

Defining contexts in context-free grammars^{*}

Mikhail Barash^{1,2} and Alexander Okhotin¹

¹ Department of Mathematics, University of Turku, Turku FI-20014, Finland,
mikhail.barash@utu.fi, alexander.okhotin@utu.fi

² Turku Centre for Computer Science, Turku FI-20520, Finland.

Abstract. Conjunctive grammars (Okhotin, 2001) are an extension of the standard context-free grammars with a conjunction operation, which maintains most of their practical properties, including many parsing algorithms. This paper introduces a further extension to the model, which is equipped with quantifiers for referring to the left context, in which the substring being defined does occur. For example, a rule $A \rightarrow a \ \< B$ defines a string a , as long as it is preceded by any string defined by B . The paper gives two equivalent definitions of the model—by logical deduction and by language equations—and establishes its basic properties, including a transformation to a normal form, a cubic-time parsing algorithm, and another recognition algorithm working in linear space.

1 Introduction

Context-free grammars are a logic for defining the syntax of languages. In this logic, the properties of strings are defined inductively, so that the properties of a string are determined by the properties of its substrings. This is how a rule $S \rightarrow aSb$ asserts, that if a string $a^{n-1}b^{n-1}$ has the property S , then the string $a^n b^n$ has the property S as well. Besides the concatenation, the formalism of this logic has an implicit disjunction operation, represented by having multiple rules for a single symbol. This logic can be further augmented with conjunction and negation operations, which was done by the second author [11,13] in *conjunctive grammars* and *Boolean grammars*, respectively. These grammars preserve the main idea of the context-free grammars—that of defining syntax inductively, as described above—maintain most of their practically important features, such as efficient parsing algorithms [13,15,16,17], and have been a subject of diverse research [1,4,7,8,10,18]. As the applicability of a rule of a Boolean grammar to a substring is independent of the context, in which the substring occurs, Boolean grammars constitute a natural general case of context-free grammars. Standard context-free grammars can be viewed as their disjunctive fragment.

When Chomsky [2] introduced the term “context-free grammar” for an intuitively obvious model of syntax, he had a further idea of a more powerful model, in which one could define rules applicable only in some particular contexts. However, Chomsky’s attempt to formalize his idea using the tools available at the

^{*} Supported by the Academy of Finland under grant 134860.

time (namely, the string-rewriting systems) led to nothing but space-bounded nondeterministic Turing machines. Even though the resulting devices are still known under the name of “context-sensitive grammars”, they have nothing to do with the syntax of languages: the nonterminal symbols of these grammars are, in general, nothing but bits in a memory of a general-purpose computer, and do not correspond to any syntactic notions. In particular, these grammars fail to implement Chomsky’s original idea of a phrase-structure rule applicable in a context.

This paper undertakes to reconsider Chomsky’s [2] idea of contexts in grammars, this time using the appropriate tools of deduction systems and language equations, and drawing from the experience of developing the conjunctive grammars. The model proposed in this paper are *grammars with one-sided contexts*, which introduce two special quantifiers for representing left contexts of a string. The first quantifier refers to the “past” of the current substring within the entire string being defined: an expression $\triangleleft\alpha$ defines any substring that is directly preceded by a prefix of the form α . This quantifier is meant to be used along with usual, unquantified specifications of the structure of the current substring, using conjunction to combine several specifications. For example, consider the rule $A \rightarrow BC \ \& \ \triangleleft D$, which represents any substring of the form BC preceded by a substring of the form D . If the grammar contains additional rules $B \rightarrow b$, $C \rightarrow c$ and $D \rightarrow d$, then the above rule for A shall specify that a substring bc of a string $w = dbc \dots$ has the property A ; however, this rule shall not produce the same substring occurring in the strings $w' = abc$ or $w'' = adbc$. The other quantifier, $\trianglelefteq\alpha$, represents the form of the current substring together with its left context, so that the rules $A \rightarrow B \ \& \ \trianglelefteq E$, $B \rightarrow b$, $E \rightarrow ab$ define that the substring b occurring in the string $w = ab$ has the property A . One can symmetrically define right contexts, denoted by quantifiers $\triangleright\alpha$ and $\trianglerighteq\alpha$.

In the literature, related ideas have occasionally arisen in connection with parsing, where right contexts— $\triangleright\alpha\Sigma^*$, in the terminology of this paper—are considered as “lookahead strings” and are used to guide a deterministic parser. If α represents a regular language, these simple forms of contexts occur in LR-regular [3], LL-regular [9] and LL(*) [19] parsers. Some software tools for engineering parsers, such as those developed by Parr and Fischer [19] and by Ford [5], allow specifying contexts $\triangleright\alpha\Sigma^*$, with α defined within the grammar, and such specifications can be used by a programmer to adjust the behaviour of a deterministic recursive descent parser.

