Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

An extension of context-free grammars with one-sided context specifications ☆

Mikhail Barash [a,b], Alexander Okhotin [a,*]

[a] Department of Mathematics and Statistics, University of Turku, Turku FI-20014, Finland
[b] Turku Centre for Computer Science, Turku FI-20520, Finland

ARTICLE INFO

Article history:
Received 13 September 2013
Received in revised form 18 February 2014
Available online xxxx

Keywords:
Context-free grammars
Conjunctive grammars
Contexts
Context-sensitive grammars
Parsing

ABSTRACT

The paper introduces an extension of context-free grammars equipped with an operator for referring to the left context of the substring being defined. For example, a rule $A \to a \& \lhd B$ defines a symbol a, as long as it is preceded by a string defined by B. The conjunction operator in this example is taken from conjunctive grammars (Okhotin, 2001), which are an extension of ordinary context-free grammars that maintains most of their practical properties, including many parsing algorithms. This paper gives two equivalent definitions of grammars with left contexts—by logical deduction and by language equations—and establishes their basic properties, including a transformation to a normal form and a cubic-time parsing algorithm, with a square-time version for unambiguous grammars.

1. Introduction

Context-free grammars are best understood as a logic for defining the syntax of languages. In this logic, definitions are inductive, so that the properties of a string are determined by the properties of its substrings. This is how a rule $S \to aSb$ asserts that if a string $a^{n-1}b^{n-1}$ has the property S, then the string a^nb^n has the property S as well. Besides the concatenation, the formalism of this logic includes a 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 [19,21] 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 [21,23,24,27], and have been a subject of diverse research [1,8,12,13,17,28,29]. 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. Ordinary context-free grammars can be viewed as their disjunctive fragment. For a detailed account of the research on conjunctive and Boolean grammars, an interested reader is referred to a recent survey paper [26].

When Chomsky [6] 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 [6, p. 142]. However, Chomsky's attempt to formalize his idea using the tools available at the time (namely, string-rewriting systems) led to nothing but space-bounded nondeterministic Turing machines; the “nonterminal symbols” in that model

☆ A preliminary version of this paper was presented at the 6th International Conference on Language and Automata Theory and Applications (LATA 2012, A Coruña, Spain, 5–9 March 2012), and its extended abstract appeared in the conference proceedings.

* Corresponding author.
E-mail addresses: mikbar@utu.fi (M. Barash), alexander.okhotin@utu.fi (A. Okhotin).

http://dx.doi.org/10.1016/j.ic.2014.03.003
0890-5401/

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

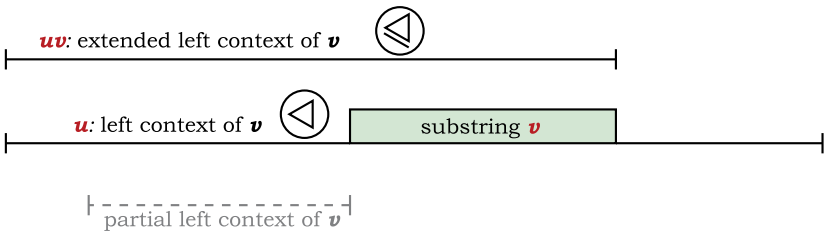

Fig. 1. A substring v with a left context u, denoted by $u\langle v\rangle$, and context operators applied to it.

no longer represent any syntactic classes, but are simply bits in the memory of those 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, and, in particular, they fail to implement Chomsky's original idea of a phrase-structure rule applicable in a context.

Even though Chomsky's terminology for formal grammars, such as the term "context-free", was generally accepted by the research community, the actual idea of a rule applicable in a context was never investigated again. None of the successful extensions of context-free grammars, such as tree-adjoining grammars [14] or multi-component grammars [36,33], allow expressing any conditions on the contexts—in spite of the nickname "mildly context-sensitive" [15]. It should also be noted that both tree-adjoining grammars and multi-component grammars define the properties of strings inductively on their length, and have nothing to do with Chomsky's "context-sensitive" string rewriting. Thus, the theory of formal grammars beyond ordinary context-free has developed in a different direction than initially pointed by Chomsky. Nevertheless, the general idea of context-sensitive rules in formal grammars remains interesting and deserves investigation.

This paper undertakes to reconsider Chomsky's [6] idea of contexts in grammars, this time using the more appropriate tools of deduction systems and language equations. The concept of a formal grammar as a logic and its semantics as logical inference was first presented in a monograph by Kowalski [18, Ch. 3], and then elaborated by Pereira and Warren [31]. Later, Rounds [32] used first-order logic over positions in a string augmented with a fixpoint operator, FO(LFP), to represent formal grammars as formulae of this logic. What is particularly important about this representation, is that all the aforementioned successful extensions of the context-free grammars, such as tree-adjoining grammars and multi-component grammars, are not only expressible in this logic, but the way they are expressed represents these grammars exactly as they are intuitively understood.

