Journal of Automata, Languages and Combinatorics **u** (v) w, x–y

CONJUNCTIVE GRAMMARS [1]

Alexander Okhotin
Faculty of Computational Mathematics and Cybernetics, Moscow State University [2]
e-mail: okhotin@aha.ru

ABSTRACT

This paper introduces a class of formal grammars made up by augmenting the formalism of context-free grammars with an explicit set-theoretic intersection operation.

It is shown that conjunctive grammars can generate some important non-context-free language constructs, including those not in the intersection closure of context-free languages, and that they can provide very succinct descriptions of some context-free languages and finite intersections of context-free languages.

On the other hand, it is proved that conjunctive grammars can still be parsed in cubic time and that the notion of the derivation tree is retained, which gives reasonable hope for their practical applicability.

Keywords: Conjunctive grammar, context-free grammar, intersection, descriptional complexity, parsing.

1. Introduction

It is a classical result that the family of context-free languages is not closed under intersection, and that most decision problems concerning intersection of context-free languages are undecidable.

However, intersection, as well as other set-theoretic operations, is an essential part of any kind of formalized reasoning, since it actually denotes an object that satisfies several conditions simultaneously.

The family of finite intersections of context-free languages (intersection closure of the context-free languages, intersective context-free languages) was introduced in [8], where it was proved that they form an infinite hierarchy. Its proper inclusion in the family of deterministic context-sensitive languages was shown in [3]. Intersection and general Boolean closures of deterministic and nondeterministic context-free languages were studied in [10, 11].

The intersective context-free languages were further investigated in [7], where they were suggested as a formal grammatical model for linguistic applications, in which

[1] Full version of a submission presented at the Second International Workshop on *Descriptional Complexity of Automata, Grammars and Related Structures* held in London, Ontario, Canada, July 27–29, 2000.

[2] Present address: Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada K7L 3N6.

several conditions, such as morphology, syntax and semantics, are imposed on a single natural language sentence. However, the two-layer structure of a tuple of context-free grammars, where finitely many intersections come on the top with normal context-free grammars operating at the bottom, limits the applicability of the concept to practical tasks; for instance, their closure under concatenation and Kleene star remains an open problem, which indicates that these operations cannot be carried out at will when needed; similarly, while the family is closed under union, this operation may lead to quadratic grow of the length of description (at least no better upper bound is known), which is also unconvenient from a practical point of view.

Conjunctive grammars represent a different approach to the problem, which is free from the mentioned drawback of the intersective context-free grammars: the context-free grammars are being generalized by introducing an explicit intersection operation within the formalism of grammar rules. While context-free grammars have all rules in the form $A \to \alpha$ ($A \in N$, $\alpha \in V^*$), rules in conjunctive grammars are written as

$$A \to \alpha_1 \& \ldots \& \alpha_n \quad (A \in N;\ n \geqslant 1;\ \text{ for each } i,\ \alpha_i \in V^*), \tag{1}$$

where the conjunction has semantics of set-theoretic intersection, i.e. the rule (1) means that if some string of terminal symbols can be derived from each α_i individually, then that string is derivable from A.

A quite similar-looking concept, the alternating context-free grammar, was introduced in [9]; in [6], a subclass of alternating context-free grammars was proved equivalent to alternating pushdown automata and therefore to exponential time languages. The set of nonterminals in an alternating context-free grammar is separated into two classes of existential and universal nonterminals, where existential nonterminals work essentially the same as in the context-free grammars, while the expansion of a universal nonterminal causes the whole derivation to be splitted into a number of independent branches, which are required to come up with the same string in the end. This subtle difference in semantics (in conjunctive grammars it is not the whole computation, but only the derivation of a substring from a single nonterminal which is being splitted) makes two generative devices entirely different.

This paper is aimed to introduce a new generative device, the conjunctive grammars, and investigate its descriptional and computational complexity. Section 2 formally defines the conjunctive grammars and gives several examples, showing that some important non-context-free language constructs can be denoted by conjunctive grammars. Section 3 introduces two normal forms: a normal form for general conjunctive grammars similar to Chomsky normal form, and another normal form for a subclass of linear conjunctive grammars; these normal forms allow to devise polynomial-time recognition algorithms, which are presented in Section 4. Section 5 deals with the descriptional complexity of the conjunctive grammars, showing that they can provide dramatically more succinct descriptions of context-free languages and of their finite intersections than the context-free grammars and the tuples of context-free grammars that denote intersection of languages. Section 6 gives a very preliminary survey of general properties of the language family generated by conjunctive grammars and poses several research problems.

2. Basic Notions

2.1. Definitions

Definition 1 *A conjunctive grammar is a quadruple $G = (\Sigma, N, P, S)$, where (i) Σ and (ii) N are disjoint finite nonempty sets of terminal and nonterminal symbols respectively; (iii) P is a finite set of grammar rules (productions), each formally defined as an ordered pair $(A, \{\alpha_1, \ldots, \alpha_n\})$ of a nonterminal and a finite set of strings over $\Sigma \cup N$, and written in the form $A \to \alpha_1 \& \ldots \& \alpha_n$, where the order of αs is considered to be in some way fixed; (iv) $S \in N$ is a nonterminal designated as the start symbol.*

Let V be the union of Σ and N. We shall use three special symbols: "(", "&" and ")"; it is assumed that none of them is in V. Define $\bar{V} = \Sigma \cup N \cup \{$ "(", "&", ")" $\}$.

For each rule $A \to \alpha_1 \& \ldots \& \alpha_n \in P$ and for each i $(1 \leqslant i \leqslant n)$, a pair of the form (A, α_i) is called a conjunct; in cases where that will not cause confusion with productions, a conjunct (A, α_i) will be denoted just as $A \to \alpha_i$.

