrow \bar{x}_1 \Rightarrow_{\bar{G}} \bar{x}'_2 \Rrightarrow \bar{x}_2 \Rightarrow_{\bar{G}} \ldots \Rrightarrow \bar{x}_g \Rightarrow_{\bar{G}} \bar{x}'_{g+1}.$$

Because we are performing $(g+1)$-step derivation, $x_{g+1} \notin T^*$. The sentential forms $\bar{x}_g$ and $\bar{x}'_{g+1}$ in the g-step simulation have to satisfy

$$\bar{x}_g \in \mathrm{split}(x_{g1}\lfloor p_g, \mathrm{len}(p_g)\rfloor' x_{g2})$$

and

$$\bar{x}'_{g+1} \in \mathrm{split}(x'_{(g+1)1}\lfloor p_{g+1}, 1\rfloor x'_{(g+1)2}),$$

where $x_g = x_{g1}x_{g2}$,

$$x_{g+1} = x'_{(g+1)1}\,\mathrm{lhs}(\lfloor p_{g+1}, 1\rfloor)x'_{(g+1)2},$$

and $\breve{\bar{x}}'_{g+1}$. Then, by Claim 3,

$$\bar{x}'_{g+1} \Rrightarrow \bar{x}_{g+1},$$

where

$$\bar{x}_{g+1} \in \mathrm{split}(x_{(g+1)1}\lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' x_{(g+1)2}),$$

$x_{(g+1)1}x_{(g+1)2} = x_{g+1}$, and $\breve{\bar{x}}_{g+1}$.

Consider the case when $\breve{\bar{x}}_{g+1}$ is not satisfied, so there is some $\langle z\rangle \in \mathrm{alph}(\bar{x}_{g+1})$ such that $\grave{z}$. By an inspection of $\bar{P}_4$ and Claim 2, observe that $\langle z\rangle$ is rewritten by $\bar{G}$'s simulation to $\langle\varepsilon\rangle$ (or $\langle\varepsilon\rangle\ldots\langle\varepsilon\rangle$ when splitted). There is, however, no production rewriting $\langle\varepsilon\rangle$ in $\bar{G}$, so the derivation is blocked. For this reason, every sentential form $\bar{x}$ in

$$\bar{x}'_1 \Rrightarrow \bar{x}_1 \Rightarrow_{\bar{G}} \bar{x}'_2 \Rrightarrow \bar{x}_2 \Rightarrow_{\bar{G}} \ldots \Rrightarrow \bar{x}_g \Rightarrow_{\bar{G}} \bar{x}'_{g+1}$$

has to satisfy $\breve{\bar{x}}$.

As

$$\bar{x}_{g+1} \in \mathrm{split}(x_{(g+1)1}\lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' x_{(g+1)2}),$$

$\lfloor p, i\rfloor' = \lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' \in \Psi'$ and $i = \mathrm{len}(p)$, only productions from $\bar{P}_2$ can be used. Let

$$x_{g+2} = u_{(g+2)1}A_{(g+2)1}\ldots u_{(g+2)j}A_{(g+2)j}u_{(g+2)(j+1)},$$

where $u_{(g+2)1},\ldots,u_{(g+2)(j+1)} \in V^*$, $A_{(g+2)1},\ldots,A_{(g+2)j} \in V - T^*$, for some $j \geq 1$. We distinguish the following three cases of $\bar{x}_{g+1}$:

1. $\mathrm{split}(r_1)\langle r_2\lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' r_3\rangle\,\mathrm{split}(r_4A_{(g+2)1}r_5)$ such that

   $$x_{g+2} = r_1r_2r_3r_4A_{(g+2)1}r_5.$$

   Then, a production constructed in (2a) can be used, and by its application we obtain a sentential form

   $$\bar{x}'_{g+2} = \mathrm{split}(r_1)\langle r_2r_3\rangle\,\mathrm{split}(r_4\lfloor p_{g+2}, 1\rfloor r_5).$$

2. $\mathrm{split}(r_1 A_{(g+2)1} r_2)\langle r_3 \lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' r_4\rangle\,\mathrm{split}(r_5)$ such that

$$x_{g+2} = r_1 A_{(g+2)1} r_2 r_3 r_4 r_5.$$

Then, a production constructed in (2b) can be used, and by its application we obtain a sentential form

$$\bar{x}'_{g+2} = \mathrm{split}(r_1 \lfloor p_{g+2}, 1\rfloor r_2)\langle r_3 r_4\rangle\,\mathrm{split}(r_5).$$

3. (a) $\mathrm{split}(r_1)\langle r_2 \lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' r_3 A_{(g+2)1} r_4\rangle\,\mathrm{split}(r_5)$ such that

$$x_{g+2} = r_1 r_2 r_3 A_{(g+2)1} r_4 r_5.$$

Then, a production constructed in (2c) can be used, and by its application we obtain a sentential form

$$\bar{x}'_{g+2} = \mathrm{split}(r_1)\langle r_2 r_3 \lfloor p_{g+2}, 1\rfloor r_4\rangle\,\mathrm{split}(r_5).$$

(b) $\mathrm{split}(r_1)\langle r_2 A_{(g+2)1} r_3 \lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' r_4\rangle\,\mathrm{split}(r_5)$ such that

$$x_{g+2} = r_1 r_2 A_{(g+2)1} r_3 r_4 r_5.$$

Then, a production constructed in (2c) can be used, and by its application we obtain a sentential form

$$\bar{x}'_{g+2} = \mathrm{split}(r_1)\langle r_2 \lfloor p_{g+2}, 1\rfloor r_3 r_4\rangle\,\mathrm{split}(r_5).$$

In the case of $x_{g+2} \in T^*$, $\bar{x}_{g+1}$ has the form

$$\mathrm{split}(r_1)\langle r_2 \lfloor p_{g+1}, \mathrm{len}(p_{g+1})\rfloor' r_3\rangle\,\mathrm{split}(r_4),$$

where $x_{g+2} = r_1 r_2 r_3 r_4 \in T^*$. Then, a production constructed in (2d) can be used, and by its application we obtain a sentential form

$$\bar{x}'_{g+2} = \mathrm{split}(r_1)\langle r_2 r_3\rangle\,\mathrm{split}(r_4).$$

Notice that a production from step (2d) is applicable also in the case of $x_{g+2} \notin T^*$ (cases 1 through 3). This production removes the symbol from Ψ' from the sentential form. However, as all productions from steps (2) through (4) require a symbol from Ψ or Ψ', its application for $x_{g+2} \notin T^*$ does not lead to a successful derivation.

As a result, there is a derivation

$$\bar{x}_{g+1} \Rightarrow_{\bar{G}} \bar{x}'_{g+2}\,[p'_{g+1}],$$

where

$$\bar{x}'_{g+2} \in \mathrm{split}(x'_{(g+2)1} \lfloor p_{g+2}, 1\rfloor x'_{(g+2)2}) \text{ for } x_{g+2} \notin T^*$$

or

$$\bar{x}'_{g+2} \in \mathrm{split}(x_{g+2}) \text{ for } x_{g+2} \in T^*$$

with $x'_{(g+2)1}\,\mathrm{lhs}(\lfloor p_{g+2}, 1\rfloor)x'_{(g+2)2} = x_{g+2}$, satisfying $\breve{x}'_{g+2}$.

If. The if part is proved by the induction on the number g of $\bar{G}$'s simulation steps in a derivation, for $g \geq 0$.

Basis. Let $g = 0$. Then, $\bar{x}'_1 \Rightarrow^0_{\bar{G}} \bar{x}'_1$ and $x_1 \Rightarrow^0_G x_1$.

Induction Hypothesis. Suppose that the claim holds for all $\bar{G}$'s derivations containing k simulation steps, where $k \leq g$, for some $g \geq 0$.

Induction Step. Consider a derivation

$$\vec{x}'_1 \Rrightarrow \bar{x}_1 \Rightarrow_{\bar{G}} \ldots \Rrightarrow \bar{x}_{g+1} \Rightarrow_{\bar{G}} \vec{x}'_{g+2}.$$

Since $g+1 \geq 1$, there exists a derivation

$$\vec{x}'_1 \Rrightarrow \bar{x}_1 \Rightarrow_{\bar{G}} \ldots \Rrightarrow \bar{x}_g \Rightarrow_{\bar{G}} \vec{x}'_{g+1} \Rrightarrow \bar{x}_{g+1} \Rightarrow_{\bar{G}} \vec{x}'_{g+2}.$$

By the induction hypothesis, there is a derivation

$$x_1 \Rightarrow_G^g x_{g+1}.$$

As

$$\vec{x}'_{g+1} \in \mathrm{split}(x'_{(g+1)1} \lfloor p_{g+1}, 1 \rfloor x'_{(g+1)2}),$$

$x'_{(g+1)1}\,\mathrm{lhs}(\lfloor p_{g+1}, 1 \rfloor)x'_{(g+1)2} = x_{g+1}$ and

$$\vec{x}'_{g+1} \Rrightarrow \bar{x}_{g+1} \Rightarrow_{\bar{G}} \vec{x}'_{g+2},$$

then, by Claim 3, there is also a derivation

$$x_{g+1} \Rightarrow_G x_{g+2}$$

such that

$$\vec{x}'_{g+2} \in \mathrm{split}(x'_{(g+2)1} \lfloor p_{g+2}, 1 \rfloor x'_{(g+2)2}),$$

$x'_{(g+2)1}\,\mathrm{lhs}(\lfloor p_{g+2}, 1 \rfloor)x'_{(g+2)2} = x_{g+2}$, or $\vec{x}'_{g+2} \in \mathrm{split}(x_{g+2})$.

□

From Claim 1,

$$\bar{S} \Rightarrow_{\bar{G}} \langle \lfloor p, 1 \rfloor \rangle.$$

As $\langle \lfloor p, 1 \rfloor \rangle \in \mathrm{split}(\lfloor p, 1 \rfloor)$, $S = \mathrm{lhs}(\lfloor p, 1 \rfloor)$, G's simulation as described in Claim 4 can be performed, so

$$\langle \lfloor p, 1 \rfloor \rangle \Rightarrow_{\bar{G}}^* u\ [\Phi],$$

where Φ is a sequence of productions from $\bar{P}_2 \cup \bar{P}_3 \cup \bar{P}_4$. If a successful derivation is simulated, then we obtain $u = \langle a_1 \rangle \langle a_2 \rangle \ldots \langle a_n \rangle$, where $n \geq 1$ and $a_1, a_2, \ldots, a_n \in T$. Finally, by the application of productions from $\bar{P}_5$, we obtain

$$u \Rightarrow_{\bar{G}}^+ v,$$

where $v = a_1 a_2 \ldots a_n$. Therefore, every string during whose generation G erases nonterminals in a generalized k-limited way can be generated by a propagating scattered context grammar $\bar{G}$. ■

Corollary 8. *For every scattered context grammar G which erases its nonterminals in a generalized k-limited way, there exists a propagating scattered context grammar $\bar{G}$ such that $L(G) = L(\bar{G})$.* ■

Corollary 9. *$\mathscr{L}(PSC) = \mathscr{L}(SC, \varepsilon, i)$ for any positive integer i.* ■

We have proved that every scattered context grammar that erases its nonterminals in a generalized k-limited way can be converted to an equivalent scattered context grammar without erasing productions. Erasing productions are often used in grammars because their presence reduces the total number of productions and makes their derivations more transparent as demonstrated in Example 3. In certain situations, however, these erasing productions are not necessary and can be removed from the grammar.

The result rises several open problems. First, we may ask if a stronger variant of Corollary 8 can be established: for a scattered context grammar G, there is a propagating scattered context grammar $\bar{G}$ such that $L(G) = L(\bar{G})$ if and only if G erases its nonterminals in a generalized k-limited way.

Open Problem 3. Does generalized k-limited erasing cover all possible types of erasing which can be performed by scattered context grammars without erasing productions?

Second, the result induces the following decidability question:

Open Problem 4. We have given an integer k and a scattered context grammar G. Is it decidable whether G erases its nonterminals in a generalized k-limited way?

We propose these open problems for further investigation.

Chapter 5

Restrictions and Extensions

This chapter introduces several restricted and extended versions of scattered context grammars. We present two types of these modifications—modifications of the definition of a derivation step and modifications of the whole concept of a scattered context grammar. The first and the second type of modifications is represented by the results discussed in Sections 5.2, 5.3, 5.4, and Section 5.1, respectively. Most importantly, we investigate the generative power of scattered context grammars modified in this way.

5.1 Non-Context-Free Components of Scattered Context Grammars

Scattered context grammars were introduced as a generalization of context-free grammars. Indeed, each scattered context production consists of k context-free components which are applied in parallel on the current sentential form. It is only natural to generalize other grammars from the Chomsky hierarchy in a similar manner as well. The main aim of this section is to study the generative power of scattered context grammars whose components are linear and right-linear. To be able to use scattered context productions, we permit the context-free starting productions to generate the initial strings containing several nonterminals and do not use these productions during the rest of the derivation. It turns out that scattered context grammars with an initial string which contains n nonterminals and productions with linear or right-linear components are as powerful as linear simple matrix grammars of degree n or right-linear simple matrix grammars of degree n, respectively. Finally, we mention several corollaries of this result and discuss the generative power of scattered context grammars with context-sensitive and unrestricted productions.

We start by defining linear and right-linear scattered context grammars formally.

Definition 28. A *linear scattered context grammar* is a scattered context grammar $G = (V, T, P, S)$, where P is a finite set of productions of the following two forms:

1. $(S) \to (x_1 A_1 \dots x_k A_k x_{k+1})$, where $A_i \in (V-T)-\{S\}$, $x_j \in T^*$ for all $1 \leq i \leq k$, $1 \leq j \leq k+1$,
2. $(A_1, \dots, A_k) \to (z_1, \dots, z_k)$, where $A_i \in (V-T)-\{S\}$, and either $z_i = x_i B_i y_i$, where $x_i, y_i \in T^*$, $B_i \in (V-T)-\{S\}$, or $z_i \in T^*$ for all $1 \leq i \leq k$, for some $k \geq 1$.

A linear scattered context grammar is *of degree n* if $(S) \to (x_1 A_1 \dots x_n A_n x_{n+1}) \in P$ is the production satisfying $n \geq m$ for all $(S) \to (y_1 A_1 \dots y_m A_m y_{m+1}) \in P$. The family of languages generated by linear scattered context grammars of degree n is denoted by $\mathscr{L}(SC, LIN, n)$, and

$$\mathscr{L}(SC, LIN) = \bigcup_{n=1}^{\infty} \mathscr{L}(SC, LIN, n).$$

Definition 29. A *right-linear scattered context grammar* is a linear scattered context grammar $G = (V, T, P, S)$, where P is a finite set of productions of the following two forms:

1. $(S) \to (x_1 A_1 \dots x_k A_k)$, where $A_i \in (V - T) - \{S\}$, $x_i \in T^*$ for all $1 \leq i \leq k$, for some $k \geq 1$,
2. $(A_1, \dots, A_k) \to (z_1, \dots, z_k)$, where $A_i \in (V - T) - \{S\}$, and either $z_i = x_i B_i$, where $x_i \in T^*$, $B_i \in (V - T) - \{S\}$, or $z_i \in T^*$ for all $1 \leq i \leq k$, for some $k \geq 1$.

The family of languages generated by right-linear scattered context grammars of degree n is denoted by $\mathscr{L}(SC, RLIN, n)$, and

$$\mathscr{L}(SC, RLIN) = \bigcup_{n=1}^{\infty} \mathscr{L}(SC, RLIN, n).$$

To prove that $\mathscr{L}(SM, LIN, n) = \mathscr{L}(SC, LIN, n)$ for each $n \geq 1$, we first give two preliminary lemmas.

Lemma 4. *For each $n \geq 1$, $\mathscr{L}(SM, LIN, n) \subseteq \mathscr{L}(SC, LIN, n)$.*

Proof. Let $\bar{G} = (\bar{V}_1, \dots, \bar{V}_n, T, \bar{P}, \bar{S})$ be a linear simple matrix grammar of degree n. Set $N = \{\langle p, i \rangle : p \in \bar{P}, 1 \leq i \leq n\}$. Define the linear scattered context grammar of degree n,

$$G = (\bar{V}_1 \cup \dots \cup \bar{V}_n \cup N \cup \{S\}, T, P, S),$$

where P is defined as follows:

1. For each $(\bar{S}) \to (x_1 A_1 \dots x_n A_n x_{n+1}) \in \bar{P}$, where $x_i \in T^*$, for all $1 \leq i \leq n+1$, and $p = (A_1, \dots, A_n) \to (y_1, \dots, y_n) \in \bar{P}$, add
   $(S) \to (x_1 \langle p, 1 \rangle x_2 A_2 \dots x_n A_n x_{n+1})$ to P;
2. For each $p = (A_1, \dots, A_i, \dots, A_n) \to (x_1 B_1 y_1, \dots, x_i B_i y_i, \dots, x_n B_n y_n) \in \bar{P}$, where $x_j, y_j \in T^*$, $A_j, B_j \in \bar{V}_j - T$ for all $1 \leq j \leq n$,
   (a) for each $i < n$, add
   $(\langle p, i \rangle, A_{i+1}) \to (x_i B_i y_i, \langle p, i+1 \rangle)$ to P;
   (b) for each $q = (B_1, \dots, B_n) \to (z_1, \dots, z_n) \in \bar{P}$, add
      i. $(B_1, \langle p, n \rangle) \to (\langle q, 1 \rangle, x_n B_n y_n)$ to P;
      ii. for $n = 1$, add
      $(\langle p, 1 \rangle) \to (x_1 \langle q, 1 \rangle y_1)$ to P;
3. For each $p = (A_1, \dots, A_i, \dots, A_n) \to (x_1, \dots, x_i, \dots, x_n) \in \bar{P}$, where $x_j \in T^*$ for all $1 \leq j \leq n$,
   (a) for each $i < n$, add
   $(\langle p, i \rangle, A_{i+1}) \to (x_i, \langle p, i+1 \rangle)$ to P;
   (b) add $(\langle p, n \rangle) \to (x_n)$ to P.

Each production introduced in step (1) simulates the initial production of $\bar{G}$ and, in addition, selects the next production, p, to be simulated. After its application, we obtain the sentential form of the form

$$w_1 \langle p, 1 \rangle w_2 A_2 \dots w_n A_n w_{n+1},$$

where $w_i \in T^*$ for all $1 \leq i \leq n$, and $p = (A_1, \dots, A_n) \to (z_1, \dots, z_n) \in \bar{P}$. Consider any derivation

$$w_1 A_1 w_2 A_2 \dots w_n A_n w_{n+1} \Rightarrow_{\bar{G}} w_1 x_1 B_1 y_1 \dots w_n x_n B_n y_n w_{n+1} \; [p],$$

where

$$p = (A_1, \ldots, A_n) \to (x_1 B_1 y_1, \ldots, x_n B_n y_n),$$

$x_i, y_i \in T^*$, $A_i, B_i \in \bar{V}_i - T$ for all $1 \leq i \leq n$. This derivation is simulated by G in n derivation steps by first applying a production from (2a) $n-1$ times and, finally, applying a production from (2bi), so

$$\begin{aligned} & w_1 \langle p, 1 \rangle w_2 A_2 \ldots w_n A_n w_{n+1} \\ \Rightarrow_G\ & w_1 x_1 B_1 y_1 w_2 \langle p, 2 \rangle \ldots w_n A_n w_{n+1} \\ & \vdots \\ \Rightarrow_G\ & w_1 x_1 B_1 y_1 w_2 x_2 B_2 y_2 \ldots w_n \langle p, n \rangle w_{n+1} \\ \Rightarrow_G\ & w_1 x_1 \langle q, 1 \rangle y_1 w_2 x_2 B_2 y_2 \ldots w_n x_n B_n y_n w_{n+1}, \end{aligned}$$

where $q = (B_1, \ldots, B_n) \to (z_1, \ldots, z_n) \in \bar{P}$. Observe that no nonterminal A_i can be skipped by a production from (2a) because the sentential form contains exactly n nonterminals and the form of the productions from (2a) requires their n applications during each simulation. For the same reason, a production from (2bi) has to select the first nonterminal in G's sentential form. If $n = 1$, a production from (2bii) is used instead of (2a) and (2bi). Finally, a production of the form $(A_1, \ldots, A_n) \to (x_1, \ldots, x_n) \in \bar{P}$, where $x_i \in T^*$ for all $1 \leq i \leq n$, is simulated by productions from (3a) and (3b) which perform the simulation analogously to the productions from (2a) and (2bi), respectively. By removing the symbol from N from the sentential form, (3b) finishes the derivation. ■

As the number of components in every production of G constructed in the proof of Lemma 4 is at most 2, we state the following corollary.

Corollary 10. *For every linear simple matrix grammar $\bar{G}$ of degree n, there is a linear scattered context grammar G of degree n such that $L(\bar{G}) = L(G)$ and* $\mathrm{mcs}(G) = 1$. ■

Lemma 5. *For each $n \geq 1$, $\mathscr{L}(SC, LIN, n) \subseteq \mathscr{L}(SM, LIN, n)$.*

Proof. Let $\bar{G} = (\bar{V}, T, \bar{P}, \bar{S})$ be a linear scattered context grammar of degree n. Set

$$\begin{aligned} V_1 &= \{\langle a, 1 \rangle : a \in (\bar{V} - \{\bar{S}\}) \cup \{\varepsilon\}\} \cup T, \\ &\vdots \\ V_n &= \{\langle a, n \rangle : a \in (\bar{V} - \{\bar{S}\}) \cup \{\varepsilon\}\} \cup T. \end{aligned}$$

For $A \in (\bar{V} - T) - \{\bar{S}\}$, set $\alpha(xAy, i) = x\langle A, i \rangle y$, where $x, y \in T^*$, for $a \in T$, set $\alpha(xay, i) = x\langle a, i \rangle y$, and $\alpha(\varepsilon, i) = \langle \varepsilon, i \rangle$, for all $1 \leq i \leq n$. Define the linear simple matrix grammar of degree n,

$$G = (V_1, \ldots, V_n, T, P, S),$$

where P is defined as follows:

1. For each $(\bar{S}) \to (x_1 A_1 \ldots x_k A_k x_{k+1}) \in \bar{P}$, where $k \leq n$, add $(S) \to (x_1 \langle A_1, 1 \rangle \ldots x_k \langle A_k, k \rangle x_{k+1} \langle \varepsilon, k+1 \rangle \ldots \langle \varepsilon, n \rangle)$ to P;

2. For each $(A_1, \ldots, A_k) \to (z_1, \ldots, z_k) \in \bar{P}$, where $k \leq n$, $A_i \in (\bar{V} - T) - \{\bar{S}\}$ for all $1 \leq i \leq k$, $c_1, \ldots, c_{n-k} \in (\bar{V} - \{\bar{S}\}) \cup \{\varepsilon\}$, $\Gamma \in \mathrm{perm}(k, n-k)$,

$$\begin{aligned} (d_1, \ldots, d_n) &= \mathrm{reorder}((A_1, \ldots, A_k, c_1, \ldots, c_{n-k}), \Gamma), \\ (u_1, \ldots, u_n) &= \mathrm{reorder}((z_1, \ldots, z_k, c_1, \ldots, c_{n-k}), \Gamma), \end{aligned}$$

add $(\langle d_1, 1 \rangle, \ldots, \langle d_n, n \rangle) \to (\alpha(u_1, 1), \ldots, \alpha(u_n, n))$ to P;

3. For each $a_i \in T \cup \{\varepsilon\}$ for all $1 \leq i \leq n$, add
$(\langle a_1, 1\rangle, \ldots, \langle a_n, n\rangle) \to (a_1, \ldots, a_n)$ to P.

Productions from (1) simulate $\bar{G}$'s productions of the form

$$(\bar{S}) \to (x_1 A_1 \ldots x_k A_k x_{k+1}),$$

where $k \leq n$, so that each $A_i \in (\bar{V} - T) - \{\bar{S}\}$ is converted to $\langle A_i, i\rangle \in V_i - T$ for all $1 \leq i \leq k$ and the string $\langle \varepsilon, k+1\rangle \ldots \langle \varepsilon, n\rangle$, which is erased in the last step of the derivation, is added at the end of the resulting sentential form so that the sentential form contains n nonterminals. Consider the sentential form of the form

$$w_1 \langle B_1, 1\rangle \ldots w_m \langle B_m, m\rangle w_{m+1} \langle \varepsilon, m+1\rangle \ldots \langle \varepsilon, n\rangle,$$

where $w_j \in T^*$ and $\langle B_i, i\rangle \in V_i - T$ for all $1 \leq j \leq m+1$, $1 \leq i \leq m$. Each $\langle B_i, i\rangle$ may be of the form

- $\langle \varepsilon, i\rangle$ which indicates that the ith nonterminal was deleted in $\bar{G}$,
- $\langle a, i\rangle$, where $a \in T$, which indicates that the ith nonterminal was rewritten to a in $\bar{G}$, or
- $\langle A, i\rangle$, where $A \in (\bar{V} - T) - \{\bar{S}\}$.

The application of a production $(A_1, \ldots, A_k) \to (z_1, \ldots, z_k) \in \bar{P}$, where $k \leq n$, $A_i \in (\bar{V} - T) - \{\bar{S}\}$ for all $1 \leq i \leq k$, can be simulated in G if $B_{j_1} \ldots B_{j_k} = A_1 \ldots A_k$, where $j_i < j_{i+1}$, $1 \leq j_i \leq m$ for all $1 \leq i \leq k$. The productions of G constructed in step (2) permute $A_1, \ldots, A_k$ while preserving their order (see the definition of perm()) with symbols from $(\bar{V} - \{\bar{S}\}) \cup \{\varepsilon\}$ (not preserving their order), and convert them to the corresponding symbols from $(V_1 \cup \ldots \cup V_n) - T$. The same is performed with the elements of the right-hand side of the production for the same permutation. As a result, every symbol $\langle B_i, i\rangle$ remains unchanged if $\langle B_i, i\rangle = \langle \varepsilon, i\rangle$, $\langle B_i, i\rangle = \langle a, i\rangle$, where $a \in T$, or the simulated production is not applied to B_i. Otherwise, $\langle B_i, i\rangle$ is rewritten to $x\langle C, i\rangle y$, $x\langle a, i\rangle y$, or $\langle \varepsilon, i\rangle$, where $x, y \in T^*$, $a \in T$, $C \in (\bar{V} - T) - \{\bar{S}\}$, depending on the right-hand side of the simulated scattered context production's component applied to B_i. Finally, a production from (3) finishes the derivation by rewriting each $\langle a, i\rangle$, where $a \in T$, to a and erasing each $\langle \varepsilon, i\rangle$. ■

The main result follows next.

Theorem 23. *For each $n \geq 1$,*

$$\begin{aligned} \mathscr{L}(SC, LIN, n) &= \mathscr{L}(SM, LIN, n). \\ \mathscr{L}(SC, LIN) &= \mathscr{L}(SM, LIN). \end{aligned}$$

Proof. Follows immediately from Lemma 4 and Lemma 5. ■

A similar result can be proved for right-linear scattered context grammars as well.

Theorem 24. *For each $n \geq 1$,*

$$\begin{aligned} \mathscr{L}(SC, RLIN, n) &= \mathscr{L}(SM, RLIN, n). \\ \mathscr{L}(SC, RLIN) &= \mathscr{L}(SM, RLIN). \end{aligned}$$

Proof. The proof is analogous to the proof of Theorem 23 and is, therefore, left to the reader. ■

The following four corollaries follow immediately from Theorems 23, 24, and Theorems 2, 3, 4, 5.

Corollary 11. *For each $n \geq 1$,*

$$\begin{aligned}\mathscr{L}(SC, LIN, n) &\subset \mathscr{L}(SC, LIN, n+1),\\ \mathscr{L}(SC, RLIN, n) &\subset \mathscr{L}(SC, RLIN, n+1),\\ \mathscr{L}(SC, RLIN, n) &\subset \mathscr{L}(SC, LIN, n).\end{aligned}$$

Corollary 12.

$$\mathscr{L}(CF) - \mathscr{L}(SC, LIN) \neq \emptyset,\ \mathscr{L}(CF) - \mathscr{L}(SC, RLIN) \neq \emptyset,$$

$$\mathscr{L}(SC, RLIN) \subset \mathscr{L}(SC, LIN) \subset \mathscr{L}(PSC).$$

Corollary 13 (Positive closure properties)**.** *Each of the language families $\mathscr{L}(SC, RLIN, n)$ and $\mathscr{L}(SC, LIN, n)$, where $n \geq 1$, is closed under union, reversal, homomorphism, inverse homomorphism, substitution with regular languages, concatenation with regular languages, intersection with regular languages, left and right quotient by regular languages. $\mathscr{L}(SC, LIN)$ and $\mathscr{L}(SC, RLIN)$ are closed under concatenation.* ■

Corollary 14 (Negative closure properties)**.** *Each family $\mathscr{L}(SC, LIN, n)$, where $n \geq 1$, is not closed under concatenation with linear languages. Each family $\mathscr{L}(SC, RLIN, n)$, where $n \geq 1$, is not closed under concatenation with $\mathscr{L}(SC, RLIN, 2)$. $\mathscr{L}(SC, LIN)$ and $\mathscr{L}(SC, RLIN)$ are not closed under intersection, complement and Kleene star. $\mathscr{L}(SC, LIN)$ is not closed under substitution with linear languages. $\mathscr{L}(SC, RLIN)$ is not closed under substitution with $\mathscr{L}(SC, RLIN, 2)$.* ■

We have proved that (right) linear scattered context grammars characterize the same family of languages as (right) linear simple matrix grammars. The main difference of these grammars lies in the way how their productions are used. While in the case of (right) linear simple matrix grammars every production contains exactly n components and each component rewrites symbols over its own alphabet, the number of components in (right) linear scattered context grammars may be different in every production and all components share a single alphabet. In addition, we have proved that the generative power of (right) linear scattered context grammars does not depend on the number of components in scattered context productions (see Corollary 10) but on the number of nonterminals appearing in the starting production. As a result, (right) linear scattered context grammars seem to be more convenient for describing languages than (right) linear simple matrix grammars as the total number of their nonterminals is lower and their productions can avoid unnecessary rewriting and capture only true context dependencies.

An interesting aspect of our result is the fact that when restricting propagating scattered context grammars and simple matrix grammars to their right and right-linear variants, the power of the resulting grammars is equal. However, ordinary propagating scattered context grammars are more powerful than simple matrix grammars (see Theorem 13).

With respect to the Chomsky hierarchy, there remain two possible types of components of scattered context grammars: context-sensitive and unrestricted. The generative power of scattered context grammars with these components is obvious—they characterize the family of context-sensitive and unrestricted languages, respectively. Therefore, as scattered context grammars with context-sensitive and unrestricted components do not provide a higher generative power than their components, their practical use is questionable.

