Languages Generated by Context-Free and Type AB → BA Rules

Benedek Nagy

Faculty of Informatics, University of Debrecen, H-4010 PO Box 12, Hungary,
Research Group on Mathematical Linguistics, Rovira i Virgili University,
Tarragona, Spain
nbenedek@inf.unideb.hu

Abstract: Derivations using branch-interchanging and language family obtained by context-free and interchange (AB → BA) rules are analysed. This language family is between the context-free and context-sensitive families helping to fill the gap between them. Closure properties are analysed. Only semi-linear languages can be generated in this way.

Keywords: formal languages, Chomsky hierarchy, derivation trees, interchange (permutation) rule, semi-linear languages, mildly context-sensitivity

1 Introduction

The Chomsky type grammars and the generated language families are one of the most basic and most important fields of theoretical computer science. The field is fairly old, the basic concepts and results are from the middle of the last century (see, for instance, [2, 6, 7]). The context-free grammars (and languages) are widely used due to their generating power and simple way of derivation. The derivation trees represent the context-free derivations. There is a big gap between the efficiency of context-free and context-sensitive grammars. There are very 'simple' non-context-free languages, for instance $\{a^{n\wedge 2} | n \in N\}$, $\{a^n b^n c^n | n \in N\}$, etc.

It was known in the early 70's that every context-sensitive language can be generated by rules only of the following types AB → AC, AB → BA, A → BC, A → B and A → a (where A,B,C are nonterminals and a is a terminal symbol). In 1974 Penttonen showed that onesided context-sensitivity is enough to obtain the whole context-sensitive language class [5], so grammars with only rules of type AB → AC, A → BC, A → B, A → a are enough. In Turing-machine simulations the rules of type AB → BA are frequently used representing the movement of the head of the machine. The problem about the generating power of grammars having non-context-free rules only in the form AB → BA remained open.

In this paper we consider the language family generated by generative grammars allowing only permutation/interchange rule (type AB → BA) as non-context-free ones. These rules are monotone rules having exactly the same letters in both sides. We will show that the context-free rules with only interchange rules (type AB → BA) are more efficient than the context-free ones, but they are not enough to get all context-sensitive languages. We use the term interchange rule for these rules indicating that they allow to interchange some letters in the sentential form (i.e. some branches of the derivation tree).

The structure of the paper is as follows. In the next section we present some motivations, the presented grammar and language family is related to several other things. In Section 3 recall some basic definitions and facts that we need later on. After this, Section 4 is about the permutation languages. Some examples and properties, mainly closure properties will be presented.

2 Motivations

In formal language (and classical computing) theory there is a big gap between the context-free and context-sensitive languages. The word problem is to decide weather a given word is in the generated language of the given grammar. For context-free languages the word problem can be solved in deterministic polynomial time with small polynomial coefficient. The word problem can be solved by Cocke-Younger-Kasami (CYK) algorithm in deterministic cubic time for grammars in Chomsky normal form, while in about O($n^{2.7}$) time with Earley algorithm. The problem can be solved in linear time in a non-deterministic way using Greibach normal form. Oppsite to this the word problem in context-sensitive case is much harder. However the problem can be solved in linear space in non-deterministic manner, the word problem is PSPACE-complete.

Several phenomenons are shown to be non context-free, such as the language of tautologies of Boolean logic, natural and programming languages, developmental biology, economic modelling, semiotics of fairy-tales, music and visual arts, etc. ([1]). So, context-free grammars are not enough to describe several phenomenons of the world, but the context-sensitive family is too large. In applications usually only one of its subset is needed. Therefore several branches of extensions of context-free grammars were introduced by controlling the derivations in another way, such as, for instance, priority relation among the rules (see [1]).

2.1 Motivations from Concurrency Theory

Our investigation is interest for concurrency and parallelism theory as well, where the order of some processes can be interchanged. Usually when two independent

process can be processed in the same time, then both of their order is allowed. Even if they can be processed in a parallel way by the architecture, in the description usually their order can be written in a sequential way.