This paper derives the general outlook on grammars from the cited work [31,32], and draws from the experience of developing the conjunctive grammars [26] to define the desired grammar model. The new model proposed in this paper are *grammars with one-sided contexts*, which are based on conjunctive grammars and introduce two special operators for referring to the context of a substring being defined. The *left context operator* refers to the "past" of the current substring: an expression $\lhd\alpha$ defines any substring that is directly preceded by a prefix of the form α. This operator is meant to be used together with usual specifications of the structure of the current substring, using conjunction to combine several specifications. For example, consider the rule $A \to BC \,\&\, \lhd D$, which represents any substring of the form BC preceded by a substring of the form D. If the grammar contains additional rules $B \to b$, $C \to c$ and $D \to d$, then the above rule for A specifies that a substring bc of a string $w = dbc\ldots$ has the property A; however, this rule will not produce the same substring occurring in the strings $w' = abc$ or $w'' = adbc$. The other *extended left context operator* $\trianglelefteq\alpha$ represents the form of the left context of the current substring concatenated with the substring itself, so that the rules $A \to B \,\&\, \trianglelefteq E$, $B \to b$, $E \to ab$ state that the substring b occurring in the string $w = ab$ has the property A. Fig. 1 illustrates how context operators refer to a substring and its left context. One can symmetrically define operators for right contexts $\rhd\alpha$ and extended right contexts $\trianglerighteq\alpha$.

Note that the proposed context operators apply to the whole left context, which begins with the first symbol of the entire string. One could argue for using a partial left context instead, that is, any substring ending where the substring being defined begins (illustrated at the bottom of Fig. 1). Such a partial context of the form D can be easily described using the proposed operator as $\lhd\Sigma^*D$, where Σ^* stands for an arbitrary string. On the other hand, the whole left context can be simulated by a partial left context, provided that the entire string begins with a special marker symbol: if a partial context beginning with that symbol is requested, then it can only be the whole left context. Thus, a partial left context operator would be less convenient to use, as well as not any easier to implement than the proposed operator for the whole left context.

In the literature, related ideas have occasionally arisen in connection with parsing, where (extended) right contexts of the form $\trianglerighteq\alpha\Sigma^*$ (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 [7], LL-regular [11] and LL($*$) [30] parsers. Some software tools for engineering parsers, such as those developed by Parr and Fischer [30] and by Ford [9], allow specifying contexts $\trianglerighteq\alpha\Sigma^*$, with α defined within the grammar, and such specifications can be used by a programmer for *ad hoc* adjustment of 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 possibility, pursued in Section 2, is to consider deduction of elementary propositions of the form $A(u\langle v\rangle)$, where

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

$u\langle v\rangle$ 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 $A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd\beta_1 \,\&\, \ldots \,\&\, \lhd\beta_m \,\&\, \trianglelefteq\gamma_1 \,\&\, \ldots \,\&\, \trianglelefteq\gamma_n$, 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 such strings w that $S(\varepsilon\langle w\rangle)$ can be deduced. A standard proof tree of such a deduction constitutes a parse tree of the string w. This definition generalizes the representation of ordinary context-free grammars by deduction—assumed, for instance, by Pereira and Warren [31] and Sikkel [34]—as well as the extension of this representation to conjunctive grammars [22].

An alternative, equivalent definition given in Section 3 uses a generalization of language equations, in which the unknowns are *sets of strings with contexts* $u\langle v\rangle$. All connectives in the rules of a grammar—that is, concatenation, disjunction, conjunction and both context operators—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 ordinary context-free grammars [10] and conjunctive grammars [20]. 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 of the contexts, in which these substrings occur. Nothing like any bit manipulation in "sentential forms" is possible in the proposed model, and hence the 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.

The purpose of this paper is to settle the basic properties of grammars with one-sided contexts, and thus to justify that these grammars make some sense as a model of syntax. First, a transformation to a normal form generalizing the Chomsky normal form is devised in Section 4; the construction proceeds by first suppressing the generation of the empty string (as in a rule $A \to \varepsilon$), then by eliminating empty context specifications ($\lhd\varepsilon$) in any rules, and finally by removing chain dependencies (as in $A \to B$). 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 $\mathcal{O}(n^3)$, where n is the length of the input string. A variant of this algorithm works in time $\mathcal{O}(n^2)$ for unambiguous grammars with one-sided contexts; this algorithm extends a similar algorithm for unambiguous Boolean grammars [25], which is in turn based upon the algorithm of Kasami and Torii [16].

2. Definition by deduction

Formal grammars deal with finite strings over a finite alphabet Σ. The set of all such strings is denoted by Σ^*. The length of a string $w = a_1 \ldots a_\ell$, with $a_i \in \Sigma$, is the number of symbols in it, denoted by $|w| = \ell$. The unique string of length 0 is known as the *empty string* and denoted by ε. The concatenation of two strings $u, v \in \Sigma^*$ is the string $u \cdot v = uv$. For any strings $u, v, w \in \Sigma^*$, consider their concatenation uvw: then the string v is known as a *substring* of uvw, the string u is its *left context* and w is its *right context*.

This paper primarily deals with left contexts of strings, and a substring v occurring in a left context u shall be denoted by $u\langle v\rangle$. Formally, this is a pair of a string and its left context. It shall occasionally be referred to as simply a string, so that a reference to "a string $u\langle v\rangle$" assumes a substring occurring in a given context.

Any subset of Σ^* is a *(formal) language* over Σ. Since languages are sets, all Boolean operations on sets apply to them. In addition, for any two languages $K, L \subseteq \Sigma^*$, their concatenation is $KL = \{uv \mid u \in K,\ v \in L\}$. The concatenation of arbitrarily many strings from the same language $L \subseteq \Sigma^*$, defined by $L^* = \{w_1 \ldots w_\ell \mid \ell \geqslant 0,\ w_i \in L\}$, is known as the *Kleene star* of a language, or simply the *star*.