The following notation will be used for a collection of rules for a single nonterminal: $A \to \alpha_{11} \& \ldots \& \alpha_{1n_1} \mid \ldots \mid \alpha_{m1} \& \ldots \& \alpha_{mn_m}$.

A derivation in conjunctive grammar is a nondeterministic rewriting of formulae over the basis of concatenation, conjunction and symbols from V. We shall write these formulae as strings over $\bar{V}$, formally defined as follows:

Definition 2 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar.*

i. Any string over V is a conjunctive formula.

ii. If $\mathcal{A}$ and $\mathcal{B}$ are formulae, then $\mathcal{AB}$ is a formula.

iii. If $\mathcal{A}_1, \ldots, \mathcal{A}_n$ $(n \geqslant 1)$ are formulae, then $(\mathcal{A}_1 \& \ldots \& \mathcal{A}_n)$ is a formula.

We shall use script capital Latin letters from the beginning of the alphabet to denote conjunctive formulae: $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$.

Definition 3 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar. Define $\overset{G}{\Longrightarrow}$, a relation of immediate derivability on the set of conjunctive formulae:*

i. A nonterminal can be rewritten by the body of some rule enclosed in parentheses, i.e. for all $s_1, s_2 \in \bar{V}^*$ and for all $A \in N$, if $s_1 A s_2$ is a formula, then for all $A \to \alpha_1 \& \ldots \& \alpha_n \in P$

$$s_1 A s_2 \overset{G}{\Longrightarrow} s_1 (\alpha_1 \& \ldots \& \alpha_n) s_2 \tag{2}$$

ii. Conjunction of several identical terminal strings enclosed in parentheses can be replaced by one such string, i.e. for all $s_1, s_2 \in \bar{V}^*$, for all $w \in \Sigma^*$ and for all $n \geqslant 1$, if $s_1 (\underbrace{w \& \ldots \& w}_{n}) s_2$ is a formula, then

$$s_1 (\underbrace{w \& \ldots \& w}_{n}) s_2 \overset{G}{\Longrightarrow} s_1 w s_2 \tag{3}$$

Let $\overset{G}{\Longrightarrow}^*$ *be the reflexive transitive closure of* $\overset{G}{\Longrightarrow}$.

It can easily be proved that right parts of (2) and (3) are indeed formulae, which makes the definition consistent.

Definition 4 *Let* $G = (\Sigma, N, P, S)$ *be a grammar, let* $\mathcal{A}$ *be a formula. The language generated by* $\mathcal{A}$ *is a set of all strings over* Σ *derivable from* $\mathcal{A}$*:*

$$L_G(\mathcal{A}) = \{w \mid w \in \Sigma^*,\ \mathcal{A} \overset{G}{\Longrightarrow}^* w\} \tag{4}$$

The language generated by the grammar is a set of all strings over Σ *derivable from its start symbol:* $L(G) = L_G(S)$.

A language L *is called conjunctive, if it is generated by some conjunctive grammar. Let* $\mathcal{L}_{\&}$ *be the set of all conjunctive languages.*

2.2. Language Generated by a Formula

The language generated by a formula inductively depends on its structure, as shown in the following theorem:

Theorem 1 *Let* $G = (\Sigma, N, P, S)$ *be a conjunctive grammar. Let* $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{B}$ *be formulae, let* $A \in N$*, let* $u \in \Sigma^*$*. Then,*

$$L_G((\mathcal{A}_1 \& \ldots \& \mathcal{A}_n)) = \bigcap_{i=1}^{n} L_G(\mathcal{A}_i) \tag{5a}$$

$$L_G(\mathcal{A}\mathcal{B}) = L_G(\mathcal{A}) \cdot L_G(\mathcal{B}) \tag{5b}$$

$$L_G(A) = \bigcup \{L_G((\alpha_1 \& \ldots \& \alpha_m)) \mid A \to \alpha_1 \& \ldots \& \alpha_m \in P\} \tag{5c}$$

$$L_G(u) = \{u\} \tag{5d}$$

The proof is purely technical and is omitted. It follows that conjunction has semantic of intersection of languages, concatenation of formulae concatenates languages generated by them, nonterminal symbols generate what is generated by their rules, and terminal strings generate themselves. This corresponds to the intuitive meaning of these symbols and operations.

One important corollary of Theorem 1 is that the order of conjuncts has no influence on the language generated by a formula, and consequently that the order of conjuncts in a grammar rule does not affect the generated language; this immediately follows from (5a).

It should also be observed that if every rule in some conjunctive grammar consists of a single conjunct, then this grammar generates the same language as a similarly notated context-free grammar.

2.3. Examples

We shall consider two classical examples of non-context-free languages (see, for instance, [1], p.179-180) and show that each of them is generated by a conjunctive grammar.

Example 1 *A conjunctive grammar for the language* $\{a^n b^n c^n \mid n \geqslant 0\}$*:*

$S \to AB\&DC$
$A \to aA \mid \epsilon$
$B \to bBc \mid \epsilon$
$C \to cC \mid \epsilon$
$D \to aDb \mid \epsilon$

It is obvious that A, B, C and D generate the languages $\{a^i \mid i \geqslant 0\}$, $\{b^j c^j \mid j \geqslant 0\}$, $\{c^l \mid l \geqslant 0\}$ and $\{a^k b^k \mid k \geqslant 0\}$ respectively. Therefore, S generates

$$\{a^i b^j c^j \mid i, j \geqslant 0\} \cap \{a^k b^k c^l \mid k, l \geqslant 0\} = \{a^n b^n c^n \mid n \geqslant 0\} \tag{6}$$

Example 2 *A conjunctive grammar for the language* $\{wcw \mid w \in \{a, b\}^*\}$*:*

$S \to C\&D$
$C \to aCa \mid aCb \mid bCa \mid bCb \mid c$
$D \to aA\&aD \mid bB\&bD \mid cE$
$A \to aAa \mid aAb \mid bAa \mid bAb \mid cEa$
$B \to aBa \mid aBb \mid bBa \mid bBb \mid cEb$
$E \to aE \mid bE \mid \epsilon$

C ensures that the string consists of two equal-length parts separated by a center marker; formally we say that C generates $\{xcy \mid x, y \in \{a, b\}^*; |x| = |y|\}$.