In this paper, the above intuitive definition of grammars with one-sided contexts is formalized in two equivalent ways. The first formalization, pursued in Section 2, uses *deduction* of elementary propositions of the form $[A, \langle u \rangle v]$, where $\langle u \rangle v$ denotes a substring v in left context u (that is, occurring in a string uvw) and A is a syntactic property defined by the grammar (“nonterminal symbol” in Chomsky’s terminology); this proposition asserts that v has the property A in the context u . Then, each rule of the grammar, which is of the general form $A \rightarrow \alpha_1 \ \& \ \dots \ \& \ \alpha_k \ \& \ \triangleleft\beta_1 \ \& \ \dots \ \& \ \triangleleft\beta_m \ \& \ \trianglelefteq\gamma_1 \ \& \ \dots \ \& \ \trianglelefteq\gamma_n$, becomes a deduction scheme for inferring elementary propositions of this form from each other,

and the language generated by the grammar is ultimately defined as the set of all strings w , such that $[S, \langle \varepsilon \rangle w]$ can be deduced. The standard proof tree of such a deduction constitutes a parse tree of the string w . This definition generalizes the representation of standard context-free grammars by deduction—assumed, for instance, in the monograph by Sikkel [20]—as well as the extension of this representation to conjunctive grammars [14].

An alternative, equivalent definition given in Section 3 uses a generalization of *language equations*, in which the unknowns are *sets of pairs* of a string and its left contexts. All connectives and quantifiers in the rules of a grammar—that is, concatenation, disjunction, conjunction and both context quantifiers—are then interpreted as operations on such sets, and the resulting system of equations is proved to have a least fixpoint, as in the known cases of standard context-free grammars [6] and conjunctive grammars [12]. This least solution defines the language generated by the grammar.

These definitions ensure that the proposed grammars with one-sided contexts define the properties of strings inductively from the properties of their substrings and the contexts, in which these substrings occur. There is no rewriting of “sentential forms” involved, and hence the proposed model avoids falling into the same pit as Chomsky’s “context-sensitive grammars”, that of being able to simulate computations of space-bounded Turing machines.

This paper settles the basic properties of grammars with one-sided contexts. First, a transformation to a normal form generalizing the Chomsky normal form is devised in Section 4; the construction proceeds in the usual way, first by eliminating empty strings, and then by removing cyclic dependencies. This normal form is then used to extend the basic Cocke–Kasami–Younger parsing algorithm to grammars with one-sided contexts; the algorithm, described in Section 5, works in time $O(n^3)$, where n is the length of the input string. Finally, in Section 6, it is demonstrated that every language defined by a grammar with one-sided contexts can be recognized in deterministic linear space.

In this extended abstract, most proofs are omitted due to space constraints.

2 Definition by deduction

A grammar with one-sided contexts uses concatenation, conjunction and disjunction, as well as quantifiers, either only $\{\langle, \triangleleft\}$ for left contexts, or only $\{\triangleright, \triangleright\}$ for right contexts. Though left contexts are assumed throughout this paper, all results symmetrically hold for grammars with right contexts.

Definition 1. *A grammar with left contexts is a quadruple $G = (\Sigma, N, R, S)$, where*

- Σ is the alphabet of the language being defined;
- N is a finite set of auxiliary symbols (“nonterminal symbols” in Chomsky’s terminology), disjoint with Σ , which denote the properties of strings defined in the grammar;

– R is a finite set of grammar rules, each of the form

$$A \rightarrow \alpha_1 \& \dots \& \alpha_k \& \triangleleft \beta_1 \& \dots \& \triangleleft \beta_m \& \trianglelefteq \gamma_1 \& \dots \& \trianglelefteq \gamma_n, \quad (1)$$

with $A \in N$, $k, m, n \geq 0$, $\alpha_i, \beta_i, \gamma_i \in (\Sigma \cup N)^*$;

– $S \in N$ is a symbol representing correct sentences (“start symbol” in the common jargon).

A grammar with left contexts degenerates to a conjunctive grammar, if the context quantifiers are never used, that is, if $m = n = 0$ for every rule (1); and further to a standard context-free grammar, if conjunction is never used, that is, if $k = 1$ in every rule.

For each grammar rule (1), each term α_i , $\triangleleft \beta_i$ and $\trianglelefteq \gamma_i$ is called a conjunct. Each unquantified conjunct α_i gives a representation of the string being defined. A conjunct $\triangleleft \beta_i$ similarly describes the form of the *left context* or the *past*, relative to the string being defined. Conjuncts of the form $\trianglelefteq \gamma_i$ refer to the form of the left context and the current string, concatenated into a single string. Intuitively, such a rule asserts that every substring v occurring in the left context u , such that v is representable as each α_i , u is representable as each β_i and uv is representable as each γ_i , therefore has the property A .

Formally, the semantics of grammars with contexts are defined by a deduction system of elementary propositions (items) of the form “a string $v \in \Sigma^*$ written in left context $u \in \Sigma^*$ has the property $\alpha \in (\Sigma \cup N)^*$ ”, denoted by $[\alpha, \langle u \rangle v]$.