5.2 n-Limited Derivations

As formal language theory has always introduced and studied various left restrictions placed on grammatical derivations, we investigate this classical topic in terms of propagating scattered context

grammars. More specifically, we discuss the language families generated by propagating scattered context grammars whose derivations are n-limited, where n is a positive integer. In these derivations, a scattered context production is always applied within the first n occurrences of nonterminals in the current sentential form. It demonstrates that this restriction gives rise to an infinite hierarchy of language families. In addition, it proves that each family of this hierarchy is properly included in the family of context-sensitive languages.

Based upon the proper inclusion, we obtain several conclusions and formulate new open problems. Perhaps most importantly, we point out that the language family generated by propagating scattered context grammars that make derivations in the above n-limited way is properly contained in the context-sensitive language family, so in this sense, we partially contribute to the solving of Open Problem 1.

The result has also one practical aspect—when constructing a compiler based on a grammatical model, we usually need to restrict this model in order to make the compiler more effective. The presented result shows that when the model is based on propagating scattered context grammars, by limiting the width of the window in which the context dependency is checked, we also limit the power of this compiler. In certain situations, such as the parsing of streamed data, limiting the context dependency check to a finite window is necessary as we do not know the exact length of the input and we cannot determine which symbols appear on the input next.

First, we define n-limited derivations formally.

Definition 30. Let $G = (V,T,P,S)$ be a propagating scattered context grammar. If $(A_1,\ldots,A_k) \to (x_1,\ldots,x_k) \in P$, $u = u_1A_1u_2\ldots u_kA_ku_{k+1}$, and $v = u_1x_1u_2\ldots u_kx_ku_{k+1}$, where $u_i \in V^*$ for all $1 \leq i \leq k+1$, and $u \Rightarrow_G v$ satisfies

$$|u_1A_1\ldots u_kA_k|_{V-T} \leq n,$$

then the derivation step is *n-limited* and we write

$$u \overset{n}{\underset{\lim}{\Rightarrow}}_G v.$$

An *n-limited derivation*, denoted by $x \overset{n}{\underset{\lim}{\Rightarrow}}{}^*_G y$, is a derivation in which every derivation step $u \overset{j}{\underset{\lim}{\Rightarrow}}_G v$ satisfies $j \leq n$. Define the *language of order n* generated by G as

$$L(G,\lim,n) = \{x \in T^* : S \overset{n}{\underset{\lim}{\Rightarrow}}{}^*_G x\}.$$

The family of languages of order n generated by propagating scattered context grammars is denoted by $\mathscr{L}(PSC,\lim,n)$, and

$$\mathscr{L}(PSC,\lim,\infty) = \bigcup_{i=1}^{\infty} \mathscr{L}(PSC,\lim,i).$$

We prove the main result, $\mathscr{L}(PSC,\lim,n) = \mathscr{L}(ST,n)$ for all $n \geq 1$, by demonstrating that $\mathscr{L}(ST,n) \subseteq \mathscr{L}(PSC,\lim,n)$ and $\mathscr{L}(PSC,\lim,n) \subseteq \mathscr{L}(ST,n)$ in Lemmas 6 and 7, respectively.

Lemma 6. *$\mathscr{L}(ST,n) \subseteq \mathscr{L}(PSC,\lim,n)$ for all $n \geq 1$.*

Proof. Let $G = (V,T,K,P,S,p_0)$ be a state grammar of order n. Set

$$N_1 = \{\langle A,p,k\rangle : A \in V-T, p \in K, 1 \leq k \leq n\},$$
$$N_2 = \{\langle \hat{A},p,k\rangle : A \in V-T, p \in K, 1 \leq k \leq n\},$$
$$N_3 = \{\langle A',p,n-1\rangle : A \in V-T, p \in K\},$$

and $N_4 = \{\hat{A} : A \in V - T\}$. Set $\alpha(p) = \{A : (A,p) \to (x,q) \in P\}$ for each $p \in K$. Define the propagating scattered context grammar

$$\bar{G} = (V \cup N_1 \cup N_2 \cup N_3 \cup N_4 \cup \{\bar{S}\}, T, \bar{P}, \bar{S})$$

with $\bar{P}$ constructed as follows:

1. Add $(\bar{S}) \to (\langle \hat{S}, p_0, 1 \rangle)$ to $\bar{P}$;

2. For each $A_1, \ldots, A_k \in V - T$, where $1 \leq k \leq n$ (number of nonterminals in a sentential form if there are less than n nonterminals, otherwise $k = n$), each

$$(A_r, p) \to (x_1 B_1 \ldots x_t B_t x_{t+1}, q) \in P,$$

where $1 \leq r \leq k$ (the G's production simulates the rewriting of the rth nonterminal in a sentential form), $B_1, \ldots, B_t \in V - T$, $x_1, \ldots, x_{t+1} \in T^*$ for some $t \geq 0$ (number of nonterminals appearing on the right-hand side of the simulated G's production), $A_i \notin \alpha(p)$ for each $1 \leq i < r$, each $A_{n+1} \in (V - T) \cup N_4$,

   (a) and $r + t - 1 > n$, add

      i. (used when the sentential form contains more than n nonterminals)

$$\begin{aligned} &(\langle A_1, p, n \rangle, \ldots, \langle A_{r-1}, p, n \rangle, \langle A_r, p, n \rangle, \\ &\ \langle A_{r+1}, p, n \rangle, \ldots, \langle A_n, p, n \rangle) \\ \to\ &(\langle A_1, q, n \rangle, \ldots, \langle A_{r-1}, q, n \rangle, x_1 \langle B_1, q, n \rangle \ldots x_{n-r+1} \langle B_{n-r+1}, q, n \rangle \\ &\ x_{n-r+2} B_{n-r+2} \ldots x_t B_t x_{t+1}, A_{r+1}, \ldots, A_n) \end{aligned}$$

      to $\bar{P}$;

      ii. (used when the sentential form contains at most n nonterminals and A_r is not the last nonterminal)
      if $r < k$, add

$$\begin{aligned} &(\langle A_1, p, k \rangle, \ldots, \langle A_{r-1}, p, k \rangle, \langle A_r, p, k \rangle, \\ &\ \langle A_{r+1}, p, k \rangle, \ldots, \langle A_{k-1}, p, k \rangle, \langle \hat{A}_k, p, k \rangle) \\ \to\ &(\langle A_1, q, n \rangle, \ldots, \langle A_{r-1}, q, n \rangle, x_1 \langle B_1, q, n \rangle \ldots x_{n-r+1} \langle B_{n-r+1}, q, n \rangle \\ &\ x_{n-r+2} B_{n-r+2} \ldots x_t B_t x_{t+1}, A_{r+1}, \ldots, A_{k-1}, \hat{A}_k) \end{aligned}$$

      to $\bar{P}$;

      iii. (used when the sentential form contains at most n nonterminals and A_r is the last nonterminal)
      if $r = k$, add

$$\begin{aligned} &(\langle A_1, p, k \rangle, \ldots, \langle A_{k-1}, p, k \rangle, \langle \hat{A}_k, p, k \rangle) \\ \to\ &(\langle A_1, q, n \rangle, \ldots, \langle A_{k-1}, q, n \rangle, x_1 \langle B_1, q, n \rangle \ldots x_{n-k+1} \langle B_{n-k+1}, q, n \rangle \\ &\ x_{n-k+2} B_{n-k+2} \ldots x_{t-1} B_{t-1} x_t \hat{B}_t x_{t+1}) \end{aligned}$$

      to $\bar{P}$;

   (b) and $r + t - 1 \leq n$, $k + t - 1 > n$, add

i. (used when the sentential form contains more than n nonterminals)

$$\begin{aligned}&(\langle A_1,p,n\rangle,\ldots,\langle A_{r-1},p,n\rangle,\langle A_r,p,n\rangle,\\&\ \langle A_{r+1},p,n\rangle,\ldots,\langle A_{n-t+1},p,n\rangle,\langle A_{n-t+2},p,n\rangle,\ldots,\langle A_n,p,n\rangle)\\\rightarrow\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,x_1\langle B_1,q,n\rangle\ldots x_t\langle B_t,q,n\rangle x_{t+1},\\&\ \langle A_{r+1},q,n\rangle,\ldots,\langle A_{n-t+1},q,n\rangle,A_{n-t+2},\ldots,A_n)\end{aligned}$$

to $\bar{P}$;

ii. (used when the sentential form contains at most n nonterminals and A_r is not the last nonterminal)
if $r<k$, add

$$\begin{aligned}&(\langle A_1,p,k\rangle,\ldots,\langle A_{r-1},p,k\rangle,\langle A_r,p,k\rangle,\\&\ \langle A_{r+1},p,k\rangle,\ldots,\langle A_{n-t+1},p,k\rangle,\\&\ \langle A_{n-t+2},p,k\rangle,\ldots,\langle A_{k-1},p,k\rangle,\langle\hat{A}_k,p,k\rangle)\\\rightarrow\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,x_1\langle B_1,q,n\rangle\ldots x_t\langle B_t,q,n\rangle x_{t+1},\\&\ \langle A_{r+1},q,n\rangle,\ldots,\langle A_{n-t+1},q,n\rangle,A_{n-t+2},\ldots,A_{k-1},\hat{A}_k)\end{aligned}$$

to $\bar{P}$;

(c) and $k+t-1\leq n$, and

i. if $t=0$, add

A. (used when the sentential form contains more than n nonterminals and A_r is rewritten to $x_1\in T^*$)

$$\begin{aligned}&(\langle A_1,p,n\rangle,\ldots,\langle A_{r-1},p,n\rangle,\langle A_r,p,n\rangle,\\&\ \langle A_{r+1},p,n\rangle,\ldots,\langle A_n,p,n\rangle)\\\rightarrow\ &(\langle A'_1,q,n-1\rangle,\ldots,\langle A'_{r-1},q,n-1\rangle,x_1,\\&\ \langle A'_{r+1},q,n-1\rangle,\ldots,\langle A'_n,q,n-1\rangle),\end{aligned}$$

B. (used immediately after (2.c.i.A))

$$\begin{aligned}&(\langle A'_1,q,n-1\rangle,\ldots,\langle A'_{r-1},q,n-1\rangle,\\&\ \langle A'_{r+1},q,n-1\rangle,\ldots,\langle A'_n,q,n-1\rangle,A_{n+1})\\\rightarrow\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,\\&\ \langle A_{r+1},q,n\rangle,\ldots,\langle A_n,q,n\rangle,\langle A_{n+1},q,n\rangle)\end{aligned}$$

to $\bar{P}$;

ii. (used when the sentential form contains at most n nonterminals and A_r is not the last nonterminal)
if $r<k$, add

$$\begin{aligned}&(\langle A_1,p,k\rangle,\ldots,\langle A_{r-1},p,k\rangle,\\&\ \langle A_r,p,k\rangle,\langle A_{r+1},p,k\rangle,\ldots,\langle A_{k-1},p,k\rangle,\langle\hat{A}_k,p,k\rangle)\\\rightarrow\ &(\langle A_1,q,k+t-1\rangle,\ldots,\langle A_{r-1},q,k+t-1\rangle,\\&\ x_1\langle B_1,q,k+t-1\rangle\ldots x_t\langle B_t,q,k+t-1\rangle x_{t+1},\\&\ \langle A_{r+1},q,k+t-1\rangle,\ldots,\langle A_{k-1},q,k+t-1\rangle,\langle\hat{A}_k,q,k+t-1\rangle)\end{aligned}$$

to $\bar{P}$;

iii. (used when the sentential form contains at most n nonterminals and A_r is the last nonterminal)
if $r=k$

A. and $k > 1$ or $t \neq 0$, add

$$\begin{aligned}&(\langle A_1, p, k\rangle, \ldots, \langle A_{k-1}, p, k\rangle, \langle \hat{A}_k, p, k\rangle)\\ \rightarrow\ &(\langle A_1, q, k+t-1\rangle, \ldots, \langle A_{k-1}, q, k+t-1\rangle,\\ &x_1\langle B_1, q, k+t-1\rangle \ldots x_{t-1}\langle B_{t-1}, q, k+t-1\rangle\\ &x_t\langle \hat{B}_t, q, k+t-1\rangle x_{t+1})\end{aligned}$$

to $\bar{P}$;

B. (simulates the last derivation step of G)
and $k = 1$, $t = 0$, add
$(\langle \hat{A}_1, p, 1\rangle) \rightarrow (x_1)$ to $\bar{P}$.

Claim 5. Each G's sentential form

$$(y_1 A_1 \ldots y_m A_m y_{m+1}, p),$$

where $p \in K$, $y_1, \ldots, y_{m+1} \in T^*$, $A_1, \ldots, A_m \in V - T$ for some $m \geq 0$, corresponds to one of the following sentential forms in $\bar{G}$:

1. For $m \leq n$, $y_1\langle A_1, p, m\rangle \ldots y_{m-1}\langle A_{m-1}, p, m\rangle y_m\langle \hat{A}_m, p, m\rangle y_{m+1}$;
2. For $m > n$, $y_1\langle A_1, p, n\rangle \ldots y_n\langle A_n, p, n\rangle y_{n+1}A_{n+1} \ldots y_{m-1}A_{m-1}y_m\hat{A}_m y_{m+1}$.

Proof. Every derivation in $\bar{G}$ starts by the production from step (1) of the construction and this production is not used during the rest of the derivation process. So,

$$S \Rightarrow_{\bar{G}} \langle \hat{S}, p_0, 1\rangle.$$

The rest of the claim is proved by induction on the length h of derivations.

Basis. Let $h = 0$. Then, $(S, p_0) \Rightarrow_G^0 (S, p_0)$ corresponds to $\langle \hat{S}, p_0, 1\rangle \Rightarrow_{\bar{G}}^0 \langle \hat{S}, p_0, 1\rangle$.

Induction Hypothesis. Suppose that the claim holds for all derivations of length h or less, for some $h \geq 0$.

Induction Step. First, consider a G's sentential form $(y_1 A_1 \ldots y_m A_m y_{m+1}, p)$, where $m \leq n$, and a production

$$(A_r, p) \rightarrow (x_1 B_1 \ldots x_t B_t x_{t+1}, q) \in P,$$

where $1 \leq r \leq m$, $B_1, \ldots, B_t \in V - T$, $x_1, \ldots, x_{t+1} \in T^*$ for some $t \geq 0$, which is applicable to the above sentential form (that is, $A_i \notin \alpha(p)$ for each $1 \leq i < r$ and $A_r \in \alpha(p)$). By its application, we obtain

$$\begin{aligned}&(y_1 A_1 \ldots y_{r-1}A_{r-1}y_r A_r y_{r+1}A_{r+1} \ldots y_m A_m y_{m+1}, p)\\ \Rightarrow_G\ &(y_1 A_1 \ldots y_{r-1}A_{r-1}y_r x_1 B_1 \ldots x_t B_t x_{t+1} y_{r+1}A_{r+1} \ldots y_m A_m y_{m+1}, q).\end{aligned}$$

By the induction hypothesis, for $m \leq n$, the $\bar{G}$'s sentential form corresponding to

$$(y_1 A_1 \ldots y_{r-1}A_{r-1}y_r A_r y_{r+1}A_{r+1} \ldots y_m A_m y_{m+1}, p)$$

is of the form

$$y_1\langle A_1, p, m\rangle \ldots y_r\langle A_r, p, m\rangle \ldots y_{m-1}\langle A_{m-1}, p, m\rangle y_m\langle \hat{A}_m, p, m\rangle y_{m+1}.$$

Then, only one of the productions from steps (2.a.ii), (2.a.iii), (2.b.ii), (2.c.ii), and (2.c.iii) is applicable (productions from other steps require $m > n$). Notice that for $m \leq n$, all first m nonterminals in the $\bar{G}$'s sentential form contain m and p. This allows the grammar to use only the production which rewrites all m nonterminals and record the state corresponding to the simulated derivation in G. Furthermore, notice that all productions are constructed so that the last nonterminal in every sentential form is from $N_2 \cup N_4$. Which production is applied depends on m, n, r, and t:

1. If $r+t-1>n$ and $r<m$, then

$$\begin{aligned}&(\langle A_1,p,m\rangle,\ldots,\langle A_{r-1},p,m\rangle,\langle A_r,p,m\rangle,\\&\ \langle A_{r+1},p,m\rangle,\ldots,\langle A_{m-1},p,m\rangle,\langle\hat{A}_m,p,m\rangle)\\\rightarrow\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,x_1\langle B_1,q,n\rangle\ldots x_{n-r+1}\langle B_{n-r+1},q,n\rangle\\&\ x_{n-r+2}B_{n-r+2}\ldots x_tB_tx_{t+1},A_{r+1},\ldots,A_{m-1},\hat{A}_m)\end{aligned}$$

introduced by (2.a.ii) is applied, so

$$\begin{aligned}&y_1\langle A_1,p,m\rangle\ldots y_{r-1}\langle A_{r-1},p,m\rangle y_r\langle A_r,p,m\rangle\\&y_{r+1}\langle A_{r+1},p,m\rangle\ldots y_{m-1}\langle A_{m-1},p,m\rangle y_m\langle\hat{A}_m,p,m\rangle y_{m+1}\\\overset{n}{\lim}\Rightarrow_{\bar{G}}\ &y_1\langle A_1,q,n\rangle\ldots y_{r-1}\langle A_{r-1},q,n\rangle y_r\\&x_1\langle B_1,q,n\rangle\ldots x_{n-r+1}\langle B_{n-r+1},q,n\rangle x_{n-r+2}B_{n-r+2}\ldots x_tB_tx_{t+1}\\&y_{r+1}A_{r+1}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}.\end{aligned}$$

2. If $r+t-1>n$ and $r=m$, then

$$\begin{aligned}&(\langle A_1,p,m\rangle,\ldots,\langle A_{m-1},p,m\rangle,\langle\hat{A}_m,p,m\rangle)\\\rightarrow\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{m-1},q,n\rangle,x_1\langle B_1,q,n\rangle\ldots x_{n-m+1}\langle B_{n-m+1},q,n\rangle\\&\ x_{n-m+2}B_{n-m+2}\ldots x_{t-1}B_{t-1}x_t\hat{B}_tx_{t+1})\end{aligned}$$

introduced by (2.a.iii) is applied, so

$$\begin{aligned}&y_1\langle A_1,p,m\rangle\ldots y_{m-1}\langle A_{m-1},p,m\rangle y_m\langle\hat{A}_m,p,m\rangle y_{m+1}\\\overset{n}{\lim}\Rightarrow_{\bar{G}}\ &y_1\langle A_1,q,n\rangle\ldots y_{m-1}\langle A_{m-1},q,n\rangle y_m\\&x_1\langle B_1,q,n\rangle\ldots x_{n-m+1}\langle B_{n-m+1},q,n\rangle\\&x_{n-m+2}B_{n-m+2}\ldots x_{t-1}B_{t-1}x_t\hat{B}_tx_{t+1}.\end{aligned}$$

3. If $r+t-1\leq n$, $m+t-1>n$, and $r<m$, then

$$\begin{aligned}&(\langle A_1,p,m\rangle,\ldots,\langle A_{r-1},p,m\rangle,\langle A_r,p,m\rangle,\\&\ \langle A_{r+1},p,m\rangle,\ldots,\langle A_{n-t+1},p,m\rangle,\\&\ \langle A_{n-t+2},p,m\rangle,\ldots,\langle A_{m-1},p,m\rangle,\langle\hat{A}_m,p,m\rangle)\\\rightarrow\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,x_1\langle B_1,q,n\rangle\ldots x_t\langle B_t,q,n\rangle x_{t+1},\\&\ \langle A_{r+1},q,n\rangle,\ldots,\langle A_{n-t+1},q,n\rangle,A_{n-t+2},\ldots,A_{m-1},\hat{A}_m)\end{aligned}$$

introduced by (2.b.ii) is applied, so

$$\begin{aligned}&y_1\langle A_1,p,m\rangle\ldots y_{r-1}\langle A_{r-1},p,m\rangle y_r\langle A_r,p,m\rangle\\&y_{r+1}\langle A_{r+1},p,m\rangle\ldots y_{m-1}\langle A_{m-1},p,m\rangle y_m\langle\hat{A}_m,p,m\rangle y_{m+1}\\\overset{n}{\lim}\Rightarrow_{\bar{G}}\ &y_1\langle A_1,q,n\rangle\ldots y_{r-1}\langle A_{r-1},q,n\rangle y_rx_1\langle B_1,q,n\rangle\ldots x_t\langle B_t,q,n\rangle x_{t+1}\\&y_{r+1}\langle A_{r+1},q,n\rangle\ldots y_{n-t+1}\langle A_{n-t+1},q,n\rangle\\&y_{n-t+2}A_{n-t+2}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}.\end{aligned}$$

4. If $m+t-1\leq n$ and $r<m$, then

$$\begin{aligned}&(\langle A_1,p,m\rangle,\ldots,\langle A_{r-1},p,m\rangle,\langle A_r,p,m\rangle,\\&\ \langle A_{r+1},p,m\rangle,\ldots,\langle A_{m-1},p,m\rangle,\langle\hat{A}_m,p,m\rangle)\\\rightarrow\ &(\langle A_1,q,m+t-1\rangle,\ldots,\langle A_{r-1},q,m+t-1\rangle,\\&\ x_1\langle B_1,q,m+t-1\rangle\ldots x_t\langle B_t,q,m+t-1\rangle x_{t+1},\\&\ \langle A_{r+1},q,m+t-1\rangle,\ldots,\langle A_{m-1},q,m+t-1\rangle,\langle\hat{A}_m,q,m+t-1\rangle)\end{aligned}$$

introduced by (2.c.ii) is applied, so

$$\begin{aligned}
& y_1\langle A_1,p,m\rangle \dots y_{r-1}\langle A_{r-1},p,m\rangle y_r\langle A_r,p,m\rangle \\
& y_{r+1}\langle A_{r+1},p,m\rangle \dots y_{m-1}\langle A_{m-1},p,m\rangle y_m\langle \hat{A}_m,p,m\rangle y_{m+1} \\
\overset{n}{\lim}\Rightarrow_{\bar{G}}\ & y_1\langle A_1,q,m+t-1\rangle \dots y_{r-1}\langle A_{r-1},q,m+t-1\rangle y_r \\
& x_1\langle B_1,q,m+t-1\rangle \dots x_t\langle B_t,q,m+t-1\rangle x_{t+1} \\
& y_{r+1}\langle A_{r+1},q,m+t-1\rangle \dots y_{m-1}\langle A_{m-1},q,m+t-1\rangle \\
& y_m\langle \hat{A}_m,q,m+t-1\rangle y_{m+1}.
\end{aligned}$$

5. If $m+t-1 \leq n$, $r=m$, and $m>1$ or $t\neq 0$, then

$$\begin{aligned}
& (\langle A_1,p,m\rangle,\dots,\langle A_{m-1},p,m\rangle,\langle \hat{A}_m,p,m\rangle) \\
\to\ & (\langle A_1,q,m+t-1\rangle,\dots,\langle A_{m-1},q,m+t-1\rangle, \\
& x_1\langle B_1,q,m+t-1\rangle \dots x_{t-1}\langle B_{t-1},q,m+t-1\rangle \\
& x_t\langle \hat{B}_t,q,m+t-1\rangle x_{t+1})
\end{aligned}$$

introduced by (2.c.iii.A) is applied, so

$$\begin{aligned}
& y_1\langle A_1,p,m\rangle \dots y_{m-1}\langle A_{m-1},p,m\rangle y_m\langle \hat{A}_m,p,m\rangle y_{m+1} \\
\overset{n}{\lim}\Rightarrow_{\bar{G}}\ & y_1\langle A_1,q,m+t-1\rangle \dots y_{m-1}\langle A_{m-1},q,m+t-1\rangle y_m \\
& x_1\langle B_1,q,m+t-1\rangle \dots x_{t-1}\langle B_{t-1},q,m+t-1\rangle \\
& x_t\langle \hat{B}_t,q,m+t-1\rangle x_{t+1}y_{m+1}.
\end{aligned}$$

6. If $m=1$, $t=0$, then

$$(\langle \hat{A}_1,p,1\rangle) \to (x_1)$$

introduced by (2.c.iii.B) is applied, so

$$y_1\langle \hat{A}_1,p,1\rangle y_2 \overset{n}{\lim}\Rightarrow_{\bar{G}} y_1x_1y_2.$$

Notice that this production can only be used in the ultimate derivation step because it removes the last symbol from $N_1\cup N_2$ from the sentential form and this symbol is on the left-hand side of every production introduced in step (2).

Observe that as all of these productions satisfy $A_i\notin\alpha(p)$ for each $1\leq i<r$ and all nonterminals in the sentential form of $\bar{G}$ are rewritten, the simulation of G's production is proper.

Second, consider a G's sentential form $(y_1A_1\dots y_mA_my_{m+1},p)$, where $m>n$, and a production

$$(A_r,p)\to(x_1B_1\dots x_tB_tx_{t+1},q)\in P,$$

where $1\leq r\leq m$, $B_1,\dots,B_t\in V-T$, $x_1,\dots,x_{t+1}\in T^*$ for some $t\geq 0$, which is applicable to the above sentential form. By its use, we obtain

$$\begin{aligned}
& (y_1A_1\dots y_{r-1}A_{r-1}y_rA_ry_{r+1}A_{r+1}\dots y_nA_n\dots y_mA_my_{m+1},p) \\
\Rightarrow_G\ & (y_1A_1\dots y_{r-1}A_{r-1}y_rx_1B_1\dots x_tB_tx_{t+1} \\
& y_{r+1}A_{r+1}\dots y_nA_n\dots y_mA_my_{m+1},q).
\end{aligned}$$

By the induction hypothesis, for $m>n$, the $\bar{G}$'s sentential form corresponding to

$$(y_1A_1\dots y_{r-1}A_{r-1}y_rA_ry_{r+1}A_{r+1}\dots y_nA_n\dots y_mA_my_{m+1},p)$$

is of the form

$$y_1\langle A_1,p,n\rangle \dots y_r\langle A_r,p,n\rangle \dots y_n\langle A_n,p,n\rangle y_{n+1}A_{n+1}\dots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}.$$

Then, only one of the productions from (2.a.i), (2.b.i), and (2.c.i.A) is applicable, depending on m, n, r, and t:

1. If $r+t-1>n$, then

$$\begin{aligned}&(\langle A_1,p,n\rangle,\ldots,\langle A_{r-1},p,n\rangle,\langle A_r,p,n\rangle,\langle A_{r+1},p,n\rangle,\ldots,\langle A_n,p,n\rangle)\\ \to\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,x_1\langle B_1,q,n\rangle\ldots x_{n-r+1}\langle B_{n-r+1},q,n\rangle\\ &x_{n-r+2}B_{n-r+2}\ldots x_tB_tx_{t+1},A_{r+1},\ldots,A_n)\end{aligned}$$

introduced by (2.a.i) is applied, so

$$\begin{aligned}&y_1\langle A_1,p,n\rangle\ldots y_{r-1}\langle A_{r-1},p,n\rangle y_r\langle A_r,p,n\rangle\\ &y_{r+1}\langle A_{r+1},p,n\rangle\ldots y_n\langle A_n,p,n\rangle\\ &y_{n+1}A_{n+1}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}\\ \overset{n}{\lim}\Rightarrow_{\bar G}\ &y_1\langle A_1,q,n\rangle\ldots y_{r-1}\langle A_{r-1},q,n\rangle y_r\\ &x_1\langle B_1,q,n\rangle\ldots x_{n-r+1}\langle B_{n-r+1},q,n\rangle x_{n-r+2}B_{n-r+2}\ldots x_tB_tx_{t+1}\\ &y_{r+1}A_{r+1}\ldots y_nA_ny_{n+1}A_{n+1}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}.\end{aligned}$$

2. If $r+t-1\leq n$ and $m+t-1>n$, then

$$\begin{aligned}&(\langle A_1,p,n\rangle,\ldots,\langle A_{r-1},p,n\rangle,\langle A_r,p,n\rangle,\\ &\langle A_{r+1},p,n\rangle,\ldots,\langle A_{n-t+1},p,n\rangle,\langle A_{n-t+2},p,n\rangle,\ldots,\langle A_n,p,n\rangle)\\ \to\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,x_1\langle B_1,q,n\rangle\ldots x_t\langle B_t,q,n\rangle x_{t+1},\\ &\langle A_{r+1},q,n\rangle,\ldots,\langle A_{n-t+1},q,n\rangle,A_{n-t+2},\ldots,A_n)\end{aligned}$$

introduced by (2.b.i) is applied, so

$$\begin{aligned}&y_1\langle A_1,p,n\rangle\ldots y_{r-1}\langle A_{r-1},p,n\rangle y_r\langle A_r,p,n\rangle\\ &y_{r+1}\langle A_{r+1},p,n\rangle\ldots y_{n-t+1}\langle A_{n-t+1},p,n\rangle\\ &y_{n-t+2}\langle A_{n-t+2},p,n\rangle\ldots y_n\langle A_n,p,n\rangle\\ &y_{n+1}A_{n+1}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}\\ \overset{n}{\lim}\Rightarrow_{\bar G}\ &y_1\langle A_1,q,n\rangle\ldots y_{r-1}\langle A_{r-1},q,n\rangle y_r\\ &x_1\langle B_1,q,n\rangle\ldots x_t\langle B_t,q,n\rangle x_{t+1}\\ &y_{r+1}\langle A_{r+1},q,n\rangle\ldots y_{n-t+1}\langle A_{n-t+1},q,n\rangle\\ &y_{n-t+2}A_{n-t+2}\ldots y_nA_ny_{n+1}A_{n+1}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}.\end{aligned}$$

3. If $m+t-1\leq n$ and $t=0$, then

$$\begin{aligned}&(\langle A_1,p,n\rangle,\ldots,\langle A_{r-1},p,n\rangle,\langle A_r,p,n\rangle,\langle A_{r+1},p,n\rangle,\ldots,\langle A_n,p,n\rangle)\\ \to\ &(\langle A'_1,q,n-1\rangle,\ldots,\langle A'_{r-1},q,n-1\rangle,x_1,\\ &\langle A'_{r+1},q,n-1\rangle,\ldots,\langle A'_n,q,n-1\rangle)\end{aligned}$$

introduced by (2.c.i.A) is applied, so

$$\begin{aligned}&y_1\langle A_1,p,n\rangle\ldots y_{r-1}\langle A_{r-1},p,n\rangle y_r\langle A_r,p,n\rangle\\ &y_{r+1}\langle A_{r+1},p,n\rangle\ldots y_n\langle A_n,p,n\rangle\\ &y_{n+1}A_{n+1}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}\\ \overset{n}{\lim}\Rightarrow_{\bar G}\ &y_1\langle A'_1,q,n-1\rangle\ldots y_{r-1}\langle A'_{r-1},q,n-1\rangle y_rx_1\\ &y_{r+1}\langle A'_{r+1},q,n-1\rangle\ldots y_n\langle A'_n,q,n-1\rangle\\ &y_{n+1}A_{n+1}\ldots y_{m-1}A_{m-1}y_m\hat{A}_my_{m+1}.\end{aligned}$$

Recall that the last nonterminal in every sentential form of $\bar G$ is from $N_2\cup N_4$. As $\langle A_n,p,n\rangle\notin N_2\cup N_4$, there is at least one nonterminal in the sentential form following $\langle A_n,p,n\rangle$. Therefore, the production

$$\begin{aligned}&(\langle A'_1,q,n-1\rangle,\ldots,\langle A'_{r-1},q,n-1\rangle,\\ &\langle A'_{r+1},q,n-1\rangle,\ldots,\langle A'_n,q,n-1\rangle,A_{n+1})\\ \to\ &(\langle A_1,q,n\rangle,\ldots,\langle A_{r-1},q,n\rangle,\\ &\langle A_{r+1},q,n\rangle,\ldots,\langle A_n,q,n\rangle,\langle A_{n+1},q,n\rangle)\end{aligned}$$

from (2.c.i.B) can be used. This production rewrites a nonterminal $A \in (V-T) \cup N_4$ in its last component. Because we generate a language of order n, $A = A_{n+1}$, so either

$$\begin{aligned} & y_1\langle A'_1,q,n-1\rangle \dots y_{r-1}\langle A'_{r-1},q,n-1\rangle y_r x_1 \\ & y_{r+1}\langle A'_{r+1},q,n-1\rangle \dots y_n\langle A'_n,q,n-1\rangle \\ & y_{n+1}A_{n+1}y_{n+2}A_{n+2}\dots y_{m-1}A_{m-1}y_m\hat{A}_m y_{m+1} \\ \overset{n}{\underset{\lim}{\Rightarrow}}_{\bar{G}} \; & y_1\langle A_1,q,n\rangle \dots y_{r-1}\langle A_{r-1},q,n\rangle y_r x_1 \\ & y_{r+1}\langle A_{r+1},q,n\rangle \dots y_n\langle A_n,q,n\rangle \\ & y_{n+1}\langle A_{n+1},q,n\rangle y_{n+2}A_{n+2}\dots y_{m-1}A_{m-1}y_m\hat{A}_m y_{m+1} \end{aligned}$$

if $A_{n+1} \in V-T$ or

$$\begin{aligned} & y_1\langle A'_1,q,n-1\rangle \dots y_{r-1}\langle A'_{r-1},q,n-1\rangle y_r x_1 \\ & y_{r+1}\langle A'_{r+1},q,n-1\rangle \dots y_n\langle A'_n,q,n-1\rangle y_{n+1}\hat{A}_{n+1}y_{n+2} \\ \overset{n}{\underset{\lim}{\Rightarrow}}_{\bar{G}} \; & y_1\langle A_1,q,n\rangle \dots y_{r-1}\langle A_{r-1},q,n\rangle y_r x_1 \\ & y_{r+1}\langle A_{r+1},q,n\rangle \dots y_n\langle A_n,q,n\rangle y_{n+1}\langle \hat{A}_{n+1},q,n\rangle y_{n+2}, \end{aligned}$$

if $\hat{A}_{n+1} \in N_4$.