D takes one symbol from the left and uses A or B to compare it to the corresponding symbol at the right. At the same time, D recursively calls itself in order to process the rest of the string in the same way.

Formally, A generates $\{xcvay \mid x, v, y \in \{a, b\}^*,\ |x| = |y|\}$, B generates $\{xcvby \mid x, v, y \in \{a, b\}^*,\ |x| = |y|\}$ and therefore D produces $\{uczu \mid u, z \in \{a, b\}^*\}$ (the last result may be obtained by a straightforward induction over the length of the string). Finally,

$$\{xcy \mid x, y \in \{a, b\}^*, |x| = |y|\} \cap \{uczu\} = \{wcw \mid w \in \{a, b\}^*\} \tag{7}$$

It has been proved in [10] that the language $\{wcw \mid w \in \{a, b\}^*\}$ cannot be expressed as a finite intersection of context-free languages. On the other hand, it is clear that every such intersection is generated by some conjunctive grammar. Thus, the family of conjunctive languages properly contains the intersection closure of context-free languages.

It is important to note that the grammar from Example 2 essentially uses the center marker, and therefore this method cannot be applied to writing a conjunctive grammar for the language $\{ww \mid w \in \{a, b\}^*\}$.

Example 3 *For each Turing machine M over the work alphabet Σ that cannot print a blank, let $ID_i(w)$ be the instantaneous description of M on the input w after i steps of computation, written as $-uaqv-$, $--qv-$ or $-u-q-$, where $u, v \in \Sigma^*$, $a \in \Sigma$, q is a state of M, "$-$" is a symbol that denotes a blank tape square, and at the moment the machine scans the square that precedes the symbol q in the string.*

The language of all valid accepting computations of M,

$$VALC[M] = \{ID_0(w)\#ID_1(w)\#\ldots\#ID_k(w) \mid w \in \Sigma^*, \\ ID_k(w) \text{ is an accepting configuration}\}, \tag{8}$$

is a conjunctive language.

The construction of the grammar is generally based upon the idea given in Example 2 and is omitted.

2.4. Derivation Tree

The representation of a context-free derivation in the form of a tree is a very important property of context-free grammars, since it gives birth to the syntactical analysis.

It turns out that the notion of derivation tree can be extended to conjunctive grammars, where it will actually be a tree with shared leaves.

Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar, let $w \in L(G)$. Consider some derivation from S to w; assume that it is organized as follows: first, rules from P are applied until there are no nonterminals left in the formula; second, rules of the form $(u\& \ldots \&u) \to u$ are applied until all parentheses and conjunction signs are removed. There is no loss of generality in this assumption, because an existing derivation can always be reconstructed to match the required sequence of steps.

The construction of the derivation tree is done in two steps: initially, a context-free derivation tree is made from the first part of the derivation (as if there was a context-free rule $A \to \alpha_1\alpha_2\ldots\alpha_m$ instead of each conjunctive rule $A \to \alpha_1\&\alpha_2\&\ldots\&\alpha_m$); then, terminal leaves of the tree are glued in accordance with the second part of the derivation. Thus, terminal leaves of the resulting tree can possibly have more than one incoming arc. Internal nodes of the tree are labeled with rules rather than with nonterminals.

Example 4 *Consider the grammar for $\{a^n b^n c^n \mid n \geqslant 0\}$ from Example 1, the string $w = abc$ and its derivation:*

$$\begin{aligned}
S &\overset{G}{\Longrightarrow} (AB\&DC) \overset{G}{\Longrightarrow} ((aA)B\&DC) \overset{G}{\Longrightarrow} ((a())B\&DC) \overset{G}{\Longrightarrow} \\
&((a())(bBc)\&DC) \overset{G}{\Longrightarrow} ((a())(b()c)\&DC) \overset{G}{\Longrightarrow} \\
&((a())(b()c)\&(aDb)C) \overset{G}{\Longrightarrow} ((a())(b()c)\&(a()b)C) \overset{G}{\Longrightarrow} \\
&((a())(b()c)\&(a()b)(cC)) \overset{G}{\Longrightarrow} \underline{((a())(b()c)\&(a()b)(c()))} \overset{G}{\Longrightarrow} \\
&((a)(b()c)\&(a()b)(c())) \overset{G}{\Longrightarrow} (a(b()c)\&(a()b)(c())) \overset{G}{\Longrightarrow} \\
&(a(bc)\&(a()b)(c())) \overset{G}{\Longrightarrow} (abc\&(a()b)(c())) \overset{G}{\Longrightarrow} (abc\&(ab)(c())) \overset{G}{\Longrightarrow} \\
&(abc\&ab(c())) \overset{G}{\Longrightarrow} (abc\&ab(c)) \overset{G}{\Longrightarrow} (abc\&abc) \overset{G}{\Longrightarrow} abc
\end{aligned}$$

The corresponding derivation tree is shown in Figure 1(b). Figure 1(a) shows an intermediate tree corresponding to the first part of the derivation (from S to the underlined formula).