One of the most known modelling technique of concurrent and parallel events is the usage of Petri nets. Their theory is well-developed. One can also use them in language generation. In these cases the alphabet is the set of actions/events. The words are representing possible sequential runs of the system. When some actions are independent (i.e. can be processed parallely) then all possible permutations of their order will appear in the same place of various words of the language.

2.2 Linguistic Motivations

The work has some linguistic motivations as well: in some morphologically rich languages (as, for instance, Japanese, Finnish and Hungarian) the word order is not strict in a sentence. There is a Hungarian example:

'A kutya hangosan ugat.' 'Hangosan ugat a kutya.' 'A kutya ugat hangosan.' 'Hangosan a kutya ugat.' 'Ugat a kutya hangosan.' 'Ugat hangosan a kutya.' are all correct sentences about the same meaning: The dog (a kutya) barks (ugat) loudly (hangosan). So, usually some of the parts of the sentences can freely be interchanged.

Similar examples can be found in other languages as well.

2.3 Natural Computing

In membrane computing the multisets are used in computations [4]. One type of representation of multisets by strings to have all possible words that can be obtained using the letters of the given multisets. That means that the commutaive closure of a language should be used.

Membrane systems also can be used to generate languages in the traditional sense (set of words). A run of a system can generating strings in the following way: the objects (letters) sent out will be attached to the generated string (which was the empty string at the start). The objects sent out at the same time (same step) can be occurr in any order. Therefore the objects sent out at the same time are freely permutable in the generated words.

A rule is called cooperating rule if more than one symbol occurrs in the left hand side. If there is no cooperating rule in the membrane sytem, then semilinear computations can be obtained.

3 Basic Notions, Definitions and Preliminaries

First some definitions about Chomsky-type grammars and generated languages are recalled and our notations are fixed ([2, 3, 6, 7]).

A grammar is a construct G= (N,T,S,H), where N, T are the non-terminal and terminal alphabets, N and T are disjoint finite sets. S is a non-terminal symbol, it is a special symbol, called initial (or start) letter. H is a finite set of pairs, where a pair uses to be written in the form $v \to w$ with $v \in (N \cup T)^* N (N \cup T)^*$ and $w \in (N \cup T)^*$. (We used the well-known notation of Kleene-star.) H is the set of derivation rules; $v \Rightarrow w$ $(v,w \in (N \cup T)^*)$ is a direct derivation if there exist $x,y,v',w' \in (N \cup T)^*$ such that $v = x\ v'\ y$, $w = x\ w'\ y$ and $v' \to w' \in H$. The transitive and reflexive closure of the direct derivation is the derivation denoted by $v \Rightarrow^* u$.

We say that $v \in (N \cup T)^*$ is a sentential form if $S \Rightarrow^* v$ holds.

The language generated by a grammar G is the set of terminal words which can be derived from the initial letter: $L(G) = \{ w \mid S \Rightarrow^* w , w \in T^* \}$.

We use λ to sign the empty word. For any word and sentential form u we will use $|u|$ to sign its length, i.e. the number of letters it contains.

Two grammars are equivalent if they generate the same language modulo the empty word, so from now on, we do not care about the fact whether $\lambda \in L$.

Depending on the possible structures of the derivation rules various classes of grammars are defined. We recall the most important classes.

- monotone grammars: each rule $v \to u$ satisfies the condition $|v| \leq |u|$ but the possible rule $S \to \lambda$, in which case the initial symbol S does not occur on any right hand side of a rule.
- context-free grammars: for every rule the next scheme holds: $A \to v$ with $A \in N$ and $v \in (N \cup T)^*$.
- regular grammars: each derivation rule is one of the following forms: $A \to w$, $A \to wB$; where $A,B \in N$ and $w \in T^*$.

