Grammars with two-sided contexts[*]

Mikhail Barash
Department of Mathematics and Statistics, University of Turku, Turku FI-20014, Finland
Turku Centre for Computer Science, Turku FI-20520, Finland
mikbar@utu.fi

Alexander Okhotin
Department of Mathematics and Statistics, University of Turku, Turku FI-20014, Finland
alexander.okhotin@utu.fi

In a recent paper (M. Barash, A. Okhotin, "Defining contexts in context-free grammars", LATA 2012), the authors introduced an extension of the context-free grammars equipped with an operator for referring to the left context of the substring being defined. This paper proposes a more general model, in which context specifications may be two-sided, that is, both the left and the right contexts can be specified by the corresponding operators. The paper gives the definitions and establishes the basic theory of such grammars, leading to a normal form and a parsing algorithm working in time $\mathscr{O}(n^4)$, where n is the length of the input string.

1 Introduction

The context-free grammars are a logic for representing the syntax of languages, in which the properties of longer strings are defined by concatenating shorter strings with known properties. Disjunction of syntactic conditions is represented in this logic as multiple alternative rules defining a single symbol. One can further augment this logic with conjunction and negation operations, leading to *conjunctive grammars* [13] and *Boolean grammars* [15]. These grammars are context-free in the general sense of the word, as they define the properties of each substring independently of the context, in which it occurs. Furthermore, most of the practically important features of ordinary context-free grammars, such as efficient parsing algorithms, are preserved in their conjunctive and Boolean variants [15, 18]. These grammar models have been a subject of recent theoretical studies [1, 8, 10, 12, 24].

Not long ago, the authors [3, 4] proposed an extension of the context-free grammars with special operators for expressing the form of the *left context*, in which the substring occurs. For example, a rule $A \to BC \,\&\, \lhd D$ asserts that every string representable as BC in a left context of the form described by D therefore has the property A. These grammars were motivated by Chomsky's [6, p. 142] well-known idea of a phrase-structure rule applicable only in some particular contexts. Chomsky's own attempt to implement this idea by string rewriting resulted in a model equivalent to linear-space Turing machines, in which the "nonterminal symbols", meant to represent syntactic categories, could be freely manipulated as tape symbols. In spite of the name "context-sensitive grammars", the resulting model was unsuitable for describing the syntax of languages, and thus failed to represent the idea of a rule applicable in a context.

Taking a new start with this idea, the authors [4] defined *grammars with one-sided contexts*, following the logical outlook on grammars, featured in the work of Kowalski [11, Ch. 3] and of Pereira and

[*]Supported by the Academy of Finland under grant 257857.

© M. Barash, A. Okhotin
This work is licensed under the
Creative Commons Attribution License.

Warren [19], and later systematically developed by Rounds [21]. A grammar defines the truth value of statements of the form “a certain string has a certain property”, and these statements are deduced from each other according to the rules of the grammar. The resulting definition maintains the underlying logic of the context-free grammars, and many crucial properties of grammars are preserved: grammars with one-sided contexts have parse trees, can be transformed to a normal form and have a cubic-time parsing algorithm [4]. However, the model allowed specifying contexts only on one side, and thus it implemented, so to say, only one half of Chomsky’s idea.

This paper continues the development of formal grammars with context specifications by allowing contexts in both directions. The proposed *grammars with two-sided contexts* may contain such rules as $A \to BC \,\&\, \lhd D \,\&\, \rhd E$, which define any substring of the form BC preceded by a substring of the form D and followed by a substring of the form E. If the grammar contains additional rules $B \to b, C \to c, D \to d$ and $E \to e$, then the above rule for A asserts that a substring bc of a string $w = dbce$ has the property A. However, this rule will not produce the same substring bc occurring in another string $w' = dbcd$, because its right context does not satisfy the conjunct $\rhd E$. Furthermore, the grammars allow expressing the so-called *extended right context* ($\trianglerighteq \alpha$), which defines the form of the current substring concatenated with its right context, as well as the symmetrically defined *extended left context* ($\trianglelefteq \alpha$).

In Section 2, this intuitive definition is formalized by deduction of propositions of the form $A\big(u\langle w\rangle v\big)$, which states that the substring w occurring in the context between u and v has the property A, where A is a syntactic category defined by the grammar (“nonterminal symbol” in Chomsky’s terminology). Then, each rule of the grammar becomes a schema for deduction rules, and a string w is generated by the grammar, if there is a proof of the proposition $S\big(\varepsilon\langle w\rangle\varepsilon\big)$. A standard proof tree of such a deduction constitutes a parse tree of the string w.

The next Section 3 presents basic examples of grammars with two-sided contexts. These examples model several types of cross-references, such as declaration of identifiers before or after their use.

The paper then proceeds with developing a normal form for these grammars, which generalizes the Chomsky normal form for ordinary context-free grammars. In the normal form, every rule is a conjunction of one or more *base conjuncts* describing the form of the current substring (either as a concatenation of the form BC or as a single symbol a), with any context specifications ($\lhd D$, $\trianglelefteq E$, $\trianglerighteq F$, $\rhd H$). The transformation to the normal form, presented in Section 4, proceeds in three steps. First, all rules generating the empty string in any contexts are eliminated. Second, all rules with an explicit empty context specification ($\lhd\varepsilon$, $\rhd\varepsilon$) are also eliminated. The final step is elimination of any rules of the form $A \to B \,\&\, \ldots$, where the dependency of A on B potentially causes cycles in the definition.

Once the normal form is established, a simple parsing algorithm for grammars with two-sided contexts with the running time $\mathcal{O}(n^4)$ is presented in Section 5. While this paper has been under preparation, Rabkin [20] has developed a more efficient and more sophisticated parsing algorithm for grammars with two-sided contexts, with the running time $\mathcal{O}(n^3)$.

2 Definition

Ordinary context-free grammars allow using the concatenation operation to express the form of a string, and disjunction to define alternative forms. In conjunctive grammars, the conjunction operation may be used to assert that a substring being defined must conform to several conditions at the same time. The grammars studied in this paper further allow operators for expressing the form of the left context ($\lhd$, $\trianglelefteq$) and the right context ($\rhd$, $\trianglerighteq$) of a substring being defined.

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

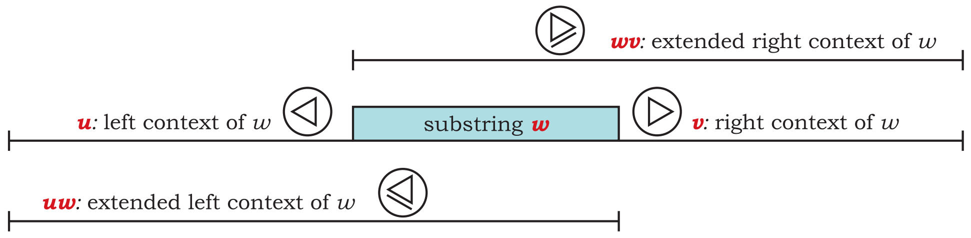

wv: extended right context of w
u: left context of w
substring w
v: right context of w
uw: extended left context of w

Figure 1: A substring w of a string uwv: four types of contexts.

- *Σ is the alphabet of the language being defined;*
- *N is a finite set of auxiliary symbols ("nonterminal symbols" in Chomsky's terminology), which denote the properties of strings defined in the grammar;*
- *R is a finite set of grammar rules, each of the form*

$$\begin{aligned} A \to \alpha_1 \,\&\, \dots \,\&\, \alpha_k \,\&\, \lhd\beta_1 \,\&\, \dots \,\&\, \lhd\beta_m \,\&\, \trianglelefteq\gamma_1 \,\&\, \dots \,\&\, \trianglelefteq\gamma_n \,\&\, \\ \&\, \trianglerighteq\kappa_1 \,\&\, \dots \,\&\, \trianglerighteq\kappa_{m'} \,\&\, \rhd\delta_1 \,\&\, \dots \,\&\, \rhd\delta_{n'}, \end{aligned} \qquad (1)$$

  with $A \in N$, $k \geqslant 1$, $m,n,m',n' \geqslant 0$ and $\alpha_i, \beta_i, \gamma_i, \kappa_i, \delta_i \in (\Sigma \cup N)^*$;
- *$S \in N$ is a symbol representing well-formed sentences of the language.*

If all rules in a grammar have only left contexts (that is, if $m' = n' = 0$), then this is a grammar with one-sided contexts [4]. If no context operators are ever used ($m = n = m' = n' = 0$), this is a conjunctive grammar, and if the conjunction is also never used ($k = 1$), this is an ordinary context-free grammar.