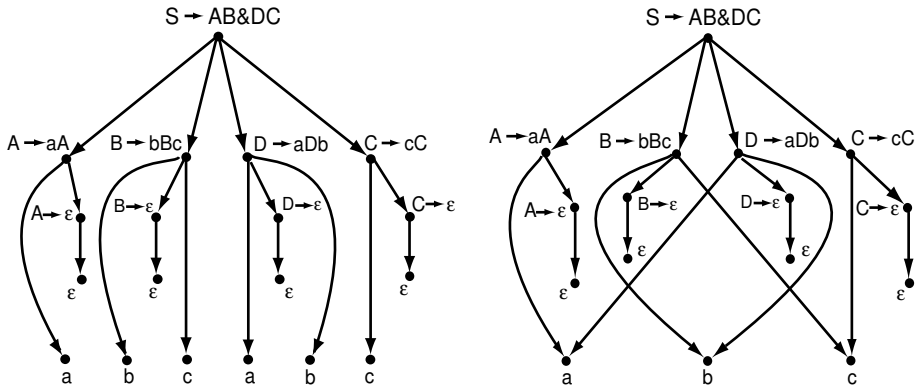

Figure 1: Derivation tree: (a) before gluing; (b) after gluing

2.5. Linear Conjunctive Grammars

The notion of linearity of a context-free grammar can be naturally extended to the conjunctive grammars:

Definition 5 *A conjunctive grammar $G = (\Sigma, N, P, S)$ is said to be linear, if each rule in P is of the form*

$$A \to u_1 B_1 v_1 \& \ldots \& u_m B_m v_m \quad (u_i, v_i \in \Sigma^*,\ B_i \in N) \tag{9a}$$

$$A \to w \quad (w \in \Sigma^*) \tag{9b}$$

The grammar from Example 2 is linear; the grammar from Example 1 is not. It will be shown later in Section 4.2 that linear conjunctive grammars can be parsed as efficiently as linear context-free grammars.

3. Normal Forms

3.1. Epsilon Conjuncts

Definition 6 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar. A conjunct of the form $A \to \epsilon$, where $A \in N$, is called an epsilon conjunct.*

In this section we shall show that, exactly like in the context-free case, for any given conjunctive grammar one can construct an equivalent (with the exception of the membership of ϵ) grammar without epsilon conjuncts.

Let us consider the set of all nonterminals capable of generating the empty string:

Definition 7 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar. Define*

$$\textsc{Nullable}(G) = \{A \mid A \in N, A \overset{G}{\Longrightarrow}{}^{*} \epsilon\}, \tag{10}$$

There is a algorithm to compute the set $\textsc{Nullable}(G)$ for any given conjunctive grammar G, which uses the same nested-set technique as the classical algorithm for the context-free case.

Using the set $\textsc{Nullable}(G)$, it is possible to remove all epsilon conjuncts from the grammar. The construction again follows the context-free case: nullable nonterminals are removed from rule bodies in all possible combinations, and, afterwards, all rules containing an epsilon conjunct are discarded.

Theorem 2 (Elimination of epsilon conjuncts) *For any conjunctive grammar $G = (\Sigma, N, P, S)$ there exists a conjunctive grammar $G' = (\Sigma, N, P', S)$, free of epsilon conjuncts (i.e. without rules of the form $A \to \alpha_1 \& \ldots \& \alpha_{k-1} \& \epsilon \& \alpha_{k+1} \& \ldots \& \alpha_m$, where $m \geqslant 1$, $1 \leqslant k \leqslant m$ and $\alpha_i \in \Sigma^{+}$), such that*

$$L(G') = L(G) \setminus \{\epsilon\} \tag{11}$$

3.2. Unit Conjuncts

The context-free notion of a chain rule (or unit rule) $A \to B$ is inherited by the conjunctive grammars in the form of so-called *unit conjunct*:

Definition 8 (Unit conjunct) *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar. A conjunct of the form $A \to B$, where $A, B \in N$, is called a unit conjunct.*

In the context-free case, the task of removing chain rules from the grammar is usually solved by first precomputing all chain derivations, storing them in the form of the sets $\textsc{Chain}(A) = \{B \mid A \Longrightarrow^{*} B\}$ (for all $A \in N$) and then using these sets to replace each chain rule by immediately determinable non-chain rules.

In the case of conjunctive grammars, this task is a bit more involved, since a rule may contain multiple unit conjuncts (consider $A \to B \& C \& \alpha$), and thus there is no direct equivalent of the sets $\textsc{Chain}(A)$ and the corresponding transformation technique. A slightly different approach is suggested, which does not use any global information about the grammar, such as precomputed derivations, and is confined to local substitutions.

Lemma 1 (Substitution of unit conjuncts) *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar. Let*

$$A \to \alpha_1 \& \ldots \& \alpha_{k-1} \& B \& \alpha_{k+1} \& \ldots \& \alpha_m \quad (m \geqslant 1,\ 1 \leqslant k \leqslant m,\ \alpha_i \in \Sigma^{*}) \tag{12}$$

be some rule for A that contains a unit conjunct. Let $B \to \beta_{11} \& \ldots \& \beta_{1l_1}$, $\ldots$, $B \to \beta_{r1} \& \ldots \& \beta_{rl_r}$ be all rules for B that do not contain the unit conjunct $B \to B$ ($\beta_{ij} \neq B$).

If $A \neq B$, then the rule (12) can be replaced with the collection of rules

$$A \to \alpha_1 \& \ldots \& \alpha_{k-1} \& \beta_{i1} \& \ldots \& \beta_{il_i} \& \alpha_{k+1} \& \ldots \& \alpha_m, \quad (1 \leqslant i \leqslant r), \tag{13}$$

without altering the language generated by the grammar.

If $A = B$, then the rule (12) may be just removed.

Definition 9 *Let $G = (\Sigma, N, P, S)$ be a grammar without epsilon conjuncts. A directed graph $\Gamma = (N, \{(A, B) \mid \text{there is a conjunct } A \to B \text{ in } G\})$ is called a graph of immediate reachability by unit conjuncts.*

Lemma 2 *Let $\Gamma = (\{A_1 \ldots A_n\}, E)$ be a directed graph without multiple arcs. Then, using the transformation rules*

i. Any arc (A, B) can be substituted with a possibly empty set of arcs $\{(A, C) \mid (B, C) \in E,\ C \neq B\}$,

ii. Any loop (A, A) can be removed,

it is possible to remove all arcs from the graph in finite number of steps.