Definition 2. Let $G = (\Sigma, N, R, S)$ be a grammar with contexts, and define the following deduction system of items of the form $[X, \langle u \rangle v]$, with $X \in \Sigma \cup N$ and $u, v \in \Sigma^*$. There is a single axiom scheme:

$$\vdash_G [a, \langle x \rangle a] \quad (\text{for all } a \in \Sigma \text{ and } x \in \Sigma^*).$$

Each rule $A \rightarrow \alpha_1 \& \dots \& \alpha_k \& \triangleleft \beta_1 \& \dots \& \triangleleft \beta_m \& \trianglelefteq \gamma_1 \& \dots \& \trianglelefteq \gamma_n$ in the grammar defines a scheme $I \vdash_G [A, \langle u \rangle v]$ for deduction rules, for all $u, v \in \Sigma^*$ and for every set of items I satisfying the below properties:

- For every unquantified conjunct $\alpha_i = X_1 \dots X_\ell$ with $\ell \geq 0$ and $X_j \in \Sigma \cup N$, there should exist a partition $v = v_1 \dots v_\ell$ with $[X_j, \langle uv_1 \dots v_{j-1} \rangle v_j] \in I$ for all $j \in \{1, \dots, \ell\}$;
- For every conjunct $\triangleleft \beta_i = \triangleleft X_1 \dots X_\ell$ with $\ell \geq 0$ and $X_j \in \Sigma \cup N$, there should be such a partition $u = u_1 \dots u_\ell$, that $[X_j, \langle u_1 \dots u_{j-1} \rangle u_j] \in I$ for all $j \in \{1, \dots, \ell\}$;
- Every conjunct $\trianglelefteq \gamma_i = \trianglelefteq X_1 \dots X_\ell$ with $\ell \geq 0$ and $X_j \in \Sigma \cup N$ should have a corresponding partition $uv = w_1 \dots w_\ell$ with $[X_j, \langle w_1 \dots w_{j-1} \rangle w_j] \in I$ for all $j \in \{1, \dots, \ell\}$.

Note, that if $\alpha_i = \varepsilon$ in the first case, then the given condition implies $v = \varepsilon$, and similarly, $\beta_i = \varepsilon$ implies $u = \varepsilon$, and $\gamma_i = \varepsilon$ implies $u = v = \varepsilon$.

Then the language generated by a nonterminal symbol A is defined as

$$L_G(A) = \{ \langle u \rangle v \mid u, v \in \Sigma^*, \vdash_G [A, \langle u \rangle v] \}.$$

The language generated by the grammar G is the set of all strings with left context ε generated by S :

$$L(G) = \{ w \mid w \in \Sigma^*, \vdash_G [S, \langle \varepsilon \rangle w] \}.$$

Using both kinds of past quantifiers (\triangleleft and \trianglelefteq) is actually redundant, because each of them can be expressed through the other as follows:

- $\triangleleft D$ is equivalent to $D' \Sigma^*$, for a new symbol D' with a rule $D' \rightarrow \varepsilon \& \triangleleft D$;
- $\trianglelefteq E$ can be replaced by $\Sigma^* E''$, where E'' has the unique rule $E'' \rightarrow \varepsilon \& \triangleleft E$.

Using both types of quantifiers together shall be essential later, when transforming a grammar into a normal form, where the empty string cannot be defined.

The following sample grammar with contexts defines a rather simple language, which is an intersection of two standard context-free languages. The value of this example is in demonstrating the machinery of contexts in action.

Example 1. The following grammar generates the language $\{ a^n b^n c^n d^n \mid n \geq 0 \}$:

$$\begin{aligned} S &\rightarrow aSd \mid bSc \mid \varepsilon \& \triangleleft A \\ A &\rightarrow aAb \mid \varepsilon \end{aligned}$$

The symbol A generates all strings $a^n b^n$ with $n \geq 0$ in any context. Without the context specification $\triangleleft A$, the symbol S would define all strings of the form $wh(w^R)$, where $w \in \{a, b\}^*$ and the homomorphism h maps a to d and b to c . However, the rule $S \rightarrow \varepsilon \& \triangleleft A$ ensures that the first half of the string (the prefix ending with the last b) is of the form $a^n b^n$ for some $n \geq 0$, and therefore S generates $\{ a^n b^n c^n d^n \mid n \geq 0 \}$. Consider the following logical derivation of the fact that the string $abcd$ with the left context ε is defined by S .

$$\begin{aligned} &\vdash [a, \langle \varepsilon \rangle a] && (axiom) \\ &\vdash [b, \langle a \rangle b] && (axiom) \\ &\vdash [c, \langle ab \rangle c] && (axiom) \\ &\vdash [d, \langle abc \rangle d] && (axiom) \\ &\vdash [A, \langle a \rangle \varepsilon] && (A \rightarrow \varepsilon) \\ [a, \langle \varepsilon \rangle a], [A, \langle a \rangle \varepsilon], [b, \langle a \rangle b] &\vdash [A, \langle \varepsilon \rangle ab] && (A \rightarrow aAb) \\ &[A, \langle \varepsilon \rangle ab] \vdash [S, \langle ab \rangle \varepsilon] && (S \rightarrow \varepsilon \& \triangleleft A) \\ [b, \langle a \rangle b], [S, \langle ab \rangle \varepsilon], [c, \langle ab \rangle c] &\vdash [S, \langle a \rangle bc] && (S \rightarrow bSc) \\ [a, \langle \varepsilon \rangle a], [S, \langle a \rangle bc], [d, \langle abc \rangle d] &\vdash [S, \langle \varepsilon \rangle abcd] && (S \rightarrow aSd) \end{aligned}$$

The tree corresponding to this deduction is given in Figure 1, where the dependence upon a context is marked by a dotted arrow.