For each rule (1), each term α_i, $\lhd\beta_i$, $\trianglelefteq\gamma_i$, $\trianglerighteq\kappa_i$ and $\rhd\delta_i$ is called a *conjunct*. Denote by $u\langle w\rangle v$ a substring $w \in \Sigma^*$, which is preceded by $u \in \Sigma^*$ and followed by $v \in \Sigma^*$, as illustrated in Figure 1. Intuitively, such a substring is generated by a rule (1), if

- each *base conjunct* $\alpha_i = X_1 \dots X_\ell$ gives a representation of w as a concatenation of shorter substrings described by $X_1, \dots, X_\ell$, as in context-free grammars;
- each conjunct $\lhd\beta_i$ similarly describes the form of the *left context u*;
- each conjunct $\trianglelefteq\gamma_i$ describes the form of the *extended left context uw*;
- each conjunct $\trianglerighteq\kappa_i$ describes the *extended right context wv*;
- each conjunct $\rhd\delta_i$ describes the *right context v*.

The semantics of grammars with two-sided contexts are defined by a deduction system of elementary propositions (items) of the form "a string $w \in \Sigma^*$ written in a left context $u \in \Sigma^*$ and in a right context $v \in \Sigma^*$ has the property $X \in \Sigma \cup N$", denoted by $X\big(u\langle w\rangle v\big)$. The deduction begins with axioms: any symbol $a \in \Sigma$ written in any context has the property a, denoted by $a\big(u\langle a\rangle v\big)$ for all $u, v \in \Sigma^*$. Each rule in R is then regarded as a schema for deduction rules. For example, a rule $A \to BC$ allows making deductions of the form

$$B\big(u\langle w\rangle w'v\big), C\big(uw\langle w'\rangle v\big) \vdash_G A\big(u\langle ww'\rangle v\big) \qquad \text{(for all } u, w, w', v \in \Sigma^*\text{)},$$

which is essentially a concatenation of w and w' that respects the contexts. If the rule is of the form $A \to BC \,\&\, \lhd D$, this deduction requires an extra premise:

$$B\big(u\langle w\rangle w'v\big), C\big(uw\langle w'\rangle v\big), D\big(\varepsilon\langle u\rangle ww'v\big) \vdash_G A\big(u\langle ww'\rangle v\big).$$

And if the rule is $A \to BC \,\&\, \trianglerighteq F$, the deduction proceeds as follows:

$$B\big(u\langle w\rangle w'v\big), C\big(uw\langle w'\rangle v\big), F\big(u\langle ww'v\rangle\varepsilon\big) \vdash_G A\big(u\langle ww'\rangle v\big).$$

The general form of deduction schemata induced by a rule in R is defined below.

Definition 2. *Let $G = (\Sigma, N, R, S)$ be a grammar with two-sided contexts. Define the following deduction system of items of the form $X\big(u\langle w\rangle v\big)$, with $X \in \Sigma \cup N$ and $u, w, v \in \Sigma^*$. There is a single axiom scheme $\vdash_G a\big(u\langle a\rangle v\big)$, for all $a \in \Sigma$ and $u, v \in \Sigma^*$. Each rule (1) in R defines the following scheme for deduction rules:*

$$I \vdash_G A\big(u\langle w\rangle v\big),$$

for all $u, w, v \in \Sigma^*$ and for every set of items I satisfying the below properties:

- *For every base conjunct $\alpha_i = X_1 \dots X_\ell$, with $\ell \geqslant 0$ and $X_j \in \Sigma \cup N$, there should exist a partition $w = w_1 \dots w_\ell$ with $X_j\big(uw_1 \dots w_{j-1}\langle w_j\rangle w_{j+1} \dots w_\ell v\big) \in I$ for all $j \in \{1, \dots, \ell\}$.*
- *For every conjunct $\lhd\beta_i = \lhd X_1 \dots X_\ell$ there should be such a partition $u = u_1 \dots u_\ell$, that $X_j\big(u_1 \dots u_{j-1}\langle u_j\rangle u_{j+1} \dots u_\ell wv\big) \in I$ for all $j \in \{1, \dots, \ell\}$.*
- *Every conjunct $\trianglelefteq\gamma_i = \trianglelefteq X_1 \dots X_\ell$ should have a corresponding partition $uw = x_1 \dots x_\ell$ with $X_j\big(x_1 \dots x_{j-1}\langle x_j\rangle x_{j+1} \dots x_\ell v\big) \in I$ for all $j \in \{1, \dots, \ell\}$.*
- *For every conjunct $\rhd\delta_i$ and $\trianglerighteq\kappa_i$, the condition is defined symmetrically.*

Then the language generated by a symbol $A \in N$ is defined as

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

The language generated by the grammar G is the set of all strings with empty left and right contexts generated by S: $L(G) = \{\, w \mid w \in \Sigma^*, \vdash_G S\big(\varepsilon\langle w\rangle\varepsilon\big) \,\}$.

The following trivial example of a grammar is given to illustrate the definitions.

Example 1. Consider the grammar with two-sided contexts that defines the singleton language $\{abca\}$:

$$\begin{aligned} S &\to aS \mid Sa \mid BC \\ A &\to a \\ B &\to b \,\&\, \lhd A \\ C &\to c \,\&\, \rhd A \end{aligned}$$

The deduction given below proves that the string $abca$ has the property S.

$$\begin{array}{rll}
 & \vdash a\big(\varepsilon\langle a\rangle bca\big) & (axiom) \\
 & \vdash b\big(a\langle b\rangle ca\big) & (axiom) \\
 & \vdash c\big(ab\langle c\rangle a\big) & (axiom) \\
 & \vdash a\big(abc\langle a\rangle\varepsilon\big) & (axiom) \\
a\big(\varepsilon\langle a\rangle bca\big) & \vdash A\big(\varepsilon\langle a\rangle bca\big) & (A \to a) \\
b\big(a\langle b\rangle ca\big), A\big(\varepsilon\langle a\rangle bca\big) & \vdash B\big(a\langle b\rangle ca\big) & (B \to b \,\&\, \lhd A) \\
a\big(abc\langle a\rangle\varepsilon\big) & \vdash A\big(abc\langle a\rangle\varepsilon\big) & (A \to a) \\
c\big(ab\langle c\rangle a\big), A\big(abc\langle a\rangle\varepsilon\big) & \vdash C\big(ab\langle c\rangle a\big) & (C \to c \,\&\, \rhd A) \\
B\big(a\langle b\rangle ca\big), C\big(ab\langle c\rangle a\big) & \vdash S\big(a\langle bc\rangle a\big) & (S \to BC) \\
a\big(\varepsilon\langle a\rangle bca\big), S\big(a\langle bc\rangle a\big) & \vdash S\big(\varepsilon\langle abc\rangle a\big) & (S \to aS) \\
S\big(\varepsilon\langle abc\rangle a\big), a\big(abc\langle a\rangle\varepsilon\big) & \vdash S\big(\varepsilon\langle abca\rangle\varepsilon\big) & (S \to Sa)
\end{array}$$

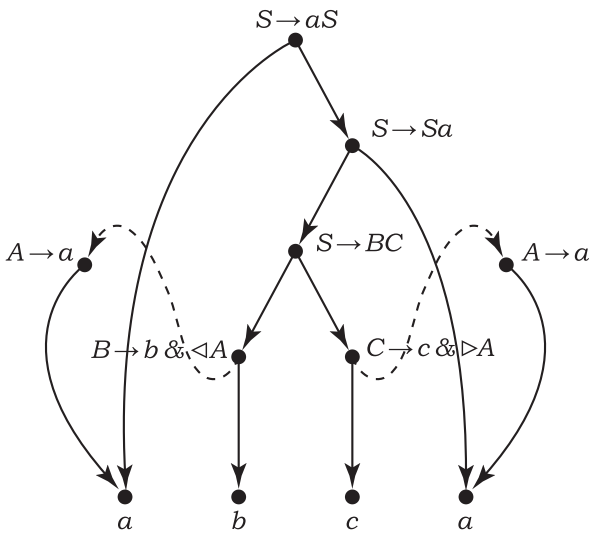


Figure 2: A parse tree of the string *abca* according to the grammar in Example 1.