Proof. First, all the arcs (A_i, A_j), such that $i > j$, are removed by the first transformation rule in the following order:

$$\begin{array}{l} (A_2, A_1), \\ (A_3, A_1), (A_3, A_2), \\ \qquad \vdots \\ (A_n, A_1), \ldots, (A_n, A_{n-1}). \end{array} \tag{14}$$

Let us prove that the removal of neither of them leads to restoration of any previously removed arc; the proof is an induction over the position of the arc in the list (14).

Basis. Since the arc (A_2, A_1) is the first in the list (14), no other arc could have been removed prior to it.

Induction step. Consider the removal of an arc (A_i, A_j) $(i > j)$. A new arc (A_i, A_k) appears in course of such a removal iff $k \neq j$ and the arc (A_j, A_k) is in the graph. Let us consider the following three cases:

i. If $k < j < i$, then the arc (A_j, A_k) was removed earlier, and, by induction hypothesis, it has not been restored since then; therefore, this case is impossible.

ii. If $j < k < i$, then the arc (A_i, A_k) appears in the list (14) *after* the current arc (A_i, A_j), and hence it could not have been removed earlier.

iii. If $i \leqslant k$, then the arc (A_i, A_k) is not in the list (14).

After all the arcs (14) are removed, all loops are discarded by the second rule. Afterwards there will be no oriented cycles left in the graph, and it is possible to remove arcs leading to sink nodes one by one, until none will remain in the graph. □

Theorem 3 *For each conjunctive grammar $G = (\Sigma, N, P, S)$ without epsilon conjuncts there exists a conjunctive grammar $G' = (\Sigma, N, P', S)$ without epsilon and unit conjuncts, such that $L(G') = L(G)$.*

Proof. Note that grammar transformations carried out by Lemma 1 are equivalent to graph transformations done by Lemma 2. Since the latter provides a way to remove all arcs from the graph of immediate reachability by unit conjuncts in finite number of steps, there is a corresponding sequence of grammar transformations by Lemma 1, which leads to removal of all unit conjuncts from the grammar. No epsilon conjuncts will appear, because right parts of conjuncts are not altered. □

3.3. Binary Normal Form

We propose a normal form that naturally extends Chomsky normal form for the case of conjunctive grammars.

Definition 10 *A conjunctive grammar $G = (\Sigma, N, P, S)$ is said to be in the binary normal form, if each rule in P is in form*

$$A \to B_1C_1\& \ldots \&B_mC_m, \quad \text{where } m \geqslant 1;\ A, B_i, C_i \in N \tag{15a}$$

$$A \to a, \quad \text{where } A \in N, a \in \Sigma, \tag{15b}$$

$$S \to \epsilon, \quad \text{only if } S \text{ does not appear in right parts of rules} \tag{15c}$$

Theorem 4 *For each conjunctive grammar $G = (\Sigma, N, P, S)$ there exists and can be effectively constructed a conjunctive grammar $G' = (\Sigma, N', P', S')$ in the binary normal form, such that $L(G) = L(G')$.*

Proof. First, epsilon conjuncts are removed by Theorem 2 and then unit conjuncts are removed by Theorem 3. At this point each conjunct in the grammar is of the form

(a) $A \to a$, where $A \in N$, $a \in \Sigma$, or

(b) $A \to \alpha$, where $A \in N$, $\alpha \in V^*$, $|\alpha| \geqslant 2$.

All rules that contain conjuncts of both types generate empty language and therefore are removed. The same applies to the rules that contain more than one conjunct of type (a) (recall that the conjuncts in every rule are distinct and therefore $A \to a\&a$ cannot be the case). Every appearance of a terminal in a conjunct of type (b) is eliminated by moving that terminal into a separate rule. All conjuncts of type (b) that are more than two nonterminals long are splitted in two by introducing new nonterminals. Finally, if $\epsilon \in L(G)$, then a new start symbol S' is introduced, $S' \to \epsilon$ is added to P', and for each rule $S \to \sigma_1\& \ldots \&\sigma_k \in P'$, a rule $S' \to \sigma_1\& \ldots \&\sigma_k$ is added to P' as well. □

3.4. Linear Normal Form

A linear conjunctive grammar $G = (\Sigma, N, P, S)$ is said to be in the linear normal form, if each rule in P is of the form

$$A \to bB_1\& \ldots \&bB_m\&C_1c\& \ldots \&C_nc \quad (m+n \geqslant 1;\ A, B_i, C_j \in N;\ b, c \in \Sigma), \tag{16a}$$

$$A \to a \quad (A \in N, a \in \Sigma), \tag{16b}$$

$$S \to \epsilon, \quad \text{only if } S \text{ does not appear in right parts of rules} \tag{16c}$$

Every linear conjunctive grammar can be effectively transformed to an equivalent grammar in the linear normal form [**?**] by first removing ϵ conjuncts and unit conjuncts using the same transformations of grammar as in Section 3.3 (these transformations are known to preserve linearity of the grammar), and then cutting long conjuncts until all of them are of the form $A \to a$, $A \to bB$ and $A \to Cc$, and finally removing the rules that ***, which obviously cannot be using in any successful derivation.

4. Recognition and Parsing

4.1. Algorithm for Grammars in Binary Normal Form

Conjunctive grammars in the binary normal form can be parsed by a variation of Cocke–Kasami–Younger algorithm. Like in the context-free case, we define a so-called *recognition matrix*, an upper-triangular matrix of sets of nonterminals that derive the substrings of the input string.