Example 2. The following grammar generates the language $\{ u_1 \dots u_n \mid \text{for every } i, u_i \in a^*c, \text{ or there exist } j, k \text{ with } u_j = b^k c \text{ and } u_k = a^k c \}$.

$$\begin{aligned} S &\rightarrow AcS \mid CcS \mid BcS \& DcE \mid \varepsilon \\ A &\rightarrow aA \mid \varepsilon && C \rightarrow B \& \trianglelefteq EF && E \rightarrow AcE \mid BcE \mid \varepsilon \\ B &\rightarrow bB \mid \varepsilon && D \rightarrow bDa \mid cE && F \rightarrow aFb \mid cE \end{aligned}$$

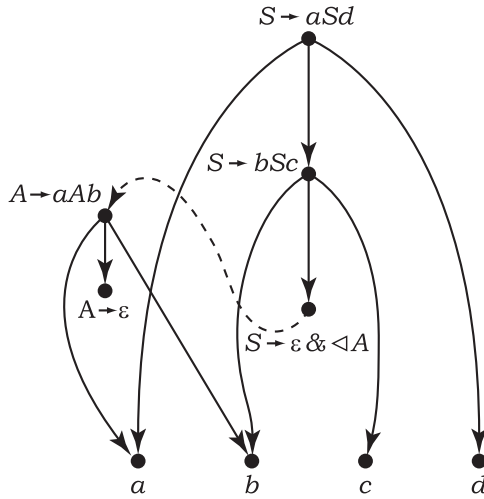


Fig. 1: A parse tree of the string $abcd$ according to the grammar in Example 1.

This is an abstract language representing declaration of identifiers *before or after* their use. Substrings of the form $a^k c$ represent declarations, while every substring of the form $b^k c$ is a reference to a declaration of the form $a^k c$.

The idea of the grammar is that S should generate a string $\langle u_1 \dots u_\ell \rangle u_{\ell+1} \dots u_n$ with $u_i \in a^* c \cup b^* c$ if every “reference” in the suffix $u_{\ell+1} \dots u_n$ has a corresponding “declaration” in the whole string $u_1 \dots u_n$. This condition is defined inductively on ℓ . The rule $S \rightarrow \varepsilon$ is the basis of induction: the string $\langle u_1 \dots u_n \rangle \varepsilon$ has the desired property. The rule $S \rightarrow CcS$ appends a reference of the form $(b^* \& \leq EF)c$, where the context specification ensures that this “reference” has a matching *earlier* “declaration”. The possibility of a *later* “declaration” is checked by another rule $S \rightarrow BcS \& DcE$.

For the language in Example 2, no Boolean grammar is known (and hence no conjunctive grammar either). At the time of writing, this is the only such example known to the authors. This is due to the simple reason that conjunctive grammars are already quite powerful, and can define most of the standard examples of formal languages. No methods of proving that a language does not have a conjunctive grammar are currently known, and hence there is no proof that grammars with contexts generate a larger class of languages.

3 Definition by language equations

The representation of standard context-free grammars by language equations, introduced by Ginsburg and Rice [6], is one of the several ways of defining their semantics. For example, a grammar $S \rightarrow aSb \mid \varepsilon$ is regarded as an equation

$S = (\{a\} \cdot S \cdot \{b\}) \cup \{\varepsilon\}$, which has a unique solution $S = \{a^n b^n \mid n \geq 0\}$. Conjunctive grammars inherit the same definition by equations [12], with the conjunction represented by the intersection operation.

This important representation can be extended to grammars with contexts. However, in order to include the contexts in the equations, the whole model has to be extended from ordinary formal languages to sets of pairs of the form $\langle u \rangle v$, that is, to languages of pairs $L \subseteq \Sigma^* \times \Sigma^*$. All usual operations on languages used in equations are redefined for languages of pairs as follows. For all $K, L \subseteq \Sigma^* \times \Sigma^*$, consider their

- union $K \cup L = \{\langle u \rangle v \mid \langle u \rangle v \in K \text{ or } \langle u \rangle v \in L\}$,
- intersection $K \cap L = \{\langle u \rangle v \mid \langle u \rangle v \in K, \langle u \rangle v \in L\}$,
- concatenation $K \cdot L = \{\langle u \rangle vw \mid \langle u \rangle v \in K, \langle uv \rangle w \in L\}$,
- \triangleleft -context $\triangleleft L = \{\langle u \rangle v \mid \langle \varepsilon \rangle u \in L, v \in \Sigma^*\}$, and
- \trianglelefteq -context $\trianglelefteq L = \{\langle u \rangle v \mid \langle \varepsilon \rangle w \in L, w = uv\}$.

With respect to the partial order of inclusion, all these operations are monotone and continuous. Hence, every system of equations $X_i = \varphi_i(X_1, \dots, X_n)$ with $i \in \{1, \dots, n\}$, in which X_i are unknown languages of pairs and the right-hand sides φ_i are comprised of the above operations, has a least solution. A grammar is represented as such a system in the most direct way.