Observe that as all of these productions satisfy $A_i \notin \alpha(p)$ for each $1 \leq i < r$ and all first n nonterminals in the sentential form of $\bar{G}$ are rewritten, the simulation of G's production is proper.

Finally, notice that if the sentential form of $\bar{G}$ is of the form (1) or (2) as described in Claim 5, the sentential form obtained after performing a derivation step is of one of these forms as well. As the right-hand side of the production introduced in step (1) of the construction is of the form (1), every sentential form obtained during the derivation process satisfies the properties given in Claim 5. □

From Claim 5 and the derivations described in its proof, it is easy to see that $\bar{G}$ rewrites at most n first nonterminals in a sentential form and that $L(G,n) = L(\bar{G},\lim,n)$. ■

Lemma 7. *$\mathscr{L}(PSC,\lim,n) \subseteq \mathscr{L}(ST,n)$ for all $n \geq 1$.*

Proof. Let $L(G,\lim,n)$ be a language of order n generated by a propagating scattered context grammar $G = (V,T,P,S)$. Set

$$N = \{\langle A,i\rangle : A \in V-T, 1 \leq i \leq n\}.$$

Further, set $K_1 = \{\langle p,i\rangle : p \in P, 0 \leq i < n\}$ and

$$K_2 = \{\langle p,i,j\rangle : p = (A_1,\dots,A_k) \to (x_1,\dots,x_k) \in P, 0 \leq i \leq n, 0 \leq j \leq k\}.$$

Define the state grammar

$$\bar{G} = (V \cup N \cup \{\bar{S}\}, T, K_1 \cup K_2 \cup \{p_0\}, \bar{P}, \bar{S}, p_0)$$

with $\bar{P}$ constructed as follows:

1. For each $p = (S) \to (x) \in P$, add
   $(\bar{S},p_0) \to (S,\langle p,0\rangle)$ to $\bar{P}$;
2. For each $A \in V-T$, $p = (A_1,\dots,A_k) \to (x_1,\dots,x_k) \in P$, $0 \leq i < n$, add
   (a) $(A,\langle p,i\rangle) \to (\langle A,i+1\rangle,\langle p,i+1\rangle)$,
   (b) $(A,\langle p,i\rangle) \to (\langle A,i+1\rangle,\langle p,i+1,k\rangle)$ to $\bar{P}$;

3. For each $p = (A_1,\ldots,A_j,\ldots,A_k) \to (x_1,\ldots,x_j,\ldots,x_k) \in P$, $q \in P$, $A \in V - T$, $1 \le i \le n$, $0 \le j \le k$, add

   (a) $(\langle A,i\rangle,\langle p,i,j\rangle) \to (A,\langle p,i-1,j\rangle)$,

   (b) if $j \ge 1$, add
   $(\langle A_j,i\rangle,\langle p,i,j\rangle) \to (x_j,\langle p,i-1,j-1\rangle)$,

   (c) $(A,\langle p,0,0\rangle) \to (A,\langle q,0\rangle)$ to $\bar{P}$.

The derivation starts in $\bar{G}$ by a production introduced in step (1) and as no production contains $\bar{S}$ on its right-hand side, none of the productions form (1) is used during the rest of the derivation. Consider now a G's sentential form $u_1A_1\ldots u_kA_ku_{k+1}$, where $u_1,\ldots,u_{k+1} \in V^*$, and a production

$$p = (A_1,\ldots,A_k) \to (x_1,\ldots,x_k) \in P.$$

Obviously, for a sentential form satisfying $|u_1A_1\ldots u_kA_k|_{V-T} \le n$,

$$u_1A_1\ldots u_kA_ku_{k+1} \overset{n}{\underset{\text{lim}}{\Rightarrow}}_G u_1x_1\ldots u_kx_ku_{k+1}\ [p].$$

Consider now a $\bar{G}$'s sentential form

$$(u_1A_1\ldots u_kA_ku_{k+1},\langle p,0\rangle)$$

corresponding to the above sentential form of G. (Notice that also $(S,\langle q,0\rangle)$, where $q = (S) \to (x) \in P$, obtained by the application of a production from (1) is a sentential form of this kind.) The above described G's derivation step is simulated in $\bar{G}$ by productions from (2) and (3). First, productions from (2) are used. For a nondeterministically chosen $m \ge k$, the first m nonterminals in $u_1A_1\ldots u_kA_ku_{k+1}$ are processed from the left to the right and each nonterminal is assigned its ordinal number. Then, productions from (3) are used to simulate p's context-free components form the right to the left. To describe this derivation more formally, we express

$$(u_1A_1\ldots u_kA_ku_{k+1},\langle p,0\rangle)$$

as

$$(w_1B_1\ldots w_mB_m\ldots w_nB_n\ldots w_tB_tw_{t+1},\langle p,0\rangle),$$

where $w_1,\ldots,w_{t+1} \in T^*$, $B_1,\ldots,B_t \in V - T$, $t = |u_1A_1\ldots u_kA_ku_{k+1}|_{V-T}$, and where $B_{l_i} = A_i$ for some $1 \le l_i \le m$, for all $1 \le i \le k$, and $l_j < l_{j+1}$ for all $1 \le j \le k-1$. (Discussion of other possible kinds of this sentential form, for example for $t < n$, is left to the reader.) Then, the derivation performed by productions from (2a) can be expressed as

$$\begin{aligned}
&(w_1B_1\ldots w_mB_m\ldots w_tB_tw_{t+1},\langle p,0\rangle)\\
\Rightarrow_{\bar{G}}\ &(w_1\langle B_1,1\rangle w_2B_2\ldots w_mB_m\ldots w_tB_tw_{t+1},\langle p,1\rangle)\\
&\vdots\\
\Rightarrow_{\bar{G}}\ &(w_1\langle B_1,1\rangle\ldots w_{m-1}\langle B_{m-1},m-1\rangle w_mB_m\ldots w_tB_tw_{t+1},\langle p,m-1\rangle).
\end{aligned}$$

Finally, a production from (2b) is used, so

$$\begin{aligned}
&(w_1\langle B_1,1\rangle\ldots w_{m-1}\langle B_{m-1},m-1\rangle w_mB_m\ldots w_tB_tw_{t+1},\langle p,m-1\rangle)\\
\Rightarrow_{\bar{G}}\ &(w_1\langle B_1,1\rangle\ldots w_m\langle B_m,m\rangle w_{m+1}B_{m+1}\ldots w_tB_tw_{t+1},\langle p,m,k\rangle).
\end{aligned}$$

Next, productions from (3) are used to simulate all context-free components of p in the reversed order. The simulation of $A_k \to x_k$ is performed as follows:

$$\begin{aligned}
&(w_1\langle B_1,1\rangle \dots w_m\langle B_m,m\rangle w_{m+1}B_{m+1}\dots w_tB_tw_{t+1},\langle p,m,k\rangle)\\
\Rightarrow_{\bar{G}}\ &(w_1\langle B_1,1\rangle \dots w_{m-1}\langle B_{m-1},m-1\rangle w_mB_m\dots w_tB_tw_{t+1},\langle p,m-1,k\rangle)\\
&\vdots\\
\Rightarrow_{\bar{G}}\ &(w_1\langle B_1,1\rangle \dots w_{l_k}\langle B_{l_k},l_k\rangle w_{l_k+1}B_{l_k+1}\dots w_tB_tw_{t+1},\langle p,l_k,k\rangle)\\
\Rightarrow_{\bar{G}}\ &(w_1\langle B_1,1\rangle \dots w_{l_k-1}\langle B_{l_k-1},l_k-1\rangle\\
&\quad w_{l_k}x_kw_{l_k+1}B_{l_k+1}\dots w_tB_tw_{t+1},\langle p,l_k-1,k-1\rangle).
\end{aligned}$$

The context-free components $A_{k-1}\to x_{k-1},\dots,A_1\to x_1$ are simulated analogously, until a sentential form

$$(u_1x_1\dots u_kx_ku_{k+1},\langle p,0,0\rangle)$$

is obtained. Notice that when a state $\langle p,0,i\rangle$, $i\geq 1$ is reached, the derivation is blocked. This means that either in the nondeterministic part of the derivation the value of m was chosen too low so the whole scattered context production cannot be simulated or more than n first nonterminals need to be rewritten to simulate the scattered context production. Finally, a production from (3c) is used to finish the simulation of p and to start the simulation of $q\in P$:

$$\begin{aligned}
&(u_1x_1\dots u_kx_ku_{k+1},\langle p,0,0\rangle)\\
\Rightarrow_{\bar{G}}\ &(u_1x_1\dots u_kx_ku_{k+1},\langle q,0\rangle).
\end{aligned}$$

This simulation continues until the sentential form $(w,\langle q,0,0\rangle)$, where $w\in T^*$, $q\in P$, is obtained. ■

Theorem 25. $\mathscr{L}(PSC,\lim,n)=\mathscr{L}(ST,n)$.

Proof. Follows immediately from Lemmas 6 and 7. ■

The following corollaries follow immediately from Theorems 7, 8 and 25.

Corollary 15.

$$\mathscr{L}(CF)=\mathscr{L}(PSC,\lim,1)\subset\mathscr{L}(PSC,\lim,2)\subset\ldots\subset\mathscr{L}(PSC,\lim,\infty)\subset\mathscr{L}(CS).$$

■

Corollary 16. *Every $\mathscr{L}(PSC,\lim,n)$, where $n\geq 1$, is an abstract family of languages.* ■

We have demonstrated that limiting derivations performed by propagating scattered context grammars to the first n nonterminals gives rise to an infinite hierarchy of languages. The definition of n-limited derivations, however, induces the following problem:

Open Problem 5. Can we construct a propagating scattered context grammar which rewrites the first n nonterminals without restricting the derivations explicitly (that is to define a propagating scattered context grammar of order n in the way analogous to a state grammar of order n) and obtain the same results?

We propose this open problem for further study.

5.3 Leftmost Derivations

As the exact relation of $\mathscr{L}(PSC)$ with respect to $\mathscr{L}(CS)$ is unknown, there have been several attempts to modify the basic definition of propagating scattered context grammars to obtain the family of context-sensitive languages. The approach discussed in [82] is one of them. In [82] it was proved that propagating scattered context grammars that use leftmost derivations are as powerful as context-sensitive grammars. This result is of some interest as the use of context-free, context-sensitive, and unrestricted productions in a leftmost way in the corresponding grammars of the Chomsky hierarchy does not have any impact on their generative power.

The proof presented in [82] consists of two parts; first, two preliminary lemmas (Lemma 2 and Lemma 3) are given and then, the main result, stated in Theorem 2, is presented as a straightforward corollary of these two lemmas. In Lemma 2 it is demonstrated how any sentence of a context-sensitive language can be derived by a propagating scattered context grammar which uses leftmost derivations. Every sentence generated in such a way contains, however, some additional symbols. Lemma 3 shows how these symbols can be removed. Together, the proof consists of six-page construction part and not even one-page basic idea of the construction which makes it extremely hard to follow. A more formal proof of the correctness of the construction is missing.

Our aim is to present the proof of this result in much simpler and more readable way. The main difference of our proof lies in (1) the way how the symbols to be rewritten are selected and (2) the way how context-sensitive productions are simulated. Furthermore, the proof is based on only one construction instead of two. All this leads to a significantly simpler and more transparent proof.

We start by defining propagating scattered context grammars which use leftmost and rightmost derivations.

Definition 31. A *propagating scattered context grammar which uses leftmost* or *rightmost derivations* is a propagating scattered context grammar $G = (V,T,P,S)$ whose language is defined as

$$L(G,\mathrm{lm}) = \{x \in T^* : S \,{}_{\mathrm{lm}}\Rightarrow^*_G x\} \text{ or } L(G,\mathrm{rm}) = \{x \in T^* : S \,{}_{\mathrm{rm}}\Rightarrow^*_G x\},$$

respectively. The family of languages generated by propagating scattered context grammars which use leftmost or rightmost derivations is denoted by $\mathscr{L}(PSC,\mathrm{lm})$ or $\mathscr{L}(PSC,\mathrm{rm})$, respectively.

The following theorem and its proof demonstrate how for every context-sensitive grammar G in Kuroda normal form a propagating scattered context grammar $\bar{G}$ which uses leftmost derivations can be constructed so that $L(G) = L(\bar{G},\mathrm{lm})$.

Theorem 26. $\mathscr{L}(PSC,\mathrm{lm}) = \mathscr{L}(CS)$.

Proof. Let $G = (V,T,P,S)$ be a context-sensitive grammar in Kuroda normal form. Set $N_1 = (V - T) \cup \{\bar{a} : a \in T\}$ and $\hat{N}_1 = \{\hat{A} : A \in N_1\}$. Let $n = |N_1|$; then, we denote the elements of N_1 as $\{A_1,\ldots,A_n\}$. Define the homomorphism α from V^* to N_1^* as $\alpha(A) = A$ for each $A \in V - T$, and $\alpha(a) = \bar{a}$ for each $a \in T$. Set $N'_2 = \{A' : A \in V - T\}$, $N_3 = \{\langle ab\rangle : a,b \in V\}$,

$$N'_4 = \{\langle Aa\rangle' : A \in V - T, a \in V\},$$

and

$$\begin{aligned} N_5 = {} & \{\langle a,0\rangle, \langle ab,0\rangle : a,b \in V\} \\ & \cup \{\langle a,i,j\rangle : a \in V - T, 1 \le i \le 3, 1 \le j \le n\} \\ & \cup \{\langle ab,4\rangle : a,b \in T\}. \end{aligned}$$

Define the propagating scattered context grammar

$$\bar{G} = (N_1 \cup \hat{N}_1 \cup N'_2 \cup N_3 \cup N'_4 \cup N_5 \cup \{\bar{S},X\} \cup T, T, \bar{P}, \bar{S}),$$

where $\bar{P}$ is constructed as follows:

1. (a) For each $a \in L(G)$, where $a \in T$, add $(\bar{S}) \to (a)$ to $\bar{P}$;
   (b) For each $S \Rightarrow_G ab$, where $a, b \in V$, add $(\bar{S}) \to (\langle ab, 0\rangle X)$ to $\bar{P}$;

2. For each $a, b, c \in V$, add
   (a) $(\langle a, 0\rangle, \alpha(b)) \to (\alpha(a), \langle b, 0\rangle)$,
   (b) $(\alpha(a), \langle b, 0\rangle) \to (\langle a, 0\rangle, \alpha(b))$,
   (c) $(\langle a, 0\rangle, \langle bc\rangle) \to (\alpha(a), \langle bc, 0\rangle)$,
   (d) $(\alpha(a), \langle bc, 0\rangle) \to (\langle a, 0\rangle, \langle bc\rangle)$ to $\bar{P}$;

3. For each $A \to a \in P$ and $b \in V$, add
   (a) $(\langle A, 0\rangle) \to (\langle a, 0\rangle)$,
   (b) $(\langle Ab, 0\rangle) \to (\langle ab, 0\rangle)$,
   (c) $(\langle bA, 0\rangle) \to (\langle ba, 0\rangle)$ to $\bar{P}$;

4. For each $A \to BC \in P$ and $a \in V$, add
   (a) $(\langle A, 0\rangle) \to (B\langle C, 0\rangle)$,
   (b) $(\langle Aa, 0\rangle) \to (B\langle Ca, 0\rangle)$,
   (c) $(\langle aA, 0\rangle) \to (\alpha(a)\langle BC, 0\rangle)$ to $\bar{P}$;

5. For each $AB \to CD \in P$, $a \in V$, $E \in N_3 \cup N_4'$, $F' \in \{B', \langle Ba\rangle'\}$, $1 \leq i \leq n$, and $1 \leq j \leq n-1$, add
   (a) $(\langle AB, 0\rangle) \to (\langle CD, 0\rangle)$,
   (b) i. $(\langle A, 0\rangle, B, X) \to (\langle A, 1, 1\rangle, B', A_1)$,
   ii. $(\langle A, 0\rangle, \langle Ba\rangle, X) \to (\langle A, 1, 1\rangle, \langle Ba\rangle', A_1)$,
   (c) i. $(\langle A, 1, i\rangle, A_i) \to (\langle A, 2, i\rangle, \hat{A}_i)$,
   ii. $(\langle A, 2, i\rangle, F', \hat{A}_i) \to (\langle A, 3, i\rangle, F', A_i)$,
   iii. $(\langle A, 3, j\rangle, E, A_j) \to (\langle A, 1, j+1\rangle, E, A_{j+1})$,
   (d) i. $(\langle A, 3, n\rangle, B', E, A_n) \to (\langle C, 0\rangle, D, E, X)$,
   ii. $(\langle A, 3, n\rangle, \langle Ba\rangle', A_n) \to (\langle C, 0\rangle, \langle Da\rangle, X)$ to $\bar{P}$;

6. For each $a, b, c \in T$, add
   (a) $(\langle ab, 0\rangle) \to (\langle ab, 4\rangle)$,
   (b) $(\bar{c}, \langle ab, 4\rangle) \to (c, \langle ab, 4\rangle)$,
   (c) $(\langle ab, 4\rangle, X) \to (a, b)$ to $\bar{P}$.

Basic Idea. In short, productions introduced in (1) initiate the derivation, productions from (2) are used to select the nonterminal to be rewritten, productions from (3), (4), and (5) simulate G's productions of the form $A \to a$, $A \to BC$, and $AB \to CD$, respectively, and, finally, productions from (6) finish the derivation.

In greater detail, consider a sentential form $a_1 \dots a_m A B a_{m+1} \dots a_k$ of G and G's production $AB \to CD$. This sentential form can be expressed as

$$b_1 \dots b_{r-1} \langle a_r, 0 \rangle b_{r+1} \dots b_m A B b_{m+1} \dots b_{k-2} \langle a_{k-1} a_k \rangle X$$

in $\bar{G}$, where each $b_j = \alpha(a_j)$. First, to simulate the application of $AB \to CD$ in $\bar{G}$, productions from (2) have to be used to select the nonterminal A in the sentential form so that

$$b_1 \dots b_m \langle A, 0 \rangle B b_{m+1} \dots b_{k-2} \langle a_{k-1} a_k \rangle X.$$

By a production from (5bi), the second nonterminal to be rewritten, B, is selected. It needs to be verified, however, whether there is no symbol between A and B in the sentential form. This verification is performed for each symbol A_i which may appear in $\{b_{m+1}, \dots, b_{k-2}\}$. By a production from (5ci), the first A_i following A is tagged and by a production from (5cii), the first tagged A_i following B is untagged. Therefore, if there is some A_i between A and B, no production from (5cii) can be used and the derivation is blocked. To ensure that there is at least one A_i behind B, A_i is inserted at the end of the sentential form by a production from (5ciii) before the verification. Finally, after it has been verified that there is no symbol between A and B, a production from (5di) rewrites AB to CD, and the simulation of other productions may continue. As the simulation of productions of the form $A \to BC$ and $A \to a$ is much simpler than the above case, we do not describe it in greater detail. Notice that the simulation can be performed also in the penultimate nonterminal of the sentential form. This is allowed by productions from (2c), (2d), (3b), (3c), (4b), (4c), (5a), (5bii), and (5dii).

Finally, when the sentential form has the form $\bar{a}_1 \dots \bar{a}_{k-2} \langle a_{k-1} a_k, 0 \rangle X$, where all $a_j \in T$, the derivation enters the final phase by a production from (6a) in which each $\bar{a}_j$ is replaced with a_j by a production from (6b). Ultimately, a production from (6c) replaces the nonterminal X with a_k, so we obtain the G's sentence $a_1 \dots a_k$.

Formal Proof. Every derivation starts either by a production introduced in (1a) to generate sentences $a \in L(G)$, where $a \in T$, or by a production introduced in (1b) to generate sentences $x \in L(G)$, $|x| \geq 2$. As $\bar{S}$ does not occur on a right-hand side of any production, productions from (1) are not used during the rest of the derivation.

Consider G's sentential form $a_1 \dots a_k$, where $a_1, \dots, a_k \in V$, for some $k \geq 2$. In $\bar{G}$, this sentential form corresponds either to

$$b_1 \dots b_{r-1} \langle a_r, 0 \rangle b_{r+1} \dots b_{k-2} \langle a_{k-1} a_k \rangle X,$$

where $b_i = \alpha(a_i)$ for all $i \in \{1, \dots, r-1, r+1, \dots, k-2\}$, for some $1 \leq r \leq k-2$, or to

$$b_1 \dots b_{k-2} \langle a_{k-1} a_k, 0 \rangle X,$$

where $b_i = \alpha(a_i)$ for all $1 \leq i \leq k-2$ (observe that every right-hand side of a production from (1b) represents a sentential form of this kind). To simulate any G's production, the leftmost nonterminal from its left-hand side has to be selected in the sentential form of $\bar{G}$. This is done by appending 0 to the symbol to be selected by productions from (2). Specifically, for a symbol $a \in V$, (2a) selects the leftmost symbol a immediately following the currently selected symbol and (2b) selects the leftmost symbol a preceding the currently selected symbol. Productions from (2c) and (2d) are used to select and unselect the penultimate nonterminal in $\bar{G}$'s sentential form which is composed of two symbols from V. Observe that in this way, any symbol (except for the final X) in every sentential form of $\bar{G}$ can be selected. Further, observe that during a derivation, always one symbol is selected.

After the required nonterminal is selected, the use of G's production can be simulated. Productions of the form $A \to a$ are simulated by (3a) for every selected nonterminal $a_1, \dots, a_{k-2}$ and

by (3b), (3c) if the penultimate nonterminal (which contains a_{k-1}, a_k) of $\bar{G}$'s sentential form is selected. Analogously, productions of the form $A \to BC$ are simulated by productions from (4), and (5a) is used to simulate the use of $AB \to CD$ inside the penultimate nonterminal.

In what follows, we demonstrate how an application of a production of the form $AB \to CD$ within $a_1 \dots a_{k-2}$ is simulated by $\bar{G}$. Suppose that the sentential form in $\bar{G}$ is of the form

$$b_1 \dots b_{r-1} \langle a_r, 0 \rangle b_{r+1} \dots b_{k-2} \langle a_{k-1} a_k \rangle X$$

and we simulate the production $a_r a_{r+1} \to c_r c_{r+1} \in P$. Recall that $N_1 = \{A_1, \dots, A_n\}$ denotes the set of all symbols which may appear in the set $\{b_{r+1}, \dots, b_{k-2}\}$. First, to select b_{r+1}, the production

$$(\langle a_r, 0 \rangle, b_{r+1}, X) \to (\langle a_r, 1, 1 \rangle, b'_{r+1}, A_1)$$

from (5bi) is applied in a successful derivation, so

$$\begin{aligned} & b_1 \dots b_{r-1} \langle a_r, 0 \rangle b_{r+1} \dots b_{k-2} \langle a_{k-1} a_k \rangle X \\ \underset{\text{lm}}{\Rightarrow}_{\bar{G}} \; & b_1 \dots b_{r-1} \langle a_r, 1, 1 \rangle b'_{r+1} b_{r+2} \dots b_{k-2} \langle a_{k-1} a_k \rangle A_1. \end{aligned}$$

Observe that if b_{r+1} does not immediately follow $\langle a_r, 0 \rangle$, the leftmost $b \in \{b_{r+1}, \dots, b_{k-2}\}$ satisfying $b = b_{r+1}$ can be selected by a production from (5bi). The purpose of productions from (5c) is to verify that the nonterminal immediately following $\langle a_r, 0 \rangle$ has been selected. First, the production

$$(\langle a_r, 1, 1 \rangle, A_1) \to (\langle a_r, 2, 1 \rangle, \hat{A}_1)$$

from (5ci) is applied to tag the first A_1 following $\langle a_r, 1, 1 \rangle$, so

$$\begin{aligned} & b_1 \dots b_{r-1} \langle a_r, 1, 1 \rangle b'_{r+1} b_{r+2} \dots b_{k-2} \langle a_{k-1} a_k \rangle A_1 \\ \underset{\text{lm}}{\Rightarrow}_{\bar{G}} \; & b_1 \dots b_{r-1} \langle a_r, 2, 1 \rangle b'_{r+1} y_1 \langle a_{k-1} a_k \rangle d_1, \end{aligned}$$

where either

$$y_1 = b_{r+2} \dots b_{m-1} \hat{b}_m b_{m+1} \dots b_{k-2},$$

$\hat{b}_m = \hat{A}_1$, $d_1 = A_1$, for some $1 \leq m \leq k-2$, or $y_1 = b_{r+2} \dots b_{k-2}$, $d_1 = \hat{A}_1$. Then, the production

$$(\langle a_r, 2, 1 \rangle, b'_{r+1}, \hat{A}_1) \to (\langle a_r, 3, 1 \rangle, b'_{r+1}, A_1)$$

from (5cii) is applied to untag the first symbol $\hat{A}_1$ following b'_{r+1}, so

$$\begin{aligned} & b_1 \dots b_{r-1} \langle a_r, 2, 1 \rangle b'_{r+1} y_1 \langle a_{k-1} a_k \rangle d_1, \\ \underset{\text{lm}}{\Rightarrow}_{\bar{G}} \; & b_1 \dots b_{r-1} \langle a_r, 3, 1 \rangle b'_{r+1} b_{r+2} \dots b_{k-2} \langle a_{k-1} a_k \rangle A_1. \end{aligned}$$

This means that if A_1 occurs between $\langle a_r, 2, 1 \rangle$ and b'_{r+1}, it is tagged by the production from (5ci) but it cannot be untagged by any production from (5cii) so the derivation is blocked. Finally, the production

$$(\langle a_r, 3, 1 \rangle, \langle a_{k-1} a_k \rangle, A_1) \to (\langle a_r, 1, 2 \rangle, \langle a_{k-1} a_k \rangle, A_2)$$

from (5ciii) is applied,

$$\begin{aligned} & b_1 \dots b_{r-1} \langle a_r, 3, 1 \rangle b'_{r+1} b_{r+2} \dots b_{k-2} \langle a_{k-1} a_k \rangle A_1 \\ \underset{\text{lm}}{\Rightarrow}_{\bar{G}} \; & b_1 \dots b_{r-1} \langle a_r, 1, 2 \rangle b'_{r+1} b_{r+2} \dots b_{k-2} \langle a_{k-1} a_k \rangle A_2, \end{aligned}$$

and the same verification continues for A_2. This verification proceeds for all symbols from the set $\{A_1,\ldots,A_n\}$ so this part of derivation can be expressed as

$$\begin{aligned} & u_1\ [p_{11}]\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} v_1\ [p_{12}]\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} w_1\ [p_{13}] \\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} & u_2\ [p_{21}]\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} v_2\ [p_{22}]\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} w_2\ [p_{23}] \\ & \vdots \\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} & u_n\ [p_{n1}]\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} v_n\ [p_{n2}] \end{aligned}$$

with

$$\begin{aligned} u_i &= b_1\ldots b_{r-1}\langle a_r,1,i\rangle b'_{r+1}b_{r+2}\ldots b_{k-2}\langle a_{k-1}a_k\rangle A_i, \\ v_i &= b_1\ldots b_{r-1}\langle a_r,2,i\rangle b'_{r+1}y_i\langle a_{k-1}a_k\rangle d_i, \\ w_j &= b_1\ldots b_{r-1}\langle a_r,3,j\rangle b'_{r+1}b_{r+2}\ldots b_{k-2}\langle a_{k-1}a_k\rangle A_j, \end{aligned}$$

p_{i1}, p_{i2}, and p_{j3} are productions from (5ci), (5cii), and (5ciii), respectively, for all $1\leq i\leq n$, $1\leq j\leq n-1$, and either

$$y_i = b_{r+2}\ldots b_{i_{m-1}}\hat{b}_{i_m}b_{i_{m+1}}\ldots b_{k-2},$$

$\hat{b}_{i_m}=\hat{A}_{i_m}$, $d_i=A_i$, for some $1\leq i_m\leq k-2$, or $y_i=b_{r+2}\ldots b_{k-2}$, $d_i=\hat{A}_i$. After the verification, the application of $a_ra_{r+1}\rightarrow c_rc_{r+1}\in P$ is simulated by

$$(\langle a_r,3,n\rangle,b'_{r+1},\langle a_{k-1}a_k\rangle,A_n)\rightarrow(\langle c_r,0\rangle,c_{r+1},\langle a_{k-1}a_k\rangle,X)$$

from (5ciii), so

$$\begin{aligned} & b_1\ldots b_{r-1}\langle a_r,3,n\rangle b'_{r+1}b_{r+2}\ldots b_{k-2}\langle a_{k-1}a_k\rangle A_n \\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} & b_1\ldots b_{r-1}\langle c_r,0\rangle c_{r+1}b_{r+2}\ldots b_{k-2}\langle a_{k-1}a_k\rangle X. \end{aligned}$$

Observe that in order to simulate a production of the form $AB\rightarrow CD$ within $a_{k-2}a_{k-1}$, productions from (5bii) and (5dii) have to be used instead of productions from (5bi) and (5di) in the simulation described above. The details are left to the reader.