Definition 11 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar in the binary normal form. Let $w = a_1 \ldots a_n \in \Sigma^+$ $(n \geqslant 1)$ be some string. Let $1 \leqslant i \leqslant j \leqslant n$. Define*

$$T_{ij} = \{A \mid A \in N,\ A \overset{G}{\Longrightarrow}{}^* a_i \ldots a_j\} \tag{17}$$

It is clear that $w \in L(G)$ if and only if $S \in T_{1n}$. The task of computing each T_{ij} $(i < j)$ can be reduced to computing all T_{ik} $(i \leqslant k < j)$ and T_{lj} $(i < l \leqslant j)$, as shown in the following theorem.

Theorem 5 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar in the binary normal form. Let $w = a_1 \ldots a_n \in \Sigma^*$ $(n \geqslant 2)$ be some string. Let $1 \leqslant i < j \leqslant n$. Then,*

$$\begin{aligned} T_{ij} = \{A \mid A \in N,\ &\text{there is a rule } A \to B_1C_1\& \ldots \&B_mC_m \in P,\ \text{such that} \\ &\text{for any } p\ (1 \leqslant p \leqslant m)\ \text{there exists } l\ (i \leqslant l < j)\text{: } B_p \in T_{i,l},\ C_p \in T_{l+1,j}\} \end{aligned} \tag{18}$$

Proof. Nonterminal A is in T_{ij} if and only if $A \overset{G}{\Longrightarrow}{}^* a_i \ldots a_j$. The first step of the derivation cannot be an application of rule $A \to a$, because $i < j$. Hence, $A \Longrightarrow (B_1C_1\& \ldots \&B_mC_m) \Longrightarrow^* a_i \ldots a_j$ for some rule $A \to B_1C_1\& \ldots \&B_mC_m \in P$. By the first part of Theorem 1, that holds if and only if for any p $(1 \leqslant p \leqslant m)$ $B_pC_p \Longrightarrow^* a_i \ldots a_j$, which, by the second part of Theorem 1, holds if and only if for any p there is l $(i \leqslant l < j)$, such that $B_p \Longrightarrow^* a_i \ldots a_l$ (i. e. $B_p \in T_{i,l}$) and $C_p \Longrightarrow^* a_{l+1} \ldots a_j$ (i. e. $C_p \in T_{l+1,j}$), which completes the proof. □

The following algorithm computes all T_{ij} starting from $T_{11}, \ldots, T_{nn}$ and ending with T_{1n}:

Algorithm 1 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar in the binary normal form. For each $R \subseteq N \times N$ denote*

$$f(R) = \{A \mid A \in N, \text{ there is a rule } A \to B_1C_1\& \ldots \&B_mC_m \in P, \text{ such that for any } p \ (1 \leqslant p \leqslant m) \ (B_p, C_p) \in R\} \tag{19}$$

Let $w = a_1 \ldots a_n \in \Sigma^*$ $(n \geqslant 1)$ be the input string. For all i, j, such that $1 \leqslant i \leqslant j \leqslant n$, compute T_{ij}.

for $i = 1$ *to* n
    $T_{ii} = \{A \mid A \in N, \ A \to a_i \in P\}$
for $k = 1$ *to* $n - 1$
    for $i = 1$ *to* $n - k$
    {
        let $j = k + i$
        let $R = \emptyset$ $(\ R \subseteq N \times N\ )$
        for $l = i$ *to* $j - 1$
            $R = R \cup T_{il} \times T_{l+1,j}$
        $T_{ij} = f(R)$
    }

The first loop of the algorithm computes the diagonal of the matrix. Each k-th iteration of the second outer loop deals with the computation of all T_{ij} such that $j - i = k$, and the nested loop by i goes through all $T_{i,k+i}$. For each of these T_{ij} we compute the set R of pairs of nonterminals (B, C), such that $BC \Longrightarrow^* a_i \ldots a_j$; this is being done by the inner loop, which considers all factorizations of $a_i \ldots a_j$ into two nonempty substrings.

Then, the mapping f is used to determine all nonterminals that have rules comprised entirely from the conjuncts whose right parts are contained in R. This set $f(R)$ is, by Theorem 5, exactly the set of nonterminals that derive $a_i \ldots a_j$.

The difference between Algorithm 1 and the original Cocke–Kasami–Younger algorithm is that in the case of conjunctive grammars one has to accumulate all the pairs (B, C) from the different factorizations of the current substring, and only the full set of such pairs can be used to determine the membership of nonterminals in T_{ij}.

However, that does not increase the complexity of the algorithm.

Theorem 6 *For each conjunctive grammar $G = (\Sigma, N, P, S)$, the Algorithm 1 computes $\{T_{ij}\}$ as in Definition 11 for any input string $w \in \Sigma^+$, and $w \in L(G)$ iff S is in the constructed set $T_{1,|w|}$.*

When implemented on a RAM, the algorithm works in $O(n^3)$ time and uses $O(n^2)$ space.

Once the string is recognized, all of its derivation trees may be constructed by analyzing the recognition matrix in the way similar to the context-free case.