Definition 3. For every grammar with contexts $G = (\Sigma, N, R, S)$, the associated system of language equations is a system of equations in variables N , in which each variable assumes a value of a language of pairs $L \subseteq \Sigma^* \times \Sigma^*$, and which contains the following equations for every variable A :

$$A = \bigcup_{\substack{A \rightarrow \alpha_1 \& \dots \& \alpha_k \& \\ \& \triangleleft \beta_1 \& \dots \& \triangleleft \beta_m \& \\ \& \trianglelefteq \gamma_1 \& \dots \& \trianglelefteq \gamma_n \in R}} \left[\bigcap_{i=1}^k \alpha_i \cap \bigcap_{i=1}^m \triangleleft \beta_i \cap \bigcap_{i=1}^n \trianglelefteq \gamma_i \right]. \quad (2)$$

Each instance of a symbol $a \in \Sigma$ in such a system defines the language $\{\langle x \rangle a \mid x \in \Sigma^*\}$, and each empty string denotes the language $\{\langle x \rangle \varepsilon \mid x \in \Sigma^*\}$.

Let $(L_{A_1}, \dots, L_{A_n})$ with $L_{A_i} \subseteq \Sigma^* \times \Sigma^*$ be the least solution of the system. Define $L_G(A) = L_A$, and let $L(G) = \{w \mid \langle \varepsilon \rangle w \in L_S\}$.

For instance, the grammar in Example 1 induces the system

$$\begin{aligned} S &= \langle \Sigma^* \rangle a \cdot S \cdot \langle \Sigma^* \rangle d \cup \langle \Sigma^* \rangle b \cdot S \cdot \langle \Sigma^* \rangle c \cup (\langle \Sigma^* \rangle \varepsilon \cap \triangleleft A) \\ A &= \langle \Sigma^* \rangle a \cdot A \cdot \langle \Sigma^* \rangle b \cup \langle \Sigma^* \rangle \varepsilon \end{aligned}$$

with a unique solution $S = \{\langle a^i \rangle a^{n-i} b^n c^n d^n \mid n \geq i \geq 0\} \cup \{\langle a^n b^i \rangle b^{n-i} c^n d^n \mid n \geq i \geq 0\}$, $A = \{\langle x \rangle a^n b^n \mid x \in \{a, b\}^*\}$.

Definitions 2 and 3 are proved equivalent as follows.

Theorem 1. Let $G = (\Sigma, N, R, S)$ be a grammar with contexts, let $X = \varphi(X)$ be the associated system of equations. For every $A \in N$ and $\langle u \rangle v \in \Sigma^* \times \Sigma^*$,

$$\langle u \rangle v \in \left[\bigsqcup_{k \geq 0} \varphi^k(\emptyset, \dots, \emptyset) \right]_A \quad \text{if and only if} \quad \vdash_G [A, \langle u \rangle v].$$

4 Normal form

Consider the Chomsky normal form for standard context-free grammars, in which all rules are of the form $A \rightarrow BC$ and $A \rightarrow a$. Its extension to conjunctive grammars allows rules of the form $A \rightarrow B_1C_1 \& \dots \& B_kC_k$. The transformation to this normal form is done by first eliminating ε -conjuncts, that is, rules of the form $A \rightarrow \varepsilon \& \dots$, and then removing *unit conjuncts*, or rules of the form $A \rightarrow B \& \dots$ [11]. This transformation shall now be further extended to grammars with contexts.

The task of eliminating ε -conjuncts is formulated in the same way: for any given grammar with contexts, the goal is to construct an equivalent (modulo the membership of ε) grammar without ε -conjuncts. A similar construction for context-free grammars (as well as for conjunctive grammars) begins with determining the set of nonterminals that generate the empty string, which is obtained as a least upper bound of an ascending sequence of sets of nonterminals. For grammars with contexts, it is necessary to consider pairs of the form (A, Z) , with $A \in N$ and $Z \subseteq N$, representing the intuitive idea that A generates ε in the context of the form described by all nonterminals in Z . The set of all such pairs is obtained as a limit of a sequence of sets as follows.

Definition 4. Let $G = (\Sigma, N, R, S)$ be a grammar with contexts. Assume, without loss of generality, that in each rule of the grammar, the context quantifiers are applied only to single nonterminal symbols rather than concatenations. Construct the sequence of sets $\text{NULLABLE}_i(G) \subseteq N \times 2^N$, with $i \geq 0$, by setting $\text{NULLABLE}_0(G) = \emptyset$ and

$$\begin{aligned} \text{NULLABLE}_{i+1}(G) = \{ & (A, \{D_1, \dots, D_m\} \cup \{E_1, \dots, E_n\} \cup Z_1 \cup \dots \cup Z_k) \mid \\ & A \rightarrow \alpha_1 \& \dots \& \alpha_k \& \triangleleft D_1 \& \dots \& \triangleleft D_m \& \trianglelefteq E_1 \& \dots \& \trianglelefteq E_n \in R, \\ & Z_1, \dots, Z_k \subseteq N : (\alpha_1, Z_1), \dots, (\alpha_k, Z_k) \in \text{NULLABLE}_i^*(G) \}, \end{aligned}$$

where \mathcal{S}^* , for any set $\mathcal{S} \subseteq N \times 2^N$, denotes the set of all pairs $(A_1 \dots A_\ell, Z_1 \cup \dots \cup Z_\ell)$ with $\ell \geq 0$ and $(A_i, Z_i) \in \mathcal{S}$.