Finally, consider a G's sentence $a_1\ldots a_k\in T^+$. This corresponds to

$$\bar{a}_1\ldots\bar{a}_{r-1}\langle a_r,0\rangle\bar{a}_{r+1}\ldots\bar{a}_{k-2}\langle a_{k-1}a_k\rangle X$$

in $\bar{G}$ after finishing the simulation. To enter the final phase in $\bar{G}$, $\langle a_{k-1}a_k\rangle$ has to be selected by a production from (2c), so we obtain

$$\bar{a}_1\ldots\bar{a}_{k-2}\langle a_{k-1}a_k,0\rangle X.$$

The rest of the derivation can be expressed as

$$\begin{aligned} & \bar{a}_1\ldots\bar{a}_{k-2}\langle a_{k-1}a_k,0\rangle X \\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} & \bar{a}_1\ldots\bar{a}_{k-2}\langle a_{k-1}a_k,4\rangle X\ [p_{6a}] \\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}}^{k-2} & a_1\ldots a_{k-2}\langle a_{k-1}a_k,4\rangle X\ [\Xi_{6b}] \\ {}_{\mathrm{lm}}\!\Rightarrow_{\bar{G}} & a_1\ldots a_{k-2}a_{k-1}a_k\ [p_{6c}], \end{aligned}$$

where p_{6a} and p_{6c} are productions introduced in steps (6a) and (6c), respectively, and Ξ_{6b} is a sequence of $k-2$ productions from (6b). As a result, every $x\in L(\bar{G},\mathrm{lm})$ if and only if $x\in L(G)$, so the theorem holds. ■

Next, we state the following corollary.

Corollary 17. $\mathscr{L}(PSC,\mathrm{rm})=\mathscr{L}(CS)$.

Proof. This corollary can be proved by a straightforward modification of the proof of Theorem 26 and is, therefore, left to the reader. ■

5.4 Maximal and Minimal Rewriting

In this section we introduce two natural modifications of propagating scattered context grammars to be able to describe all context-sensitive languages. As a matter of fact, these simple modifications only change the way propagating scattered context grammars perform their derivations while keeping their grammatical concept unchanged. More specifically, this modification requires that during every derivation step, a production containing the maximal or the minimal number of nonterminals on its left-hand side is chosen from the set of all applicable productions. This kind of modification is different from the other previously introduced modifications; while the other modifications restrict the way in which the context-free components of a scattered context production operate, maximal and minimal derivations prescribe the way a scattered context production is applied as a whole. We demonstrate that these grammars characterize the family of context-sensitive languages if they work in this modified way.

Formally, we define the two new kinds of derivations next.

Definition 32. Let $G = (V,T,P,S)$ be a scattered context grammar. Define the *maximal direct derivation* as

$$u \ {}_{\max}\Rightarrow_G v \ [p]$$

if and only if $u \Rightarrow_G v \ [p]$ and there is no $r \in P$ satisfying $\mathrm{len}(r) > \mathrm{len}(p)$ such that $u \Rightarrow_G w \ [r]$. Similarly, define the *minimal direct derivation* as

$$u \ {}_{\min}\Rightarrow_G v \ [p]$$

if and only if $u \Rightarrow_G v \ [p]$ and there is no $r \in P$ satisfying $\mathrm{len}(r) < \mathrm{len}(p)$ such that $u \Rightarrow_G w \ [r]$. Define the transitive closure and the reflexive and transitive closure of maximal and minimal direct derivations in the standard way. The *language of a scattered context grammar G which uses maximal* and *minimal derivations* is denoted by $L(G,\max)$ and $L(G,\min)$ and defined as

$$L(G,\max) = \{x \in T^* : S \ {}_{\max}\Rightarrow^*_G x\}$$

and

$$L(G,\min) = \{x \in T^* : S \ {}_{\min}\Rightarrow^*_G x\},$$

respectively. The corresponding language families of propagating scattered context grammars are denoted by $\mathscr{L}(PSC,\max)$ and $\mathscr{L}(PSC,\min)$.

Next, we demonstrate that propagating scattered context grammars which use maximal and minimal derivations characterize the family of context-sensitive languages.

Theorem 27. $\mathscr{L}(CS) = \mathscr{L}(PSC,\max)$.

Proof. Let L be a context-sensitive language. As by Theorem 7 state grammars characterize the family of context-sensitive languages, we suppose that L is described by a state grammar $\bar{G} = (\bar{V},T,K,\bar{P},\bar{S},p_0)$. Set

$$Y = \{\langle A,q\rangle : A \in \bar{V} - T, q \in K\}$$

and $Z = \{\bar{a} : a \in T\}$. Define the homomorphism α form $\bar{V}^*$ to $((\bar{V}-T)\cup Z)^*$ as $\alpha(A) = A$ for all $A \in \bar{V} - T$ and $\alpha(a) = \bar{a}$ for all $a \in T$. Set $V = \bar{V} \cup Y \cup Z \cup \{S,X\}$. Define the propagating scattered context grammar G as

$$G = (V,T,P,S),$$

where P is constructed as follows:

1. For each $x \in L(\bar{G})$, where $|x| \leq 2$, add
   $(S) \to (x)$ to P;

2. For each

$$(x,q) \in \{(x,q) : (\bar{S},p_0) \Rightarrow_{\bar{G}}^{+} (x,q) \text{ for some } q \in K \\ \text{and } 3 \leq |x| \leq \min(\{3, \max(\{|y| : (B,p) \to (y,p') \in \bar{P}\})\})\},$$

   where

   (a) $x \in T^*$, add
   $(S) \to (x)$ to P;

   (b) $x = x_1 A x_2$, $A \in \bar{V} - T$, $x_1, x_2 \in \bar{V}^*$, add
   $(S) \to (\alpha(x_1)\langle A,q\rangle\alpha(x_2))$ to P;

3. For each $(A,p) \to (x,q)$, $(B,p) \to (y,r) \in \bar{P}$, $C \in \bar{V}$, $\Gamma_{21} \in \text{perm}(2,1)$,

$$z = \text{reorder}((B, \langle A,p\rangle, \alpha(C)), \Gamma_{21}),$$

   add
   $z \to (X,X,X)$ to P;

4. For each $(A,p) \to (x,q) \in \bar{P}$, $B \in \bar{V} - T$, $C \in \bar{V}$, $\Gamma_{11} \in \text{perm}(1,1)$,

$$y = \text{reorder}((\langle A,p\rangle, \alpha(C)), \Gamma_{11}),$$

   add

   (a) $(B, \langle A,p\rangle) \to (\langle B,q\rangle, \alpha(x))$,

   (b) $(\langle A,p\rangle, B) \to (\alpha(x), \langle B,q\rangle)$ to P;

   (c) If $x = vBw$, $v,w \in \bar{V}^*$, for each

$$z = \text{reorder}((\alpha(v)\langle B,q\rangle\alpha(w), \alpha(C)), \Gamma_{11}),$$

   add
   $y \to z$ to P;

   (d) For each

$$u = \text{reorder}((\alpha(x), \alpha(C)), \Gamma_{11}),$$

   add
   $y \to u$ to P;

5. For each $a \in T$, add
   $(\bar{a}) \to (a)$ to P.

Basic Idea. The state grammar $\bar{G}$ is simulated by the propagating scattered context grammar G which performs maximal derivations. Productions from (1) are used to generate a sentence $w \in L(\bar{G})$, where $|w| \leq 2$, while the productions introduced in (2) start the simulation of the derivation of a $\bar{G}$'s sentence, w, $|w| \geq 3$. Let $(A,p) \to (x,q)$ be a production of $\bar{G}$ which is applicable to a sentential form $(w_1 A w_2, p)$ generated by $\bar{G}$. The sentential form $(w_1 A w_2, p)$ in $\bar{G}$ corresponds to the sentential form $\alpha(w_1)\langle A,p\rangle\alpha(w_2)$ in G. To simulate the application of $(A,p) \to (x,q)$ in G, it is checked first, whether the production is applied to the leftmost nonterminal of the sentential form

for the given state p. If not, some production from (3) is applicable. This production is applied because it has the highest priority of all productions, and its application introduces the symbol X to the sentential form, which blocks the derivation. The successful derivation proceeds by a production from (4a), (4b), and (4c) which nondeterministically selects the following nonterminal to be rewritten and appends the new state to it. The production which finishes the derivation of a sentence in $\bar{G}$ is simulated by a production from (4d) which removes the compound nonterminal $\langle \ldots \rangle$ from the sentential form. Finally, each symbol $\bar{a}, a \in T$ is rewritten to a.

Formal Proof.

Claim 6. Each $x \in L(\bar{G})$, where $|x| \leq 2$ is generated by G as follows:

$$S \Rightarrow_{G\,\max} x\ [p_1],$$

where p_1 is one of the productions introduced in step (1) of the construction. □

Claim 7. Every

$$(\bar{S}, p_0) \Rightarrow_{\bar{G}}^{+} (x, q),$$

where $q \in K$, $x \in T^+$,

$$3 \leq |x| \leq \min(\{3, \max(\{|y| : (B,p) \rightarrow (y,p') \in \bar{P}\})\})$$

is generated by G as follows:

$$S \Rightarrow_{G\,\max} x\ [p_{2a}],$$

where p_{2a} is one of the productions introduced in step (2a) of the construction. □

Claim 8. Every

$$(\bar{S}, p_0) \Rightarrow_{\bar{G}}^{+} (x, q) \Rightarrow_{\bar{G}}^{+} (u, r),$$

where $q, r \in K$, $u \in T^+$, $x = v_0 A w_0$, $A \in \bar{V} - T$, $v_0, w_0 \in \bar{V}^*$,

$$3 \leq |x| \leq \min(\{3, \max(\{|y| : (B,p) \rightarrow (y,p') \in \bar{P}\})\}),$$

can only be generated by G as follows:

$$\begin{array}{lll} S \Rightarrow_{G\,\max} & \alpha(v_0)\langle A, q\rangle\alpha(w_0) & [p_{2b}] \\ \Rightarrow_{G\,\max}^{*} & y & [\Xi_4] \\ \Rightarrow_{G\,\max} & z & [p_{4d}] \\ \Rightarrow_{G\,\max}^{|u|} & u & [\Xi_5], \end{array}$$

where $y \in Z^*YZ^*$, $z = \alpha(u)$; p_{2b} and p_{4d} denote one of the productions introduced in steps (2b) and (4d), respectively, and Ξ_4 and Ξ_5 are sequences of productions introduced in steps (4a), (4b), (4c), and (5), respectively.

Proof. Observe that the productions from (1) and (2) are the only productions containing S on their left-hand sides and no other productions contain S on their right-hand sides. To generate a sentence u, $|u| \geq 3$, the derivation has to start with

$$S \Rightarrow_{G\,\max} \alpha(v_0)\langle A, q\rangle\alpha(w_0)\ [p_{2b}],$$

and productions from (1) and (2) are not used during the rest of the derivation.

Further observe that none of the productions introduced in (3) can be applied during a successful derivation as no productions rewrite the nonterminal X which is contained on the right-hand side of each production from step (3).

To generate a sentence over T, all symbols from $\bar{V} - T$ have to be removed from the sentential form. Only productions from step (4) can be used for their replacement as they contain symbols from $\bar{V} - T$ on their left-hand sides. Further, productions (4a), (4b), (4c) contain one symbol from Y both on their left and their right-hand sides, while productions from (4d) contain a symbol form Y only on their left-hand sides. Therefore, after the application of a production from (4d), none of the productions from step (4) is applicable. Because for each production p_4 and p_5 introduced in step (4) and (5), respectively, it holds that $\mathrm{len}(p_4) > \mathrm{len}(p_5)$, no production from step (5) is applied while some production from step (4) is applicable. As a result, the corresponding part of the derivation looks as follows:

$$\begin{aligned}\alpha(v_0)\langle A, q\rangle\alpha(w_0) \ &{}_{\max}\Rightarrow_G^* y\ [\Xi_4] \\ &{}_{\max}\Rightarrow_G z\ [p_{4d}].\end{aligned}$$

At this point, $z = \alpha(u)$ in a successful derivation. Productions from step (5) replace each $\bar{a} \in \mathrm{alph}(z)$ with a in $|u|$ steps, so we obtain

$$z\ {}_{\max}\Rightarrow_G^{|u|}\ u\ [\Xi_5].$$

Putting together the previous observations, we obtain the formulation of Claim 8, so the claim holds. □

Claim 9. In a successful derivation, every

$$\begin{aligned}&\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0) \\ {}_{\max}\Rightarrow_G\ &\alpha(v_1)\langle B_1, q_1\rangle\alpha(w_1)\ [p_0] \\ &\vdots \\ {}_{\max}\Rightarrow_G\ &\alpha(v_n)\langle B_n, q_n\rangle\alpha(w_n)\ [p_{n-1}]\end{aligned}$$

is performed in G if and only if

$$\begin{aligned}&(v_0B_0w_0, q_0) \\ \Rightarrow_{\bar{G}}\ &(v_1B_1w_1, q_1)\ [(B_0, q_0) \to (x_1, q_1)] \\ &\vdots \\ \Rightarrow_{\bar{G}}\ &(v_nB_nw_n, q_n)\ [(B_{n-1}, q_{n-1}) \to (x_n, q_n)]\end{aligned}$$

is performed in $\bar{G}$, where $v_i, w_i \in \bar{V}^*$, $B_i \in \bar{V} - T$, $q_i \in K$ for all $0 \leq i \leq n$, for some $n \geq 0$, $x_1, \ldots, x_n \in \bar{V}^+$, and $p_0, \ldots, p_{n-1}$ are productions introduced in steps (4a), (4b), and (4c).

Proof.

Only If. We show that

$$\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0)\ {}_{\max}\Rightarrow_G^m\ \alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m)$$

implies

$$(v_0B_0w_0, q_0) \Rightarrow_{\bar{G}}^m (v_mB_mw_m, q_m)$$

by induction on m.

Basis. Let $m = 0$. Then,

$$\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0) \underset{\max}{\Rightarrow}{}^{0}_{G} \alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0)$$

and, clearly,

$$(v_0 B_0 w_0, q_0) \Rightarrow^{0}_{\bar{G}} (v_0 B_0 w_0, q_0).$$

Induction Hypothesis. Suppose that the claim holds for all k-step derivations, where $k \leq m$, for some $m \geq 0$.

Induction Step. Let us consider a derivation

$$\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0) \underset{\max}{\Rightarrow}{}^{m+1}_{G} \alpha(v_{m+1})\langle B_{m+1}, q_{m+1}\rangle\alpha(w_{m+1}).$$

Since $m+1 \geq 1$, there is some

$$\alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m) \in ((\bar{V} - T) \cup Z)^* Y ((\bar{V} - T) \cup Z)^*$$

and a production p_m such that

$$\begin{aligned}\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0) &\underset{\max}{\Rightarrow}{}^{m}_{G} \alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m) \\ &\underset{\max}{\Rightarrow}_{G} \alpha(v_{m+1})\langle B_{m+1}, q_{m+1}\rangle\alpha(w_{m+1})\ [p_m].\end{aligned}$$

By the induction hypothesis, there is a derivation

$$(v_0 B_0 w_0, q_0) \Rightarrow^{m}_{\bar{G}} (v_m B_m w_m, q_m).$$

The production p_m is one of the productions introduced in steps (4a) through (4c) and may be of the following three forms, depending on the placement of B_{m+1}:

- $(B_{m+1}, \langle B_m, q_m\rangle) \to (\langle B_{m+1}, q_{m+1}\rangle, \alpha(x_{m+1}))$ for $v_m = v'_m B_{m+1} v''_m$,
- $(\langle B_m, q_m\rangle, B_{m+1}) \to (\alpha(x_{m+1}), \langle B_{m+1}, q_{m+1}\rangle)$ for $w_m = w'_m B_{m+1} w''_m$,
- $(\langle B_m, q_m\rangle, \alpha(A)) \to (\alpha(x'_{m+1})\langle B_{m+1}, q_{m+1}\rangle\alpha(x''_{m+1}), \alpha(A))$ or
  $(\alpha(A), \langle B_m, q_m\rangle) \to (\alpha(A), \alpha(x'_{m+1})\langle B_{m+1}, q_{m+1}\rangle\alpha(x''_{m+1}))$ for
  $x_{m+1} = x'_{m+1} B_{m+1} x''_{m+1}$,

where $A \in \bar{V}$ and $x_{m+1}, x'_{m+1}, x''_{m+1} \in \bar{V}^*$. Their construction is based on $\bar{P}$, so there is a production $(B_m, q_m) \to (x_{m+1}, q_{m+1}) \in \bar{P}$.

As we simulate G's derivation by $\bar{G}$, we have to demonstrate that for the given state q_m, the leftmost nonterminal in the sentential form is rewritten in G. We prove this by contradiction. Suppose that there is a production $p'_m \in P$ from step (4) which rewrites some $B'_m \in \bar{V} - T$ in a state q_m, and $B'_m \in \mathrm{alph}(v_m)$. Then there exists $(B'_m, q_m) \to (x'_{m+1}, q'_{m+1}) \in \bar{P}$ and, as a result, there also exist productions from (3) which are based on $(B_m, q_m) \to (x_{m+1}, q_{m+1})$ and $(B'_m, q_m) \to (x'_{m+1}, q'_{m+1})$. These productions have the following forms:

- $(B'_m, \langle B_m, q_m\rangle, \alpha(A)) \to (X, X, X)$,
- $(B'_m, \alpha(A), \langle B_m, q_m\rangle) \to (X, X, X)$,
- $(\alpha(A), B'_m, \langle B_m, q_m\rangle) \to (X, X, X)$,

where $A \in \bar{V}$. Because $|\alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m)| \geq 3$, one of these productions is applicable. As productions introduced in step (3) have higher precedence than productions introduced in step (4), one of them is applied, which introduces X to the sentential form. This symbol, however, can never be removed from the sentential form, so the derivation is not successful.

As a result, the leftmost nonterminal for a state q_m is rewritten in G, so $(B_m, q_m) \to (x_{m+1}, q_{m+1})$ is used in $\bar{G}$ and we obtain

$$(v_m B_m w_m, q_m) \Rightarrow_{\bar{G}} (v_{m+1} B_{m+1} w_{m+1}, q_{m+1}) \; [(B_m, q_m) \to (x_{m+1}, q_{m+1})].$$

If. We demonstrate that

$$(v_0 B_0 w_0, q_0) \Rightarrow_{\bar{G}}^{m} (v_m B_m w_m, q_m)$$

implies

$$\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0) \; {}_{\max}\Rightarrow_{G}^{m} \alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m)$$

by induction on m.

Basis. Let $m = 0$. Then

$$(v_0 B_0 w_0, q_0) \Rightarrow_{\bar{G}}^{0} (v_0 B_0 w_0, q_0).$$

Clearly,

$$\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0) \; {}_{\max}\Rightarrow_{G}^{0} \alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0).$$

Induction Hypothesis. Suppose that the claim holds for all k-step derivations, where $k \leq m$, for some $m \geq 0$.

Induction Step. Consider a derivation

$$(v_0 B_0 w_0, q_0) \Rightarrow_{\bar{G}}^{m+1} (v_{m+1} B_{m+1} w_{m+1}, q_{m+1}).$$

Since $m + 1 \geq 1$, there is some $(v_m B_m w_m, q_m)$, where $v_m, w_m \in \bar{V}^*$, $B_m \in \bar{V} - T$, and a production $(B_m, q_m) \to (x_{m+1}, q_{m+1})$ such that

$$\begin{aligned}(v_0 B_0 w_0, q_0) &\Rightarrow_{\bar{G}}^{m} (v_m B_m w_m, q_m) \\ &\Rightarrow_{\bar{G}} (v_{m+1} B_{m+1} w_{m+1}, q_{m+1}) \; [(B_m, q_m) \to (x_{m+1}, q_{m+1})].\end{aligned}$$

By the induction hypothesis, there is a derivation

$$\alpha(v_0)\langle B_0, q_0\rangle\alpha(w_0) \; {}_{\max}\Rightarrow_{G}^{m} \alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m).$$

Because $(B_m, q_m) \to (x_{m+1}, q_{m+1})$ rewrites the leftmost rewritable symbol B_m for a given state q_m, there is no production $(B'_m, q_m) \to (x'_{m+1}, q'_{m+1})$ satisfying $B'_m \in \text{alph}(v_m)$. As a result, none of the productions from step (3) is applicable.

For each $(B_m, q_m) \to (x_{m+1}, q_{m+1}) \in \bar{P}$, there are productions of the following three forms in G whose use depends on the placement of B_{m+1}:

1. $(B_{m+1}, \langle B_m, q_m\rangle) \to (\langle B_{m+1}, q_{m+1}\rangle, \alpha(x_{m+1}))$ for $v_m = v'_m B_{m+1} v''_m$,
2. $(\langle B_m, q_m\rangle, B_{m+1}) \to (\alpha(x_{m+1}), \langle B_{m+1}, q_{m+1}\rangle)$ for $w_m = w'_m B_{m+1} w''_m$,
3. $(\langle B_m, q_m\rangle, \alpha(A)) \to (\alpha(x'_{m+1})\langle B_{m+1}, q_{m+1}\rangle\alpha(x''_{m+1}), \alpha(A))$ or
   $(\alpha(A), \langle B_m, q_m\rangle) \to (\alpha(A), \alpha(x'_{m+1})\langle B_{m+1}, q_{m+1}\rangle\alpha(x''_{m+1}))$ for
   $x_{m+1} = x'_{m+1} B_{m+1} x''_{m+1}$,

where $A \in \bar{V}$ and $x_{m+1}, x'_{m+1}, x''_{m+1} \in \bar{V}^*$. As $|\alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m)| \geq 3$, one of them is applicable in G, so we obtain

$$\alpha(v_m)\langle B_m, q_m\rangle\alpha(w_m) \ {}_{\max}\Rightarrow_G \ \alpha(v_{m+1})\langle B_{m+1}, q_{m+1}\rangle\alpha(w_{m+1}).$$

□

By Claims 6 through 9 it follows that $\mathscr{L}(CS) \subseteq \mathscr{L}(PSC, \max)$. As propagating scattered context grammars do not contain ε-productions, their derivations can be simulated by linear bounded automata. As a result, $\mathscr{L}(PSC, \max) \subseteq \mathscr{L}(CS)$. Therefore, $\mathscr{L}(CS) = \mathscr{L}(PSC, \max)$. ■

Theorem 28. $\mathscr{L}(CS) = \mathscr{L}(PSC, \min)$.

Proof. Let L be a context-sensitive language described by a state grammar, $\bar{G} = (\bar{V}, T, K, \bar{P}, \bar{S}, p_0)$. Set

$$Y = \{\langle A, q\rangle : A \in \bar{V} - T, q \in K\},$$

and $Z = \{\bar{a} : a \in T\}$. Define the homomorphism α form $\bar{V}^*$ to $((\bar{V} - T) \cup Z)^*$ as $\alpha(A) = A$ for all $A \in \bar{V} - T$ and $\alpha(a) = \bar{a}$ for all $a \in T$. Set $V = \bar{V} \cup Y \cup Z \cup \{S, X\}$. Define the propagating scattered context grammar G' as

$$G' = (V, T, P', S),$$

where P' is constructed as follows:

1. For each $x \in L(\bar{G})$, where $|x| \leq 3$, add
   $(S) \to (x)$ to P';

2. For each

$$(x, q) \in \{(x, q) : (\bar{S}, p_0) \Rightarrow^+_{\bar{G}} (x, q) \text{ for some } q \in K \\ \text{and } 4 \leq |x| \leq \min(\{4, \max(\{|y| : (B, p) \to (y, p') \in \bar{P}\})\})\},$$

   where

   (a) $x \in T^*$, add
   $(S) \to (x)$ to P';

   (b) $x = x_1 A x_2$, $A \in \bar{V} - T$, $x_1, x_2 \in \bar{V}^*$, add
   $(S) \to (\alpha(x_1)\langle A, q\rangle\alpha(x_2))$ to P';

3. For each $(A, p) \to (x, q)$, $(B, p) \to (y, r) \in \bar{P}$, add
   $(B, \langle A, p\rangle) \to (X, X)$ to P';

4. For each $(A, p) \to (x, q) \in \bar{P}$, $B \in \bar{V} - T$, $D, E \in \bar{V}$, $\Gamma_{21} \in \text{perm}(2, 1)$, $\Gamma_{12} \in \text{perm}(1, 2)$,

$$u = \text{reorder}((B, \langle A, p\rangle, \alpha(D)), \Gamma_{21}), \ u' = \text{reorder}((\langle B, q\rangle, \alpha(x), \alpha(D)), \Gamma_{21}),$$
$$r = \text{reorder}((\langle A, p\rangle, B, \alpha(D)), \Gamma_{21}), \ r' = \text{reorder}((\alpha(x), \langle B, q\rangle, \alpha(D)), \Gamma_{21}),$$

$$y = \text{reorder}((\langle A, p\rangle, \alpha(D), \alpha(E)), \Gamma_{12}),$$

   add

   (a) $u \to u'$,

   (b) $r \to r'$ to P';

(c) If $x = vBw$, $v, w \in \bar{V}^*$, for each

$$z = \text{reorder}((\alpha(v)\langle B, q\rangle\alpha(w), \alpha(D), \alpha(E)), \Gamma_{12}),$$

add
$y \to z$ to P';

(d) For each

$$u = \text{reorder}((\alpha(x), \alpha(D), \alpha(E)), \Gamma_{12}),$$

add
$y \to u$ to P';

5. For each $a, b, c, d \in T$, add

(a) $(\bar{a}, \bar{b}, \bar{c}, \bar{d}) \to (a, \bar{b}, \bar{c}, \bar{d})$,

(b) $(\bar{a}, \bar{b}, \bar{c}, \bar{d}) \to (a, b, c, d)$ to P'.

Claim 10. Every

$$(\bar{S}, p_0) \Rightarrow_{\bar{G}}^{+} (x, q) \Rightarrow_{\bar{G}}^{+} (u, r),$$

where $q, r \in K$, $u \in T^+$, $x = v_0 A w_0$, $A \in \bar{V} - T$, $v_0, w_0 \in \bar{V}^*$,

$$4 \leq |x| \leq \min(\{4, \max(\{|y| : (B, p) \to (y, p') \in \bar{P}\})\}),$$

can only be generated by G' as follows:

$$\begin{array}{lll} S & \alpha(v_0)\langle A, q\rangle\alpha(w_0) & [p_{2b}] \\ \min\Rightarrow_G & y & [\Xi_4] \\ \min\Rightarrow_G^* & z & [p_{4d}] \\ \min\Rightarrow_G^{|u|-4} & v & [\Xi_5] \\ \min\Rightarrow_G & u & [p_{5b}], \end{array}$$

where $y \in Z^*YZ^*$, $z = \alpha(u)$, $v \in (T \cup Z)^+$; p_{2b}, p_{4d}, and p_{5b} represent one of the productions introduced in steps (2b), (4d), and (5b), respectively, and Ξ_4 and Ξ_5 are sequences of productions introduced in steps (4a), (4b), (4c), and (5a), respectively.

Proof. The proof of the beginning of the derivation,

$$\begin{array}{lll} S & \alpha(v_0)\langle A, q\rangle\alpha(w_0) & [p_{2b}] \\ \min\Rightarrow_G^* & y & [\Xi_4] \\ \min\Rightarrow_G & z & [p_{4d}], \end{array}$$

is analogous to the proof of Claim 8 (in terms of minimal derivations) and is left to the reader.

Recall that z satisfies $z = \alpha(u)$. Each of the productions from (5a) replaces one occurrence of $\bar{a}$ with a for some $a \in T$ and, finally, the application of a production from step (5b) replaces the remaining four nonterminals with their terminal variants. Therefore,

$$\begin{array}{lll} z & \min\Rightarrow_G^{|u|-4} & v\ [\Xi_5] \\ & \min\Rightarrow_G & u\ [p_{5b}], \end{array}$$

so the claim holds. □

Notice that $\text{len}(p_3) < \text{len}(p_4) < \text{len}(p_5)$ for each production p_3, p_4, and p_5 introduced in steps (3), (4), and (5), respectively, so the priorities (and the use) of the productions from the individual steps are the same as in the case of grammars which use maximal derivations. As a result, formulations of Claim 6, 7, and 9 can be changed in terms of minimal derivations. As their proofs resemble the proofs of the claims mentioned above, they are left to the reader. Therefore, $\mathscr{L}(\boldsymbol{CS}) \subseteq \mathscr{L}(\boldsymbol{PSC}, \min)$ and for the same reason as in the proof of Theorem 27, $\mathscr{L}(\boldsymbol{PSC}, \min) \subseteq \mathscr{L}(\boldsymbol{CS})$, so $\mathscr{L}(\boldsymbol{CS}) = \mathscr{L}(\boldsymbol{PSC}, \min)$.

■

We have demonstrated that propagating scattered context grammars which use maximal or minimal derivations characterize the family of context-sensitive languages. Consequently, if in the future formal language theory proves that any propagating scattered context grammar making maximal or minimal derivations can be transformed to an equivalent propagating scattered context grammar making ordinary derivations, it also proves that these grammars generate the family of all context-sensitive languages and, thereby, solves the long-standing open problem.

Chapter 6

Generators of Sentences with Their Parses

Parsing is important to all scientific areas that grammatically analyze and process languages, ranging from compiler design through linguistics to molecular biology. As obvious, *parses*—that is, the sequences of grammatical rules according to which sentences are generated—usually represent the goal information we want to achieve by parsing. (Let us note that the notion of a *parse* represents a synonym of several other notions, including a *derivation word*, a *Szilard word*, and a *control word*—see page 18 in [84].) We demonstrate that for every recursively enumerable language L, there exists a propagating scattered context grammar whose language consists of L's sentences followed by their parses. That is, if we eliminate all the suffixes representing the parses, we obtain precisely the recursively enumerable language L.