Another possible definition of grammars with contexts is by directly expressing them in first-order logic over positions in a string [21]. Nonterminal symbols become *binary predicates*, with the arguments referring to positions in the string. Each predicate $A(x,y)$ is defined by a formula $\varphi_A(x,y)$ that states the condition of a substring delimilited by positions x and y having the property A. There are built-in unary predicates $a(x)$, for each $a \in \Sigma$, which assert that the symbol in position x in the string is a, and binary predicates $x < y$ and $x = y$ for comparing positions. Arguments to predicates are given as *terms*, which are either variables ($t = x$) or constants referring to the first and the last positions ($t = \underline{\text{begin}}$, $t = \underline{\text{end}}$), and which may be incremented ($t+1$) or decremented ($t-1$). Each formula is constructed from predicates using conjunction, disjunction and first-order existential quantification.

Example 2. The grammar from Example 1 is expressed by the following formulae defining predicates $S(x,y)$, $B(x,y)$, $A(x,y)$ and $C(x,y)$.

$$\begin{aligned}
S(x,y) &= (a(x) \wedge S(x+1,y)) \vee (S(x,y-1) \wedge a(y)) \vee (\exists z\,(x < z < y \wedge B(x,z) \wedge C(z,y))) \\
A(x,y) &= a(x) \wedge x+1 = y \\
B(x,y) &= b(x) \wedge x+1 = y \wedge A(\underline{\text{begin}},x) \\
C(x,y) &= c(x) \wedge x+1 = y \wedge A(y,\underline{\text{end}})
\end{aligned}$$

The membership of a string w is expressed by the statement $S(\underline{\text{begin}},\underline{\text{end}})$, which may be true of false.

3 Examples

This section presents several examples of grammars with two-sided contexts generating important syntactic constructs. All examples use ordinary context-free elements, such as a grammar for $\{\, a^n b^n \mid n \geqslant 0 \,\}$, and combine these elements using the new context operators. This leads to natural specifications of languages in the style of classical formal grammars.

Consider the problem of checking declaration of identifiers before their use: this construct can be found in all kinds of languages, and it can be expressed by a conjunctive grammar [16, Ex. 3]. The variant of this problem, in which the identifiers may be declared *before or after* their use, is also fairly common: consider, for instance, the declaration of classes in C++, where an earlier defined method can refer to a class member defined later. However, no conjunctive grammar expressing this construct is known.

A grammar with one-sided contexts for declarations before or after use has recently been constructed by the authors [4]. That grammar used context specifications, along with iterated conjunction, to express what would be more naturally expressed in terms of two-sided contexts. In the model proposed in this paper, the same language can be defined in a much more natural way.

Example 3 (cf. grammar with one-sided contexts [4, Ex. 4])**.** Consider the language

$$\{\, u_1 \dots u_n \mid \text{for every } u_i, \textbf{ either } u_i \in a^*c, \textbf{ or } u_i = b^k c \text{ and there exists } j \in \{1,\dots,n\} \text{ with } u_j = a^k c \,\}.$$

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 $a^k c$.

This language is generated by the following grammar.

$$\begin{array}{rcl@{\qquad\qquad}rcl}
S & \to & AS \mid CS \mid DS \mid \varepsilon & C & \to & B \,\&\, \trianglelefteq EFc \\
A & \to & aA \mid c & D & \to & B \,\&\, \trianglerighteq HcE \\
B & \to & bB \mid c & F & \to & aFb \mid cE \\
E & \to & AE \mid BE \mid \varepsilon & H & \to & bHa \mid cE
\end{array}$$

The idea of the grammar is that S should generate a substring $u_1 \dots u_\ell \langle u_{\ell+1} \dots u_n \rangle \varepsilon$, with $0 \leqslant \ell \leqslant n$ and $u_i \in a^*c \cup b^*c$, if and only if every reference in $u_{\ell+1} \dots u_n$ has a corresponding declaration somewhere in the whole string $u_1 \dots u_n$. The rules for S define all substrings satisfying this condition inductively on their length, until the entire string $\varepsilon\langle u_1 \dots u_n \rangle \varepsilon$ is defined. The rule $S \to \varepsilon$ defines the base case: the string $u_1 \dots u_n \langle \varepsilon \rangle \varepsilon$ has the desired property. The rule $S \to CS$ appends a reference of the form b^*c, restricted by an extended left context $\trianglelefteq EFc$, which ensures that this reference has a matching *earlier* declaration; here E represents the prefix of the string up to that earlier declaration, while F matches the symbols a in the declaration to the symbols b in the reference. The possibility of a *later* declaration is checked by another rule $S \to DS$, which adds a reference of the form b^*c with an extended right context $\trianglerighteq HcE$, where H is used to match the bs forming this reference to the as in the later declaration.

The next example abstracts another syntactic mechanism—*function prototypes*—found in the C programming language and, under the name of *forward declarations*, in the programming language Pascal.

Example 4. Consider the language

$$\Big\{ u_1 \dots u_n \;\Big|\; \text{for every } u_i, \textbf{ either } u_i = a^k c \text{ and there exists } j > i, \text{ such that } u_j = d^k c, \tag{2a}$$
$$\textbf{or } u_i = b^k c \text{ and there exists } j < i, \text{ for which } u_j = a^k c \Big\}. \tag{2b}$$

A substring of the form $a^k c$ represents a function prototype and a substring $d^k c$ represents its body. Calls to functions are expressed as substrings $b^k c$. Condition (2a) means that every prototype must be followed by its body, and restriction (2b) requires that references are only allowed to declared prototypes.

This language can be generated by the following grammar with two-sided contexts.

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl}
S & \to & US \mid VS \mid DS \mid \varepsilon & D & \to & dD \mid c & E & \to & AE \mid BE \mid DE \mid \varepsilon \\
A & \to & aA \mid c & U & \to & A \,\&\, \trianglerighteq HcE & H & \to & aHd \mid cE \\
B & \to & bB \mid c & V & \to & B \,\&\, \trianglelefteq EFc & F & \to & aFb \mid cE
\end{array}$$

The rules $S \to US$ and $U \to A \,\&\, \trianglerighteq HcE$ append a prototype $a^k c$ and the extended right context of the form $a^k c \dots d^k c \dots$ ensures that this prototype has a matching body somewhere *later* within the string. The rules $S \to VS$ and $V \to B \,\&\, \trianglelefteq EFc$ append a reference $b^k c$, and the context specification $\dots a^k c \dots b^k c$ checks that it has a matching prototype *ealier* in the string. Function bodies $d^k c$ are added by the rule $S \to DS$. Using these rules, S generates substrings of the form $u_1 \dots u_\ell \langle u_{\ell+1} \dots u_n \rangle \varepsilon$, with $0 \leqslant \ell \leqslant n$ and $u_i \in a^* c \cup b^* c \cup d^* c$, such that every prototype $u_i = a^k c$ in $u_{\ell+1} \dots u_n$ has a corresponding body $d^k c$ in $u_{i+1} \dots u_n$ and every reference $u_i = b^k c$ in $u_{\ell+1} \dots u_n$ has a corresponding prototype $a^k c$ in $u_1 \dots u_{i-1}$.

The next example gives a grammar with contexts that defines reachability on graphs. Sudborough [22] defined a linear context-free grammar for a special encoding of the graph reachability problem on acyclic graphs, in which every arc goes from a lower-numbered vertex to a higher-numbered vertex. The grammar presented below allows any graphs and uses a direct encoding. This example illustrates the ability of grammars with contexts to define various kinds of cross-references.