4.2. Algorithm for Grammars in Linear Normal Form

Let T_{ij} be as in Definition 11. Exactly like in the case of linear context-free grammars, T_{ij} $(i < j)$ is a function of $T_{i+1,j}$, $T_{i,j-1}$, a_i and a_j. The following theorem is proved very much like Theorem 5:

Theorem 7 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar in the linear normal form. Let $w = a_1 \dots a_n \in \Sigma^*$ $(n \geqslant 2)$ be some string. Let $1 \leqslant i < j \leqslant n$. Then,*

$$\begin{aligned} T_{ij} = \{A \mid A \in N,\ &\text{there is a rule } A \to bB_1\& \dots \& bB_m \& C_1 c \& \dots \& C_n c \in P, \\ &\text{such that } b = a_i,\ c = a_j,\ \text{for all } p\,(1 \leqslant p \leqslant m)\ B_p \in T_{i+1,j} \text{ and} \\ &\text{for all } q\,(1 \leqslant q \leqslant n)\ C_q \in T_{i,j-1}\} \end{aligned} \tag{20}$$

Algorithm 2 *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar in the linear normal form. Let $w = a_1 \dots a_n \in \Sigma^*$ $(n \geqslant 1)$ be the string being recognized. Compute T_{1n}.*

for $i = 1$ *to* n
    $T_{ii} = \{A \mid A \in N,\ A \to a_i \in P\}$
for $k = 1$ *to* $n - 1$
    for $i = 1$ *to* $n - k$
    {
        let $j = k + i$
        $T_{ij} =$ (as in (20))
    }

It is easily seen that each iteration of the outer loop depends only on the input string and the products of the previous iteration. That allows us to discard earlier portions of the matrix in course of the computation and thus use only $O(n)$ space. The time complexity of the algorithm is quadratic, since the computation of (20) takes constant time.

5. Descriptional Complexity

In this section we shall investigate the relative succinctness of representation of context-free languages by context-free grammars and conjunctive grammars, as well as the succinctness of representation of finite intersections of context-free languages by conjunctive grammars.

We shall consider two descriptional complexity measures: the cardinality of the set of nonterminals and the total number of symbols used in the description of the grammar. The latter is denoted $|G|$ and defined as

$$|G| = \sum_{A \to \alpha_1 \& \dots \& \alpha_n \in P} (n + 1 + \sum_{i=1}^{n} |\alpha_i|) \tag{21}$$

for any conjunctive grammar $G = (\Sigma, N, P, S)$.

Let us consider the following sequence of context-free languages:

$$L^{(k)} = \{wc^n wd^n \mid n \geqslant 1, w \in \{a,b\}^*, |w| = k\}, \quad k > 0 \tag{22}$$

Theorem 8 *For any $k > 0$ the minimal context-free grammar for $L^{(k)}$ has exactly $2^k + 1$ nonterminals.*

Proof. The construction of such a grammar is straightforward and is omitted.

Let us prove that any grammar generating $L^{(k)}$ must have at least $2^k + 1$ nonterminals. Consider an arbitrary context-free grammar $G = (\{a,b,c,d\}, N, P, S)$, such that $L(G) = L^{(k)}$. For each string $u \in \{a,b\}^k$, define the set

$$N_u = \{A \mid A \in N,\ A \Longrightarrow^* c^i A d^i\ (i > 0),\ A \Longrightarrow^* c^p u d^q\ (p,q > 0)\} \tag{23}$$

The collection of sets (23) has the following properties:

- For each $u \in \{a,b\}^k$, $N_u \neq \emptyset$. In order to prove that, we fix some string u and consider a sufficiently large n, such that the conditions of the pumping lemma hold for $uc^n ud^n$. It is easily seen that the nonterminal and the factorization promised by the lemma satisfy (23).
- For each $u, v \in \{a,b\}^k$ $(u \neq v)$, $N_u \cap N_v = \emptyset$ (otherwise the grammar would generate strings of the form $uc^i vd^j$).
- S does not belong to any N_u (otherwise the grammar would generate strings which begin with c).

Since there are 2^k different strings in $\{a,b\}^k$, it holds that $|N| \geqslant 2^k + 1$. □

On the other hand, for every k there exists a conjunctive grammar for $L^{(k)}$, in which the number of nonterminals does not depend upon k. Consider a quite natural representation of $L^{(k)}$ as a language of strings satisfying three conditions:

$$L^{(k)} = \{uc^i vd^j \mid i = j\} \cap \{uc^i vd^j \mid u = v\} \cap \{uc^i vd^j \mid |u| = k\}, \tag{24}$$

The first language is context-free; the second can be obtained by modifying the grammar from Example 2; the third can be generated by a grammar with a constant number of nonterminals and $O(k)$ length of description.

This allows to conclude that the transition from conjunctive grammars to context-free grammars may lead to unbounded growth in the number of nonterminals.

In order to investigate the succinctness of the total length of description, we shall use the method of [4, 5] to show that the length of description of intersective context-free languages by the tuples of context-free grammars (which denote intersection of languages) is not bounded by any recursive function of the length of their description by conjunctive grammars.

First we shall prove an auxiliary result about the language of all valid computations of a given Turing machine, which was discussed in Example 3.

Lemma 3 *Let M be a deterministic Turing machine that cannot print a blank and makes at least one move before halting. Then VALC$[M]$ is in the intersection closure of the context-free languages iff $L(M)$ is finite.*

Proof. If $L(M)$ is finite, then there are only finitely many accepting computations of M. On the other hand, if $L(M)$ is infinite, then for input strings of unbounded length the first two configurations of the Turing machine must be related, which gives an unbounded number of cross-dependencies and therefore implies that VALC$[M]$ cannot be represented as a finite intersection of context-free languages. □

Let us consider some effective enumeration $\{G_i\}$ of the conjunctive grammars. Let $\mathcal{L}_{ICF}$ denote the intersection closure of the context-free languages.

Lemma 4 *The language* $L = \{G_i \mid L(G_i) \notin \mathcal{L}_{ICF}\}$ *is not recursively enumerable.*

Proof. Due to Lemma 3, if L were recursively enumerable, then the set of Turing machines generating infinite languages would also be recursively enumerable, which contradicts Rice's theorem. □

In accordance with a theorem by Hartmanis ([5], Theorem 4), our Lemma 4, together with the decidability of the membership problem for conjunctive languages shown in Sections 3 and 4, implies the following result:

Theorem 9 *The relative succinctness of representing languages from* $\mathcal{L}_{ICF}$ *by conjunctive grammars and by tuples of context-free grammars is not bounded by any recursive function.*

6. General Properties of Conjunctive Languages

6.1. Decision Problems

Since many decision problems are known to be unsolvable for finite intersections of context-free languages, the same negative results also hold for conjunctive languages. Namely, emptiness, finiteness, regularity, context-freeness, inclusion and equivalence are undecidable.