A formal grammar defines a language, viewed as a set of syntactically correct strings, by specifying their syntactic structure. The universally used *context-free grammars* [2,6] allow specifying concatenation of strings and disjunction of syntactical definitions. *Conjunctive grammars* [19] further allow the use of a conjunction operation. The new model introduced in this paper, *grammars with one-sided contexts*, use concatenation, conjunction and disjunction, as well as the new *context operators*, either only for left contexts ($\lhd$, $\trianglelefteq$), or only for right contexts ($\rhd$, $\trianglerighteq$). 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

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd\beta_1 \,\&\, \ldots \,\&\, \lhd\beta_m \,\&\, \trianglelefteq\gamma_1 \,\&\, \ldots \,\&\, \trianglelefteq\gamma_n, \tag{1}$$

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

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

For each grammar rule (1), each term α_i, $\lhd\beta_i$ and $\lessdot\gamma_i$ is called a *conjunct*. Each *base conjunct* α_i gives a representation of the substring being defined as a concatenation of shorter substrings. A conjunct $\lhd\beta_i$ similarly describes the form of the *left context* or the *past* of the substring being defined. Conjuncts of the form $\lessdot\gamma_i$ refer to the form of the left context and the current substring, concatenated into a single string.

A grammar with left contexts degenerates to a conjunctive grammar [19] if no context operators are ever used, that is, if $m = n = 0$ for every rule (1); and further to an ordinary context-free grammar if conjunction is never used, that is, if $k = 1$ in every rule.

Intuitively, a rule (1) can be read as follows: a substring v occurring in a left context u has the property A, if

- for each base conjunct $\alpha_i = X_1 \dots X_\ell$ in the rule, with $X_1, \dots, X_\ell \in \Sigma \cup N$, the string v is representable as a concatenation $X_1 \dots X_\ell$, that is, $v = v_1 \dots v_\ell$, where each substring v_j has the property X_i, *and*
- for each conjunct $\lhd\beta_i$, the left context u is similarly representable as β_i, *and*
- for each conjunct $\lessdot\gamma_i$, the string uv is representable as γ_i.

As in an ordinary context-free grammar, multiple rules $A \to \Phi_1, \dots, A \to \Phi_\ell$ for a single symbol A represent logical disjunction of conditions, and are denoted by $A \to \Phi_1 \mid \dots \mid \Phi_\ell$.

Formally, the semantics of grammars with contexts are defined by a deduction system, which deals with elementary propositions (items) of the form "a string $v \in \Sigma^*$ written in a left context $u \in \Sigma^*$ has the property $X \in \Sigma \cup N$", denoted by $X(u\langle v\rangle)$. The deduction begins with axioms, such as that any symbol $a \in \Sigma$ written in any context has the property a, which is denoted by $a(x\langle a\rangle)$ for all $x \in \Sigma^*$. 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

$$B\big(u\langle v\rangle\big), C\big(uv\langle w\rangle\big) \vdash_G A\big(u\langle vw\rangle\big) \quad \big(\text{for all } u, v, w \in \Sigma^*\big),$$

which is essentially a concatenation of v and w that respects the left contexts. If the rule is of the form $A \to BC \,\&\, \lhd D$, then this deduction requires an extra premise ensuring that the left context u has the property D:

$$B\big(u\langle v\rangle\big), C\big(uv\langle w\rangle\big), D\big(\varepsilon\langle u\rangle\big) \vdash_G A\big(u\langle vw\rangle\big) \quad \big(\text{for all } u, v, w \in \Sigma^*\big).$$

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

$$B\big(u\langle v\rangle\big), C\big(uv\langle w\rangle\big), E\big(\varepsilon\langle uvw\rangle\big) \vdash_G A\big(u\langle vw\rangle\big) \quad \big(\text{for all } u, v, w \in \Sigma^*\big).$$

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

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

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

Each rule $A \to \alpha_1 \,\&\, \dots \,\&\, \alpha_k \,\&\, \lhd\beta_1 \,\&\, \dots \,\&\, \lhd\beta_m \,\&\, \lessdot\gamma_1 \,\&\, \dots \,\&\, \lessdot\gamma_n$ in the grammar defines the following scheme for deduction rules:

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

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

- 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 $v = v_1 \dots v_\ell$ with $X_j(uv_1 \dots v_{j-1}\langle v_j\rangle) \in I$ for all $j \in \{1, \dots, \ell\}$;
- for every conjunct $\lhd\beta_i = \lhd X_1 \dots X_\ell$ with $\ell \geqslant 0$ and $X_j \in \Sigma \cup N$, there should be such a partition $u = u_1 \dots u_\ell$, that $X_j(u_1 \dots u_{j-1}\langle u_j\rangle) \in I$ for all $j \in \{1, \dots, \ell\}$;
- every conjunct $\lessdot\gamma_i = \lessdot X_1 \dots X_\ell$ with $\ell \geqslant 0$ and $X_j \in \Sigma \cup N$ should have a corresponding partition $uv = w_1 \dots w_\ell$ with $X_j(w_1 \dots w_{j-1}\langle w_j\rangle) \in I$ for all $j \in \{1, \dots, \ell\}$.

Note that if $\alpha_i = \varepsilon$ in the first case, that is, if $\ell = 0$, then the given condition implies $v = \varepsilon$, and deduction is possible only if $v = \varepsilon$; similarly, $\beta_i = \varepsilon$ implies $u = \varepsilon$, and $\gamma_i = \varepsilon$ implies $u = v = \varepsilon$.