Finally, let $\text{NULLABLE}(G) = \bigcup_{i \geq 0} \text{NULLABLE}_i(G)$.

In the definition of \mathcal{S}^* , note that $\emptyset^* = \{(\varepsilon, \emptyset)\}$. This value of $\text{NULLABLE}_i^*(G)$ is used in the construction of $\text{NULLABLE}_1(G)$.

The next lemma explains how the set $\text{NULLABLE}(G)$ represents the generation of ε by different nonterminals in different contexts.

Lemma 1. Let $G = (\Sigma, N, R, S)$ be a grammar with contexts, let $A \in N$ and $u \in \Sigma^*$. Then, $\langle u \rangle \varepsilon \in L_G(A)$ if and only if there exist $K_1, \dots, K_t \in N$, such that $(A, \{K_1, \dots, K_t\}) \in \text{NULLABLE}(G)$ and $\langle \varepsilon \rangle u \in L_G(K_1), \dots, \langle \varepsilon \rangle u \in L_G(K_t)$.

It is convenient to define the elimination of ε -conjuncts for a grammar with contexts $G = (\Sigma, N, R, S)$, in which every symbol $A \in N$ either has one or more rules of the form $A \rightarrow B_1 \& \dots \& B_k \& \triangleleft D_1 \& \dots \& \triangleleft D_m \& \trianglelefteq E_1 \& \dots \& \trianglelefteq E_n$

with $B_i, D_i, E_i \in N$, or a unique rule of the form $A \rightarrow BC$ (with $B, C \in N$), $A \rightarrow a$ (with $a \in \Sigma$) or $A \rightarrow \varepsilon$. The pre-processing necessary to reach this intermediate form is straightforward, and then an equivalent grammar $G' = (\Sigma, N, R', S)$ without ε -conjuncts is constructed as follows:

1. All rules of the form $A \rightarrow a$ in R are added to R' .
2. Every rule $A \rightarrow B_1 \& \dots \& B_k \& \triangleleft D_1 \& \dots \& \triangleleft D_m \& \trianglelefteq E_1 \& \dots \& \trianglelefteq E_n$ in R is added to R' . Additionally, if $\langle \varepsilon \rangle \varepsilon \in L_G(D_1) \cap \dots \cap L_G(D_m)$, then another rule $A \rightarrow B_1 \& \dots \& B_k \& E_1 \& \dots \& E_n \& \triangleleft \varepsilon$ is added to R' .
3. Every rule $A \rightarrow BC \in R$ is added to R' , along with the following ones:
 - $A \rightarrow B \& \trianglelefteq K_1 \& \dots \& \trianglelefteq K_t$, for all $(C, \{K_1, \dots, K_t\}) \in \text{NULLABLE}(G)$;
 - $A \rightarrow C \& \triangleleft K_1 \& \dots \& \triangleleft K_t$, for all $(B, \{K_1, \dots, K_t\}) \in \text{NULLABLE}(G)$;
 - $A \rightarrow C \& \triangleleft \varepsilon$, if there exists a nonempty set $\{K_1, \dots, K_t\} \subseteq N$, such that $(B, \{K_1, \dots, K_t\}) \in \text{NULLABLE}(G)$ and $\langle \varepsilon \rangle \varepsilon \in L_G(K_1) \cap \dots \cap L_G(K_t)$.

Lemma 2. *Let $G = (\Sigma, N, R, S)$ be a grammar with contexts. Then the grammar $G' = (\Sigma, N', R', S)$ constructed above generates the language $L(G) \setminus \{\varepsilon\}$.*

The second stage of the transformation to the normal form is removing the *unit conjuncts* in rules of the form $A \rightarrow B \& \dots$. Already for conjunctive grammars, the only known transformation involves substituting all rules for B into all rules for A , and results in a worst-case exponential blowup [11]. The same construction applies for grammars with contexts as it is.

These transformations lead to the following normal form theorem.

Theorem 2. *For each grammar with contexts $G = (\Sigma, N, R, S)$ there exists and can be effectively constructed a grammar with contexts $G' = (\Sigma, N', R', S)$ generating the same language, in which all rules are of the form*

$$\begin{aligned}
A &\rightarrow B_1 C_1 \& \dots \& B_k C_k \& \triangleleft D_1 \& \dots \& \triangleleft D_m \& \trianglelefteq E_1 \& \dots \& \trianglelefteq E_n \\
A &\rightarrow a \& \triangleleft D_1 \& \dots \& \triangleleft D_m \& \trianglelefteq E_1 \& \dots \& \trianglelefteq E_n \\
A &\rightarrow B_1 C_1 \& \dots \& B_k C_k \& \triangleleft \varepsilon \\
A &\rightarrow a \& \triangleleft \varepsilon \qquad \qquad \qquad (k \geq 1, m, n \geq 0, a \in \Sigma, B_i, C_i, D_i, E_i \in N')
\end{aligned}$$

The size of the resulting grammar, measured in the total number of symbols used to describe it, is at most exponential in the size of the given grammar.