A language is regular/ context-free/ context-sensitive if it can be generated by a regular/ context-free/ monotone grammar, respectively.

For these families the notations **L-reg**, **L-CF** and **L-CS** are used. It is well known that the following strict relations hold: **L-reg** $\subset$ **L-CF** $\subset$ **L-CS**.

Let the terminal alphabet T be ordered. For each word its Parikh-vector is assigned (Parikh-mapping). The elements of this vector are the occurrences of the letters of the alphabet in the word. Formally, using alphabet $T=(a_1, a_2, ..., a_n)$ let $\Psi : T^* \to N^n$, $\Psi (w) = (|w|_{a_1}, |w|_{a_2},...,|w|_{a_n})$, where $w \in T^*$ and $|w|_{a_i}$ is the

number of occurrences of the letter a_i in w. The set of Parikh-vectors of words of a language is called the Parikh-set of the language. Formally:
$\Psi(L) = \{ \Psi(w) \mid w \in L \}$. Two languages are letter-equivalent if and only if their Parikh-sets are identical.

A language is linear (in Parikh-sense) if its Parikh set can be written in the form of linear set: $\left\{ \underline{v}_0 + \sum_{i=1}^{m} x_i \underline{v}_i \middle| x_i \in N \right\}$ for some vectors $\underline{v}_j \in N^n$, $0 \le j \le m$.

A language is semi-linear (in Parikh-sense) if its Parikh set can be written as a finite union of linear sets. We will use shortly the term semi-linear for this property.

Every context-free language is semi-linear. Non semi-linear context-sensitive languages are known (for instance $L_{\square} = \{a^{n\hat{}2} \mid a \in T\}$).

The context-sensitive languages can be generated rule set in Kuroda normal form, i.e. using only rules type $AB \rightarrow CD$, $A \rightarrow BC$, $A \rightarrow B$, $A \rightarrow a$ ($A,B,C,D \in N$, $a \in T$) [3].

Moreover, due to Révész there is a normal form is containing rules of type $AB \rightarrow AC$, $AB \rightarrow BA$, $A \rightarrow BC$, $A \rightarrow B$, $A \rightarrow a$ (see, for instance [6]).

There was a very nice open problem ([7]) whether one-sided context-sensitivity has the same generating power as both sided context-sensitivity. Finally, Penttonen solved the problem.

Every context-sensitive language can be generated by a grammar whose derivation rules are of the form $AB \rightarrow AC$, $A \rightarrow BC$, $A \rightarrow B$, $A \rightarrow a$, where A, B and C are nonterminals and a is a terminal. This normal form is from ([5]), where it was called one-sided normal form.

So the interchange rules (type $AB \rightarrow BA$) can be omitted from context-sensitive grammars.

The context-free grammars are very popular ones because the concept of derivation trees fits very well in these derivations. It is an important property of the (context-free) derivations that the direction left-to-right is preserved. The letters in the beginning of the sentential form refer for the beginning of the derived word, and have no influence to the end-part.

4 The Permutation Languages

First we are defining formally the grammar and language class we are dealing with.

Definition 1 *A grammar* G = (N,T,S,H) *is a context-free grammar with interchange rules (or shortly permutation grammar) if* H *contains only special type of non-context-free rules, the interchange (permutation) rules, which are in the form* AB → BA (A,B ∈ N). *We denote the language family generated in this way by* **L-perm** *and call it as* permutation languages.

In the derivations the new non context-free rules allow to permute some branches of the derivation tree.

4.1 Find the Position in the Chomsky-Hierarchy

First let us see an example.

Example 1 Let G = ({S, A, B, C}, {a, b, c}, S, H) be a context-free grammar with interchange rule: H = {S → ABC, S → SABC, AB → BA, BA → AB, AC → CA, CA → AC, BC → CB, CB → BC, A → a, B → b, C → c}.
Fig. 1 shows the 'derivation-tree' of the word 'aaccbb' in this system.