Then the language generated by a symbol $A \in N$ is defined as the set of all strings with contexts $u\langle v\rangle$, for which the item $A(u\langle v\rangle)$ can be deduced from the axioms in one or more steps:

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

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

$$L(G) = \big\{w \;\big|\; w \in \Sigma^*, \vdash_G S\big(\varepsilon\langle w\rangle\big)\big\}.$$

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

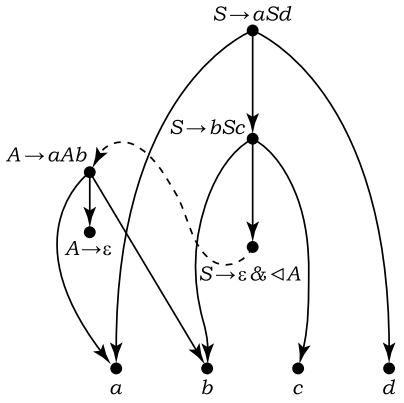

Fig. 2. A parse tree of the string *abcd* according to the grammar in Example 1.

Using both kinds of left context operators (proper $\lhd$ and extended $\leqslant$) is actually redundant, since each of them can be expressed through the other as follows:

- a proper left context $\lhd D$ is equivalent to a concatenation $D'\Sigma^*$, for a new nonterminal symbol D' with a single rule $D' \to \varepsilon \,\&\, \leqslant D$;
- an extended left context $\leqslant E$ can be replaced by a concatenation $\Sigma^* E'$, where the new nonterminal E' has a single rule $E' \to \varepsilon \,\&\, \lhd E$.

Using both context operators shall become essential later in Section 4, when transforming the grammar to a normal form, in which the empty string is prohibited.

The following sample grammar with contexts defines a rather simple language, which is an intersection of two ordinary context-free languages, and hence could be defined by a conjunctive grammar without contexts. The value of this example is in demonstrating the machinery of contexts in action.

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

$$S \to aSd \mid bSc \mid \varepsilon \,\&\, \lhd A$$

$$A \to aAb \mid \varepsilon$$

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

$$\begin{aligned}
&\vdash a\big(\varepsilon\langle a\rangle\big) && \text{(axiom)}\\
&\vdash b\big(a\langle b\rangle\big) && \text{(axiom)}\\
&\vdash c\big(ab\langle c\rangle\big) && \text{(axiom)}\\
&\vdash d\big(abc\langle d\rangle\big) && \text{(axiom)}\\
&\vdash A\big(a\langle\varepsilon\rangle\big) && (A \to \varepsilon)\\
a\big(\varepsilon\langle a\rangle\big), A\big(a\langle\varepsilon\rangle\big), b\big(a\langle b\rangle\big) &\vdash A\big(\varepsilon\langle ab\rangle\big) && (A \to aAb)\\
A\big(\varepsilon\langle ab\rangle\big) &\vdash S\big(ab\langle\varepsilon\rangle\big) && (S \to \varepsilon \,\&\, \lhd A)\\
b\big(a\langle b\rangle\big), S\big(ab\langle\varepsilon\rangle\big), c\big(ab\langle c\rangle\big) &\vdash S\big(a\langle bc\rangle\big) && (S \to bSc)\\
a\big(\varepsilon\langle a\rangle\big), S\big(a\langle bc\rangle\big), d\big(abc\langle d\rangle\big) &\vdash S\big(\varepsilon\langle abcd\rangle\big) && (S \to aSd)
\end{aligned}$$

The tree corresponding to this deduction is given in Fig. 2, where the reference to a context is marked by a dotted arrow. This tree represents a parse of the string according to the grammar.

The deductions according to Definition 2 are just a formalization of such parse trees as the one in Fig. 2, in the same way as Chomsky's string rewriting formalizes context-free parse trees. Even though deductions look bulkier than string rewriting, they ultimately represent the same logical content, which is as simple as a parse tree.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

The next example defines an abstract language representing declaration of identifiers before their use. Though the same language can be defined by a conjunctive grammar [26, Ex. 3], the grammar with contexts given here provides a more natural definition of this construct.

Example 2. The following grammar defines the language

$$\left\{u_1 \dots u_n \;\middle|\; \text{for every } u_i,\ \textbf{either } u_i \in a^*c,\ \textbf{or } u_i = b^k c \text{ and there exists } j < i,\ \text{for which } u_j = a^k c\right\}. \tag{2}$$

$$S \to AS \mid CS \mid \varepsilon$$

$$A \to aA \mid c$$

$$B \to bB \mid c$$

$$C \to B \,\&\, {\lhd}EFc$$

$$E \to AE \mid BE \mid \varepsilon$$

$$F \to aFb \mid cE$$

Substrings of the form $a^k c$ stand for declarations, while every substring of the form $b^k c$ is a reference to a declaration of the form $a^k c$. The claim is that S generates a string $u_1 \dots u_\ell \langle u_{\ell+1} \dots u_n \rangle$, with $u_i \in a^*c \cup b^*c$, if and only if every reference u_i in the suffix $u_{\ell+1} \dots u_n$ has a corresponding earlier declaration in the prefix $u_1 \dots u_{i-1}$. The grammar defines all strings satisfying this condition inductively on ℓ. The base case is given by the rule $S \to \varepsilon$: the string $u_1 \dots u_n \langle \varepsilon \rangle$ has the desired property. The rule $S \to AS$ appends any declaration (A), while the rules $S \to CS$ and $C \to B \,\&\, {\lhd}EFc$ append a reference (B), which has a matching earlier declaration (${\lhd}EFc$). The latter context specification checks the declaration as follows: the symbol E represents the prefix of the string up to that declaration, while F matches the symbols a in the declaration to the symbols b in the reference.

The next grammar defines a variant of the language from Example 2, for which no conjunctive or Boolean grammar is known.