Example 5. Consider encodings of directed graphs as strings of the form $b^s a^{i_1} b^{j_1} a^{i_2} b^{j_2} \dots a^{i_n} b^{j_n} a^t$, with $s, t \geqslant 1$, $n \geqslant 0$, $i_k, j_k \geqslant 1$, where each block $a^i b^j$ denotes an arc from vertex number i to vertex number j, while the prefix b^s and the suffix a^t mark s as the source vertex and t as the target. Then the following grammar defines all graphs with a path from s to t.

$$\begin{array}{rcl@{\qquad}rcl}
S & \to & FDCA \mid F & & & \\
A & \to & aA \mid c & D & \to & B \,\&\, \trianglelefteq BCE \mid B \,\&\, \trianglerighteq FDCA \mid B \,\&\, \trianglerighteq F \\
B & \to & bB \mid c & E & \to & aEb \mid DCA \\
C & \to & ABC \mid \varepsilon & F & \to & bFa \mid bCa
\end{array}$$

The grammar is centered around the nonterminal D, which generates all such substrings $b^s a^{i_1} b^{j_1} \dots a^{i_k} \langle b^{j_k} \rangle a^{i_{k+1}} b^{j_{k+1}} \dots a^{i_n} b^{j_n} a^t$ that there is a path from j_k to t in the graph. If this path is empty, then $j_k = t$. Otherwise, the first arc in the path can be listed either to the left or to the right of b^k. These three cases are handled by the three rules for D. Each of these rules generates b^{j_k} by the base conjunct B, and then uses an extended left or right context operator to match b^{j_k} to the tail of the next arc or to a^t.

The rule $D \to B \,\&\, \trianglelefteq BCE$ considers the case when the next arc in the path is located to the left of b^{j_k}. Let this arc be $a^{i_\ell} b^{j_\ell}$, for some $\ell < k$. Then the extended left context BCE covers the substring $b^s a^{i_1} b^{j_1} \dots a^{i_\ell} b^{j_\ell} \dots a^{i_k} b^{j_k}$. The concatenation BC skips the prefix $b^s a^{i_1} b^{j_1} \dots a^{i_{\ell-1}} b^{j_{\ell-1}}$, and then the nonterminal E matches a^{i_ℓ} to b^{j_k}, verifying that $i_\ell = j_k$. After this, the rule $E \to DCA$ ensures that the substring b^{j_ℓ} is generated by D, that is, that there is a path from j_ℓ to t. The concatenation CA skips the inner substring $a^{i_{\ell+1}} b^{j_{\ell+1}} \dots a^{i_k}$.

The second rule $D \to B \,\&\, \trianglerighteq FDCA$ searches for the next arc to the right of b^{j_k}. Let this be an ℓ-th arc in the list, with $\ell > k$. The extended right context $FDCA$ should generate the suffix $b^{j_k} \dots a^{i_\ell} b^{j_\ell} \dots a^{i_n} b^{j_n} a^t$. The symbol F covers the substring $b^{j_k} \dots a^{i_\ell}$, matching b^{j_k} to a^{i_ℓ}. Then, D generates the substring b^{j_ℓ}, checking that there is a path from j_ℓ to t. The concatenation CA skips the rest of the suffix.

Finally, if the path is of length zero, that is, $j_k = t$, then the rule $D \to B \,\&\, \trianglerighteq F$ uses F to match b^{j_k} to the suffix a^t in the end of the string.

Once the symbol D checks the path from any vertex to the vertex t, for the initial symbol S, it is sufficient to match b^s in the beginning of the string to any arc $a^{j_k} b^{j_k}$, with $j_k = s$. This is done by the rule $S \to FDCA$, which operates in the same way as the second rule for D. The case of s and t being the same node is handled by the rule $S \to F$.

All the above examples use identifiers given in unary, which are matched by rules of the same kind as the rules defining the language $\{ a^n b^n \mid n \geqslant 0 \}$. These examples can be extended to use identifiers over an arbitrary alphabet Σ, owing to the fact that there is a conjunctive grammar generating the language $\{ w\#w \mid w \in \Sigma^* \}$, for some separator $\# \notin \Sigma$ [13, 16].

4 Normal form

An ordinary context-free grammar can be transformed to the Chomsky normal form, with the rules restricted to $A \to BC$ and $A \to a$, with $B, C \in N$ and $a \in \Sigma$. This form has the following generalization to grammars with contexts.

Definition 3. *A grammar with two-sided contexts $G = (\Sigma, N, R, S)$ is said to be in the binary normal form, if each rule in R is of one of the forms*

$$A \to B_1C_1 \,\&\, \dots \,\&\, B_kC_k \,\&\, \lhd D_1 \,\&\, \dots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \dots \,\&\, \trianglelefteq E_n \,\&\, \trianglerighteq F_1 \,\&\, \dots \,\&\, \trianglerighteq F_{n'} \,\&\, \rhd H_1 \,\&\, \dots \,\&\, \rhd H_{m'},$$
$$A \to a \,\&\, \lhd D_1 \,\&\, \dots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \dots \,\&\, \trianglelefteq E_n \,\&\, \trianglerighteq F_1 \,\&\, \dots \,\&\, \trianglerighteq F_{n'} \,\&\, \rhd H_1 \,\&\, \dots \,\&\, \rhd H_{m'},$$

where $k \geqslant 1$, $m, n, n', m' \geqslant 0$, $B_i, C_i, D_i, E_i, F_i, H_i \in N$, $a \in \Sigma$.

The transformation to the normal form consists of three stages: first, removing all *empty conjuncts* ε; secondly, eliminating *empty contexts* ($\lhd\varepsilon$, $\rhd\varepsilon$); finally, getting rid of *unit conjuncts* of the form B, with $B \in N$.

The first step is the removal of all rules of the form $A \to \varepsilon \,\&\, \dots$, so that no symbols generate ε, while all non-empty strings are generated as before. As generation of longer strings may depend on the generation of ε, already for ordinary context-free grammars, such a transformation requires adding extra rules that simulate the same dependence without actually generating any empty strings.

Example 6. Consider the following context-free grammar, which defines the language $\{abc, ab, ac, a, bcd, bd, cd, d\}$.

$$\begin{array}{rcl} S & \to & aA \mid Ad \\ A & \to & BC \\ B & \to & \varepsilon \mid b \\ C & \to & \varepsilon \mid c \end{array}$$

Since B generates the empty string, the rule $A \to BC$ can be used to generate just C; therefore, once the rule $B \to \varepsilon$ is removed, one should add a new rule $A \to C$, in which B is omitted. Similarly one can remove the rule $C \to \varepsilon$ and add a "compensatory" rule $A \to B$. Since both B and C generate ε, so does A by the rule $A \to BC$. Hence, extra rules $S \to a$ and $S \to d$, where A is omitted, have to be added.

An algorithm for carrying out such a transformation first calculates the set of nonterminals that generate the empty string, known as $\textsc{Nullable}(G) \subseteq N$, and then uses it to reconstruct the rules of the grammar.

This set is calculated as a least upper bound of an ascending sequence of sets $\textsc{Nullable}_i(G)$. The set $\textsc{Nullable}_1(G) = \{A \in N \mid A \to \varepsilon \in R\}$ contains all nonterminals which directly define the empty string. Every next set $\textsc{Nullable}_{i+1}(G) = \{A \in N \mid A \to \alpha \in R,\ \alpha \in \textsc{Nullable}_i^*(G)\}$ contains nonterminals that generate ε by the rules referring to other nullable nonterminals. This knowledge is given by the Kleene star of $\textsc{Nullable}_i(G)$.

For the grammar in Example 6, the calculation of the set $\textsc{Nullable}(G)$ proceeds as follows:

$$\begin{array}{rcl} \textsc{Nullable}_0(G) & = & \varnothing, \\ \textsc{Nullable}_1(G) & = & \{B, C\}, \\ \textsc{Nullable}_2(G) & = & \{B, C, A\}, \end{array}$$

and $\textsc{Nullable}(G) = \textsc{Nullable}_2(G)$.

The same idea works for conjunctive grammars as well [13]. For grammars with contexts [4], the generation of the empty string additionally depends on the left contexts, in which the string occurs. This requires an elaborated version of the set $\textsc{Nullable}(G)$, formed of nonterminals along with the information about the left contexts in which they may define ε.