The language containing all words with the same number of a, b and c is generated in the previous example. This language is a non-context-free one. So, we can state, that generating power of the grammar increasing if we allow interchange rules. (Obviously without any (applicable) interchange rule one can generate any context-free language.)

Note, that the rules of this example and also of every permutation grammar can be written in Kuroda normal form, where each non-context-free rule is an interchange rule.

Definition 2 *The context-free grammar* G' *and language* L' *obtained from a permutation grammar* G *and language* L *by deleting all the non-context-free rules are a basis-grammar and a basis-language of L.*

The basis language is usually not uniquely defined. For instance for the language generated by Example 1 it can be any context-free language containing exactly words with Parikh-vectors (n, n, n) with every $n \in$ N.

Every basis language is letter equivalent to the original one. Since the interchange rules do not modify the multiset of the symbols of a sentential form, the Parikh-set of the generated language is the same as the Parikh-set of the context-free language obtained without interchanging branches.

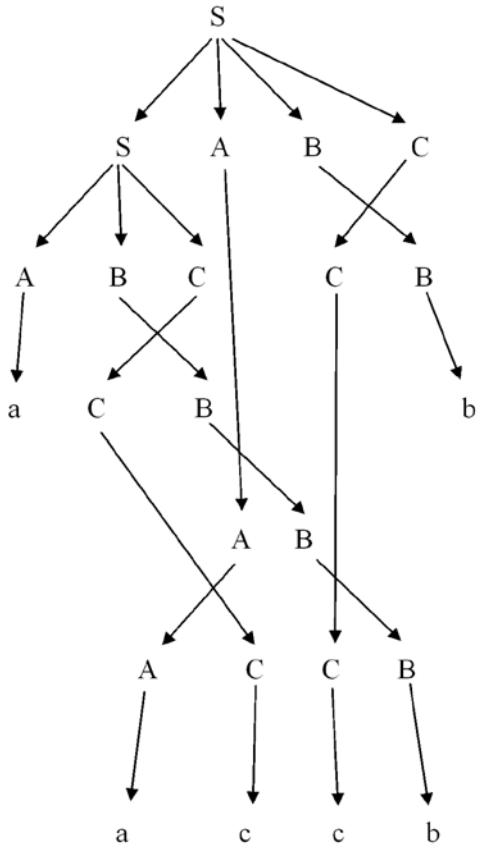

Figure 1
Derivation in a context-free grammar with interchange rule obtaining a non-CF language

By the previous observations we have the next theorem.

Theorem 1 *All languages which can be generated with context free and permutation rules are semi-linear.*

This theorem implies that not all context-sensitive languages can be generated by context-free and permutation rules. For example the context-sensitive language $L_{\square}$ is not semi-linear, therefore it cannot be generated using only permutations as non-context-free rules.

Now we can place the language family **L-perm** in the Chomsky hierarchy:

$$\textbf{L-reg} \subset \textbf{L-CF} \subset \textbf{L-perm} \subset \textbf{L-CS},$$

where each inclusion is strict.

Now, we are detailing some further result about **L-perm**.

4.2 Closure Properties

In this section closure properties under several language operation will be analysed.

4.2.1 Regular Operations

Let us see the closure properties of **L-perm** under the three regular operations (union, concatenation, Kleene-star).

Theorem 2 *The language family* **L-perm** *is closed under union.*

Theorem 3 *The language family* **L-perm** *is closed under concatenation.*

Theorem 4 *The language family* **L-perm** *is closed under Kleene-star iteration.*

The proof of all these theorems goes in the standard constructive way (as it goes for **L-CF** and for **L-CS**). We note here that all classical language classes (regular, context-free, context-sensitive, recursive enumerable) are closed under the regular operations.

4.2.2 Set-Theoretical Operations

It is interesting to analyse the closure not only under union, but under other set-theoretical operations. It is known [2] that regular and context-sensitive languages are closed under intersection and complement (using the universal language T*), but the context-free languages are not closed under these two operations.