Example 3. Consider the language $\{u_1 \dots u_n \mid \text{for every } u_i,\ \textbf{either } u_i \in a^*c,\ \textbf{or } u_i = b^k c \text{ and there exists } j \in \{1, \dots, n\} \text{ with } u_j = a^k c\}$. This is an abstract language representing declaration of identifiers *before or after* their use. As in Example 2, substrings $a^k c$ are declarations and substrings $b^k c$ are references. This language is generated by the following grammar.

$$S \to AS \mid CS \mid BS \,\&\, DcE \mid \varepsilon$$

$$A \to aA \mid c$$

$$B \to bB \mid c$$

$$C \to B \,\&\, {\lhd}EFc$$

$$D \to bDa \mid cE$$

$$E \to AE \mid BE \mid \varepsilon$$

$$F \to aFb \mid cE$$

As compared with the grammar in Example 2, this grammar allows matching a reference to a declaration located *later* in the string, which is done by the rule $S \to BS \,\&\, DcE$. Here the first conjunct BS appends a reference, while the other conjunct DcE uses D to match the bs forming this reference to the as in the later declaration.

The ideas of Example 2 have been implemented in an extensive example of a grammar with left contexts, which defines the complete syntax of a simple typed programming language [4].

Consider an important property of formal grammars in general: the *syntactic ambiguity*, that is, having multiple parses of a single string. A grammar that defines a unique parse of each string it generates is known as unambiguous. The following formal definition of this notion is adapted from the case of conjunctive and Boolean grammars [25].

Definition 3. A grammar with contexts $G = (\Sigma, N, R, S)$ is said to be unambiguous, if it has the following two properties:

Unambiguous choice of a rule. For every item $\vdash_G A(u\langle v \rangle)$ with $A \in N$ and $u, v \in \Sigma^*$, there exists a unique rule, by which this item is deduced. In other words, different rules generate disjoint languages.

Unambiguous concatenation. For every conjunct $X_1 \dots X_\ell$, ${\lhd}X_1 \dots X_\ell$ or ${\lhd}X_1 \dots X_\ell$ that occurs in any rule, and for all $u, v \in \Sigma^*$, there exists at most one partition $v = v_1 \dots v_\ell$ with $uv_1 \dots v_{i-1}\langle v_i \rangle \in L_G(X_i)$ for all i.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

The grammar in Example 1, which generates the language $\{a^n b^n c^n d^n \mid n \geqslant 0\}$, is unambiguous. To see that there is no ambiguity of choice, consider that each string in $L_G(S)$ either begins with a and is generated by the rule $S \to aSd$, or begins with b and is generated by $S \to bSc$, or is empty and is generated by the last rule $S \to \varepsilon \,\&\, \lhd A$; hence, the rules for S generate disjoint languages. Similarly, one of the two rules for A generates non-empty strings that begin with a, and the other generates the empty string. Concatenations in this grammar are unambiguous, because all of them are of the form xTy, with $x, y \in \Sigma^*$ and $T \in N$.

On the other hand, the grammar in Example 2, representing the language of declarations before use, is ambiguous, because it directly transcribes the definition of the language (2), and that definition contains syntactic ambiguity in itself. Whenever for a substring $b^k c$ representing a "reference" there exist multiple earlier "declarations" $u_j = u_{j'} = a^k c$ with $j \neq j'$, the grammar will allow different parses for these two cases. In terms of Definition 3, this constitutes ambiguity in the concatenation EFc in the rule $C \to B \,\&\, \leqslant EFc$. The grammar in Example 3 has two further instances of ambiguity: every time a "reference" has matching "declarations" before and after it, there is an ambiguity of choice between the rules $S \to CS$ and $S \to BS \,\&\, DcE$, and a "reference" followed by multiple "declarations" makes the concatenation DcE ambiguous.

Nevertheless, it is possible to construct unambiguous grammars generating the same languages as in Examples 2 and 3. Such grammars would need to fix, which of the multiple "declarations" should be matched to each "reference"; for instance, this could be the leftmost "declaration". Constructing unambiguous grammars for these languages is left as an exercise for the reader.

3. Definition by language equations

The representation of ordinary context-free grammars by language equations, introduced by Ginsburg and Rice [10], is one of the several ways of defining their semantics. For example, a grammar $S \to aSb \mid \varepsilon$ is regarded as an equation $S = (\{a\} \cdot S \cdot \{b\}) \cup \{\varepsilon\}$, which has the unique solution $S = \{a^n b^n \mid n \geqslant 0\}$. Conjunctive grammars inherit the same definition by equations [20], with the conjunction represented by the intersection operation. This important representation shall now be extended to grammars with contexts.

In order to represent contexts in the equations, the whole model has to be extended from ordinary formal languages to languages of strings with contexts $u\langle v\rangle$, that is, to sets of pairs of strings $L \subseteq \Sigma^* \times \Sigma^*$. All usual operations on languages used in equations will have to be extended to languages of strings with contexts. For all $K, L \subseteq \Sigma^* \times \Sigma^*$, consider their

- *union* $K \cup L = \{u\langle v\rangle \mid u\langle v\rangle \in K \text{ or } u\langle v\rangle \in L\}$,
- *intersection* $K \cap L = \{u\langle v\rangle \mid u\langle v\rangle \in K,\ u\langle v\rangle \in L\}$,
- *concatenation* $K \cdot L = \{u\langle vw\rangle \mid u\langle v\rangle \in K,\ uv\langle w\rangle \in L\}$,
- *(proper) left context* $\lhd L = \{u\langle v\rangle \mid \varepsilon\langle u\rangle \in L,\ v \in \Sigma^*\}$ and
- *extended left context*, $\leqslant L = \{u\langle v\rangle \mid \varepsilon\langle uv\rangle \in L\}$.