This characterization of recursively enumerable languages is of some interest because it is based on propagating scattered context grammars whose languages are included in the family of context-sensitive languages, which is properly contained in the family of recursively enumerable languages. Similar kind of characterization was already studied in [7] (see Theorem 14). This result was later improved by [37] showing that the same characterization can be achieved by using only leftmost derivations. We present a more general way of characterizing recursively enumerable languages by propagating scattered context grammars. Instead appending a sequence of useless symbols at the beginning or at the end of every sentence as in the above results, we add a useful information about the process of the generation—the parse.

Noteworthy, languages consisting of sentences followed by their parses were discussed in terms of matrix grammars in Section 7.2 of [6], which refers to these languages as *extended Szilard languages*. Apart from using different grammars, this discussion concentrated its attention on different areas of investigation, excluding any study of descriptional complexity, canonical derivations, or the characterization of the family of recursively enumerable languages.

Based upon scattered context grammars, we introduce scattered context generators that produce their sentences followed by the corresponding parses. As canonical leftmost and rightmost derivations fulfill a crucial role in parsing, we then modify these generators to their canonical versions that do the same job except that they only perform either leftmost or rightmost derivations. In addition, we reduce the grammatical size of these generators.

As we record which productions were used during the derivation, we need to refer to these productions somehow. This is done by production labels.

Definition 33. We assume that for every scattered context grammar $G = (V,T,P,S)$ there is a set of *production labels*, denoted by $\mathrm{lab}(G)$, such that $|\mathrm{lab}(G)| = |P|$; as usual, $\mathrm{lab}(G)^*$ denotes the set of

all strings over lab(G). Furthermore, there is a bijection from P to lab(G) such that if this bijection maps a production $(A_1,\ldots,A_n) \to (x_1,\ldots,x_n) \in P$ to a label $l \in \text{lab}(G)$, we say that $(A_1,\ldots,A_n) \to (x_1,\ldots,x_n)$ *is labeled with* l, symbolically written as

$$l : (A_1,\ldots,A_n) \to (x_1,\ldots,x_n).$$

To make the following text more readable, we use lhs(l) and rhs(l) instead of lhs$((A_1,\ldots,A_n) \to (x_1,\ldots,x_n))$ and rhs$((A_1,\ldots,A_n) \to (x_1,\ldots,x_n))$, respectively. By analogy with labeling each production in every scattered context grammar, we label each production (a,b,x,c) in every queue grammar as $l : (a,b,x,c)$. To express that $x \Rightarrow_G^* y$, where $x,y \in V^*$, by using a sequence of productions labeled with $p_1, p_2, \ldots, p_n$, we write $x \Rightarrow_G^* y\ [\rho]$, where $\rho = p_1 \ldots p_n \in \text{lab}(G)^*$.

As we also use canonical generators, we define the derivations performed in a leftmost and a rightmost way.

Definition 34. If every derivation step in every successful derivation in a scattered context grammar G is leftmost, G *generates* $L(G)$ *in a leftmost way*. If every step in every successful derivation in G is rightmost, G *generates* $L(G)$ *in a rightmost way*.

Finally, we introduce the most important definition of this chapter—a proper generator of its sentences with their parses.

Definition 35. Let $G = (V,T,P,S)$ be a scattered context grammar and let $S \Rightarrow_G^* x\ [\rho]$, where $x \in T^*$ and $\rho \in \text{lab}(G)^*$; then, x is a *sentence generated by* G *according to parse* ρ. Let lab$(G) \subseteq T$. G is a *proper generator of its sentences with their parses* if and only if

$$L(G) = \{x : x = y\rho, y \in (T - \text{lab}(G))^*, \rho \in \text{lab}(G)^*, S \Rightarrow_G^* x\ [\rho]\};$$

in addition, if G generates $L(G)$ in a leftmost or a rightmost way, G is a *proper leftmost* or a *proper rightmost generator of its sentences with their parses*. Similarly, G is a *proper generator of its sentences preceded by their parses* if and only if

$$L(G) = \{x : x = \rho y, y \in (T - \text{lab}(G))^*, \rho \in \text{lab}(G)^*, S \Rightarrow_G^* x\ [\rho]\};$$

in addition, if G generates $L(G)$ in a leftmost way, G is a *proper leftmost generator of its sentences preceded by their parses*.

Notice that these definitions impose no restrictions on derivations in a proper leftmost generator. However, every proper leftmost generator G satisfies the property that if G makes a non-leftmost step during a derivation, then this derivation cannot generate a member of $L(G)$. The same applies for proper rightmost generators.

Example 4. We illustrate these definitions by four scattered context grammars, each of which has its set of production labels equal to $\{1,2,3,4\}$.

1. Consider the scattered context grammar

   $$G_1 = (\{S,A,B,C,a,b,c\},\{a,b,c\},P_1,S)$$

   with P_1 containing

   $$1 : (S) \to (\varepsilon),$$
   $$2 : (S) \to (ABC),$$
   $$3 : (A,B,C) \to (aA,bB,cC),$$
   $$4 : (A,B,C) \to (a,b,c).$$

   As $\{1,2,3,4\} \not\subseteq \{a,b,c\}$, G_1 is not a proper generator of its sentences with their parses.

2. Consider the scattered context grammar

$$G_2 = (\{S,A,B,C,a,b,c,1,2,3,4\},\{a,b,c,1,2,3,4\},P_2,S)$$

with P_2 containing

$$\begin{array}{l} 1 : (S) \to (1), \\ 2 : (S) \to (ABC2), \\ 3 : (A,B,C) \to (aA,bB,cC3), \\ 4 : (A,B,C) \to (a,b,c4). \end{array}$$

Notice that $\{1,2,3,4\} \subseteq \{a,b,c,1,2,3,4\}$. However,

$$\begin{array}{rl} L(G_2) = & \{a^n b^n c^n \operatorname{rev}(\rho) : n \geq 0, S \Rightarrow^*_{G_2} a^n b^n c^n \operatorname{rev}(\rho)\ [\rho]\} \\ \neq & \{a^n b^n c^n \rho : n \geq 0, S \Rightarrow^*_{G_2} a^n b^n c^n \rho\ [\rho]\}, \end{array}$$

so G_2 is not a proper generator of its sentences with their parses either.

3. Consider the scattered context grammar

$$G_3 = (\{S,A,B,C,\$,a,b,c,1,2,3,4\},\{a,b,c,1,2,3,4\},P_3,S)$$

with P_3 containing

$$\begin{array}{l} 1 : (S) \to (1), \\ 2 : (S) \to (ABC2\$), \\ 3 : (A,B,C,\$) \to (AA,BB,CC,3\$), \\ 4 : (A,B,C,\$) \to (a,b,c,4). \end{array}$$

Observe that

$$L(G_3) = \{a^n b^n c^n \rho : n \geq 0, S \Rightarrow^*_{G_3} a^n b^n c^n \rho\ [\rho]\},$$

so G_3 is a proper generator of its sentences with their parses. However, as G_3 does not have to generate every sentence in a leftmost or a rightmost way, G_3 is neither a proper leftmost nor a proper rightmost generator of its sentences with their parses.

4. Consider the scattered context grammar

$$G_4 = (\{S,A,B,C,\$,a,b,c,1,2,3,4\},\{a,b,c,1,2,3,4\},P_4,S)$$

with P_4 containing

$$\begin{array}{l} 1 : (S) \to (1), \\ 2 : (S) \to (ABC2\$), \\ 3 : (A,B,C,\$) \to (aA,bB,cC,3\$), \\ 4 : (A,B,C,\$) \to (a,b,c,4). \end{array}$$

Observe that

$$L(G_4) = \{a^n b^n c^n \rho : n \geq 0, S \Rightarrow^*_{G_4} a^n b^n c^n \rho\ [\rho]\}$$

and every derivation step in G_4 is both leftmost and rightmost, so G_4 is both a proper leftmost and a proper rightmost generator of its sentences with their parses.

6.1 General Generators

Next, we demonstrate that for every recursively enumerable language L, there is a propagating scattered context grammar G which represents a proper generator of its sentences with their parses such that L results from $L(G)$ by eliminating all production labels in $L(G)$.

Theorem 29. *For every recursively enumerable language L, there exists a propagating scattered context grammar G such that G is a proper generator of its sentences with their parses and $L = L(G)//\mathrm{lab}(G)^+$.*

Proof. Let L be a recursively enumerable language. Then, there is a scattered context grammar $\bar{G} = (\bar{V}, T, \bar{P}, \bar{S})$ such that $L = L(\bar{G})$ (see Theorem 16). Set $\Phi = \{\hat{a} : a \in T\}$. Define the homomorphism γ from $\bar{V}$ to $(\Phi \cup (\bar{V} - T) \cup \{Y\})^+$ as $\gamma(a) = \hat{a}$ for all $a \in T$ and $\gamma(A) = A$ for all $A \in \bar{V} - T$. Extend the domain of γ to $\bar{V}^+$ in the standard manner; non-standardly, however, define $\gamma(\varepsilon) = Y$ rather than $\gamma(\varepsilon) = \varepsilon$. (Let us note that at this point γ does not, strictly speaking, represent a homomorphism.) Finally, set $\Gamma = \{\$_1, \$_2, \$_3\}$ and $V = \bar{V} \cup \mathrm{lab}(G) \cup \Phi \cup \Gamma \cup \{S, X, Y, Z\}$. Define the propagating scattered context grammar

$$G = (\bar{V} \cup \mathrm{lab}(G) \cup \Phi \cup \Gamma \cup \{S, X, Y, Z\}, T \cup \mathrm{lab}(G), P, S)$$

with

$$\mathrm{lab}(G) = \{\lfloor 1 \rfloor, \lfloor 1_\varepsilon \rfloor, \lfloor 2 \rfloor, \lfloor 2_\varepsilon \rfloor, \lfloor 3 \rfloor, \lfloor 4 \rfloor\} \cup \Xi_1 \cup \Xi_2 \cup \Xi_3,$$

where $\Xi_1 = \{\lfloor 1p \rfloor : p \in \mathrm{lab}(\bar{G})\}$, $\Xi_2 = \{\lfloor 2a \rfloor : a \in T\}$, $\Xi_3 = \{\lfloor 3a \rfloor : a \in T\}$; without loss of generality, assume $\mathrm{lab}(G) \cap \mathrm{alph}(L) = \emptyset$. P is constructed as follows:

1. Add
   (a) $\lfloor 1 \rfloor : (S) \to (X \lfloor 1 \rfloor \$_1 Z \bar{S})$ and
   (b) $\lfloor 1_\varepsilon \rfloor : (S) \to (\lfloor 1_\varepsilon \rfloor \$_1 \bar{S})$ to P;
2. (a) For each $p : (A_1, \ldots, A_n) \to (x_1, \ldots, x_n) \in \bar{P}$, add
   $\lfloor 1p \rfloor : (\$_1, A_1, \ldots, A_n) \to (\lfloor 1p \rfloor \$_1, \gamma(x_1), \ldots, \gamma(x_n))$ to P;
   (b) Add
      i. $\lfloor 2 \rfloor : (\$_1) \to (\lfloor 2 \rfloor \$_2)$ and
      ii. $\lfloor 2_\varepsilon \rfloor : (\$_1) \to (\lfloor 2_\varepsilon \rfloor \$_3)$ to P;
3. For each $a \in T$, add
   (a) $\lfloor 2a \rfloor : (X, \$_2, Z, \hat{a}) \to (aX, \lfloor 2a \rfloor \$_2, Y, Z)$ and
   (b) $\lfloor 3a \rfloor : (X, \$_2, Z, \hat{a}) \to (a, \lfloor 3a \rfloor \$_3, Y, Y)$ to P;
4. Add $\lfloor 3 \rfloor : (\$_3, Y) \to (\lfloor 3 \rfloor, \$_3)$ to P;
5. Add $\lfloor 4 \rfloor : (\$_3) \to (\lfloor 4 \rfloor)$ to P.

Basic Idea. First, we explain how G makes the derivation of a nonempty sentence followed by its parse; then, we explain the derivation of the empty sentence followed by its parse.

G makes the derivation of $a_1 a_2 \ldots a_n \rho$, where $n \geq 1$, each $a_i \in T$ and ρ is the corresponding parse, by productions introduced in steps (1) through (5) in this order. After starting this derivation

by using the production from (1a), it applies productions introduced in (2a), which simulate the applications of productions from $\bar{P}$. More precisely, it simulates the application of $p : (A_1, \ldots, A_n) \to (x_1, \ldots, x_n) \in \bar{P}$ by using

$$\lfloor 1p \rfloor : (\$_1, A_1, \ldots, A_n) \to (\lfloor 1p \rfloor \$_1, \gamma(x_1), \ldots, \gamma(x_n)) \in P$$

so that it places its own label, $\lfloor 1p \rfloor$, right behind the previously generated production labels; this substring of labels occurs between the leftmost symbol, X, and $\$_1$, in the sentential form. Otherwise,

$$\lfloor 1p \rfloor : (\$_1, A_1, \ldots, A_n) \to (\lfloor 1p \rfloor \$_1, \gamma(x_1), \ldots, \gamma(x_n))$$

is analogical to $p : (A_1, \ldots, A_n) \to (x_1, \ldots, x_n)$ except that (1) the former has the fill-in symbol Y where the latter has ε and (2) the former has $\hat{a}$ where the latter has terminal a_i. After using productions introduced in (2), G has its current sentential form of the form

$$X\tau\$_2 Z u_0 \hat{a}_1 u_1 \hat{a}_2 u_2 \ldots u_{n-1} \hat{a}_n u_n,$$

where τ is a prefix of ρ and $u_i \in \{Y\}^*$. By using productions from (3), it places $a_1 \ldots a_n$ at the beginning of the sentential form while replacing each $\hat{a}_i$ with Y and generating the production labels. By using productions labeled with $\lfloor 3 \rfloor$, G replaces each Y with $\lfloor 3 \rfloor$ while shifting $\$_3$ to the right. Finally, the application of the production labeled with $\lfloor 4 \rfloor$ completes the derivation of $a_1 a_2 \ldots a_n \rho$.

Finally, let us explain how G makes the derivation of the empty sentence ε followed by its parse. By use of productions labeled with $\lfloor 1_\varepsilon \rfloor$ and $\lfloor 2_\varepsilon \rfloor$ instead of $\lfloor 1 \rfloor$ and $\lfloor 2 \rfloor$, respectively, the process of placing terminal symbols at the beginning of the sentential form by productions from step (3) is skipped; otherwise, the derivation proceeds as above.

Formal Proof.

Claim 11. G generates each $w \in L(G) - \mathrm{lab}(G)^+$ in the following way:

$$\begin{aligned} S &\Rightarrow_G X\lfloor 1 \rfloor \$_1 Z \bar{S} && [\lfloor 1 \rfloor] \\ &\Rightarrow_G^+ x && [\rho] \\ &\Rightarrow_G y && [\lfloor 2 \rfloor] \\ &\Rightarrow_G^* z && [\sigma] \\ &\Rightarrow_G u && [\lfloor 3a \rfloor] \\ &\Rightarrow_G^+ v && [\tau] \\ &\Rightarrow_G w && [\lfloor 4 \rfloor], \end{aligned}$$

where $x, y, z, u, v, w \in V^*$, $\lfloor 3a \rfloor \in \Xi_3$, $\tau \in \{\lfloor 3 \rfloor\}^+$, $\rho \in \Xi_1^+$, and $\sigma \in \Xi_2^*$.

Proof. First, let us make these four observations:

1. Since the only productions with S on their left-hand sides are productions introduced in step (1) of the construction, $S \Rightarrow_G^+ w$ surely starts with a step made by one of these productions. Further, $S \notin \mathrm{alph}(\mathrm{rhs}(p))$ for any $p \in P$. Therefore, these productions are not used during the rest of the derivation. Notice that $\mathrm{alph}(w) \cap T \neq \emptyset$ and only productions labeled with $p \in \Xi_2 \cup \Xi_3$ satisfy $a \in \mathrm{alph}(\mathrm{rhs}(p))$, where $a \in T$. This derivation ends by applying the production labeled with $\lfloor 4 \rfloor$ because it is the only production with its right-hand side over $(T \cup \mathrm{lab}(G))^*$. Thus, $S \Rightarrow_G^+ w$ can be expressed as

$$\begin{aligned} S &\Rightarrow_G X\lfloor 1 \rfloor \$_1 Z \bar{S} && [\lfloor 1 \rfloor] \\ &\Rightarrow_G^+ v \\ &\Rightarrow_G w && [\lfloor 4 \rfloor]. \end{aligned}$$

2. Let p be a label of any production introduced in steps (2) through (4) of the construction; then, $|\mathrm{lhs}(p)|_\Gamma = |\mathrm{rhs}(p)|_\Gamma = 1$. In greater detail, for each $\lfloor 1p \rfloor \in \Xi_1$, $\lfloor 2a \rfloor \in \Xi_2$, and $\lfloor 3a \rfloor \in \Xi_3$, productions introduced in step (2) satisfy

$$\begin{aligned} |\mathrm{lhs}(\lfloor 1p \rfloor)|_{\{\$_1\}} &= |\mathrm{rhs}(\lfloor 1p \rfloor)|_{\{\$_1\}} = 1, \\ |\mathrm{lhs}(\lfloor 2 \rfloor)|_{\{\$_1\}} &= |\mathrm{rhs}(\lfloor 2 \rfloor)|_{\{\$_2\}} = 1, \\ |\mathrm{lhs}(\lfloor 2\varepsilon \rfloor)|_{\{\$_1\}} &= |\mathrm{rhs}(\lfloor 2\varepsilon \rfloor)|_{\{\$_3\}} = 1. \end{aligned}$$

   Similarly, productions introduced in step (3) satisfy

$$\begin{aligned} |\mathrm{lhs}(\lfloor 2a \rfloor)|_{\{\$_2\}} &= |\mathrm{rhs}(\lfloor 2a \rfloor)|_{\{\$_2\}} = 1, \\ |\mathrm{lhs}(\lfloor 3a \rfloor)|_{\{\$_2\}} &= |\mathrm{rhs}(\lfloor 3a \rfloor)|_{\{\$_3\}} = 1. \end{aligned}$$

   Finally, the production introduced in step (4) satisfies

$$|\mathrm{lhs}(\lfloor 3 \rfloor)|_{\{\$_3\}} = |\mathrm{rhs}(\lfloor 3 \rfloor)|_{\{\$_3\}} = 1.$$

3. Because $X \in \mathrm{alph}(x)$ and only productions labeled with $p \in \Xi_3$ satisfy $X \in \mathrm{alph}(\mathrm{lhs}(p))$ and $X \notin \mathrm{alph}(\mathrm{rhs}(p))$, the production labeled with $\lfloor 2\varepsilon \rfloor$ cannot be used.

4. Let p be the label of any production introduced in steps (1) through (5); then, $\mathrm{alph}(\mathrm{rhs}(p)) \cap \mathrm{lab}(G) = \{p\}$ and $|\mathrm{rhs}(p)|_{\{p\}} = 1$.

Based on these observations, notice that G generates each $w \in L(G) - \mathrm{lab}(G)^+$ in the way described in the formulation of Claim 11. □

Claim 12. Consider the derivation from Claim 11. In its beginning,

$$\begin{aligned} S &\Rightarrow_G X\lfloor 1 \rfloor \$_1 Z \bar{S} && [\lfloor 1 \rfloor] \\ &\Rightarrow_G^+ x && [\rho] \\ &\Rightarrow_G y && [\lfloor 2 \rfloor], \end{aligned}$$

every sentential form s in $X\lfloor 1 \rfloor \$_1 Z \bar{S} \Rightarrow_G^+ x$ satisfies

$$s \in \{X\}\,\mathrm{lab}(G)^+\{\$_1\}\{Z\}(\Phi \cup (\bar{V} - T) \cup \{Y\})^+$$

and

$$y \in \{X\}\,\mathrm{lab}(G)^+\{\$_2\}\{Z\}(\Phi \cup \{Y\})^+.$$

Proof. By the definition of the homomorphism γ, productions labeled with $\lfloor 1p \rfloor$, where $p \in \mathrm{lab}(\bar{G})$, rewrite symbols over $\Phi \cup (\bar{V} - T) \cup \{Y\}$ and change $\$_1$ to $\lfloor 1p \rfloor \$_1$. Since $\bar{V} \cap \{X, \$_1, Z\} = \emptyset$, every sentential form s in $X\lfloor 1 \rfloor \$_1 Z \bar{S} \Rightarrow_G^+ x$ satisfies

$$s \in \{X\}\,\mathrm{lab}(G)^+\{\$_1\}\{Z\}(\Phi \cup (\bar{V} - T) \cup \{Y\})^+.$$

Only productions labeled with $\lfloor 1p \rfloor \in \Xi_1$ satisfy

$$\mathrm{alph}(\mathrm{lhs}(\lfloor 1p \rfloor)) \cap (\bar{V} - T) \neq \emptyset.$$

Therefore, to generate $w \in (T \cup \mathrm{lab}(G))^*$, productions labeled with $\lfloor 1p \rfloor$ have to be applied until

$$s \in \{X\}\,\mathrm{lab}(G)^+\{\$_1\}\{Z\}(\Phi \cup \{Y\})^+.$$

Finally, the production labeled with $\lfloor 2 \rfloor$ is used, so

$$y \in \{X\}\,\mathrm{lab}(G)^+\{\$_2\}\{Z\}(\Phi \cup \{Y\})^+$$

and the claim holds. □

Claim 13. In

$$\begin{aligned} y &\Rightarrow_G^* z \; [\lfloor \sigma \rfloor] \\ &\Rightarrow_G u \; [\lfloor 3a \rfloor] \end{aligned}$$

of the derivation from Claim 11, every sentential form o in $y \Rightarrow_G^* z$ can be expressed as

$$o \in T^*\{X\}\,\mathrm{lab}(G)^+\{\$_2\}\{Y\}^*\{Z\}(\Phi \cup \{Y\})^+$$

and $u \in T^+\,\mathrm{lab}(G)^+\{\$_3\}\{Y\}^+$. In greater detail,

$$\begin{aligned} & X\lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \$_2 Z Y^{i_0} \hat{b}_1 Y^{i_1} \hat{b}_2 Y^{i_2} \dots \hat{b}_m Y^{i_m} = y \\ \Rightarrow_G \; & b_1 X \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \$_2 Y^{i_0+1} Z Y^{i_1} \hat{b}_2 Y^{i_2} \dots \hat{b}_m Y^{i_m} && [\lfloor 2b_1 \rfloor] \\ \Rightarrow_G \; & b_1 b_2 X \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \lfloor 2b_2 \rfloor \$_2 Y^{i_0+1} Y^{i_1+1} Z Y^{i_2} \dots \hat{b}_m Y^{i_m} && [\lfloor 2b_2 \rfloor] \\ \Rightarrow_G^{m-3} \; & b_1 \dots b_{m-1} X \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \dots \lfloor 2b_{m-1} \rfloor \$_2 Y^{i_0+1} \\ & \dots Y^{i_{m-2}+1} Z Y^{i_{m-1}} \hat{b}_m Y^{i_m} && [\bar{\sigma}] \\ \Rightarrow_G \; & b_1 \dots b_m \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \dots \lfloor 2b_{m-1} \rfloor \lfloor 3b_m \rfloor \$_3 Y^{i_0+1} \\ & \dots Y^{i_m+1} = u && [\lfloor 3b_m \rfloor], \end{aligned}$$

where $\lfloor p_1 \rfloor, \dots, \lfloor p_n \rfloor \in \mathrm{lab}(G)$ are labels that denote productions introduced in steps (1) and (2), $b_1, \dots, b_m \in T$, $\bar{\sigma} = \lfloor 2b_3 \rfloor \dots \lfloor 2b_{m-1} \rfloor$, $i_0, \dots, i_m \geq 0$, $m = |s|$, where $s \in L(\bar{G})$ is the corresponding sentence of the scattered context grammar $\bar{G}$.

Proof. Notice that

$$|\mathrm{lhs}(\lfloor 2a \rfloor)|_{\{X\}} = |\mathrm{rhs}(\lfloor 2a \rfloor)|_{\{X\}} = 1$$

and

$$|\mathrm{lhs}(\lfloor 2a \rfloor)|_{\{Y\}} = |\mathrm{rhs}(\lfloor 2a \rfloor)|_{\{Y\}} = 1,$$

where $\lfloor 2a \rfloor \in \Xi_2$. In every derivation step of $y \Rightarrow_G^* z$, the the first symbol $\hat{b} \in \Phi$ following Z is replaced with Z, X is changed to bX, and $\$_2$ is changed to $\lfloor 2a \rfloor \$_2$, where $\lfloor 2a \rfloor \in \Xi_2$. As only productions from step (3), labeled with p, satisfy $\mathrm{alph}(\mathrm{lhs}(p)) \cap \Phi \neq \emptyset$, $\mathrm{alph}(\mathrm{rhs}(p)) \cap \Phi = \emptyset$, Z can replace only the first occurrence of $\hat{b} \in \Phi$ behind Z to generate $w \in (T \cup \mathrm{lab}(G))^*$. Productions labeled with $\lfloor 2a \rfloor$ are used $m-1$ times. Thus, $y \Rightarrow_G^* z$ has the form

$$\begin{aligned} & X\lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \$_2 Z Y^{i_0} \hat{b}_1 Y^{i_1} \hat{b}_2 Y^{i_2} \dots \hat{b}_m Y^{i_m} \\ \Rightarrow_G \; & b_1 X \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \$_2 Y^{i_0+1} Z Y^{i_1} \hat{b}_2 Y^{i_2} \dots \hat{b}_m Y^{i_m} && [\lfloor 2b_1 \rfloor] \\ \Rightarrow_G \; & b_1 b_2 X \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \lfloor 2b_2 \rfloor \$_2 Y^{i_0+1} Y^{i_1+1} Z Y^{i_2} \dots \hat{b}_m Y^{i_m} && [\lfloor 2b_2 \rfloor] \\ \Rightarrow_G^{m-3} \; & b_1 \dots b_{m-1} X \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \lfloor 2b_{m-1} \rfloor \$_2 Y^{i_0+1} \\ & \dots Y^{i_{m-2}+1} Z Y^{i_{m-1}} \hat{b}_m Y^{i_m} && [\bar{\sigma}], \end{aligned}$$

where every sentential form satisfies

$$T^*\{X\}\,\mathrm{lab}(G)^+\{\$_2\}\{Y\}^*\{Z\}(\Phi \cup \{Y\})^+.$$

Finally, a production labeled with $\lfloor 3a \rfloor$ is applied; therefore, $z \Rightarrow_G u$ can be expressed as

$$\begin{aligned} & b_1 \dots b_{m-1} X \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \dots \lfloor 2b_{m-1} \rfloor \$_2 Y^{i_0+1} \\ & \dots Y^{i_{m-2}+1} Z Y^{i_{m-1}} \hat{b}_m Y^{i_m} \\ \Rightarrow_G \; & b_1 \dots b_m \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 2b_1 \rfloor \dots \lfloor 2b_{m-1} \rfloor \lfloor 3b_m \rfloor \$_3 Y^{i_0+1} \dots Y^{i_m+1} \; [\lfloor 3b_m \rfloor] \end{aligned}$$

with $u \in T^+\,\mathrm{lab}(G)^+\{\$_3\}\{Y\}^+$.

Putting together the previous parts of derivation, we obtain the formulation of Claim 13. Thus, Claim 13 holds. □

Claim 14. In

$$\begin{aligned} u &\Rightarrow_G^+ v \ [\tau] \\ &\Rightarrow_G w \ [\lfloor 4 \rfloor] \end{aligned}$$

of the derivation from Claim 11, every sentential form s of $u \Rightarrow_G^+ v$ satisfies $s \in T^+ \operatorname{lab}(G)^+ \{\$_3\}\{Y\}^*$ and $w \in T^+ \operatorname{lab}(G)^+$. In greater detail, this derivation can be expressed as

$$\begin{aligned} & b_1 \dots b_m \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \{\$_3\} Y^i \\ \Rightarrow_G \ & b_1 \dots b_m \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 3 \rfloor \{\$_3\} Y^{i-1} \ [\lfloor 3 \rfloor] \\ \Rightarrow_G^{i-2} \ & b_1 \dots b_m \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 3 \rfloor^{i-1} \{\$_3\} Y \ [\bar{\tau}] \\ \Rightarrow_G \ & b_1 \dots b_m \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 3 \rfloor^{i} \{\$_3\} \qquad [\lfloor 3 \rfloor] \\ \Rightarrow_G \ & b_1 \dots b_m \lfloor p_1 \rfloor \dots \lfloor p_n \rfloor \lfloor 3 \rfloor^{i} \lfloor 4 \rfloor \qquad [\lfloor 4 \rfloor], \end{aligned}$$

where each $b_j \in T$ for all $1 \leq j \leq m$, $\lfloor p_k \rfloor \in \operatorname{lab}(G)$ for all $1 \leq k \leq n$ are labels that denote productions introduced in steps (1) through (3) of the construction, and $\bar{\tau} \in \{\lfloor 3 \rfloor\}^*$.

Proof. Observe that in order to generate $w \in (T \cup \operatorname{lab}(G))^*$, the first occurrence of Y following $\$_3$ has to be rewritten by the production labeled with $\lfloor 3 \rfloor$ in every derivation step. Finally, the production labeled with $\lfloor 4 \rfloor$ is applied. At this point, w satisfies $w \in T^+ \operatorname{lab}(G)^+$. □

The following claim formally demonstrates how G generates the empty sentence ε followed by its parse.

Claim 15. G generates each $w \in \operatorname{lab}(G)^+$ in the following way:

$$\begin{aligned} S &\Rightarrow_G \lfloor 1_\varepsilon \rfloor \$_1 \bar{S} \ [\lfloor 1_\varepsilon \rfloor] \\ &\Rightarrow_G^+ x \qquad [\rho] \\ &\Rightarrow_G y \qquad [\lfloor 2_\varepsilon \rfloor] \\ &\Rightarrow_G^+ v \qquad [\tau] \\ &\Rightarrow_G w \qquad [\lfloor 4 \rfloor], \end{aligned}$$

where $\rho \in \Xi_1^+$ and $\tau \in \{\lfloor 3 \rfloor\}^+$.

Proof. Note that $\operatorname{alph}(w) \cap T = \emptyset$ and only productions labeled with $p \in \Xi_3$ satisfy $X \in \operatorname{alph}(\operatorname{lhs}(p))$, $X \notin \operatorname{alph}(\operatorname{rhs}(p))$, and $a \in \operatorname{alph}(\operatorname{rhs}(p))$, where $a \in T$. Therefore, X cannot appear in any sentential form of $S \Rightarrow_G^* w$, so the derivation starts with a step made by the production labeled with $\lfloor 1_\varepsilon \rfloor$. As $X \notin \operatorname{alph}(x)$ and for $p \in \Xi_2 \cup \Xi_3$, $X \in \operatorname{alph}(\operatorname{lhs}(p))$, the production labeled with $\lfloor 2_\varepsilon \rfloor$ has to be used. Observe that other derivation steps are made in the way described in Claim 12 and Claim 14. □

From Claims 11 through 15, it follows that for every recursively enumerable language L, there exists a propagating scattered context grammar G such that G is a proper generator of its sentences with their parses and $L = L(G) /\!/ \operatorname{lab}(G)^+$. ■

In addition, Theorem 29 immediately implies Corollary 5. Clearly, by constructing a proper generator of its sentences with their parses G whose language satisfies $L = L(G) /\!/ \operatorname{lab}(G)^+$ and defining h so that it erases every symbol from $\operatorname{lab}(G)$ and preserves all other symbols, we get a constructive proof of Corollary 5.