In order to eliminate null conjuncts in case of grammars with two-sided contexts, one has to consider yet another variant of the set $\textsc{Nullable}(G)$, which respects both left and right contexts.

Example 7. Consider the following grammar with two-sided contexts, obtained by adding context restrictions to the grammar in Example 6; this grammar defines the language $L = \{abc, ac, bcd, bd\}$.

$$\begin{array}{rcl} S & \to & aA \mid Ad \\ A & \to & BC \\ B & \to & \varepsilon \,\&\, \lhd D \mid b \\ C & \to & \varepsilon \,\&\, \rhd E \mid c \\ D & \to & a \\ E & \to & d \end{array}$$

In this grammar, the nonterminal B generates the empty string only in a left context of the form defined by D, while C defines the empty string only in a right context of the form E. In those contexts where *both* B and C generate ε, so can A, by the rule $A \to BC$.

The information about the left and right contexts, in which a nonterminal generates the empty string, is to be stored in the set $\textsc{Nullable}(G)$, which is defined as a subset of $2^N \times N \times 2^N$. An element (U, A, V) of this set represents an intuitive idea that A defines ε in a left context of the form described by each nonterminal in U, and in a right context of the form given by nonterminals in V.

For the grammar in Example 7, such a set $\textsc{Nullable}(G)$ is constructed as follows.

$$\begin{array}{rcl} \textsc{Nullable}_0(G) & = & \varnothing \\ \textsc{Nullable}_1(G) & = & \big\{(\{D\}, B, \varnothing), (\varnothing, C, \{E\})\big\} \\ \textsc{Nullable}_2(G) & = & \big\{(\{D\}, B, \varnothing), (\varnothing, C, \{E\}), (\{D\}, A, \{E\})\big\} \end{array}$$

Then $\textsc{Nullable}(G) = \textsc{Nullable}_2(G)$. The elements $(\{D\}, B, \varnothing)$ and $(\varnothing, C, \{E\})$ are obtained directly from the rules of the grammar, and the element $(\{D\}, A, \{E\})$ represents the "concatenation" BC in the rule for A. Note the similarity of this construction to the one for the ordinary grammar in Example 6: the construction given here is different only in recording information about the contexts.

The above "concatenation" of triples $(\{D\}, B, \varnothing)$ and $(\varnothing, C, \{E\})$ should be defined to accumulate both left and right contexts. This can be regarded as a generalization of the Kleene star to sets of triples, denoted by $\textsc{Nullable}^\star(G)$. Formally, $\textsc{Nullable}^\star(G)$ is the set of all triples $(U_1 \cup \ldots \cup U_\ell,\, A_1 \ldots A_\ell,\, V_1 \cup \ldots \cup V_\ell)$ with $\ell \geqslant 0$ and $(U_i, A_i, V_i) \in \textsc{Nullable}(G)$. The symbols A_i are concatenated, while their left and right contexts are accumulated. In the special case when $\ell = 0$, the concatenation of zero symbols is the empty string, and thus $\varnothing^\star = \big\{(\varnothing, \varepsilon, \varnothing)\big\}$.

Before giving a formal definition of the set $\textsc{Nullable}(G)$, assume, for the sake of simplicity, that context operators are only applied to single nonterminal symbols, that is, every rule is of the form

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n \,\&\, \trianglerighteq F_1 \,\&\, \ldots \,\&\, \trianglerighteq F_{m'} \,\&\, \rhd H_1 \,\&\, \ldots \,\&\, \rhd H_{n'}, \quad (3)$$

with $A \in N$, $k \geqslant 1$, $m, n, m', n' \geqslant 0$, $\alpha_i \in (\Sigma \cup N)^*$ and $D_i, E_i, F_i, H_i \in N$. As will be shown in Lemma 3, there is no loss of generality in this assumption.

Definition 4. *Let $G = (\Sigma, N, R, S)$ be a grammar with two-sided contexts with all rules of the form (3). Construct the sequence of sets $\textsc{Nullable}_i(G) \subseteq 2^N \times N \times 2^N$, for $i \geqslant 0$, as follows.*

Let $\textsc{Nullable}_0(G) = \varnothing$. Every next set $\textsc{Nullable}_{i+1}(G)$ contains the following triples: for every rule (3) and for every k triples (U_1, α_1, V_1), …, (U_k, α_k, V_k) in $\textsc{Nullable}_i^\star(G)$, the triple $\big(\{D_1, \dots, D_m, E_1, \dots, E_n\} \cup \{U_1, \dots, U_k\},\ A,\ \{F_1, \dots, F_{m'}, H_1, \dots, H_{n'}\} \cup \{V_1, \dots, V_k\}\big)$ is in $\textsc{Nullable}_{i+1}(G)$.

Finally, let $\textsc{Nullable}(G) = \bigcup_{i \geqslant 0} \textsc{Nullable}_i(G)$.

The next lemma explains how exactly the set $\textsc{Nullable}(G)$ represents the generation of the empty string by different nonterminals in different contexts.

Lemma 1. *Let $G = (\Sigma, N, R, S)$ be a grammar with contexts, let $A \in N$ and $u, v \in \Sigma^*$. Then, $u\langle\varepsilon\rangle v \in L_G(A)$ if and only if there is a triple $(\{J_1, \dots, J_s\}, A, \{K_1, \dots, K_t\})$ in $\textsc{Nullable}(G)$, such that $\varepsilon\langle u\rangle v \in L_G(J_i)$ for all i and $u\langle v\rangle\varepsilon \in L_G(K_j)$ for all j.*

The plan is to reconstruct the grammar, so that for every triple $(\{J_1, \dots, J_s\}, A, \{K_1, \dots, K_t\})$ in $\textsc{Nullable}(G)$, and for every occurrence of A in the right-hand side of any rule, the new grammar contains a companion rule, in which A is omitted and context operators for J_i and K_i are introduced.

The following case requires special handling in the new grammar. Assume that A generates ε in the empty left context (that is, $u = \varepsilon$ in Lemma 1). This is reflected by a triple $(\{J_1, \dots, J_s\}, A, \{K_1, \dots, K_t\})$ in $\textsc{Nullable}(G)$, in which all symbols J_i also generate ε in the left context ε. The latter generation may in turn involve some further right context operators. In the new grammar, the left context will be explicitly set to be empty ($\lhd\varepsilon$), whereas all those right contexts should be assembled together with the set $\{K_1, \dots, K_t\}$, and used in the new rules, where A is omitted. This calculation of right contexts is done in the following special variant of the set $\textsc{Nullable}$.

Definition 5. *Let $G = (\Sigma, N, R, S)$ be a grammar. Define sets $\lhd\varepsilon\text{-}\textsc{Nullable}_i(G) \subseteq N \times 2^N$, with $i \geqslant 0$:*

$$\lhd\varepsilon\text{-}\textsc{Nullable}_0(G) = \{\, (A, V) \mid (\varnothing, A, V) \in \textsc{Nullable}(G) \,\},$$
$$\lhd\varepsilon\text{-}\textsc{Nullable}_{i+1}(G) = \Big\{ (A, V \cup V_1 \cup \ldots \cup V_s) \,\Big|\, (\{J_1, \dots, J_s\}, A, V) \in \textsc{Nullable}(G),$$
$$\exists V_1, \dots, V_s \subseteq N : (J_i, V_i) \in \lhd\varepsilon\text{-}\textsc{Nullable}_i(G) \Big\}.$$

Let $\lhd\varepsilon\text{-}\textsc{Nullable}(G) = \bigcup_{i \geqslant 0} \lhd\varepsilon\text{-}\textsc{Nullable}_i(G)$.

Lemma 2. *Let $G = (\Sigma, N, R, S)$ be a grammar, let $A \in N$ and $v \in \Sigma^*$. Then $\varepsilon\langle\varepsilon\rangle v \in L_G(A)$ if and only if there is a pair $(A, \{K_1, \dots, K_t\})$ in $\lhd\varepsilon\text{-}\textsc{Nullable}(G)$, such that $\varepsilon\langle v\rangle\varepsilon \in L_G(K_i)$ for all i.*

There is a symmetrically defined set $\rhd\varepsilon\text{-}\textsc{Nullable}(G) \subseteq 2^N \times N$, which characterizes the generation of ε in an empty right context.