A language of strings with contexts shall be occasionally called *a language with contexts* or simply *a language*. Given a grammar with contexts, the plan is to define a system of equations, in which the unknowns are languages (of strings with contexts) defined by the nonterminal symbols of the grammar. A solution of the system is any such assignment of languages to variables that turns each equation into an equality. The system will be defined in the way that it always has solutions, and among them there is a *least solution* with respect to the partial order $\sqsubseteq$ of componentwise inclusion, and this solution will be used to define the language generated by the grammar.

The order of componentwise inclusion is defined on the set $(2^{\Sigma^* \times \Sigma^*})^n$ of n-tuples of languages, where n is the number of nonterminal symbols in the grammar. For any two such n-tuples, let $(K_1, \ldots, K_n) \sqsubseteq (L_1, \ldots, L_n)$ if and only if $K_1 \subseteq L_1$, $\ldots$, $K_n \subseteq L_n$. The least element under this order is the vector of empty sets $\bot = (\varnothing, \ldots, \varnothing)$.

Definition 4. 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 every variable assumes a value of a language with contexts $L \subseteq \Sigma^* \times \Sigma^*$, and which contains the following equation for every variable A:

$$A = \bigcup_{\substack{A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\\ \&\, \lhd\beta_1 \,\&\, \ldots \,\&\, \lhd\beta_m \,\&\\ \&\, \leqslant\gamma_1 \,\&\, \ldots \,\&\, \leqslant\gamma_{m'} \in R}} \left[\bigcap_{i=1}^{k} \alpha_i \cap \bigcap_{i=1}^{m} \lhd\beta_i \cap \bigcap_{i=1}^{m'} \leqslant\gamma_i \right]. \tag{3}$$

Each string α_i, β_i or γ_i in such a system denotes a concatenation of variables and symbols, where each instance of a symbol $a \in \Sigma$ defines a constant language $\{x\langle a\rangle \mid x \in \Sigma^*\}$. If any of α_i, β_i or γ_i is an empty string, it denotes the language $\{x\langle\varepsilon\rangle \mid x \in \Sigma^*\}$.

Let $N = \{A_1, \ldots, A_n\}$ and let $(L_{A_1}, \ldots, L_{A_n})$, with $L_{A_i} \subseteq \Sigma^* \times \Sigma^*$, be the least solution of the system (3). Define the language generated by each nonterminal symbol $A \in N$ as the corresponding component of this solution: $L_G(A) = L_A$. Let $L(G) = \{w \mid \varepsilon\langle w\rangle \in L_S\}$.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

For this definition to be correct, it should be proved that any such system has a *least solution* with respect to the partial order $\sqsubseteq$ of componentwise inclusion. Let $N = \{A_1, \ldots, A_n\}$ be the set of variables. Then the system of language equations (3) can be denoted by

$$\begin{cases} A_1 = \varphi_1(A_1, \ldots, A_n) \\ \quad \vdots \\ A_n = \varphi_n(A_1, \ldots, A_n) \end{cases} \tag{4}$$

where the functions $\varphi_i \colon (2^{\Sigma^* \times \Sigma^*})^n \to 2^{\Sigma^* \times \Sigma^*}$ in the right-hand sides are defined as a composition of the operations of concatenation, union, intersection, left contexts ($\triangleleft$) and extended left contexts ($\leftthreetimes$). The right-hand sides of such a system can be represented as a vector function $\varphi = (\varphi_1, \ldots, \varphi_n)$, which has the following properties.

Lemma 1. *For every grammar with contexts, the vector function $\varphi = (\varphi_1, \ldots, \varphi_n)$ in the associated system of language equations is monotone, in the sense that for any two vectors K and L, the inequality $K \sqsubseteq L$ implies $\varphi(K) \sqsubseteq \varphi(L)$.*

The proof of Lemma 1 follows directly from the definitions of operations, like in the case of conjunctive grammars [20].

Another property of functions in the right-hand sides of these equations is their *continuity*: whenever a sequence of arguments $\{L^{(i)}\}_{i=1}^{\infty}$ converges to L, the sequence formed by applying φ to all elements of the sequence converges to $\varphi(L)$. In this paper, it is sufficient to prove the weaker property of $\sqcup$-*continuity*, defined with respect to *ascending sequences* of the form $L^{(1)} \sqsubseteq L^{(2)} \sqsubseteq \cdots \sqsubseteq L^{(k)} \sqsubseteq \cdots$.

Lemma 2. *For every grammar with contexts, the vector function $\varphi = (\varphi_1, \ldots, \varphi_n)$ in the associated system of language equations is $\sqcup$-continuous, in the sense that for every ascending sequence of vectors of languages with contexts $\{L^{(i)}\}_{i=1}^{\infty}$, it holds that*

$$\bigsqcup_{i=1}^{\infty} \varphi\big(L^{(i)}\big) = \varphi\left(\bigsqcup_{i=1}^{\infty} L^{(i)}\right).$$

This lemma is proved by a classical argument, based on the fact that the membership of any element $u\langle v\rangle$ in an n-tuple of sets $\varphi(L^{(i)})$ depends on finitely many elements in the argument $L^{(i)}$.

The next result follows by the standard method of fixpoint iteration, as explained, for instance, in the handbook chapter by Autebert et al. [2]. According to this method, the vector function φ on the right-hand side of the system is iteratively applied to the vector of empty sets, and after countably many iterations, the process converges to the least solution of the system. The method applies to every system of Eqs. (3) corresponding to a grammar with contexts, due to monotonicity and continuity of its right-hand sides (Lemmata 1 and 2).