By a straightforward modification of the construction, we can also prove an analogous result for sentences which are preceded by their parses.

Theorem 30. *For every recursively enumerable language L, there exists a propagating scattered context grammar G such that G is a proper generator of its sentences preceded by their parses and $L = \operatorname{lab}(G)^+ \backslash\backslash L(G)$.* ■

Clearly, Theorem 14 is then an immediate corollary of Theorem 30 because instead of generating a production label in every derivation step, we can generate the fill-in symbol \$ so the resulting sentence is preceded by a sequence of \$'s which is exactly what Theorem 14 says.

6.2 Canonical Generators

Next, we establish two characterizations based on leftmost and rightmost generators. We demonstrate that for every recursively enumerable language L there is a propagating scattered context grammar G which represents a proper leftmost (rightmost) generator of its sentences with their parses so that L results from $L(G)$ by eliminating all production labels in $L(G)$. Notice that there exists a significant advantage of canonical generators over general generators—in a sentence generated by leftmost generators, the recorded parse provides us with a sufficient information to reproduce its derivation while in ordinary generators, this information is insufficient because the precise specification of the occurrences of the symbols to be rewritten is missing.

Theorem 31. *For every recursively enumerable language L there exists a propagating scattered context grammar $G = (\bar{V}, \bar{T}, P, S)$ such that G is a proper leftmost generator of its sentences with their parses, $|\bar{V} - \bar{T}| \leq 6$, and $L = L(G) /\!/ \text{lab}(G)^+$.*

Proof. Let L be a recursively enumerable language. Let $Q = (V, T, W, F, R, g)$ be a queue grammar such that $L(Q) = L - \{\varepsilon\}$ and Q satisfies the properties described in Lemma 3 and Corollary 1. Recall that $\text{lab}(Q)$ is the set of Q's production labels. Define an injective homomorphism α from $\text{lab}(Q)^*$ to $\{0\}^*\{1\}$ so that α is an injective homomorphism when its domain is extended to $\text{lab}(Q)^*$ in the standard way. Further, define the substitution f on V^* so that $f(\varepsilon) = \varepsilon$ and

$$f(a) = \{\alpha(r) : r : (a, b, x, d) \in R\}$$

for all $a \in V$. Similarly, define the substitution g on W^* so that

$$g(b) = \{\alpha(r) : r : (a, b, x, d) \in R\}$$

for all $b \in W$. Set

$$\begin{aligned}
\Xi_1 &= \{\lfloor 1\bar{a}_0\bar{q}_0 \rfloor : g = a_0 q_0, \bar{a}_0 \in f(a_0), \bar{q}_0 \in g(q_0)\}, \\
\Xi_2 &= \{\lfloor 2r\bar{x}\bar{d} \rfloor : r : (a, b, x, d) \in R, x \in (V - T)^*, d \in W - F, \\
&\qquad \bar{x} \in f(x), \bar{d} \in g(d)\}, \\
\Xi_4 &= \{\lfloor 4r\bar{d} \rfloor : r : (a, b, c, d) \in R, c \in T, d \in W - F, \bar{d} \in g(d)\}, \\
\Xi_5 &= \{\lfloor 5r \rfloor : r : (a, b, c, d) \in R, c \in T, d \in F\}.
\end{aligned}$$

Define the propagating scattered context grammar G as

$$G = (\{S, A, B, \#, 0, 1\} \cup T \cup \text{lab}(G), T \cup \text{lab}(G), P, S),$$

where

$$\text{lab}(G) = \Xi_1 \cup \Xi_2 \cup \Xi_4 \cup \Xi_5 \cup \{\lfloor 3 \rfloor, \lfloor 6 \rfloor, \lfloor 7 \rfloor, \lfloor 8 \rfloor, \lfloor 9 \rfloor, \lfloor 10 \rfloor\},$$

and P is constructed as follows:

1. For each $\bar{a}_0 \in f(a_0)$, $\bar{q}_0 \in g(q_0)$ such that $g = a_0 q_0$, add $\lfloor 1\bar{a}_0\bar{q}_0 \rfloor : (S) \to (A \lfloor 1\bar{a}_0\bar{q}_0 \rfloor A A \bar{q}_0 A \bar{a}_0 A B)$ to P;

2. For each $r : (a, b, x, d) \in R$, $x \in (V - T)^*$, $d \in W - F$, and $\bar{x} \in f(x)$, $\bar{d} \in g(d)$, add $\lfloor 2r\bar{x}\bar{d} \rfloor : (A, A, A, A, A, B) \to (A, \lfloor 2r\bar{x}\bar{d} \rfloor A, \alpha(r)A, \bar{d}A, \bar{x}A, B)$ to P;

3. Add $\lfloor 3 \rfloor : (A, A, A, A, A, B) \to (A, \lfloor 3 \rfloor A, A, A, B, A)$ to P;

4. For each $r : (a, b, c, d) \in R$, $c \in T$, $d \in (W - F)$, and $\bar{d} \in g(d)$, add $\lfloor 4r\bar{d} \rfloor : (A, A, A, A, B, A) \to (cA, \lfloor 4r\bar{d} \rfloor A, \alpha(r)A, \bar{d}A, B, A)$ to P;

5. For each $r : (a,b,c,d) \in R$, $c \in T$ and $d \in F$, add
   $\lfloor 5r \rfloor : (A,A,A,A,B,A) \to (c, \lfloor 5r \rfloor A, \alpha(r)A, A, B, AA)$ to P;
6. Add
   (a) $\lfloor 6 \rfloor : (A,0,A,0,A,0,B,A,A) \to (\lfloor 6 \rfloor, A, \#, A, \#, A, B, A, A)$ and
   (b) $\lfloor 7 \rfloor : (A,1,A,1,A,1,B,A,A) \to (\lfloor 7 \rfloor, A, \#, A, \#, A, B, A, A)$ to P;
7. Add
   (a) $\lfloor 8 \rfloor : (A,A,A,B,A,A) \to (\lfloor 8 \rfloor B, \#, \#, \#, \#, \#)$,
   (b) $\lfloor 9 \rfloor : (B, \#) \to (\lfloor 9 \rfloor, B)$, and
   (c) $\lfloor 10 \rfloor : (B) \to (\lfloor 10 \rfloor)$ to P.

Basic Idea. G simulates every successful derivation of a sentence $x \in L(Q)$ and, simultaneously, records the productions it applies. At the end of the simulation, G moves the generated sentence x to the left so it occurs in front of the recorded labels. G makes all this derivation so that it represents a proper leftmost generator of its sentences with their parses and $L(Q) = L(G) /\!/ \mathrm{lab}(G)^+$.

To describe the way G works in greater detail, observe that G generates every sentence so it applies its productions in the order corresponding to steps (1) through (7) in the construction above. First, G applies a production introduced in (1) and, thereby, starts the simulation of a derivation of x by Q. That is, for $g = a_0 q_0$, G inserts the binary representation of q_0 and a_0 in front of the fourth and the fifth nonterminal A, respectively. Each production introduced in step (2) of the construction simulates a production in Q of the form $r : (a,b,x,d)$, where $x \in (V-T)^*$ and $d \in W - F$. This production places r's binary representation in front of the third nonterminal A and inserts the binary representation of d and x in front of the fourth and the fifth nonterminal A, respectively. After the application of the production labeled with $\lfloor 3 \rfloor$, G simulates only productions of the form (a,b,c,d) with $c \in T$ and $d \in W - F$ by productions introduced in step (4). This simulation places c so it precedes the first nonterminal A; otherwise, it works similarly to the simulation by productions introduced in (2). By a production constructed in (5), G completes the simulation of Q's derivation of x. To successfully complete the derivation, all the three binary substrings that follow each of the first three nonterminals A have to coincide. By the productions constructed in step (6), G verifies this coincidence so that it replaces the first coinciding binary symbol with the applied production label and the other two symbols with #'s. If G successfully completes this verification process, production labeled with $\lfloor 8 \rfloor$ replaces all A's with #'s and moves B at the very end of the sequence of the recorded labels. Then, G uses the production labeled with $\lfloor 9 \rfloor$ to replace all #'s with $\lfloor 9 \rfloor$'s. Finally, it completes the derivation by using the production labeled with $\lfloor 10 \rfloor$ to replace B with $\lfloor 10 \rfloor$.

Formal Proof.

Claim 16. Every sentence $w \in L(G)$ is generated in this way:

$$\begin{aligned}
S &\Rightarrow_G A\lfloor 1\bar{a}_0\bar{q}_0 \rfloor AA\bar{q}_0 A\bar{a}_0 AB && [\lfloor 1\bar{a}_0\bar{q}_0 \rfloor] \\
&\Rightarrow_G^* x && [\rho] \\
&\Rightarrow_G y && [\lfloor 3 \rfloor] \\
&\Rightarrow_G^* z && [\sigma] \\
&\Rightarrow_G u && [\lfloor 5r \rfloor] \\
&\Rightarrow_G^* v && [\tau] \\
&\Rightarrow_G w_1 && [\lfloor 8 \rfloor] \\
&\Rightarrow_G^* w_2 && [\omega] \\
&\Rightarrow_G w && [\lfloor 10 \rfloor],
\end{aligned}$$

where

$$x, y, z, u, v, w_1, w_2, w \in (\{S, A, B, \#, 0, 1\} \cup T \cup \mathrm{lab}(G))^*,$$

$\lfloor 1\bar{a}_0\bar{q}_0 \rfloor \in \Xi_1$, $\lfloor 5r \rfloor \in \Xi_5$, $\rho \in \Xi_2^*$, $\sigma \in \Xi_4^*$, $\tau \in \{\lfloor 6 \rfloor, \lfloor 7 \rfloor\}^*$, and $\omega \in \{\lfloor 9 \rfloor\}^*$.

Proof. Since the only productions with S on their left-hand sides are the productions introduced in step (1), $S \Rightarrow_G^* w$ surely starts with a derivation step made by one of these productions. As S does not occur on the right-hand side of any production, no production constructed in step (1) is applied later in the derivation.

All derivations end by applying the production labeled with $\lfloor 10 \rfloor$ because it is the only production with its right-hand side over $(T \cup \mathrm{lab}(G))^*$. Thus, $S \Rightarrow_G^* w$ can be expressed as

$$\begin{aligned} S &\Rightarrow_G A\lfloor 1\bar{a}_0\bar{q}_0 \rfloor AA\bar{q}_0A\bar{a}_0AB && [\lfloor 1\bar{a}_0\bar{q}_0 \rfloor] \\ &\Rightarrow_G^* w_2 \\ &\Rightarrow_G w && [\lfloor 10 \rfloor]. \end{aligned}$$

Further, notice that each production introduced in steps (2) through (6) contains exactly five A's and one B on both its left and right-hand side. As all five A's and one B are rewritten during every derivation step made by productions of (2) through (6), the position of B implies that

$$\begin{aligned} S &\Rightarrow_G A\lfloor 1\bar{a}_0\bar{q}_0 \rfloor AA\bar{q}_0A\bar{a}_0AB && [\lfloor 1\bar{a}_0\bar{q}_0 \rfloor] \\ &\Rightarrow_G^* w_2 \end{aligned}$$

can be expressed as

$$\begin{aligned} S &\Rightarrow_G A\lfloor 1\bar{a}_0\bar{q}_0 \rfloor AA\bar{q}_0A\bar{a}_0AB && [\lfloor 1\bar{a}_0\bar{q}_0 \rfloor] \\ &\Rightarrow_G^* x && [\rho] \\ &\Rightarrow_G y && [\lfloor 3 \rfloor] \\ &\Rightarrow_G^* z && [\sigma] \\ &\Rightarrow_G u && [\lfloor 5r \rfloor] \\ &\Rightarrow_G^* v && [\tau] \\ &\Rightarrow_G w_1 && [\lfloor 8 \rfloor] \\ &\Rightarrow_G^* w_2. \end{aligned}$$

The production labeled with $\lfloor 8 \rfloor$ replaces all A's with #'s. After this replacement, only productions labeled with $\lfloor 9 \rfloor$ and $\lfloor 10 \rfloor$ can be used. The production labeled with $\lfloor 9 \rfloor$ requires #'s behind B; this requirement is satisfied by the production labeled with $\lfloor 8 \rfloor$. The production labeled with $\lfloor 10 \rfloor$ is used during the very last derivation step because it removes B from the sentential form and B occurs on the left-hand sides of all other productions. Based on these observations, notice that G generates each $w \in L(G)$ in the way described in Claim 16. □

Claim 17. Consider the derivation from Claim 16. In its beginning,

$$\begin{aligned} S &\Rightarrow_G A\lfloor 1\bar{a}_0\bar{q}_0 \rfloor AA\bar{q}_0A\bar{a}_0AB && [\lfloor 1\bar{a}_0\bar{q}_0 \rfloor] \\ &\Rightarrow_G^* x && [\rho] \\ &\Rightarrow_G y && [\lfloor 3 \rfloor], \end{aligned}$$

where $\lfloor 1\bar{a}_0\bar{q}_0 \rfloor \in \Xi_1$, $\rho \in \Xi_2^*$, and every sentential form s in

$$A\lfloor 1\bar{a}_0\bar{q}_0 \rfloor AA\bar{q}_0A\bar{a}_0AB \Rightarrow_G^* x$$

satisfies

$$s \in \{A\}\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{B\},$$

and

$$y \in \{A\}\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{B\}\{A\}.$$

Proof. Productions introduced in step (2) of the construction simulate Q's productions of the form $r : (a,b,z,d)$, $z \in (V-T)^*$ and $d \in W-F$. Each production inserts its label in front of the second nonterminal A, $\alpha(r)$ in front of the third nonterminal A, $\bar{d} \in g(d)$ in front of the fourth nonterminal A, and, finally, $\bar{z}$, where $\bar{z} \in f(z)$, in front of the fifth nonterminal A. Intuitively, $\alpha(r)$ is the binary representation of currently simulated production, $\bar{d}$ is the binary representation of production r' : (a',b',z',d'), $b' = d$, which will be simulated in the following derivation step, and $\bar{z}$ is the binary representation of productions which will eventually be simulated when the first symbol of Q's sentential form becomes $c_1,\ldots,c_n$, where $c_1,\ldots,c_n = z$. As for all $l \in \mathrm{lab}(Q)$, $\alpha(l) \in \{0\}^*\{1\}$ and for each $u \in f(a)$, $v \in g(b)$ with $a \in V$, $b \in W$, $u,v \in \{0\}^*\{1\}$, every sentential form s in

$$A\lfloor 1\bar{a}_0\bar{q}_0\rfloor AA\bar{q}_0A\bar{a}_0AB \Rightarrow_G^* x$$

satisfies

$$s \in \{A\}\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{B\}.$$

Finally, after the production labeled with $\lfloor 3\rfloor$ is used,

$$y \in \{A\}\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{B\}\{A\}.$$

Therefore, the claim holds. □

Claim 18. In

$$\begin{aligned} y &\Rightarrow_G^* z \ [\sigma] \\ &\Rightarrow_G u \ [\lfloor 5r\rfloor] \end{aligned}$$

of the derivation from Claim 16, where $\sigma \in \Xi_4^*$ and $\lfloor 5r\rfloor \in \Xi_5$, every sentential form s in $y \Rightarrow_G^* z$ satisfies

$$s \in T^*\{A\}\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{B\}\{A\},$$

and

$$u \in T^+\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{B\}\{A\}\{A\}.$$

Proof. The productions introduced in step (4) simulate Q's productions of the form $r : (a,b,c,d)$, where $c \in T$ and $d \in W-F$ by analogy with the productions introduced in step (2) except that c is placed in front of the first nonterminal A because $c \in T$ is not further rewritten in Q during the rest of the derivation. Therefore, every sentential form s in $y \Rightarrow_G^* z$ satisfies

$$s \in T^*\{A\}\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{B\}\{A\}.$$

The final step in Q is made by a production with $d \in F$ that is simulated by productions introduced in step (5) of the construction. After its application,

$$u \in T^+\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{A\}\{0,1\}^*\{B\}\{A\}\{A\},$$

so the claim holds. □

Claim 19. In

$$\begin{aligned} u &\Rightarrow_G^* v \quad [\tau] \\ &\Rightarrow_G w_1 \ [\lfloor 8\rfloor] \end{aligned}$$

of the derivation from Claim 16, where $\tau \in \{\lfloor 6\rfloor,\lfloor 7\rfloor\}^*$, every sentential form s in $u \Rightarrow_G^* v$ satisfies

$$s \in T^+\,\mathrm{lab}(G)^+\{A\}\{0,1\}^*\{\#\}^*\{A\}\{0,1\}^*\{\#\}^*\{A\}\{0,1\}^*\{B\}\{A\}\{A\},$$

and $w_1 \in T^+\,\mathrm{lab}(G)^+\{B\}\{\#\}^*$.

Proof. Observe that G properly simulates Q if and only if the substrings over $\{0,1\}$ that follow each of the first three nonterminals A in the sentential form u are identical. The productions introduced in step (6) check if this property holds. As no production labeled with $\lfloor 8 \rfloor, \lfloor 9 \rfloor, \lfloor 10 \rfloor$ rewrites symbols over $\{0,1\}$, and $w \in (T \cup \mathrm{lab}(G))^*$, all symbols from $\{0,1\}$ have to be rewritten with $\lfloor 6 \rfloor$ and $\lfloor 7 \rfloor$; while the first three nonterminals A are moving right in the sentential form, the symbols following the first nonterminal A are replaced with production labels and the symbols following the second and third nonterminal A with #'s. Observe that each step in this derivation has to be leftmost. Therefore, every sentential form s in $u \Rightarrow_G^* v$ satisfies

$$s \in T^+ \mathrm{lab}(G)^+ \{A\}\{0,1\}^*\{\#\}^*\{A\}\{0,1\}^*\{\#\}^*\{A\}\{0,1\}^*\{B\}\{A\}\{A\}.$$

Finally, the production labeled with $\lfloor 8 \rfloor$ is used to rewrite each A to # and, thereby, make $w_1 \in T^+ \mathrm{lab}(G)^+\{B\}\{\#\}^*$. The claim thus holds. □

Claim 20. In

$$\begin{aligned} w_1 &\Rightarrow_G^* w_2 \ [\omega] \\ &\Rightarrow_G w \ \ [\lfloor 10 \rfloor] \end{aligned}$$

of the derivation from Claim 16, where $\omega \in \{\lfloor 9 \rfloor\}^*$, every sentential form s in $w_1 \Rightarrow_G^* w_2$ satisfies $s \in T^+ \mathrm{lab}(G)^+\{B\}\{\#\}^*$, and $w \in T^+ \mathrm{lab}(G)^+$.

Proof. Observe that in order to generate $w \in (T \cup \mathrm{lab}(G))^*$, the first occurrence of # following B is changed to $\lfloor 9 \rfloor$ in every derivation step. After each # is changed in this way, the production labeled with $\lfloor 10 \rfloor$ is applied to obtain $w \in T^+ \mathrm{lab}(G)^+$. □

By Claims 16 through 20, $L(Q) = L(G) /\!/ \mathrm{lab}(G)^+$ and G is a proper leftmost generator of its sentences with their parses. If $\varepsilon \in L$, include $\lfloor 0 \rfloor$ into $\mathrm{lab}(G)$ and a production of the form $\lfloor 0 \rfloor : (S) \to (\lfloor 0 \rfloor)$ into P. At this point, $L = L(Q) \cup \{\varepsilon\} = L(G) /\!/ \mathrm{lab}(G)^+$. Finally, notice that G contains only six nonterminals. Therefore, Theorem 31 holds. ■

Next, we establish a result that is similar to Theorem 31 in terms of proper rightmost generators.

Theorem 32. *For every recursively enumerable language L there exists a propagating scattered context grammar $G = (\bar{V}, \bar{T}, P, S)$ such that G is a proper rightmost generator of its sentences with their parses, $|\bar{V} - \bar{T}| \leq 6$, and $L = L(G) /\!/ \mathrm{lab}(G)^+$.*

Proof. The proper rightmost generator of its sentences with their parses can be constructed analogically to the construction described in the proof of Theorem 31. The constructed grammar contains exactly the same set of nonterminal and terminal symbols with a slightly modified set of productions. Specifically, the productions introduced in steps (1) through (5) of the construction in the proof of Theorem 31 rewrite the sentential form in both the leftmost and the rightmost way and could be used without any modification. However, to construct the rightmost generator, one more nonterminal A is needed in the sentential form, so an additional symbol A has to occur in every of these productions in front of the nonterminal B.

The productions introduced in steps (6) and (7) are changed in the following way:

6. Add
   (a) $\lfloor 6 \rfloor : (A,0,A,0,A,0,A,B,A,A) \to (\lfloor 6 \rfloor A,A,\#,A,\#,A,\#,B,A,A)$,
   (b) $\lfloor 7 \rfloor : (A,1,A,1,A,1,A,B,A,A) \to (\lfloor 7 \rfloor A,A,\#,A,\#,A,\#,B,A,A)$ to P;
7. Add

(a) $\lfloor 8 \rfloor : (A,A,A,A,B,A,A) \rightarrow (\#,\#,\#,\#,\#,\#,BB)$,

(b) $\lfloor 9 \rfloor : (\#,B,B) \rightarrow (B,\lfloor 9 \rfloor,B)$, and

(c) $\lfloor 10 \rfloor : (B,B) \rightarrow (\lfloor 8 \rfloor,\lfloor 10 \rfloor)$ to P.

The productions introduced in step (6) of the construction verify, whether the substrings over $\{0,1\}$ in front of the second, third, and fourth nonterminal are identical. Observe that the productions introduced in this construction make this verification in the rightmost way. Indeed, the productions introduced in step (7) replace the #'s with the rest of the parse; the rightmost generator, however, takes always the rightmost #. The production labeled with $\lfloor 8 \rfloor$ replaces each A with # and places BB at the end of the sentential form. Notably, this production does not insert its label in the sentential form (this insertion takes place later on as explained shortly). The production labeled with $\lfloor 9 \rfloor$ moves the first nonterminal B to the left and replaces # with its label, leaving the second nonterminal B at the end of the sentential form. Finally, when there is no # in the sentential form, the first nonterminal B is replaced with $\lfloor 8 \rfloor$ and the second with $\lfloor 10 \rfloor$. Observe that this grammar is a proper rightmost generator of its sentences with their parses. ■

Concluding this section, let us compare the present result with the results published in [37]. The main result of the paper is expressed by the following theorem.

Theorem 33. *For every recursively enumerable language L, there is a propagating scattered context grammar G which generates its language in a rightmost way and $L = L(G)//\{\$\}^*$.*

Clearly, if we generate the fill-in symbol \$ instead of every production's label in the proof of Theorem 32, we obtain the same result.

Finally, let us look at another aspect of our result. If we use scattered context grammars with erasing productions and substitute each label for the empty word on every production's right-hand side, we characterize every recursively enumerable language. The proofs of Theorems 31 and 32 can, therefore, serve as a constructive proof of this result. In addition, every derivation step is leftmost (or rightmost) so at the same time, we prove the result of [82]. In fact, our result is stronger as leftmost derivations are not explicitly required but it is a property of the constructed grammar that in a successful derivation, every derivation step is leftmost.

Theorem 34. *For every recursively enumerable language L, there is a scattered context grammar G which generates its language in a leftmost way and $L = L(G)$.* ■

6.3 Reduced Generators

Naturally, we want to construct our generators as economically as possible. Therefore, we reduce both the number of their nonterminals and the amount of context checks which need to be performed during a derivation step. This is done by reducing the total number of nonterminals on every production's left-hand side. We demonstrate that for every recursively enumerable language L, there is a propagating scattered context grammar G which represents a proper leftmost generator of its sentences preceded by their parses so that L results from $L(G)$ by eliminating all production labels in $L(G)$. In fact, we achieve this result in two different ways which differ in the total number of the needed nonterminals and in the maximal number of nonterminals on every production's left-hand side.

Theorem 35. *For every recursively enumerable language L there exists a propagating scattered context grammar $G = (\bar{V},\bar{T},P,S)$ such that G is a proper leftmost generator of its sentences preceded by their parses, $|\bar{V} - \bar{T}| \leq 6$, $\mathrm{mcs}(G) = 3$, and $L = \mathrm{lab}(G)^+ \backslash\backslash L(G)$.*

Proof. Let $L \subseteq T^*$ be any recursively enumerable language over an alphabet $T = \{a_1, \ldots, a_n\}$. Then, by Theorem 9, there is an Extended Post Correspondence

$$E = (D, (z_{a_1}, \ldots, z_{a_n})),$$

where $D = \{(u_1, v_1), \ldots, (u_r, v_r)\}$, $u_i, v_i, z_{a_j} \in \{0,1\}^*$ for each $1 \leq i \leq r$, $1 \leq j \leq n$, such that $L(E) = L$. Define the propagating scattered context grammar

$$G = (\{S, A, B, 0, 1, \#\} \cup T \cup \mathrm{lab}(G), T \cup \mathrm{lab}(G), P, S),$$

where

$$\begin{aligned} \mathrm{lab}(G) = {} & \{\lfloor 1 \rfloor, \lfloor 3 \rfloor, \lfloor 3_0 \rfloor, \lfloor 3_1 \rfloor, \lfloor 4 \rfloor, \lfloor 4_0 \rfloor, \lfloor 4_1 \rfloor, \lfloor 4_2 \rfloor\} \\ & \cup \{\lfloor 1_a \rfloor : a \in T\} \cup \{\lfloor 2_{u_i v_i} \rfloor, \lfloor 2_{0u_i v_i} \rfloor : (u_i, v_i) \in D\}, \end{aligned}$$

and P is constructed as follows:

1. For each $a \in T$, add
   (a) $\lfloor 1 \rfloor : (S) \to (\lfloor 1 \rfloor AA)$,
   (b) $\lfloor 1_a \rfloor : (A, A) \to (\lfloor 1_a \rfloor A z_a, Aa)$ to P;
2. For each $(u_i, v_i) \in D$, $1 \leq i \leq r$, add
   (a) $\lfloor 2_{u_i v_i} \rfloor : (A, A) \to (\lfloor 2_{u_i v_i} \rfloor B u_i, B v_i)$,
   (b) $\lfloor 2_{0u_i v_i} \rfloor : (B, B) \to (\lfloor 2_{0u_i v_i} \rfloor B u_i, B v_i)$ to P;
3. Add
   (a) $\lfloor 3 \rfloor : (B, B) \to (\lfloor 3 \rfloor A, B)$,
   (b) $\lfloor 3_0 \rfloor : (A, 0, B, 0) \to (\lfloor 3_0 \rfloor, A, \#, B)$,
   (c) $\lfloor 3_1 \rfloor : (A, 1, B, 1) \to (\lfloor 3_1 \rfloor, A, \#, B)$ to P;
4. Add
   (a) $\lfloor 4 \rfloor : (A, B) \to (\lfloor 4 \rfloor B, A)$,
   (b) $\lfloor 4_0 \rfloor : (B, \#) \to (\lfloor 4_0 \rfloor, B)$,
   (c) $\lfloor 4_1 \rfloor : (B, A) \to (\lfloor 4_1 \rfloor, B)$,
   (d) $\lfloor 4_2 \rfloor : (B) \to (\lfloor 4_2 \rfloor)$ to P.

First, observe that each production introduced in steps (1b) through (4a) contains exactly two nonterminals from the set $\{A, B\}$ on its left and right-hand side. Therefore, there are two nonterminals from this set in every sentential form while these productions are used.

Every successful derivation starts by using the production labeled with $\lfloor 1 \rfloor$ because it is the only production with S on its left-hand side. As no other production contains S on its right-hand side, it is not used during the rest of the derivation process. Therefore, every derivation starts by

$$S \Rightarrow_G \lfloor 1 \rfloor AA \, [\lfloor 1 \rfloor].$$

At this point, only productions from steps (1b) and (2a) are applicable. Productions from (1b) nondeterministically generate a sentence so that each of them adds $a \in T$ behind the second occurrence of A and simultaneously adds a's binary representation behind the first occurrence of A in the sentential form. Finally, a production from step (2a) is used to replace both occurrences of A with

B, so after its application, no production from step (1) can be used. This part of derivation can be formally expressed as

$$\begin{aligned} \lfloor 1 \rfloor AA &\Rightarrow_G^* u\,[\Xi_1] \\ &\Rightarrow_G v\,[p_2], \end{aligned}$$

where $u \in \mathrm{lab}(G)^+\{A\}\{0,1\}^*\{A\}T^*$,

$$v \in \mathrm{lab}(G)^+\{B\}\{0,1\}^+\{B\}\{0,1\}^+T^*,$$

Ξ_1 is a sequence of labels of productions introduced in step (1b), and p_2 is the label of a production introduced in step (2a) of the construction.

Each production from (2) nondeterministically selects some $(u_i, v_i) \in D$, where $1 \leq i \leq r$, and adds u_i and v_i behind the first and the second occurrence of B, respectively. Observe that only productions introduced in step (2b) and the productions labeled with $\lfloor 3 \rfloor$ and $\lfloor 4_2 \rfloor$ are applicable at this point. However, as the production labeled with $\lfloor 4_2 \rfloor$ removes B from the sentential form, its use leads to an unsuccessful derivation. The production labeled with $\lfloor 3 \rfloor$ is used to finish this part of derivation by rewriting the first occurrence of B with A. The corresponding part of derivation is of the form

$$\begin{aligned} v &\Rightarrow_G^* w\,[\Xi_2] \\ &\Rightarrow_G x\ [\lfloor 3 \rfloor], \end{aligned}$$

where

$$\begin{aligned} w &\in \mathrm{lab}(G)^+\{B\}\{0,1\}^+\{B\}\{0,1\}^+T^*, \\ x &\in \mathrm{lab}(G)^+\{A\}\{0,1\}^+\{B\}\{0,1\}^+T^*, \end{aligned}$$

and Ξ_2 is a sequence of labels of productions introduced in step (2b) of the construction.