On the contrary, membership is polynomially decidable, as shown in Section 4.

6.2. Closure Properties

It is easily established that the family of conjunctive languages is closed under union, intersection, concatenation and Kleene star, because each of these operations is explicitly (or, in case of the star, almost explicitly) included in the grammar formalism. $\mathcal{L}_{\&}$ is also closed under reversal.

Nonclosure under homomorphism follows from [2] (Theorem 3.1), since $\mathcal{L}_{\&}$ is obviously not equal to the whole family of recursively enumerable sets. $\mathcal{L}_{\&}$ is also not closed under prefix, suffix and substring, which can easily be deduced from the conjunctiveness of a language of all valid computations of a given Turing machine. It is an open problem whether the family of conjunctive languages is closed under complement and under ϵ-free homomorphism.

It must be noted that if $L_1 = \{ww \mid w \in \{a,b\}^*\}$ is not a conjunctive language, then that would imply nonclosure of $\mathcal{L}_\&$ under both ϵ-free homomorphism and complement. If $L_2 = \{a^{n^2} \mid n \geqslant 1\}$ is not conjunctive, then $\mathcal{L}_\&$ is not closed under ϵ-free homomorphism, because L_2 is an ϵ-free homomorphic image of the conjunctive language $\{ab^2ab^4ab^6a \dots b^{2n}a \mid n \geqslant 0\}$. However, no method to prove nonconjunctiveness of a given language has been developed so far.

6.3. Relationship to Other Language Families

Obviously, the intersection closure of the context-free languages is a subset of $\mathcal{L}_\&$. It follows from [10] and Example 2 that this inclusion is strict.

On the other hand, $\mathcal{L}_\&$ is properly contained in the family of languages generated by the alternating context-free grammars, since $\mathcal{L}_\& \subset \mathbf{P} \subset \mathbf{EXPTIME}$.

The languages generated by conjunctive grammars cannot be classified as mildly context-sensitive, since they do not necessary have constant growth property. Moreover, the sequence of lengths of the strings from a conjunctive language can grow arbitrarily fast (this follows from the conjunctiveness of the language of all valid computations of a given Turing machine). In this connection, it must be noted that Parikh theorem does not hold for the conjunctive languages.

It can be proved by direct construction of a linear bounded automaton that every conjunctive language can be recognized in nondeterministic linear space and therefore is context-sensitive. Under the assumption that $\mathbf{P} \neq \mathbf{PSPACE}$, it is easily obtained that $\mathcal{L}_\&$ is properly included in the family of context-sensitive languages. But, since no particular context-sensitive language is known to be nonconjunctive, there is still no "real" proof of strictness of this inclusion.

It might be reasonable to expect that conjunctive grammars over unary alphabet generate regular languages, since context-free grammars over unary alphabet do; however, the proof from the context-free case heavily relies upon the properties of the context-free languages, which are missing in our case, and thus this question requires further study.

7. Conclusion

We have introduced a generalization of the context-free grammars, which adds an explicit intersection operation to the formalism of rules, extends the model's descriptive power beyond the intersection closure of the context-free languages, and allows to denote important language constructs, thus meeting the needs of many practical applications.

It turned out that conjunctive grammars retain the notion of derivation tree and some efficient recognition algorithms from the context-free grammar theory, which allows to carry on the existing context-free parsing to the new class of grammars.

However, many important language-theoretic properties of conjunctive grammars remain uncovered, and in light of the practical importance of the model, these issues seem to be worth further investigation.

Acknowledgements

I wish to thank Kai Salomaa, Larisa I. Stanevichene, Alexey A. Vylitok, Detlef Wotschke and Vladimir A. Zakharov for their helpful and valuable comments on different versions of this paper.

I am grateful to my referees from both DCAGRS and JALC for their pertinent remarks.

References

[1] A. V. Aho, R. Sethi, J. D. Ullman. *Compilers: principles, techniques and tools*, Addison-Wesley, Reading, Mass., 1986.

[2] S. Ginsburg, S. A. Greibach and M. A. Harrison, "One-way stack automata". *Journal of the ACM*, 14:2 (1967) 389–418.

[3] I. Gorun, "A hierarchy of context-sensitive languages", *Lecture Notes in Computer Science*, 45 (1976), 299–303.

[4] J. Hartmanis, "On the succinctness of different representations of languages", *SIAM Journal on Computing*, 9 (1980), 114–120.

[5] J. Hartmanis, "On Gödel speed-up and succinctness of language representations", *Theoretical Computer Science*, 26 (1983), 335–342.

[6] O. H. Ibarra, T. Jiang, H. Wang, "A characterization of exponential-time languages by alternating context-free grammars", *Theoretical Computer Science*, 99 (1992) 301–313.

[7] M. Latta, R. Wall, "Intersective context-free languages", *Lenguajes Naturales y Lenguajes Formales IX*, Barcelona, 1993, 15–43.

[8] L. Y. Liu and P. Weiner, "An infinite hierarchy of intersections of context-free languages". *Mathematical Systems Theory*, 7 (1973) 187–192.

[9] E. Moriya, "A grammatical characterization of alternating pushdown automata", *Theoretical Computer Science*, 67 (1989) 75–85.

[10] D. Wotschke, "The Boolean closures of deterministic and nondeterministic context-free languages", *Lecture Notes in Computer Science*, 1 (1973), 113–121.

[11] D. Wotschke, "Nondeterminism and Boolean operations in PDAs", *Journal of Computer and Systems Sciences*, 16:3 (1978), 456–461.