With the generation of the empty string represented in these three sets, a grammar with two-sided contexts is transformed to the normal form as follows. First, it is convenient to simplify the rules of the grammar, so that every concatenation is of the form BC, with $B, C \in N$, and the context operators are only applied to individual nonterminals. For this, base conjuncts α with $|\alpha| > 2$ and context operators $\lhd\alpha$, $\trianglelefteq\alpha$, $\trianglerighteq\alpha$ and $\rhd\alpha$ with $|\alpha| > 1$ are shortened by introducing new nonterminals.

Lemma 3. *For every grammar $G_0 = (\Sigma, N_0, R_0, S_0)$, there exists and can be effectively constructed another grammar $G = (\Sigma, N, R, S)$ generating the same language, with all rules of the form:*

$$A \to a \tag{4a}$$
$$A \to BC \tag{4b}$$
$$A \to B_1 \,\&\, \dots \,\&\, B_k \,\&\, \lhd D_1 \,\&\, \dots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \dots \,\&\, \trianglelefteq E_n \,\&\, \trianglerighteq F_1 \,\&\, \dots \,\&\, \trianglerighteq F_{m'} \,\&\, \rhd H_1 \,\&\, \dots \,\&\, \rhd H_{n'} \tag{4c}$$
$$A \to \varepsilon, \tag{4d}$$

with $a \in \Sigma$ and $A, B, C, D_i, E_i, F_i, H_i \in N$.

Construction 1. Let $G = (\Sigma, N, R, S)$ be a grammar with two-sided contexts, with all rules of the form (4). Consider the sets $\textsc{Nullable}(G)$, $\lhd\varepsilon\text{-}\textsc{Nullable}(G)$ and $\rhd\varepsilon\text{-}\textsc{Nullable}(G)$, and construct another grammar with two-sided contexts $G' = (\Sigma, N, R', S)$, with the following rules.

1. All rules of the form (4a) in R are added to R'.
2. Every rule of the form (4b) in R is added to R', along with the following extra rules, where a nullable nonterminal is omitted and the fact that it generates ε is expressed by context operators.

$$A \to B \,\&\, \trianglelefteq J_1 \,\&\ldots\&\, \trianglelefteq J_s \,\&\, \rhd K_1 \,\&\ldots\&\, \rhd K_t, \quad \text{for } (\{J_1,\ldots,J_s\}, C, \{K_1,\ldots,K_t\}) \in \textsc{Nullable}(G)$$
$$A \to B \,\&\, \trianglelefteq J_1 \,\&\ldots\&\, \trianglelefteq J_s \,\&\, \rhd\varepsilon, \quad \text{for } (\{J_1,\ldots,J_s\}, C) \in \rhd\varepsilon\text{-}\textsc{Nullable}(G) \text{ with } s \geqslant 1$$
$$A \to C \,\&\, \lhd J_1 \,\&\ldots\&\, \lhd J_s \,\&\, \trianglerighteq K_1 \,\&\ldots\&\, \trianglerighteq K_t, \quad \text{for } (\{J_1,\ldots,J_s\}, B, \{K_1,\ldots,K_t\}) \in \textsc{Nullable}(G)$$
$$A \to C \,\&\, \trianglerighteq K_1 \,\&\ldots\&\, \trianglerighteq K_t \,\&\, \lhd\varepsilon, \quad \text{for } (B, \{K_1,\ldots,K_t\}) \in \lhd\varepsilon\text{-}\textsc{Nullable}(G) \text{ with } t \geqslant 1$$

   In the first case, C defines ε in left contexts J_i and right contexts K_i, and this restriction is implemented by context operators in the new rule. Since the left context of C includes B, extended context operators ($\trianglelefteq J_i$) are used on the left, whereas the right context operators are proper ($\rhd K_i$).

   The second case considers the possibility of a nullable nonterminal C, which defines ε in an empty right context. This condition is simulated by the conjunct $\rhd\varepsilon$ and extended left contexts $\trianglelefteq J_i$.

   The two last rules handle symmetrical cases, when the nonterminal B defines the empty string.
3. Every rule of the form (4c) is preserved in R'. In the original grammar, this rule (4c) may generate strings in empty contexts, as long as symbols in the context operators ($\lhd D_i$, $\rhd H_i$) are nullable.

   For any collection of pairs $(D_1, V_1), \ldots, (D_m, V_m) \in \lhd\varepsilon\text{-}\textsc{Nullable}(G)$, with $m \geqslant 1$, add the rule

$$A \to B_1 \,\&\ldots\&\, B_k \,\&\, E_1 \,\&\ldots\&\, E_n \,\&\, \trianglerighteq K_1 \,\&\ldots\&\, \trianglerighteq K_t \,\&\, \trianglerighteq F_1 \,\&\ldots\&\, \trianglerighteq F_{m'} \,\&\, \rhd H_1 \,\&\ldots\&\, \rhd H_{n'} \,\&\, \lhd\varepsilon,$$

   where $\{K_1,\ldots,K_t\} = \bigcup_{i=1}^{m} V_i$. Nonterminals $D_1, \ldots, D_m$ define ε in the right contexts given in the set $\lhd\varepsilon\text{-}\textsc{Nullable}(G)$. This is represented by conjuncts $\lhd\varepsilon$ and $\trianglerighteq K_i$. Extended left contexts $\trianglelefteq E_i$ are replaced with base conjuncts E_i, because in the empty left context they have the same effect.

   Symmetrically, if $(U_1, H_1), \ldots, (U_{n'}, H_{n'}) \in \rhd\varepsilon\text{-}\textsc{Nullable}(G)$, with $n' \geqslant 1$, then there is a rule

$$A \to B_1 \,\&\ldots\&\, B_k \,\&\, F_1 \,\&\ldots\&\, F_{m'} \,\&\, \lhd D_1 \,\&\ldots\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\ldots\&\, \trianglelefteq E_n \,\&\, \trianglelefteq K_1 \,\&\ldots\&\, \trianglelefteq K_t \,\&\, \rhd\varepsilon,$$

   where $\{K_1,\ldots,K_t\} = \bigcup_{i=1}^{n'} U_i$.

   Finally, if with m, $n' \geqslant 1$ and $(D_1, V_1), \ldots, (D_m, V_m) \in \lhd\varepsilon\text{-}\textsc{Nullable}(G)$, $(U_1, H_1), \ldots, (U_{n'}, H_{n'}) \in \rhd\varepsilon\text{-}\textsc{Nullable}(G)$, then the set R' contains a rule

$$A \to B_1 \,\&\ldots\&\, B_k \,\&\, E_1 \,\&\ldots\&\, E_n \,\&\, F_1 \,\&\ldots\&\, F_{m'} \,\&\, K_1 \,\&\ldots\&\, K_t \,\&\, \lhd\varepsilon \,\&\, \rhd\varepsilon,$$

   where $\{K_1,\ldots,K_t\} = \bigcup_{i=1}^{m} V_i \cup \bigcup_{j=1}^{n'} U_j$. In this case, both left and right contexts of a string are empty. All the symbols D_i and H_i define ε in the contexts specified in $\lhd\varepsilon\text{-}\textsc{Nullable}(G)$ and $\rhd\varepsilon\text{-}\textsc{Nullable}(G)$. These contexts apply to the entire string and are explicitly stated as $K_1 \,\&\ldots\&\, K_t$ in the new rule. The null contexts $\lhd\varepsilon$, $\rhd\varepsilon$ limit the applicability of this rule to the whole string. Again, as in the two previous cases, the base conjuncts are used instead of extended context operators.