In greater detail, the sentential form x can be expressed as $x_1Ax_2x_3Bx_4x_5$, where $x_1 \in \mathrm{lab}(G)^+$, $x_2 = u_{s_1} \ldots u_{s_l}$ for some $s_1, \ldots, s_l \in \{1, \ldots, r\}$, $l \geq 1$, $x_3 = z_{b_1} \ldots z_{b_k}$ for some $b_1, \ldots, b_k \in T$, $k \geq 0$, $x_4 = v_{s_1} \ldots v_{s_l}$, and $x_5 = b_1 \ldots b_k$. From the definition of the language represented by an Extended Post Correspondence, for $b_1 \ldots b_k \in L(E)$, $u_{s_1} \ldots u_{s_l} z_{b_1} \ldots z_{b_k} = v_{s_1} \ldots v_{s_l}$, therefore, in the sentential form x, $x_2x_3 = x_4$. This equivalence is verified by the productions labeled with $\lfloor 3_0 \rfloor$ and $\lfloor 3_1 \rfloor$. The production labeled with $\lfloor 3_0 \rfloor$ ($\lfloor 3_1 \rfloor$) replaces two 0's (1's) which follow A and B with A and B and further replaces A and B with $\lfloor 3_0 \rfloor$ ($\lfloor 3_1 \rfloor$) and #, respectively. While these productions are applicable, every sentential form y can be expressed as $y_1Ay_20y_3y_4By_50y_6y_7$ or $y_1Ay_21y_3y_4By_51y_6y_7$, where $y_1 \in \mathrm{lab}(G)^+$, $y_2, y_3, y_5, y_6 \in \{0,1\}^*$, $y_4 \in \{\#\}^*$, $y_7 \in T^*$. Then, the production labeled with $\lfloor 3_0 \rfloor$ or $\lfloor 3_1 \rfloor$ can be applied so that

$$\begin{aligned} &y_1Ay_20y_3y_4By_50y_6y_7 \Rightarrow_G y_1\lfloor 3_0 \rfloor y_2Ay_3y_4\#y_5By_6y_7\ [\lfloor 3_0 \rfloor] \text{ or} \\ &y_1Ay_21y_3y_4By_51y_6y_7 \Rightarrow_G y_1\lfloor 3_1 \rfloor y_2Ay_3y_4\#y_5By_6y_7\ [\lfloor 3_1 \rfloor]. \end{aligned}$$

We demonstrate that $y_2 = y_5 = \varepsilon$ by contradiction.

- Suppose that $y_2 \neq \varepsilon$. As the nonempty string y_2 cannot be removed by any production, y_2 satisfies $y_2 = \varepsilon$.
- Suppose that $y_5 \neq \varepsilon$. Then, y_5 can be removed only by the production labeled with $\lfloor 3_0 \rfloor$ or $\lfloor 3_1 \rfloor$. After its application (in the case when $y_2 = \varepsilon$), we get a sentential form of the form $y'_1y_3y_4\#Ay'_5\#By'_6y_7$, where $y'_1 \in \mathrm{lab}(G)^+$, $y'_5, y'_6 \in \{0,1\}^*$. Then, the substring $y_3y_4\#$ cannot be removed by any production, so y_5 satisfies $y_5 = \varepsilon$.

Therefore, the nonterminals 0 and 1 immediately following A and B have to be rewritten during every derivation step performed by productions labeled with $\lfloor 3_0 \rfloor$ and $\lfloor 3_1 \rfloor$. If all 0's and 1's are removed from the sentential form, the identity $x_2x_3 = x_4$ is verified.

While the productions labeled with $\lfloor 3_0 \rfloor$ and $\lfloor 3_1 \rfloor$ are applicable, other productions cannot be used. Specifically, the use of the productions labeled with $\lfloor 4 \rfloor, \lfloor 4_2 \rfloor$ before all 0's and 1's are removed from the sentential form leads to an unsuccessful derivation. After all 0's and 1's are removed, the production labeled with $\lfloor 4 \rfloor$ is used which finishes this part of derivation. Therefore, this part of derivation can be performed only as follows:

$$\begin{aligned} x &\Rightarrow_G^+ y\ [\Xi_3] \\ &\Rightarrow_G z\ [\lfloor 4 \rfloor], \end{aligned}$$

where $y \in \mathrm{lab}(G)^+\{A\}\{\#\}^+\{B\}T^*$, $z \in \mathrm{lab}(G)^+\{B\}\{\#\}^+\{A\}T^*$, and $\Xi_3 \in \{\lfloor 3_0 \rfloor, \lfloor 3_1 \rfloor\}^+$.

The remaining part of a successful derivation can be expressed as

$$\begin{aligned} z &\Rightarrow_G^+ q_1\ [\Xi_4] \\ &\Rightarrow_G q_2\ [\lfloor 4_1 \rfloor] \\ &\Rightarrow_G q\ \ [\lfloor 4_2 \rfloor], \end{aligned}$$

where $q_1 \in \mathrm{lab}(G)^+\{B\}\{A\}T^*$, $q_2 \in \mathrm{lab}(G)^+\{B\}T^*$, $q \in \mathrm{lab}(G)^+T^*$, and $\Xi_4 \in \{\lfloor 4_0 \rfloor\}^+$. The production labeled with $\lfloor 4_2 \rfloor$ is used in the last derivation step as it removes B from the sentential form. The production labeled with $\lfloor 4_1 \rfloor$ replaces A, which occurs as the last nonterminal in the sentential form, with B so it can be used only if all #'s are removed by previous applications of the production labeled with $\lfloor 4_0 \rfloor$.

Putting together the previous observations, we obtain the derivation of any string $q \in L(E)$ preceded by its parses in the following form:

$$\begin{aligned} S &\Rightarrow_G \lfloor 1 \rfloor AA\ [\lfloor 1 \rfloor] \Rightarrow_G^* u\ \ [\Xi_1] \\ &\Rightarrow_G v\ \ \ \ [p_2]\ \ \Rightarrow_G^* w\ \ [\Xi_2] \\ &\Rightarrow_G x\ \ \ \ [\lfloor 3 \rfloor] \Rightarrow_G^+ y\ \ [\Xi_3] \\ &\Rightarrow_G z\ \ \ \ [\lfloor 4 \rfloor] \Rightarrow_G^+ q_1\ [\Xi_4] \Rightarrow_G q_2\ [\lfloor 4_1 \rfloor] \Rightarrow_G q\ [\lfloor 4_2 \rfloor]. \end{aligned}$$

By examining the derivation and the applied productions, observe that (1) each of the productions adds its label at the end of the sequence of labels in every sentential form and that (2) every derivation step is leftmost. Therefore, G is a proper leftmost generator of sentences preceded by their parses, so Theorem 35 holds. ■

Theorem 36. *For every recursively enumerable language L there exists a propagating scattered context grammar $G' = (\bar{V}', \bar{T}', P', S)$ such that G is a proper leftmost generator of its sentences preceded by their parses, $|\bar{V}' - \bar{T}'| \leq 9$, $\mathrm{mcs}(G) = 1$, and $L = \mathrm{lab}(G)^+ \backslash\backslash L(G)$.*

Proof. Consider the grammar G introduced in the proof of Theorem 35. Define the propagating scattered context grammar

$$G' = (\{S, A, B, C, 0, 1, \$_0, \$_1, \#\} \cup T \cup \mathrm{lab}(G'), T \cup \mathrm{lab}(G'), P', S),$$

where

$$\begin{aligned} \mathrm{lab}(G') = &(\mathrm{lab}(G) - \{\lfloor 3_0 \rfloor, \lfloor 3_1 \rfloor\}) \\ &\cup \{\lfloor 3_{01} \rfloor, \lfloor 3_{02} \rfloor, \lfloor 3_{03} \rfloor, \lfloor 3_{04} \rfloor, \lfloor 3_{11} \rfloor, \lfloor 3_{12} \rfloor, \lfloor 3_{13} \rfloor, \lfloor 3_{14} \rfloor\}, \end{aligned}$$

and P' contains all P's productions except for the productions introduced in step (3). Further, it contains the productions from the following construction:

3. Add

   (a) $\lfloor 3 \rfloor : (B, B) \to (\lfloor 3 \rfloor A, B)$,

(b) i. $\lfloor 3_{01} \rfloor : (B,0) \to (\#,\$_0)$,
ii. $\lfloor 3_{02} \rfloor : (A,\$_0) \to (C,\$_0)$,
iii. $\lfloor 3_{03} \rfloor : (C,0) \to (\lfloor 3_{01} \rfloor \lfloor 3_{02} \rfloor \lfloor 3_{03} \rfloor,\$_0)$,
iv. $\lfloor 3_{04} \rfloor : (\$_0,\$_0) \to (\lfloor 3_{04} \rfloor A,B)$,

(c) i. $\lfloor 3_{11} \rfloor : (B,1) \to (\#,\$_1)$,
ii. $\lfloor 3_{12} \rfloor : (A,\$_1) \to (C,\$_1)$,
iii. $\lfloor 3_{13} \rfloor : (C,1) \to (\lfloor 3_{11} \rfloor \lfloor 3_{12} \rfloor \lfloor 3_{13} \rfloor,\$_1)$,
iv. $\lfloor 3_{14} \rfloor : (\$_1,\$_1) \to (\lfloor 3_{14} \rfloor A,B)$ to P'.

Every successful derivation is performed similarly to G's derivation, only the productions labeled with $\lfloor 3_0 \rfloor$ and $\lfloor 3_1 \rfloor$ are simulated by the productions labeled with $\lfloor 3_{01} \rfloor, \lfloor 3_{02} \rfloor, \lfloor 3_{03} \rfloor, \lfloor 3_{04} \rfloor$ and $\lfloor 3_{11} \rfloor, \lfloor 3_{12} \rfloor, \lfloor 3_{13} \rfloor, \lfloor 3_{14} \rfloor$, respectively. Consider a sentential form of the form $u_1A0u_2u_3B0u_4u_5$, where $u_1 \in \mathrm{lab}(G')^+$, $u_2,u_4 \in \{0,1\}^*$, $u_3 \in \{\#\}^*$, $u_5 \in T^*$. Then,

$$u_1A0u_2u_3B0u_4u_5 \Rightarrow_G u_1\lfloor 3_0 \rfloor Au_2u_3\#Bu_4u_5\ [\lfloor 3_0 \rfloor]$$

in a successful derivation of G. This derivation step is simulated by G' as follows:

$$\begin{aligned}
u_1'A0u_2u_3B0u_4u_5 &\Rightarrow_{G'} u_1'A0u_2u_3\#\$_0u_4u_5 && [\lfloor 3_{01} \rfloor]\\
&\Rightarrow_{G'} u_1'C0u_2u_3\#\$_0u_4u_5 && [\lfloor 3_{02} \rfloor]\\
&\Rightarrow_{G'} u_1'\lfloor 3_{01} \rfloor \lfloor 3_{02} \rfloor \lfloor 3_{03} \rfloor \$_0u_2u_3\#\$_0u_4u_5 && [\lfloor 3_{03} \rfloor]\\
&\Rightarrow_{G'} u_1'\lfloor 3_{01} \rfloor \lfloor 3_{02} \rfloor \lfloor 3_{03} \rfloor \lfloor 3_{04} \rfloor Au_2u_3\#Bu_4u_5 && [\lfloor 3_{04} \rfloor],
\end{aligned}$$

where $u_1' \in \mathrm{lab}(G)^+$. The G's production labeled with $\lfloor 4_1 \rfloor$ is simulated by G' analogously.

Next, we need to demonstrate that the productions introduced in step (3) cannot be used during the rest of the derivation process. We consider two special cases of the derivation. First, consider a sentential form $v_1B0v_2B0v_3v_4$, where $v_1 \in \mathrm{lab}(G)^+$, $v_2,v_3 \in \{0,1\}^*$, $v_4 \in T^*$, which is obtained after the application of a production from (2a). Then,

$$\begin{aligned}
v_1B0v_2B0v_3v_4 &\Rightarrow_{G'} v_1\#\$_0v_2B0v_3v_4 && [\lfloor 3_{01} \rfloor]\\
&\Rightarrow_{G'} v_1\#\$_0v_2\#\$_0v_3v_4 && [\lfloor 3_{01} \rfloor]\\
&\Rightarrow_{G'} v_1\#\lfloor 3_{04} \rfloor Av_2\#Bv_3v_4 && [\lfloor 3_{04} \rfloor].
\end{aligned}$$

However, in this sentential form, the first occurrence of # cannot be removed by any production, so the derivation is not successful.

Second, consider a sentential form $z_1Az_2B00z_3$, where $z_1 \in \mathrm{lab}(G)^+$, $z_2 \in \{\#\}^+$, $z_3 \in T^*$, which is obtained when some 0's behind B are not removed by productions labeled with $\lfloor 3_0 \rfloor$. Then,

$$\begin{aligned}
z_1Az_2B00z_3 &\Rightarrow_{G'} z_1\lfloor 4 \rfloor Bz_2A00z_3 && [\lfloor 4 \rfloor]\\
&\Rightarrow^+_{G'} z_1'BA00z_3 && [\Xi_4]\\
&\Rightarrow_{G'} z_1'\#A\$_00z_3 && [\lfloor 3_{01} \rfloor]\\
&\Rightarrow_{G'} z_1'\#C\$_00z_3 && [\lfloor 3_{02} \rfloor]\\
&\Rightarrow_{G'} z_1'\#\lfloor 3_{01} \rfloor \lfloor 3_{02} \rfloor \lfloor 3_{03} \rfloor \$_0\$_0z_3 && [\lfloor 3_{03} \rfloor]\\
&\Rightarrow_{G'} z_1'\#\lfloor 3_{01} \rfloor \lfloor 3_{02} \rfloor \lfloor 3_{03} \rfloor \lfloor 3_{04} \rfloor ABz_3 && [\lfloor 3_{04} \rfloor],
\end{aligned}$$

where $z_1' \in \mathrm{lab}(G)^+$ and $\Xi_4 \in \{\lfloor 4_0 \rfloor\}^+$. Again, the symbol # cannot be removed by any production, so the derivation is not successful. Other special cases and more detailed proof of this theorem is left to the reader. ■

In conclusion of this chapter, let us recall that we have demonstrated that for every recursively enumerable language, there exists a propagating scattered context grammar that generates the language's sentences followed by their parses. In addition, we have proved analogical results for canonical and reduced versions of these generators. This kind of generation is specific to scattered context grammars—we can hardly base the generation of sentences with their parses upon classical sequential rewriting mechanisms such as context-free or context-sensitive grammars. Probably, some propagating parallel rewriting mechanisms, such as propagating PC grammar systems (see Chapter 4 in Volume 2 of [62]), can be used in this way. Furthermore, some propagating regulated grammars, such as propagating matrix grammars (see Chapter 3 in Volume 3 of [62]), seem to be suitable for this generation as well. Apart from parses, there are more kinds of information which can be stored during the derivation of a string (positions of nonterminals rewritten during the derivation, string encoded derivation trees and other). We suggest these ideas for future investigation.

Chapter 7

Applications in Linguistics

So far, we have studied theoretical properties of scattered context grammars. This section discusses their practical use in linguistics.

In sentences of natural languages such as English, we can find words which depend on each other and which, on the other hand, are not direct neighbors. For example, consider the following sentence:

He usually goes to work early.

The subject (*he*) and the predicator (*goes*) are related; sentences

He usually go to work early.

and

I usually goes to work early.

are ungrammatical because the form of the predicator depends on the subject and the above combination is illegal. Clearly, to change person, the verb has to reflect this change as well. This is the kind of dependency that can be very easily captured by scattered context grammars. Let us construct a scattered context grammar that contains the following production:

$$(\text{He}, \text{goes}) \rightarrow (\text{We}, \text{go}).$$

This production checks whether the subject is *he* and whether the verb *go* is in third person singular. If the sentence satisfies this property, it can be transformed to the grammatically correct sentence

We usually go to work early.

Observe that the related words do not necessarily have to be direct neighbors. In the above example, the word *usually* is located between the subject and the predicator. While it is fairly easy to use context-sensitive grammars to model context dependencies where only one word is located between the related words, note that the number of words between the subject and the predicator is virtually unlimited. We can say

He almost regularly goes to work early.

but also

He sometimes, but not always, goes to work early.

To model context dependency of this kind by ordinary context-sensitive grammars, many auxiliary productions have to be introduced to propagate the information about the form of one word to the other word which may be located at the opposite end of the sentence. Still, the introduced scattered context production can be used, no matter how many words are between the subject and the predicator.

We give another example demonstrating why scattered context grammars are more economical in certain situations. Consider the following two sentences:

John recommended it.

and

Did John recommend it?

There is a relation between the basic clause and its interrogative counterpart. Indeed, we get the second, interrogative clause by adding *did* in front of *John* and by changing *recommended* to *recommend* while keeping the rest of the sentence untouched. In terms of scattered context grammars, this transformation can be described by the scattered context production

$$(\text{John, recommended}) \rightarrow (\text{Did John}, \text{recommend});$$

clearly, when applied to the first sentence, this production performs exactly the same transformation as we have just described. Although this transformation is possible by an ordinary context production, the inverse transformation is more problematic. The inverse transformation can be performed by a scattered context production

$$(\text{Did, recommend}) \rightarrow (\varepsilon, \text{recommended});$$

obviously, by erasing *did* and changing *recommend* to *recommended*, we obtain the first sentence. Again, instead of *John* the subject may consist of a noun phrase containing several words. The advantage of scattered context grammars is apparent—scattered context grammars permit us to change only some words during the transformation while keeping the other words untouched. On the other hand, context-sensitive productions are not suitable for this kind of transformations because many unnecessary context-sensitive productions have to be introduced to propagate the change made at one end of the sentence to the other end.

In the following text, we study these transformations more deeply. First, we introduce the needed notation and some conventions to be able to describe the transformations in a more exact and general way. Then, we illustrate the use of scattered context grammars for transformations of one kind of English sentences to another. Finally, we mention several areas where similar approach can be used and give some ideas for further investigation in this area.

Overview of Used Linguistic Terms

This section introduces some basic linguistic terms we use later in this chapter. Regarding the used linguistic terms and related notions, we use the terminology of [19, 20].

We focus our attention on verbs and personal pronouns because the form of the words in these categories depends on the context in which these words are used. For example, *is, are, was*, and *been* are different forms of the verb *be*. We say that words in these categories *inflect* and call this property *inflection*. Verbs and personal pronouns often represent the key elements of a clause—the *subject* and the *predicate*. In simple clauses like

She loves him.

we can understand the notion of the subject and the predicate so that some information is "predicated of" the subject (*she*) by the predicate (*loves him*). In more complicated clauses, the best way to determine the subject and the predicate is by examining their syntactic properties (see [19, 20] for more details). The predicate is a *verb phrase*—the most important word of a verb phrase is the verb, also called the *predicator*. In some verb phrases, there are more verbs present. For example, in the sentence

He has been working for hours.

the verb phrase contains three verbs. The predicator is, however, always the first verb of a verb phrase (*has* in the above example). We focus on the most elementary clauses—*canonical clauses*. The subject in these clauses always precedes the predicate, the clause is positive (without negation), declarative, and without subordinate and coordinate clauses.

The following paragraphs describe the basic categorization of verbs and personal pronouns and further characterize their inflectional forms.

Verbs

We distinguish several kinds of verbs depending on their grammatical behavior. The set of all verbs is divided into two subsets: *auxiliary verbs* and *lexical verbs*. Further, the set of auxiliary verbs consists of *modal* and *non-modal verbs*. The set of modal verbs includes the following verbs: *can*, *may*, *must*, *will*, *shall*, *ought*, *need*, *dare*; the verbs *be*, *have*, and *do* are non-modal. All the remaining verbs are lexical. It needs to be said that the above defined classes overlap in certain situations—for example, there are sentences, where *do* appears as an auxiliary verb and in different situations it behaves as a lexical verb. For simplicity, we do not take into account these special cases in the following text.

Inflectional forms of verbs are called *paradigms*. In English, every verb, except of the verb *be*, may appear in each of the six paradigms described in Table 1 (see [19, 20]). Verbs in *primary form* may occur as the only verb in a clause and form the head of a verb phrase; on the other hand, verbs in *secondary form* have to be accompanied by a verb in primary form.

	Paradigm	Person	Example
Primary form	Present	3rd sg	*She walks home.*
		Other	*They walk home.*
	Preterite	—	*She walked home.*
Secondary form	Plain form	—	*They should walk home.*
	Gerund-participle		*She is walking home.*
	Past participle		*She has walked home.*

Table 1: Paradigms of English verbs

The verb *be* has nine paradigms in its neutral form. All primary forms have, in addition, their negative contracted counterparts. Compared to other verbs, there is one more verb paradigm called *irrealis*. The irrealis form *were* (and *weren't*) is used in sentences of an unrealistic nature, such as

I wish I were rich.

All these paradigms are shown in Table 2.

	Paradigm	Person	Neutral	Negative
Primary forms	Preterite	1st sg, 3rd sg	*was*	*wasn't*
		Other	*were*	*weren't*
	Present	1st sg	*am*	*aren't*
		3rd sg	*is*	*isn't*
		Other	*are*	*aren't*
	Irrealis	1st sg, 3rd sg	*were*	*weren't*
Secondary forms	Plain form	—	*be*	—
	Gerund-participle		*being*	—
	Past participle		*been*	—

Table 2: Paradigms of the verb *be*

Personal pronouns

Personal pronouns exhibit a great amount of inflectional variation as well. Table 3 summarizes all their inflectional forms. The most important for us is the class of pronouns in *nominative* because these pronouns often appear as the subject of a clause.

	Non-reflexive				Reflexive
	Nominative	Accusative	Genitive		Plain
	Plain		Dependent	Independent	
I	*I*	*me*	*my*	*mine*	*myself*
you	*you*		*your*	*yours*	*yourself*
he	*he*	*him*	*his*		*himself*
she	*she*	*her*	*her*	*hers*	*herself*
it	*it*		*its*		*itself*
we	*we*	*us*	*our*	*ours*	*ourselves*
you	*you*		*your*	*yours*	*yourselves*
they	*they*	*them*	*their*	*theirs*	*themselves*

Table 3: Personal pronouns

Transformational Scattered Context Grammars

As we have already mentioned in the introduction of this chapter, we use scattered context grammars to transform one kind of sentences to another. To be able to formally describe these transformations, we define a special type of a scattered context grammar which does not start its derivation from the start symbol but which transforms sentences over an input vocabulary to sentences over an output vocabulary.

Definition 36. A *transformational scattered context grammar* is a quadruple

$$G = (V, T, P, I),$$

where

- V is the total vocabulary,
- $T \subset V$ is the set of terminals (or the *output vocabulary*),

- P is a set of scattered context productions,
- $I \subset V$ is the *input vocabulary*.

The derivation step is defined as in the case of scattered context grammars. The *transformation T defined by G* is denoted by $T(G)$ and defined as

$$T(G) = \{(x,y) : x \Rightarrow_G^* y, x \in I^*, y \in T^*\}.$$

If $(x,y) \in T(G)$, we say that *x is transformed to y* by G; x and y are called the *input* and the *output sentence*, respectively.

Example 5. Define the transformational scattered context grammar

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

where

$$P = \{(A,B,C) \to (a,bb,c)\}.$$

For example, for the input sentence *AABBCC*, one of the possible derivations is:

$$AABBCC \Rightarrow_G aABbbcC \Rightarrow_G aabbbbcc.$$

Therefore, the input sentence $AABBCC \in I^*$ is transformed to the output sentence $aabbbbcc \in T^*$, and $(AABBCC, aabbbbcc) \in T(G)$. If we restrict the input sentences to the language $\{A^nB^nC^n : n \geq 1\}$, we get

$$\{(A^nB^nC^n, a^nb^{2n}c^n) : n \geq 1\} \subseteq T(G),$$

so every $A^nB^nC^n$, where $n \geq 1$, is transformed to $a^nb^{2n}c^n$.

Transformational Scattered Context Grammars in Linguistics

This section uses transformational scattered context grammars for transformations of English sentences. For our purposes, we assume that the English vocabulary is finite. While it is possible to create new words by modifying or combining existing words (and thus obtaining an infinite vocabulary), the set of words commonly used in books and newspapers can be regarded as fixed. Therefore, we can talk about the finite *set of all English words*, which we denote by T in what follows. Next, we subdivide this set into subsets with respect to the above mentioned classification of verbs and pronouns:

- T is the set of all words, including all their inflectional forms,
- $T_{\mathrm{V}} \subset T$ is the set of all verbs, including all their inflectional forms,
- $T_{\mathrm{VA}} \subset T_{\mathrm{V}}$ is the set of all auxiliary verbs, including all their inflectional forms,
- $T_{\mathrm{Vpl}} \subset T_{\mathrm{V}}$ is the set of all verbs in plain form,
- $T_{\mathrm{PPn}} \subset T$ is the set of all personal pronouns in nominative.

To be able to describe all possible paradigms of a verb $v \in T_{\mathrm{Vpl}}$, we use the following notation:

- $\pi_{3\mathrm{rd}}(v)$ is the verb v in third person singular present,

- $\pi_{\text{pres}}(v)$ is the verb v in present, other than third person singular,
- $\pi_{\text{pret}}(v)$ is the verb v in preterite.

There are several conventions we use throughout the text:

1. We do not take into account capitalization and punctuation. Therefore, for us,

   He is your best friend.

   and

   he is your best friend

   are equivalent.

2. To make the examples simple and readable, we expect every input sentence to be a canonical clause. In some examples, we make slight exceptions—for instance, sometimes we permit the input sentence to be negative. The first example also demonstrates a simple type of coordination.

3. The input vocabulary is the set $I = \{\langle x \rangle : x \in T\}$, where T is the set of all English words as stated above. As a result, the transformational grammar takes an input sentence over I, for example

   $$\langle \text{he} \rangle \langle \text{is} \rangle \langle \text{your} \rangle \langle \text{best} \rangle \langle \text{friend} \rangle$$

   and transforms it to an output sentence over T, for instance

   is he your best friend

   in the case of the declarative-to-interrogative transformation. As we have already mentioned, we omit punctuation and capitalization for simplicity, so in fact, the above sentence corresponds to

   Is he your best friend?

The following four examples illustrate how to transform one kind of English sentences to another by using transformational scattered context grammars.

Clauses with *neither* and *nor*

The first, simplest example shows how to use transformational scattered context grammars to negate clauses which contain the pair of the words *neither* and *nor*, such as

Neither Thomas nor his wife went to the party.

The words *neither* and *nor* are related but there is no limitation on the number of words appearing between them. The following transformational scattered context grammar G converts the above sentence to

Both Thomas and his wife went to the party.

Set

$$G = (V, T, P, I),$$

where $V = T \cup I$, and P is defined as follows:

$$\begin{aligned} P = \{&(\langle \text{neither} \rangle, \langle \text{nor} \rangle) \to (\text{both, and})\} \\ &\cup \{(\langle x \rangle) \to (x) : x \in T - \{\text{neither, nor}\}\}. \end{aligned}$$

For example, for the above sentence, the transformation can proceed as follows:

$$\begin{aligned} &\langle \text{neither} \rangle \langle \text{thomas} \rangle \langle \text{nor} \rangle \langle \text{his} \rangle \langle \text{wife} \rangle \langle \text{went} \rangle \langle \text{to} \rangle \langle \text{the} \rangle \langle \text{party} \rangle \\ \Rightarrow_G\ &\text{both } \langle \text{thomas} \rangle \text{and } \langle \text{his} \rangle \langle \text{wife} \rangle \langle \text{went} \rangle \langle \text{to} \rangle \langle \text{the} \rangle \langle \text{party} \rangle \\ \Rightarrow_G\ &\text{both thomas and } \langle \text{his} \rangle \langle \text{wife} \rangle \langle \text{went} \rangle \langle \text{to} \rangle \langle \text{the} \rangle \langle \text{party} \rangle \\ \Rightarrow_G\ &\text{both thomas and his } \langle \text{wife} \rangle \langle \text{went} \rangle \langle \text{to} \rangle \langle \text{the} \rangle \langle \text{party} \rangle \\ \Rightarrow_G^5\ &\text{both thomas and his wife went to the party.} \end{aligned}$$

The production

$$(\langle \text{neither} \rangle, \langle \text{nor} \rangle) \to (\text{both, and})$$

replaces *neither* and *nor* with *both* and *and*, respectively. Every other word $\langle w \rangle \in I$ is changed to $w \in T$.

Existential clauses

In English, clauses which indicate an existence are called *existential*. These clauses are usually formed by the dummy subject *there*, for example

There was a nurse present.

However, this dummy subject is not mandatory in all situations. For example, the above example can be rephrased as

A nurse was present.

We construct a transformational scattered context grammar G which converts an existential clause without the dummy subject *there* to an equivalent existential clause with *there*.

Set

$$G = (V, T, P, I),$$

where $V = T \cup I \cup \{X\}$, and P is defined as follows:

$$\begin{aligned} P = \{&(\langle x \rangle, \langle \text{is} \rangle) \to (\text{there is } xX, \varepsilon), \\ &(\langle x \rangle, \langle \text{are} \rangle) \to (\text{there are } xX, \varepsilon), \\ &(\langle x \rangle, \langle \text{was} \rangle) \to (\text{there was } xX, \varepsilon), \\ &(\langle x \rangle, \langle \text{were} \rangle) \to (\text{there were } xX, \varepsilon) : x \in T\} \\ \cup\ &\{(X, \langle x \rangle) \to (X, x) : x \in T\} \\ \cup\ &\{(X) \to (\varepsilon)\}. \end{aligned}$$

For the above sample sentence, we get the following derivation:

$$\begin{aligned} &\langle \text{a} \rangle \langle \text{nurse} \rangle \langle \text{was} \rangle \langle \text{present} \rangle \\ \Rightarrow_G\ &\text{there was a } X \langle \text{nurse} \rangle \langle \text{present} \rangle \\ \Rightarrow_G\ &\text{there was a } X \text{ nurse } \langle \text{present} \rangle \\ \Rightarrow_G\ &\text{there was a } X \text{ nurse present} \\ \Rightarrow_G\ &\text{there was a nurse present.} \end{aligned}$$

A production from the first set has to be applied first as there is no symbol X in the sentential form and all other productions require X to be present in the sentential form. In our case, the production

$$(\langle \text{a} \rangle, \langle \text{was} \rangle) \to (\text{there was a } X, \varepsilon)$$

is applied; the use of other productions from this set depends on tense used in the input sentence and whether the subject is in singular or plural. The production nondeterministically selects the first word of the sentence, puts *there was* in front of it and the symbol X behind it; in addition, it erases the word *was* in the middle of the sentence. Next, all words $\langle w \rangle \in I$ are replaced with $w \in T$ by productions from the second set. These productions also verify that the previously nondeterministically selected word was at the beginning of the sentence—if not, there is a word $\langle w \rangle \in I$ in front of X which cannot be deleted. Finally, the derivation ends by erasing X from the sentential form.

Interrogative Clauses

Depending on the predicator, there are two possibilities of how declarative clauses are transformed into interrogative. If the predicator is an auxiliary verb, the interrogative clause is formed by simply swapping the subject and the predicator. For example, we get the interrogative clause

Is he mowing the lawn?

by swapping *he*, which is the subject, and *is*, which is the predicator, in

He is mowing the lawn.

If the predicator is a lexical verb, the interrogative clause is formed by adding the dummy *do* at the beginning of the declarative clause—the dummy *do* has to be of the same paradigm as the predicator in the declarative clause. The predicator itself is converted to its plain form.