5 Parsing algorithm

For each grammar with one-sided contexts in the binary normal form, there exists a parsing algorithm that determines the membership of all substrings of the input string in the languages generated by all nonterminals of the grammar, storing the results in a table. This algorithm elaborates a similar procedure for conjunctive grammars [11], which in turn generalized the Cocke–Kasami–Younger algorithm for standard context-free grammars.

Let $G = (\Sigma, N, R, S)$ be a grammar with contexts in the binary normal form, and let $w = a_1 \dots a_n$ with $n \geq 1$ and $a_k \in \Sigma$ be some string. Let $0 \leq i < j \leq n$. Define

$$T_{i,j} = \{ A \mid A \in N, \vdash_G [A, \langle a_1 \dots a_i \rangle a_{i+1} \dots a_j] \}.$$

This table is constructed by the following algorithm.

- 1: $T_{0,1} = \{ A \in N \mid A \rightarrow a_1 \ \& \ \langle \varepsilon \in R \}$
- 2: **for** $j = 1, \dots, n$ **do**
- 3: **while** $T_{0,j}$ changes **do**
- 4: **for all** $A \rightarrow a \ \& \ \langle D_1 \ \& \ \dots \ \& \ \langle D_{m'} \ \& \ \leq E_1 \ \& \ \dots \ \& \ \leq E_{m''} \ \>$ **do**
- 5: **if** $a_j = a, D_1, \dots, D_{m'} \in T_{0,j-1}$ and $E_1, \dots, E_{m''} \in T_{0,j}$ **then**
- 6: $T_{j-1,j} = T_{j-1,j} \cup \{A\}$
- 7: **for** $i = j - 2$ to 0 **do**
- 8: let $U = \emptyset$ ($U \subseteq N \times N$)
- 9: **for** $k = i + 1$ to $j - 1$ **do**
- 10: $U = U \cup (T_{i,k} \times T_{k,j})$
- 11: **for all** $A \rightarrow B_1 C_1 \ \& \ \dots \ \& \ B_m C_m \ \& \ \langle D_1 \ \& \ \dots \ \& \ \langle D_{m'} \ \& \ \leq E_1 \ \& \ \dots \ \& \ \leq E_{m''} \ \>$ **do**
- 12: **if** $(B_1, C_1), \dots, (B_m, C_m) \in U, D_1, \dots, D_{m'} \in T_{0,i}$ and $E_1, \dots, E_{m''} \in T_{0,j}$ **then**
- 13: $T_{i,j} = T_{i,j} \cup \{A\}$
- 14: **for all** $A \rightarrow B_1 C_1 \ \& \ \dots \ \& \ B_m C_m \ \& \ \langle \varepsilon \in R \ \>$ **do**
- 15: **if** $(B_1, C_1), \dots, (B_m, C_m) \in U$ and $i = 0$ **then**
- 16: $T_{i,j} = T_{i,j} \cup \{A\}$

Once the table is constructed, the input is accepted if and only if $S \in T_{0,n}$.

Theorem 3. *For every grammar with one-sided contexts G in the binary normal form, there exists an algorithm that determines the membership of a given string $w = a_1 \dots a_n$ in $L(G)$, and does so in time $\mathcal{O}(|G|^2 \cdot n^3)$, using space $\mathcal{O}(|G| \cdot n^2)$.*

6 Recognition in linear space

The cubic-time algorithm in Section 5 uses quadratic space, as do its context-free and conjunctive prototypes. For conjunctive grammars, the membership of a string can be recognized in linear space and exponential time [13] by using deterministic rewriting of terms of a linearly bounded size. In this section, the latter method is extended to handle the case of grammars with one-sided contexts.

Let $G = (\Sigma, N, R, S)$ be a grammar with one-sided contexts and let $w = a_1 \dots a_n$ be an input string. The linear-space parsing algorithm presented below constructs the sets $T_{0,1}, T_{0,2}, \dots, T_{0,n}$ (as in the top row of the table in the algorithm in the last section), with

$$T_{0,i} = \{ A \in N \mid \langle \varepsilon \rangle a_1 \dots a_i \in L_G(A) \}.$$

The membership of each $A \in N$ in each set $T_{0,\ell}$ is determined using term rewriting similar to the one defined for Boolean grammars [13], which operates in

exponential time by trying all possible parses. Whenever the grammar refers to a context $\triangleleft D$, the answer is found in one of the previously computed sets $T_{0,i}$ with $i < \ell$; a reference to a context $\trianglelefteq E$ is resolved by looking in the partially computed set $T_{0,\ell}$. However, this E may not yet have been added to $T_{0,\ell}$, and the procedure may give a false negative answer. Hence, up to $|N|$ iterations (similar to those in line 3 of the cubic-time algorithm) have to be done until all dependencies are propagated and all elements of $T_{0,\ell}$ are found. Once the last set $T_{0,n}$ is constructed, the input string is accepted if S belongs to this set.