Lemma 4. *Let $G = (\Sigma, N, R, S)$ be a grammar with two-sided contexts. Then the grammar $G' = (\Sigma, N', R', S)$ obtained by Construction 1 generates the language $L(G') = L(G) \setminus \{\varepsilon\}$.*

The above construction eliminates the empty string in all base conjuncts, but the resulting grammar may still contain null context specifications ($\lhd\varepsilon$ and $\rhd\varepsilon$), which state that the current substring is a prefix or a suffix of the whole string. These operators are eliminated by the following simple transformation. First, define a new nonterminal symbol U that generates all non-empty strings in the empty left context. This is done by the following three rules:

$$\begin{array}{ll} U \to Ua & \text{(for all } a \in \Sigma) \\ U \to a \,\&\, \trianglelefteq X & \text{(for all } a \in \Sigma) \\ X \to a & \text{(for all } a \in \Sigma) \end{array}$$

Another symbol V generates all non-empty strings in the empty right context; it is defined by symmetric rules. Then it remains to replace left and right null context operators ($\lhd\varepsilon$, $\rhd\varepsilon$) with U and V, respectively.

The third stage of the transformation to the normal form is removing the *unit conjuncts* in rules of the form $A \to B \,\&\, \dots$ Already for conjunctive grammars [13], the only known transformation involves substituting all rules for B into all rules for A; in the worst case, this results in an exponential blowup. The same construction applies verbatim to grammars with contexts.

This three-stage transformation proves the following theorem.

Theorem 1. *For each grammar with two-sided contexts $G = (\Sigma, N, R, S)$ there exists and can be effectively constructed a grammar with two-sided contexts $G' = (\Sigma, N', R', S)$ in the binary normal form, such that $L(G) = L(G') \setminus \{\varepsilon\}$.*

5 Parsing algorithm

Let $G = (\Sigma, N, R, S)$ be a grammar with two-sided contexts in the binary normal form, and let $w = a_1 \dots a_n \in \Sigma^+$, with $n \geqslant 1$ and $a_i \in \Sigma$, be an input string to be parsed. For every substring of w delimited by two positions i, j, with $0 \leqslant i < j \leqslant n$, consider the set of nonterminal symbols generating this substring.

$$T_{i,j} = \big\{ A \;\big|\; A \in N,\ a_1 \dots a_i \langle a_{i+1} \dots a_j \rangle a_{j+1} \dots a_n \in L_G(A) \big\}$$

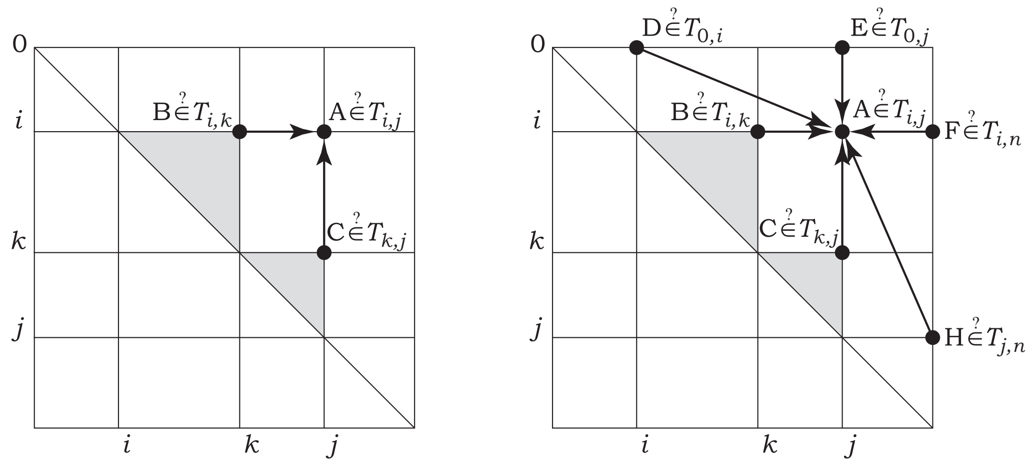
In particular, the whole string w is in $L(G)$ if and only if $S \in T_{0,n}$.

In ordinary context-free grammars, a substring $a_{i+1} \dots a_j$ is generated by A if there is a rule $A \to BC$ and a partition of the substring into $a_{i+1} \dots a_k$ generated by B and $a_{k+1} \dots a_j$ generated by C, as illustrated in Figure 3(left). Accordingly, each set $T_{i,j}$ depends only on the sets $T_{i',j'}$ with $j' - i' < j - i$, and hence all these sets may be constructed inductively, beginning with shorter substrings and eventually reaching the set $T_{0,n}$: this is the Cocke–Kasami–Younger parsing algorithm. For conjunctive grammars, all dependencies are the same, and generally the same parsing algorithm applies [13]. In grammars with only left contexts, each set $T_{i,j}$ additionally depends on the sets $T_{0,i}$ and $T_{0,j}$ via the conjuncts of the form $\lhd D$ and $\trianglelefteq E$, respectively, which still allows constructing these sets progressively for $j = 1, \dots, n$ [4].

The more complicated structure of logical dependencies in grammars with two-sided contexts is shown in Figure 3(right). The following example demonstrates how these dependencies may form circles.

Example 8. Consider the grammar with the rules

$$\begin{aligned} S &\to AB \\ A &\to a \,\&\, \rhd B \\ B &\to b \,\&\, \lhd C \\ C &\to a \end{aligned}$$



0
i
k
j
i
k
j
$B \overset{?}{\in} T_{i,k}$
$A \overset{?}{\in} T_{i,j}$
$C \overset{?}{\in} T_{k,j}$
$D \overset{?}{\in} T_{0,i}$
$E \overset{?}{\in} T_{0,j}$
$F \overset{?}{\in} T_{i,n}$
$H \overset{?}{\in} T_{j,n}$

Figure 3: How the membership of A in $T_{i,j}$ depends on other data, for rules (a) $A \to BC$ and (b) $A \to BC \,\&\, \lhd D \,\&\, \trianglelefteq E \,\&\, \trianglerighteq F \,\&\, \rhd H$.

and the input string $w = ab$. It is immediately seen that $C \in T_{0,1}$. From this, one can infer that $B \in T_{1,2}$, and that knowledge can in turn be used to show that $A \in T_{0,1}$. These data imply that $S \in T_{0,2}$. Thus, none of the sets $T_{0,1}$ and $T_{1,2}$ can be fully constructed before approaching the other.

The proposed algorithm for constructing the sets $T_{i,j}$ works as follows. At the first pass, it makes all deductions $\vdash_G A\big(a_1 \dots a_i \langle a_{i+1} \dots a_j \rangle a_{j+1} \dots a_n\big)$ that do not involve any contexts, and accordingly puts A to the corresponding $T_{i,j}$. This pass progressively considers longer and longer substrings, as done by the Cocke–Kasami–Younger algorithm for ordinary grammars. During this first pass, some symbols may be added to any sets $T_{0,j}$ and $T_{i,n}$, and thus it becomes known that some contexts are true. This triggers another pass over all entries $T_{i,j}$, from shorter substrings to longer ones, this time using the known true contexts in the deductions. This pass may result in adding more elements to $T_{0,j}$ and $T_{i,n}$, which will require yet another pass, and so on. Since a new pass is needed only if a new element is added to any of $2n-1$ subsets of N, the total number of passes is at most $(2n-1) \cdot |N| + 1$.

These calculations are implemented in Algorithm 1, which basically deduces all true statements about all substrings of the input string. For succinctness, the algorithm uses the following notation for multiple context operators. For a set $\mathscr{X} = \{X_1, \dots, X_\ell\}$, with $X_i \in N$, and for an operator $Q \in \{\lhd, \trianglelefteq, \trianglerighteq, \rhd\}$, denote $Q\mathscr{X} := QX_1 \,\&\, \dots \,\&\, QX_\ell$.

Theorem 2. *For every grammar with two-sided contexts G in the binary normal form, Algorithm 1, given an input string $w = a_1 \dots a_n$, constructs the sets $T_{i,j}$ and determines the membership of w in $L(G)$, and does so in time $\mathscr{O}(|G|^2 \cdot n^4)$, using space $\mathscr{O}(|G| \cdot n^2)$.*

While this paper was under preparation, Rabkin [20] developed a more efficient and more sophisticated parsing algorithm for grammars with two-sided contexts, with the running time $\mathscr{O}(|G| \cdot n^3)$, using space $\mathscr{O}(|G| \cdot n^2)$. Like Algorithm 1, Rabkin's algorithm works by proving all true statements about the substrings of the given string, but does so using the superior method of Dowling and Gallier [7]. Nevertheless, Algorithm 1 retains some value as the elementary parsing method for grammars with two-sided contexts—just like the Cocke–Kasami–Younger algorithm for ordinary grammars remains useful, in spite of the asymptotically superior Valiant's algorithm [23].