She usually gets up early.

is a declarative clause with the predicator *gets*, which is in third person singular, and the subject *she*. By inserting *do* in third person singular at the beginning of the sentence and converting *gets* to its plain form, we obtain

Does she usually get up early?

To simplify the following transformational scattered context grammar G which performs this conversion, we assume that the subject is a personal pronoun in nominative.

Set

$$G = (V, T, P, I),$$

where $V = T \cup I \cup \{X\}$, and P is defined as follows:

$$\begin{aligned} P = {} & \{(\langle p \rangle, \langle v \rangle) \to (vp, X) : v \in T_{\text{VA}}, p \in T_{\text{PPn}}\} \\ & \cup \{(\langle p \rangle, \langle \pi_{\text{pret}}(v) \rangle) \to (\text{did } p, vX), \\ & \quad (\langle p \rangle, \langle \pi_{\text{3rd}}(v) \rangle) \to (\text{does } p, vX), \\ & \quad (\langle p \rangle, \langle \pi_{\text{pres}}(v) \rangle) \to (\text{do } p, vX) : v \in T_{\text{Vpl}} - T_{\text{VA}}, p \in T_{\text{PPn}}\} \\ & \cup \{(\langle x \rangle, X) \to (x, X), \\ & \quad (X, \langle y \rangle) \to (X, y) : x \in T - T_{\text{V}}, y \in T\} \\ & \cup \{(X) \to (\varepsilon)\}. \end{aligned}$$

For sentences containing an auxiliary verb as the predicator, the transformation by G looks as follows:

$$\begin{aligned}
&\langle\text{he}\rangle\langle\text{is}\rangle\langle\text{mowing}\rangle\langle\text{the}\rangle\langle\text{lawn}\rangle\\
\Rightarrow_G\ &\text{is he } X\langle\text{mowing}\rangle\langle\text{the}\rangle\langle\text{lawn}\rangle\\
\Rightarrow_G\ &\text{is he } X \text{ mowing } \langle\text{the}\rangle\langle\text{lawn}\rangle\\
\Rightarrow_G\ &\text{is he } X \text{ mowing the } \langle\text{lawn}\rangle\\
\Rightarrow_G\ &\text{is he } X \text{ mowing the lawn}\\
\Rightarrow_G\ &\text{is he mowing the lawn.}
\end{aligned}$$

The derivation starts by a production from the first set which swaps the subject and the predicator and puts X behind them. Next, productions from the third set are used to rewrite every word $\langle w\rangle \in I$ to $w \in T$. Finally, X is removed from the sentential form.

The transformation of sentences in which the predicator is a lexical verb is more complicated:

$$\begin{aligned}
&\langle\text{she}\rangle\langle\text{usually}\rangle\langle\text{gets}\rangle\langle\text{up}\rangle\langle\text{early}\rangle\\
\Rightarrow_G\ &\text{does she } \langle\text{usually}\rangle \text{ get } X\langle\text{up}\rangle\langle\text{early}\rangle\\
\Rightarrow_G\ &\text{does she usually get } X\langle\text{up}\rangle\langle\text{early}\rangle\\
\Rightarrow_G\ &\text{does she usually get } X \text{ up } \langle\text{early}\rangle\\
\Rightarrow_G\ &\text{does she usually get } X \text{ up early}\\
\Rightarrow_G\ &\text{does she usually get up early.}
\end{aligned}$$

As the predicator is in third person singular, a production from

$$\{(\langle p\rangle, \langle \pi_{\text{3rd}}(v)\rangle) \to (\text{does } p, vX) : v \in T_{\text{Vpl}} - T_{\text{VA}}, p \in T_{\text{PPn}}\}$$

is applied. It inserts *does* at the beginning of the sentence, converts the predicator *gets* to its plain form *get*, and puts X behind it. Next, productions from

$$\{(\langle x\rangle, X) \to (x, X) : x \in T - T_{\text{V}}\}$$

are used to rewrite the remaining words $\langle w\rangle \in I$ in front of the predicator to $w \in T$. They do not rewrite verbs—this way the grammar ensures that the first verb in a sequence was previously chosen as the predicator. For instance, in the sentence

He has been working for hours.

has has to be selected as the predicator—otherwise the derivation is unsuccessful. Finally, the grammar rewrites all words behind X and erases X as in the previous example.

Question Tags

Question tags are special constructions which are used in spoken language. They are mini-questions that are appended to a declarative clause to ask the other person for confirmation:

Your sister is married, isn't she?

The polarity of question tags is always the opposite of the polarity of the main clause—if the main clause is positive, the question tag is negative and vice versa. If the predicator is an auxiliary verb, the question tag is formed by the same auxiliary verb. For lexical verbs, the question tag is made by using the verb *do*:

He plays the violin, doesn't he?

There are some special cases which have to be taken into account. First, the verb *be* has to be treated separately as it has more paradigms than other verbs and the question tag for the first person is irregular:

I am always right, aren't I?

Second, for the verb *have*, the question tag depends on whether it is used as an auxiliary verb or a lexical verb. In the first case *have* is used in the question tag, such as

He has been working hard, hasn't he?

in the latter case, the auxiliary *do* is used:

They have a dog, don't they?

To explain the basic concepts as simply as possible, we omit the special cases of the verb *have* in the following transformational scattered context grammar G. We also only sketch its construction and do not mention all created productions explicitly. In addition, we suppose that the subject is represented by a personal pronoun.

Set

$$G = (V, T, P, I),$$

where $V = T \cup I \cup \{X, Y\}$, and P is defined as follows:

$$\begin{aligned}
P = \{&(\langle p\rangle, \langle \text{will}\rangle, \langle x\rangle) \to (p, \text{will } X, Yx \text{ won't } p),\\
&(\langle p\rangle, \langle \text{won't}\rangle, \langle x\rangle) \to (p, \text{won't } X, Yx \text{ will } p),\\
&\ldots : p \in T_{\text{PPn}}, x \in T\}\\
\cup\ \{&(\langle \text{I}\rangle, \langle \text{am}\rangle, \langle x\rangle) \to (\text{I}, \text{am } X, Yx \text{ aren't I}),\\
&(\langle \text{you}\rangle, \langle \text{are}\rangle, \langle x\rangle) \to (\text{you}, \text{are } X, Yx \text{ aren't you}),\\
&\ldots : x \in T\}\\
\cup\ \{&(\langle p\rangle, \langle v\rangle, \langle x\rangle) \to (p, v\ X, Yx \text{ doesn't } p),\\
&(\langle q\rangle, \langle v\rangle, \langle x\rangle) \to (q, v\ X, Yx \text{ don't } q) :\\
&p \in \{\text{he, she, it}\}, q \in T_{\text{PPn}} - \{\text{he, she, it}\}, v \in T_{\text{V}} - T_{\text{VA}}, x \in T\}\\
&\vdots\\
\cup\ \{&(\langle x\rangle, X) \to (x, X),\\
&(X, \langle y\rangle, Y) \to (X, y, Y) : x \in T - T_{\text{V}}, y \in T\}\\
\cup\ \{&(X, Y) \to (\varepsilon, \varepsilon)\}.
\end{aligned}$$

First, we describe the generation of question tags for clauses in which the predicator is an auxiliary verb:

$$\begin{aligned}
&\langle \text{I}\rangle\langle \text{am}\rangle\langle \text{always}\rangle\langle \text{right}\rangle\\
\Rightarrow_G\ &\text{I am } X\langle \text{always}\rangle Y \text{ right aren't I}\\
\Rightarrow_G\ &\text{I am } X \text{ always } Y \text{ right aren't I}\\
\Rightarrow_G\ &\text{I am always right aren't I.}
\end{aligned}$$

Here, the production

$$(\langle \text{I}\rangle, \langle \text{am}\rangle, \langle \text{right}\rangle) \to (\text{I}, \text{am } X, Y \text{ right aren't I})$$

initiates the derivation. When it finds *I am* at the beginning of the sentence, it generates the question tag *aren't I* at its end. In addition, it adds X behind *I am* and Y in front of *aren't I*. Next, it rewrites all words from $\langle w\rangle \in I$ to $w \in T$. Again, it ensures that the predicator was chosen properly by productions from

$$\{(\langle x\rangle, X) \to (x, X) : x \in T - T_{\text{V}}\}.$$

In addition, productions from

$$\{(X,\langle y\rangle,Y)\rightarrow(X,y,Y) : x\in T-T_{\mathrm{V}}, y\in T\}$$

are used to check whether the question tag was placed at the very end of the sentence. If not, there remains some symbol from the input vocabulary behind Y which cannot be rewritten. Finally, the last production removes X and Y from the sentential form.

When the predicator is a lexical verb in present, the question tag is formed by *does* or *do* depending on person in which the predicator occurs:

$$\begin{aligned}&\langle\text{he}\rangle\langle\text{plays}\rangle\langle\text{the}\rangle\langle\text{violin}\rangle\\ \Rightarrow_G\ &\text{he plays } X\langle\text{the}\rangle Y \text{ violin doesn't he}\\ \Rightarrow_G\ &\text{he plays } X \text{ the violin } Y \text{ doesn't he}\\ \Rightarrow_G\ &\text{he plays the violin doesn't he.}\end{aligned}$$

Concluding Notes

The aim of this chapter was to demonstrate that in natural languages, there exist many cases of scattered context relations between individual elements of a sentence. The above examples were chosen with respect to simplicity of the resulting grammar—there exist more examples of scattered context relations than we have presented. Some of them appear when input sentences of more complicated structure are permitted as the input. For instance, relative clauses are introduced by *who* or *which* depending on the subject of the main clause. If the subject in the main clause is a person, the relative clause is introduced by *who*; otherwise it starts by *which*. Similarly, transformations of sentences in active voice to sentences in passive voice (and vice versa) require scattered context processing as well. Finally, if we do not restrict our interest to the English language, we get many more examples of scattered context dependency. For instance, in Spanish, all adjectives inflect according to gender of the noun they characterize. Again, both the noun and the adjective may appear at different parts of a sentence making it hard to capture their relationship by classical context grammars.

As we have just demonstrated, there are many uses of scattered context grammars in natural language description and processing. We recommend this area as a perspective topic for further study.

Part III

Summary and Conclusion

Chapter 8

Conclusion

The object of the present thesis was a study of scattered context grammars which represent one of the most natural formalisms of scattered information description and processing. Since their introduction, these grammars have been intensively studied over the past four decades and many research papers have been written about this topic. Our aim was to further investigate theoretical properties of scattered context context grammars to better understand, describe, and process mutually related, but not directly neighboring pieces of information. The research was carried out in four main areas.

First, we studied a classical topic of formal language theory—the removal of erasing productions. When describing a language, we use formal grammars for this description. The grammars which contain erasing productions are often simpler and the total number of the productions they contain is lower; on the other hand, parsers based on these grammars cannot be constructed as efficiently as in the case when these grammars do not contain erasing productions at all. Clearly, in general, erasing productions cannot be removed from scattered context grammars as the generative power of these grammars with erasing productions and without them is different. However, there are certain grammars which, even though they contain erasing productions, characterize a language which lies within the family of context-sensitive languages. We demonstrated that if such grammars erase their symbols in a certain way, these grammars can be converted to equivalent grammars without erasing productions. We call this erasing as generalized k-limited because during any derivation, between every two neighboring symbols which are not erased later in the derivation, there may appear at most k symbols, which are later erased.

Second, several restricted and extended versions of scattered context grammars were discussed. We started by considering scattered context grammars whose components, in contrary to the basic definition, were other than context-free; that is, right-linear, linear, context-sensitive and unrestricted. An infinite hierarchy of languages was obtained for grammars containing both right-linear and linear components. Specifically, it was proved that each family of languages generated by these grammars with starting productions containing at most n nonterminals on their right-hand sides is properly included in the family of languages generated by these grammars with starting productions containing at most $n+1$ nonterminals. It is practical to study these kinds of restrictions because usually, when creating a model of some system, we require the model to be as simple as possible. This results in a simpler description and a more efficient implementation of the resulting system. We continued by discussing derivations which may occur only within the first n nonterminals of the sentence. It was demonstrated that propagating scattered context grammars whose derivations are limited to n first nonterminals are less powerful than the grammars whose derivations are limited to $n+1$ nonterminals. This demonstrates that if a compiler based on propagating scattered context grammars restricts its context-dependency checks to only a finite part of the sentence, it will be always less powerful than if no such a restriction exists. Next, we revived the already existing

proof demonstrating that context-sensitive grammars are equivalent to propagating scattered context grammars if the latter use only leftmost or rightmost derivations and presented a much simpler proof of this result. When parsing a sentence, compiles usually simulate the leftmost or the rightmost derivation of the sentence. In fact, this requirement helps us to obtain the whole family of context-sensitive languages while without it, the exact power of propagating scattered grammars is not exactly known. Finally, the thesis studied another restriction which enabled us to describe all context-sensitive languages. This restriction requires that in every derivation step, a production which rewrites the maximal or the minimal number of nonterminals is chosen.

Third, we introduced generators of sentences with their parses. These generators are propagating scattered context grammars that generate sentences followed by a sequence of labels of productions which were used during their derivation. We established a characterization of recursively enumerable languages based upon this kind of generators by scattered context grammars without erasing productions. We also proved that there exist leftmost and rightmost generators with this property and we reduced the total number of the needed nonterminals and context-sensitive productions to a finite number no matter which recursively enumerable language is characterized.

Fourth, the use of scattered context grammars in linguistics was discussed. We demonstrated that in natural languages, context relations between not directly neighboring elements of a sentence are relatively common. We gave several examples of sentence transformations and sketched possible applications of scattered context grammars in linguistics.

Further Investigation

There are two main areas which are suitable for further investigation of scattered context grammars: their practical use in linguistics and compiler construction, and further study of their theoretical properties. We start by looking at their possible practical applications.

As we have demonstrated in Chapter 7, scattered context grammars can be used for description of natural languages and their transformations. It may be, therefore, convenient to use scattered context grammars as a device for context dependency description in processors of natural languages. These processors may be used to analyze and possibly check the correctness of their input sentences. For instance, spellcheckers, widely used in text processors, are usually based only on a dictionary which is used to compare the words from the input with the words contained in the dictionary. A more sophisticated way of spell-checking would require some model of the checked language to, for example, verify whether the combination of the subject and the predicator is legal. As scattered context grammars are ideal for capturing this kind of dependencies, they seem to be a proper candidate for language description. The transformations discussed in Chapter 7 could be even extended so that the input sentence is in one language while the output sentence is in another. This would make it possible to use scattered context grammars for computer-based translation. This approach might be successful because sentences in different languages have different word orders. German language, for instance, exhibits a tendency to put verbs at the end of the sentence while in English, verbs come right behind the subject. There are surely other examples of the use of scattered context grammars in linguistics so we recommend this area for further investigation.

Another application area of scattered context grammars might be found in construction of parallel compilers. Effective design of parallel compilers is currently under a very intensive development (see [2, 83]). Scattered context grammars could be used in parallel compilers as a model of synchronization between individual parallel branches. While every parallel branch may use context-free productions for effective parsing, scattered context productions could serve for the needed information interchange between these branches. This kind of synchronization would, for example, happen

only if a specified nonterminal appears in the sentential form of a branch. This branch would then request a simulation of the scattered context production so it would be performed in other selected branches as well. As the left-hand side of a scattered context production may contain an arbitrary number of nonterminals, it can flexibly select only those branches which are needed for the information interchange. After this communication step, all branches would continue parsing in parallel way.

Even sequential compiling benefits from additional scattered context checks as demonstrated by [63, 64]. These papers use a restricted version of propagating scattered context grammars to improve parsing of programming languages. The authors use scattered context productions to describe every variable's declaration and use. This way, they check whether a variable was declared before its use and, in addition, perform type checks at the same time. This is all done within the parsing phase of the compilation process without any need of a symbol table.

Finally, this theses, as a theoretical approach to the problematics of scattered context, introduced several new open problems to formal language theory. In Chapter 4, we eliminated erasing productions form a scattered context grammar if this grammar erased its nonterminals in a generalized k-limited way. The result rises, however, two open problems. First, are we able for given k and a grammar G to decide whether G erases its nonterminals in a generalized k-limited way? Second, does this type of erasing cover all cases in which erasing productions can be eliminated from a scattered context grammar? In Section 5.2, we restricted derivations so that at most n first nonterminals could be rewritten. Is it possible to construct a propagating scattered context grammar which satisfies this property implicitly without any modification of its definition? Finally, the long-standing open problem of whether scattered context grammars without erasing productions characterize the family of context-sensitive languages remains still open. Hopefully, this long-open problem will be solved one day in the future.

Bibliography

[1] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1988.

[2] G. Almási, C. Cascaval, and P. Wu, editors. *Languages and Compilers for Parallel Computing*. Springer, 2007.

[3] J. G. Brookshear. *Theory of Computation*. Benjamin/Cummings, 1989.

[4] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[5] A. B. Cremers. Normal forms for context-sensitive grammars. *Acta Informatica*, 3:59–73, 1973.

[6] J. Dassow and Gh. Paun. *Regulated Rewriting in Formal Language Theory*. Akademie-Verlag, 1989.

[7] A. Ehrenfeucht and G. Rozenberg. An observation on scattered grammars. *Information Processing Letters*, 9(2):84–85, 1979.

[8] S. Eilenberg. *Automata, Languages, and Machines*. Academic Press, 1976.

[9] H. Fernau. Scattered context grammars with regulation. *Annals of Bucharest University, Mathematics-Informatics Series*, 45(1):41–49, 1996.

[10] H. Fernau and A. Meduna. On the degree of scattered context-sensitivity. *Theoretical Computer Science*, 290:2121–2124, 2003.

[11] H. Fernau and A. Meduna. A simultaneous reduction of several measures of descriptional complexity in scattered context grammars. *Information Processing Letters*, 86:235–240, 2003.

[12] V. Geffert. Context-free-like forms for the phrase-structure grammars. In *Proceedings of the Mathematical Foundations of Computer Science 1988*, pages 309–317, New York, 1988.

[13] S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages*. North-Holland, 1975.

[14] S. Ginsburg and S. Greibach. Abstract families of languages. In *FOCS*, pages 128–139, 1967.

[15] J. Gonczarowski and M. K. Warmuth. Scattered versus context-sensitive rewriting. *Acta Informatica*, 27:81–95, 1989.

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

[17] M. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, 1978.

[18] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, 1979.

[19] R. Huddleston and G. Pullum. *The Cambridge Grammar of the English Language*. Cambridge University Press, 2002.

[20] R. Huddleston and G. Pullum. *A Student's Introduction to English Grammar*. Cambridge University Press, 2005.

[21] O. Ibarra. Simple matrix languages. *Information and Control*, 17(4):359–394, 1970.

[22] T. Kasai. An hierarchy between context-free and context-sensitive languages. *Journal of Computer and System Sciences*, 4(5):492–508, 1970.

[23] D. Kelley. *Automata and Formal Languages*. Prentice-Hall, 1995.

[24] H. C. M. Kleijn and G. Rozenberg. Multigrammars. *International Journal of Computer Mathematics*, 12:177–201, 1983.

[25] H. C. M. Kleijn and G. Rozenberg. On the generative power of regular pattern grammars. *Acta Informatica*, 20:391–411, 1983.

[26] J. Král. On multiple grammars. *Kybernetika*, 1:60–85, 1969.

[27] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. Springer-Verlag, 1985.

[28] S. Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, 1964.

[29] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 1981.

[30] P. M. Lewis, D. J. Rosenkrantz, and R. E. Stearns. *Compiler Design Theory*. Addison-Wesley, Reading, 1976.

[31] A. Lindenmayer. Mathematical models for cellular interactions in development, parts I–II. *Journal of Theoretical Biology*, 18:280–315, 1968.

[32] J. C. Martin. *Introduction to Languages and the Theory of Computation*. McGraw-Hill Higher Education, 1997.

[33] T. Masopust. Scattered context grammars can generate the powers of 2. In *EEICT 2007 Proceedings*, volume 1, pages 401–404, Brno, 2007.

[34] T. Masopust and A. Meduna. On the descriptional complexity of partially parallel grammars. Submitted, 2007.

[35] T. Masopust and J. Techet. Leftmost derivations of propagating scattered context grammars: A new proof. Submitted, 2007.

[36] O. Mayer. Some restrictive devices for context-free grammars. *Information and Control*, 20:69–92, 1972.

[37] A. Meduna. Canonical scattered rewriting. *International Journal of Computer Mathematics*, 51:122–129, 1993.

[38] A. Meduna. Syntactic complexity of scattered context grammars. *Acta Informatica*, 32:285–298, 1995.

[39] A. Meduna. A trivial method of characterizing the family of recursively enumerable languages by scattered context grammars. *EATCS Bulletin*, 56:104–106, 1995.

[40] A. Meduna. Four-nonterminal scattered context grammars characterize the family of recursively enumerable languages. *International Journal of Computer Mathematics*, 63:67–83, 1997.

[41] A. Meduna. Economical transformation of phrase-structure grammars to scattered context grammars. *Acta Cybernetica*, 13:225–242, 1998.

[42] A. Meduna. *Automata and Languages: Theory and Applications*. Springer, 2000.

[43] A. Meduna. Generative power of three-nonterminal scattered context grammars. *Theoretical Computer Science*, 246:276–284, 2000.

[44] A. Meduna. Terminating left-hand sides of scattered context productions. *Theoretical Computer Science*, 237:423–427, 2000.

[45] A. Meduna. Uniform generation of languages by scattered context grammars. *Fundamenta Informaticae*, 44:231–235, 2001.

[46] A. Meduna. Coincidental extension of scattered context languages. *Acta Informatica*, 39:307–314, 2003.

[47] A. Meduna. *Elements of Compiler Design*. Taylor and Francis, 2008.

[48] A. Meduna, T. Masopust, and J. Techet. Improved results on the descriptional complexity of scattered context grammars. Unpublished, 2006.

[49] A. Meduna and J. Techet. Generation of sentences with their parses: the case of propagating scattered context grammars. *Acta Cybernetica*, 17:11–20, 2005.

[50] A. Meduna and J. Techet. Canonical scattered context generators of sentences with their parses. *Theoretical Computer Science*, 389:73–81, 2007.

[51] A. Meduna and J. Techet. An infinite hierarchy of language families generated by scattered context grammars with n-limited derivations. Submitted, 2007.

[52] A. Meduna and J. Techet. Maximal and minimal scattered context rewriting. In *FCT 2007 Proceedings*, volume 4639, pages 412–423, Budapest, 2007. Springer Verlag.

[53] A. Meduna and J. Techet. Reduction of scattered context generators of sentences preceded by their leftmost parses. In *DCFS 2007 Proceedings*, pages 178–185, High Tatras, 2007.

[54] A. Meduna and J. Techet. Scattered context grammars that erase nonterminals in a generalized k-limited way. Submitted, 2007.

[55] A. Meduna and M. Švec. *Grammars with Context Conditions and Their Applications*. Wiley, 2005.

[56] D. Milgram and A. Rosenfeld. A note on scattered context grammars. *Information Processing Letters*, 1:47–50, 1971.

[57] G. Paun. Linear simple matrix languages. *Elektronische Informationsverarbeitung und Kybernetik*, 14(7/8):377–384, 1978.

[58] G. Paun. On simple matrix languages versus scattered context languages. *ITA*, 16(3):245–253, 1982.

[59] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[60] G. E. Revesz. *Introduction to Formal Language Theory*. McGraw-Hill, 1983.

[61] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, 1980.

[62] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages*. Springer, 1997.

[63] L. Rychnovský. Parsing of context-sensitive languages. In *WFM 2007 Proceedings*, pages 219–226, Hradec nad Moravicí, 2007.

[64] L. Rychnovský. Type checking by context-sensitive languages. In *EEICT 2007 Proceedings*, pages 405–409, Brno, 2007.

[65] A. Salomaa. *Theory of Automata*. Pergamon Press, 1969.

[66] A. Salomaa. *Formal Languages*. Academic Press, 1973.

[67] A. Salomaa. *Computation and Automata*. Cambridge University Press, 1985.

[68] S. Sippu and E. Soisalon-Soininen. *Parsing Theory*. Springer-Verlag, 1987.

[69] S. Sippu and E. Soisalon-Soininen. *Parsing Theory 2: LR(K) and LL(K) Parsing*. Springer, 1990.

[70] T. A. Sudkamp. *Languages and Machines: An Introduction to the Theory of Computer Science*. Addison-Wesley, Reading, 2006.

[71] J. Techet. Generation of sentences with their parses by scattered context grammars. In *International EEICT 2004 Proceedings*, volume 1, pages 113–119, Bratislava, 2004.

[72] J. Techet. Generation of sentences with their parses by scattered context grammars. In *EEICT 2004 Proceedings*, volume 1, pages 227–229, Brno, 2004.

[73] J. Techet. Generation of sentences with their parses by scattered context grammars. In *SVOČ 2004*, pages 36–36, Brno, 2004.

[74] J. Techet. Canonical scattered context generators of sentences with their parses. In *Honeywell EMI 2005 Proceedings*, pages 80–84, Brno, 2005.

[75] J. Techet. Canonical scattered context generators of sentences with their parses. In *EEICT 2005 Proceedings*, volume 1, pages 280–282, Brno, 2005.

[76] J. Techet. Scattered context generators of sentences with their parses. In *MEMICS 2005 Pre-proceedings*, pages 68–77, Znojmo, 2005.

[77] J. Techet. Částečně paralelní generování jazyků. Master's thesis, Brno, 2005.

[78] J. Techet. k-limited erasing performed by scattered context grammars. In *WFM 2007 Proceedings*, pages 227–234, Hradec nad Moravicí, 2007.

[79] J. Techet. k-limited erasing performed by scattered context grammars. In *EEICT 2007 Proceedings*, volume 4, pages 419–423, Brno, 2007.

[80] J. Techet. A note on scattered context grammars with non-context-free components. In *MEMICS 2007 Proceedings*, pages 225–232, Znojmo, 2007.

[81] G. Vaszil. On the descriptional complexity of some rewriting mechanisms regulated by context conditions. *Theoretical Computer Science*, 330:361–373, 2005.

[82] V. Virkkunen. On scattered context grammars. *Acta Universitatis Ouluensis*, 20(6):75–82, 1973.

[83] M. J. Wolfe. *High Performance Compilers for Parallel Computing*. Addison-Wesley, 1996.

[84] D. Wood. *Grammars and L Forms: An Introduction*. Springer, 1980.

[85] D. Wood. *Theory of Computation*. Harper and Row, 1987.

Symbol Index

Symbol	Page	Description
$a \in A$	13	a is a member of A
$a \notin A$	13	a is not a member of A
$\|A\|$	13	cardinality of A
$\emptyset$	13	empty set
$\{a\}$	13	a set containing a
$\{a : \pi(a)\}$	13	a set containing elements which satisfy π
$A \subseteq B$	13	A is a subset of B
$A \subset B$	13	A is a proper subset of B
2^A	14	power set of A
$A \cup B$	14	union of A and B
$A \cap B$	14	intersection of A and B
$A - B$	14	difference of A and B
$\bigcup_\pi A$	14	union of elements of A satisfying π
$\overline{A}$	14	complement of A
$\|x\|$	14	length of x
ε	14	empty sequence (string)
$\|x\|_V$	14	number of occurrences of elements from V in x
(a,b)	14	ordered pair
$(a_1,\ldots,a_n)$	14	ordered n-tuple
$A \times B$	14	Cartesian product
ρ^{-1}	14	inverse relation
$a \in \rho(b)$	15	synonym for $(a,b) \in \rho$
$a\rho b$	15	synonym for $(a,b) \in \rho$
ρ^k	15	k-fold product of ρ
ρ^+	15	transitive closure of ρ
ρ^*	15	reflexive and transitive closure of ρ
T^*	15	set of all strings over T
T^+	15	set of all non-empty strings over T
$x \cdot y$	15	concatenation of x and y
x^i	15	ith power of a string x
$L_1 \cdot L_2$	16	concatenation of languages L_1 and L_2
L_1/L_2	16	right quotient of L_1 with respect to L_2
$L_2\backslash L_1$	16	left quotient of L_1 with respect to L_2
$L_1 // L_2$	16	exhaustive right quotient of L_1 with respect to L_2
$L_2 \backslash\backslash L_1$	16	exhaustive left quotient of L_1 with respect to L_2

Symbol	Page	Description
L^i	16	ith power of a language L
L^*	16	Kleene star
L^+	16	Kleene plus
$\overline{L}$	16	complement of a language L
$\mathscr{L}$	17	family of languages
$x \to y$	17	production
$x \Rightarrow_G y\ [p]$	18	x is rewritten to y in G by using p
$x \Rightarrow_G^k y$	18	k-fold product of $\Rightarrow_G$
$x \Rightarrow_G^+ y$	18	transitive closure of $\Rightarrow_G$
$x \Rightarrow_G^* y$	18	reflexive and transitive closure of $\Rightarrow_G$
$L(G)$	18	language of grammar G
$x\ {}_{\mathrm{lm}}\Rightarrow_G y$	25	leftmost derivation step
$x\ {}_{\mathrm{rm}}\Rightarrow_G y$	25	rightmost derivation step
$\grave{x}$	35	—
$\acute{x}$	35	—
$\check{x}$	39	—
$L(G,\varepsilon,k)$	35	—
$x \Rrightarrow y$	44	—
$x\ {}^n_{\mathrm{lim}}\Rightarrow_G y$	54	n-limited derivation step
$L(G,\mathrm{lim},n)$	54	—
$L(G,\mathrm{lm})$	64	—
$L(G,\mathrm{rm})$	64	—
$x\ {}_{\max}\Rightarrow_G y$	69	maximal derivation step
$x\ {}_{\min}\Rightarrow_G y$	69	minimal derivation step
$L(G,\max)$	69	—
$L(G,\min)$	69	—
$l : p$	79	production p labeled by l
$T(G)$	101	transformation defined by G

Operator Index

Operator	Page	Description
alph(x)	15	set of symbols occurring in x
cdep(P)	32	set of context-dependent productions of P
cf($x \Rightarrow_G y$)	35	context-free simulation of $x \Rightarrow_G y$
cfree(P)	32	set of context-free productions of P
core(G)	34	core grammar of G
dcs(G)	27	degree of context sensitivity of G
domain(ρ)	14	domain of ρ
insert(x,a)	37	—
join(x)	38	—
lab(G)	78	production labels of G
len(p)	25	number of context-free productions in p
lhs(p)	18, 25	left-hand side of p
lhs-replace(x,a)	37	—
max(I)	17	maximal element of I
mcs(G)	27	maximum context sensitivity of G
min(I)	17	minimal element of I
ocs(G)	27	overall context sensitivity of G
perm(t)	17	set of t-element permutations
perm(n,m)	17	set of $(n+m)$-element permutations preserving order of first n elements
range(ρ)	14	range of ρ
reorder(v,p)	17	reordered v according to permutation p
rev(x)	15,	reversal of x
rhs(p)	18, 25	right-hand side of p
split(x)	38	—

Language Family Index

Subject Index