Theorem 4. *Every language generated by a grammar with one-sided contexts is in $\text{DSPACE}(n)$.*

7 Future work

The new model leaves many theoretical questions to ponder. For instance, is there a parsing algorithm for grammars with one-sided contexts working in less than cubic time? For standard context-free grammars, Valiant [21] discovered an algorithm that offloads the most intensive computations into calls to a Boolean matrix multiplication procedure, and thus can work in time $O(n^\omega)$, with $\omega < 3$; according to the current knowledge on matrix multiplication, ω can be reduced to 2.376. The main idea of Valiant’s algorithm equally applies to Boolean grammars, which can be parsed in time $O(n^\omega)$ as well [17]. However, extending it to grammars with contexts, as defined in this paper, seems to be inherently impossible, because the logical dependencies between the properties of substrings (that is, between the entries of the table $T_{i,j}$) now have a more complicated structure, and the order of calculating these entries apparently rules out grouping multiple operations into Boolean matrix multiplication. However, there might exist a different $o(n^3)$ -time parsing strategy for these grammars, which would be interesting to discover.

Another direction is to develop practical parsing algorithms for grammars with one-sided contexts. An obvious technique to try is the recursive descent parsing [16], where *ad hoc* restrictions resembling contexts of the form $\triangleright D \Sigma^*$ have long been used to guide deterministic computation. The Lang–Tomita Generalized LR parsing [15] is worth being investigated as well.

A more general direction for further research is to consider grammars with two-sided contexts, which would allow rules of the form $A \rightarrow BC \& \triangleleft D \& \trianglelefteq E \& \triangleright F \& \triangleright G$. Such grammars would implement Chomsky’s [2] idea of defining phrase-structure rules applicable in a context in full—which is something that was for the first time properly approached in this paper.

References

1. T. Aizikowitz, M. Kaminski, “LR(0) conjunctive grammars and deterministic synchronized alternating pushdown automata”, *Computer Science in Russia* (CSR 2011, St. Petersburg, Russia, 14–18 June 2011), LNCS 6651, 345–358.

2. N. Chomsky, “On certain formal properties of grammars”, *Information and Control*, 2:2 (1959), 137–167.
3. K. Čulík II, R. Cohen, “LR-regular grammars—an extension of LR(k) grammars”, *Journal of Computer and System Sciences*, 7:1 (1973), 66–96.
4. Z. Ésik, W. Kuich, “Boolean fuzzy sets”, *International Journal of Foundations of Computer Science*, 18:6 (2007), 1197–1207.
5. B. Ford, “Parsing expression grammars: a recognition-based syntactic foundation”, *Proceedings of POPL 2004* (Venice, Italy, January 14–16, 2004), 111–122.
6. S. Ginsburg, H. G. Rice, “Two families of languages related to ALGOL”, *Journal of the ACM*, 9 (1962), 350–371.
7. A. Jež, “Conjunctive grammars can generate non-regular unary languages”, *International Journal of Foundations of Computer Science*, 19:3 (2008), 597–615.
8. A. Jež, A. Okhotin, “Conjunctive grammars over a unary alphabet: undecidability and unbounded growth”, *Theory of Computing Systems*, 46:1 (2010), 27–58.
9. S. Jarzabek, T. Krawczyk, “LL-regular grammars”, *Information Processing Letters*, 4 (1975), 31–37.
10. V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, “Well-founded semantics for Boolean grammars”, *Information and Computation*, 207:9 (2009), 945–967.
11. A. Okhotin, “Conjunctive grammars”, *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.
12. A. Okhotin, “Conjunctive grammars and systems of language equations”, *Programming and Computer Science*, 28:5 (2002), 243–249.
13. A. Okhotin, “Boolean grammars”, *Information and Computation*, 194:1 (2004), 19–48.
14. A. Okhotin, “The dual of concatenation”, *Theoretical Computer Science*, 345:2–3 (2005), 425–447.
15. A. Okhotin, “Generalized LR parsing algorithm for Boolean grammars”, *International Journal of Foundations of Computer Science*, 17:3 (2006), 629–664.
16. A. Okhotin, “Recursive descent parsing for Boolean grammars”, *Acta Informatica*, 44:3–4 (2007), 167–189.
17. A. Okhotin, “Fast parsing for Boolean grammars: a generalization of Valiant’s algorithm”, *Developments in Language Theory (DLT 2010, London, Ontario, Canada, August 17–20, 2010)*, LNCS 6224, 340–351.
18. A. Okhotin, C. Reitwießner, “Conjunctive grammars with restricted disjunction”, *Theoretical Computer Science*, 411:26–28 (2010), 2559–2571.
19. T. Parr, K. Fisher, “LL(*): the foundation of the ANTLR parser generator”, *Programming Language Design and Implementation, PLDI 2011 (San Jose, USA, 4–8 June 2011)*, 425–436.
20. K. Sikkell, *Parsing Schemata*, Springer-Verlag, 1997.
21. L. G. Valiant, “General context-free recognition in less than cubic time”, *Journal of Computer and System Sciences*, 10:2 (1975), 308–314.