Algorithm 1. Let $G = (\Sigma, N, R, S)$ be a grammar with contexts in the binary normal form. Let $w = a_1 \dots a_n \in \Sigma^+$ (with $n \geqslant 1$ and $a_i \in \Sigma$) be the input string. Let $T_{i,j}$ with $0 \leqslant i < j \leqslant n$ be variables, each representing a subset of N, and let $T_{i,j} = \varnothing$ be their initial values.

```
1: while any of $T_{0,j}$ ($1 \leqslant j \leqslant n$) or $T_{i,n}$ ($1 \leqslant i < n$) change do
2:     for $j = 1, \dots, n$ do
3:         for all $A \to a \,\&\, \lhd\mathscr{D} \,\&\, \trianglelefteq\mathscr{E} \,\&\, \trianglerighteq\mathscr{F} \,\&\, \rhd\mathscr{H} \in R$ do
4:             if $a_j = a \wedge \mathscr{D} \subseteq T_{0,j-1} \wedge \mathscr{E} \subseteq T_{0,j} \wedge \mathscr{F} \subseteq T_{j,n} \wedge \mathscr{H} \subseteq T_{i,n}$ then
5:                 $T_{j-1,j} = T_{j-1,j} \cup \{A\}$
6:         for $i = j-2$ to $0$ do
7:             let $P = \varnothing$ ($P \subseteq N \times N$)
8:             for $k = i+1$ to $j-1$ do
9:                 $P = P \cup (T_{i,k} \times T_{k,j})$
10:            for all $A \to B_1C_1 \,\&\, \dots \,\&\, B_mC_m \,\&\, \lhd\mathscr{D} \,\&\, \trianglelefteq\mathscr{E} \,\&\, \trianglerighteq\mathscr{F} \,\&\, \rhd\mathscr{H} \in R$ do
11:                if $(B_1,C_1), \dots, (B_m,C_m) \in P \wedge \mathscr{D} \subseteq T_{0,i} \wedge \mathscr{E} \subseteq T_{0,j} \wedge \mathscr{F} \subseteq T_{j,n} \wedge \mathscr{H} \subseteq T_{i,n}$ then
12:                    $T_{i,j} = T_{i,j} \cup \{A\}$
13: accept if and only if $S \in T_{0,n}$
```

6 Conclusion

This paper has developed a formal representation for the idea of phrase-structure rules applicable in a context, featured in the early work of Chomsky [6]. This idea did not receive adequate treatment at the time, due to the unsuitable string-rewriting approach. The logical approach, adapted from Rounds [21] and his predecessors, brings it to life.

There are many theoretical questions to research about the new model: for instance, one can study the limitations of their expressive power, their closure properties, efficient parsing algorithms and subfamilies that admit more efficient parsing. Another possibility for further studies is investigating Boolean and stochastic variants of grammars with contexts, following the recent related work [8, 12, 24].

On a broader scope, there must have been other good ideas in the theory of formal grammars that were inadequately formalized before. They may be worth being re-investigated using the logical approach.

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, DOI: 10.1007/978-3-642-20712-9_27.

[2] M. Barash, "Programming language specification by a grammar with contexts", In: S. Bensch, F. Drewes, R. Freund, F. Otto (Eds.), *Fifth Workshop on Non-Classical Models of Automata and Applications* (NCMA 2013, Umeå, Sweden, 13–14 August, 2013), books@ocg.at 294, Österreichische Computer Gesellschaft (2013), 51–67, http://users.utu.fi/mikbar/kieli.

[3] M. Barash, A. Okhotin, "Defining contexts in context-free grammars", *Language and Automata Theory and Applications* (LATA 2012, A Coruña, Spain, 5–9 March 2012), LNCS 7183, 106–118, DOI: 10.1007/978-3-642-28332-1_10.

[4] M. Barash, A. Okhotin, "An extension of context-free grammars with one-sided context specifications", *Information and Computation*, in press, DOI: 10.1016/j.ic.2014.03.003.

[5] M. Barash, A. Okhotin, "Linear grammars with one-sided contexts and their automaton representation", *LATIN 2014: Theoretical Informatics* (Montevideo, Uruguay, 31 March–4 April 2014), LNCS 8392, 190–201, DOI: `10.1007/978-3-642-54423-1_17`.

[6] N. Chomsky, "On certain formal properties of grammars", *Information and Control*, 2:2 (1959), 137–167, DOI: `10.1016/S0019-9958(59)90362-6`.

[7] W. F. Dowling, J. H. Gallier, "Linear-time algorithms for testing the satisfiability of propositional Horn formulae", *Journal of Logic Programming*, 1:3 (1984), 267–284, DOI: `10.1016/0743-1066(84)90014-1`.

[8] Z. Ésik, W. Kuich, "Boolean fuzzy sets", *International Journal of Foundations of Computer Science*, 18:6 (2007), 1197–1207, DOI: `10.1142/S0129054107005248`.

[9] S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371, DOI: `10.1145/321127.321132`.

[10] A. Jeż, "Conjunctive grammars can generate non-regular unary languages", *International Journal of Foundations of Computer Science*, 19:3 (2008), 597–615, DOI: `10.1142/S012905410800584X`.

[11] R. Kowalski, *Logic for Problem Solving*, North-Holland, Amsterdam, 1979.

[12] V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, "Well-founded semantics for Boolean grammars", *Information and Computation*, 207:9 (2009), 945–967, DOI: `10.1016/j.ic.2009.05.002`.

[13] A. Okhotin, "Conjunctive grammars", *Journal of Automata, Languages and Combinatorics*, 6:4 (2001), 519–535.

[14] A. Okhotin, "Conjunctive grammars and systems of language equations", *Programming and Computer Software*, 28:5 (2002), 243–249, DOI: `10.1023/A:1020213411126`.

[15] A. Okhotin, "Boolean grammars", *Information and Computation*, 194:1 (2004), 19–48, DOI: `10.1016/j.ic.2004.03.006`.

[16] A. Okhotin, "Conjunctive and Boolean grammars: the true general case of the context-free grammars", *Computer Science Review*, 9 (2013), 27–59, DOI: `10.1016/j.cosrev.2013.06.001`.

[17] A. Okhotin, "Improved normal form for grammars with one-sided contexts", *Descriptional Complexity of Formal Systems* (DCFS 2013, London, Ontario, Canada, 22-25 July 2013), LNCS 8031, 205–216, DOI: `10.1007/978-3-642-39310-5_20`.

[18] A. Okhotin, "Parsing by matrix multiplication generalized to Boolean grammars", *Theoretical Computer Science*, 516 (2014), 101–120, DOI: `10.1016/j.tcs.2013.09.011`.

[19] F. C. N. Pereira, D. H. D. Warren, "Parsing as deduction", *21st Annual Meeting of the Association for Computational Linguistics* (ACL 1983, Cambridge, Massachusetts, USA, 15–17 June 1983), 137–144.

[20] M. Rabkin, "Recognizing two-sided contexts in cubic time", *Computer Science—Theory and Applications* (CSR 2014, Moscow, Russia, 6–12 June 2014), LNCS 8476, to appear.

[21] W. C. Rounds, "LFP: A logic for linguistic descriptions and an analysis of its complexity", *Computational Linguistics*, 14:4 (1988), 1–9.

[22] I. H. Sudborough, "A note on tape-bounded complexity classes and linear context-free languages", *Journal of the ACM*, 22:4 (1975), 499–500, DOI: `10.1145/321906.321913`.

[23] L. G. Valiant, "General context-free recognition in less than cubic time", *Journal of Computer and System Sciences*, 10:2 (1975), 308–314, DOI: `10.1016/S0022-0000(75)80046-8`.

[24] R. Zier-Vogel, M. Domaratzki, "RNA pseudoknot prediction through stochastic conjunctive grammars", *Computability in Europe 2013. Informal Proceedings*, 80–89.