Theorem 5 *The language family* **L-perm** *is not closed under intersection with regular languages.*

The language $\{a^n b^n c^n | n \in N\}$ is important from linguistical point of view. It is a well known mildly context-sensitive language. The language of Example 1 intersected with the regular set a*b*c* obtain it. Since there is no forced use of interchange rules in permutation grammars, every permutation language L must contain its the basis languages that is letter equivalent to L. But $\{a^n b^n c^n | n \in N\}$ does not contain any context-free languages that letter-equivalent to itself, so it cannot be a permutation language.

Since regular languages are context-free and therefore all of them in **L-perm**, the theorem above leads to the consequence that **L-perm** is not closed under intersection, i.e. the languages obtained by intersection of two languages of **L-perm** are not necessarily in **L-perm**.

Theorem 6 *The language family* **L-perm** *is not closed under complement.*

As we have seen the language $\{a^n b^n c^n | n \in N\}$ is not in **L-perm**, but it is known that its complement language (over alphabet {a,b,c}) is context-free, therefore it is also a permutation language. That pair of languages proves the theorem.

By these properties it seems that **L-perm** is closely related to the language family **L-CF**. The strange property, that **L-perm** is not closed under intersection with regular languages, does not occurr in the classical language classes of the Chomsky hierarchy.

4.2.3 Operations Related to Concurrency

In this part some other operations will be analysed. These operations are related to permutations.

Theorem 7 *The language family* **L-perm** *is closed under commutative closure.*

Theorem 8 *The language family* **L-perm** *is closed under shuffle.*

Both proofs are constructive. One can start with grammars in Kuroda normal form having non-context-free rules only in form AB → BA. The commutative closure can be obtained by adding all the possible permutation rules to the grammar. Before terminating a derivation one can freely move any letter to any places of the sentential form. At shuffle operation the non-terminal sets of the two grammars need to be disjoint. By adding all possible permutation rules having one non-terminal from the first and one non.terminal from the second grammar, the suffle languages will be generated.

Regarding these operations the family **L-perm** is related to the class of context-sensitive languages. The family **L-CS** is closed under these operations, but **L-CF** is not.

Conclusions

Context-free grammars were extended by permutation (interchange) rules and as it was proven their generative power are increased. Since only semi-linear languages can be genrated in this way, the generated language family **L-perm** is strictly between the context-free and context-sensitive classes. Closure properties under several operations are analysed.

Acknowledgement

The research is partly supported by Hungarian National Foundation for Scientific Research OTKA T049409 and by the Öveges programme of the Agency for Research Fund Management and Research Exploitation (KPI) and National Office for Research and Technology.

József Öveges programme

Established by the support of the National Office for Research and Technology.

NKTH
National Office for Research and Technology

References

[1] Jürgen Dassow, Gheorghe Paun: Regulated Rewriting in Formal Language Theory, (EATCS Monographs on Theoretical Computer Science 18), Springer-Verlag, Berlin, 1989

[2] John E. Hopcroft, Jeffrey D. Ullmann: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, Massachusetts, 1979

[3] Alexandru Mateescu: On Context-Sensitive Grammars, in: Carlos Martin-Vide, Victor Mitrana, Gheorghe Paun (eds.): Formal languages and applications, (Studies in Fuziness and Soft Computing 148), Springer-Verlag, Berlin, Heidelberg, 2004, pp. 139-161

[4] Gheorghe Paun: Membrane Computing. An Introduction, Springer-Verlag, Berlin, Heidelberg, 2002

[5] Martti Penttonen: One-sided and Two-sided Context in Formal Grammars, Information and Control 25 (1974), pp. 371-392

[6] György Révész: Introduction to Formal Languages, McGraw-Hill, New York, 1983

[7] Arto Salomaa: Formal Languages. Academic Press, New York, 1973