Lemma 3. *Every system of language equations (4), where φ is monotone and $\sqcup$-continuous, has a least solution, which is given by*

$$\bigsqcup_{k=0}^{\infty} \varphi^k(\bot).$$

Example 4. The following system of equations represents the grammar in Example 1.

$$S = \big(\Sigma^*\langle a\rangle \cdot S \cdot \Sigma^*\langle d\rangle\big) \cup \big(\Sigma^*\langle b\rangle \cdot S \cdot \Sigma^*\langle c\rangle\big) \cup \big(\Sigma^*\langle\varepsilon\rangle \cap \triangleleft A\big)$$
$$A = \big(\Sigma^*\langle a\rangle \cdot A \cdot \Sigma^*\langle b\rangle\big) \cup \Sigma^*\langle\varepsilon\rangle$$

Consider the sequence of iterations leading to the least solution of the system.

$$\varphi^0(\bot) = \begin{pmatrix} \varnothing \\ \varnothing \end{pmatrix}$$
$$\varphi^1(\bot) = \begin{pmatrix} \varnothing \\ \Sigma^*\langle\varepsilon\rangle \end{pmatrix}$$
$$\varphi^2(\bot) = \begin{pmatrix} \varepsilon\langle\varepsilon\rangle \\ \Sigma^*\langle\varepsilon\rangle \cup \Sigma^*\langle ab\rangle \end{pmatrix}$$
$$\varphi^3(\bot) = \begin{pmatrix} \{\varepsilon\langle\varepsilon\rangle, ab\langle\varepsilon\rangle\} \\ \Sigma^*\langle\varepsilon\rangle \cup \Sigma^*\langle ab\rangle \cup \Sigma^*\langle a^2b^2\rangle \end{pmatrix}$$
$$\varphi^4(\bot) = \begin{pmatrix} \{\varepsilon\langle\varepsilon\rangle, ab\langle\varepsilon\rangle, a\langle bc\rangle, a^2b^2\langle\varepsilon\rangle\} \\ \Sigma^*\langle\varepsilon\rangle \cup \Sigma^*\langle ab\rangle \cup \Sigma^*\langle a^2b^2\rangle \cup \Sigma^*\langle a^3b^3\rangle \end{pmatrix}$$

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

$$\varphi^5(\bot) = \begin{pmatrix} \{\varepsilon\langle\varepsilon\rangle, ab\langle\varepsilon\rangle, a^2b^2\langle\varepsilon\rangle, a^3b^3\langle\varepsilon\rangle, a\langle bc\rangle, aab\langle bc\rangle, \varepsilon\langle abcd\rangle\} \\ \Sigma^*\langle\varepsilon\rangle \cup \Sigma^*\langle ab\rangle \cup \Sigma^*\langle a^2b^2\rangle \cup \Sigma^*\langle a^3b^3\rangle \cup \Sigma^*\langle a^4b^4\rangle \end{pmatrix}, \quad \text{etc.}$$

This sequence tends to a pair $S = \{a^{n-i}\langle a^ib^nc^nd^i\rangle \mid 0 \leqslant i \leqslant n\} \cup \{a^nb^{n-i}\langle b^ic^i\rangle \mid 0 \leqslant i \leqslant n\}$, $A = \{x\langle a^nb^n\rangle \mid x \in \Sigma^*\}$, which is therefore the least solution of the system.

This proves that every system of equations corresponding to a grammar with contexts has a least solution, and hence Definition 4 is correct. It remains to prove that Definitions 2 and 4 are equivalent, that is, they define the same languages $L_G(A)$, for all $A \in N$.

Let N be the set of nonterminal symbols, let $|N| = n$. For every n-tuple of languages with contexts $L \in (2^{\Sigma^* \times \Sigma^*})^n$ and for every symbol $A \in N$, denote the A-component of this tuple by $[L]_A$.

Theorem 1. *Let $G = (\Sigma, N, R, S)$ be a grammar with left contexts, where $N = \{A_1, \ldots, A_n\}$, and let $A_i = \varphi_i(A_1, \ldots, A_n)$ $(1 \leqslant i \leqslant n)$ be the associated system of language equations. Let $u\langle v\rangle \in \Sigma^* \times \Sigma^*$ be a string with a context. Then, for every $A \in N$,*

$$u\langle v\rangle \in \left[\bigsqcup_{t \geqslant 0} \varphi^t(\varnothing, \ldots, \varnothing)\right]_A \quad \text{if and only if} \quad \vdash_G A\big(u\langle v\rangle\big).$$

Proof. The argument uses the notation $\alpha(L)$ for the substitution of an n-tuple of languages with contexts L into a concatenation $\alpha = X_1 \ldots X_\ell$, with $X_i \in \Sigma \cup N$. Its value is a language with contexts, defined as $\alpha(L) = [L]_{X_1} \ldots [L]_{X_\ell}$, where $[L]_A$ with $A \in N$ is the A-component of L, while $[L]_a = \{x\langle a\rangle \mid x \in \Sigma^*\}$ for all $a \in \Sigma$.

$\Rightarrow$ The proof is by induction on t, the number of iterations made until the string $u\langle v\rangle$ appears in $[\varphi^t(\varnothing, \ldots, \varnothing)]_A$.

Basis. For $t = 0$, there are no $A \in N$ satisfying the assumptions.

Induction step. Let the string $u\langle v\rangle$ be obtained in t iterations, that is, $u\langle v\rangle \in [\varphi^t(\bot)]_A = A(\varphi^{t-1}(\bot))$. Then, substituting $\varphi^{t-1}(\bot)$ into the equation for A yields

$$u\langle v\rangle \in \bigcup_{\substack{A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\\ \&\, \lhd\beta_1 \,\&\, \ldots \,\&\, \lhd\beta_m \,\&\\ \&\, \lhd\!\!\lhd\gamma_1 \,\&\, \ldots \,\&\, \lhd\!\!\lhd\gamma_{m'} \in R}} \left(\bigcap_{i=1}^{k} \alpha_i\big(\varphi^{t-1}(\bot)\big) \cap \bigcap_{i=1}^{m} \lhd\beta_i\big(\varphi^{t-1}(\bot)\big) \cap \bigcap_{i=1}^{m'} \lhd\!\!\lhd\gamma_i\big(\varphi^{t-1}(\bot)\big)\right).$$

Hence, there should exist such a rule

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd\beta_1 \,\&\, \ldots \lhd\beta_m \,\&\, \lhd\!\!\lhd\gamma_1 \,\&\, \ldots \,\&\, \lhd\!\!\lhd\gamma_{m'}, \tag{5}$$

that a substitution of the languages $\varphi^{t-1}(\bot)$ into each conjunct of this rule (α_i, $\lhd\beta_i$ or $\lhd\!\!\lhd\gamma_i$) evaluates to a language containing $u\langle v\rangle$.

Consider each base conjunct α_i, with $i \in \{1, \ldots, k\}$. Let $\alpha_i = X_{i,1} \ldots X_{i,\ell_i}$, where $\ell_i \geqslant 0$ and $X_{i,j} \in \Sigma \cup N$ for all $j \in \{1, \ldots, \ell_i\}$. It is known that $u\langle v\rangle \in \alpha_i(\varphi^{t-1}(\bot))$. Then, by the definition of concatenation with contexts, there is such a partition $v = v_1 \ldots v_{\ell_i}$, that

$$uv_1 \ldots v_{j-1}\langle v_j\rangle \in X_{i,j}\big(\varphi^{t-1}(\bot)\big), \tag{6}$$

for all $j \in \{1, \ldots, \ell_i\}$. For each $X_{i,j}$, if $X_{i,j}$ is a terminal symbol $X_{i,j} = a \in \Sigma$, then the substitution gives a constant language $X_{i,j}(\varphi^{t-1}(\bot)) = \{x\langle a\rangle \mid x \in \Sigma^*\}$, and the above condition (6) implies that $v_j = a$; hence, the item $X_{i,j}(uv_1 \ldots v_{j-1}\langle v_j\rangle)$ is deduced by an axiom. If $X_{i,j}$ is a nonterminal symbol $X_{i,j} = B \in N$, then the string $uv_1 \ldots v_{j-1}\langle v_j\rangle$ should be in $X_{i,j}(\varphi^{t-1}(\bot)) = [\varphi^{t-1}(\bot)]_B$; by the induction hypothesis, $\vdash_G X_{i,j}(uv_1 \ldots v_{j-1}\langle v_j\rangle)$.

The cases of conjuncts $\lhd\beta_i$ and $\lhd\!\!\lhd\gamma_i$ are analogous. Consider, for instance, $\lhd\beta_i$, and let $\beta_i = Y_{i,1} \ldots Y_{i,\ell_i'}$, where $Y_{i,j} \in \Sigma \cup N$. If $u\langle v\rangle \in \beta_i(\varphi^{t-1}(\bot))$, then there is a partition $u = u_1 \ldots u_{\ell_i'}$, for which $u_1 \ldots u_{j-1}\langle u_j\rangle \in Y_{i,j}(\varphi^{t-1}(\bot))$, for all $j \in \{1, \ldots, \ell_i'\}$. For every $Y_{i,j} = B \in N$, the induction hypothesis can be used to show that $\vdash_G Y_{i,j}(u_1 \ldots u_{j-1}\langle u_j\rangle)$; for a terminal symbol $Y_{i,j} = a \in \Sigma$, this item is an axiom.

For each extended context $\lhd\!\!\lhd\gamma_i$, let $\gamma_i = Z_{i,1} \ldots Z_{i,\ell_i''}$. Here $u\langle v\rangle \in \gamma_i(\varphi^{t-1}(\bot))$ implies that there is a partition $uv = w_1 \ldots w_{\ell_i''}$, such that the string $w_1 \ldots w_{j-1}\langle w_j\rangle$ is in $Z_{i,j}(\varphi^{t-1}(\bot))$, for all $j \in \{1, \ldots, \ell_i''\}$. Again, it is proved that $\vdash_G Z_{i,j}(w_1 \ldots w_{j-1}\langle w_j\rangle)$, using the induction hypothesis for nonterminal symbols and the axiom for terminal symbols.

Thus all the premises for deducing the item $A(u\langle v\rangle)$ according to the rule (5) have been obtained, and one can make the desired step of deduction:

$$\big\{X_{i,j}\big(uv_1 \ldots v_{j-1}\langle v_j\rangle\big)\big\}_{i,j}, \big\{Y_{i,j}\big(u_1 \ldots u_{j-1}\langle u_j\rangle\big)\big\}_{i,j}, \big\{Z_{i,j}\big(w_1 \ldots w_{j-1}\langle w_j\rangle\big)\big\}_{i,j} \vdash_G A\big(u\langle v\rangle\big).$$

$\Leftarrow$ Assume that $A(u\langle v\rangle)$ is deducible in the grammar, and let p be the number of steps in its deduction. It is claimed that $u\langle v\rangle \in [\varphi^t(\bot)]_A$ for some t depending on p. The proof is by induction on p.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

Basis. Let $p = 1$, that is, the item $A(u\langle v\rangle)$ is deduced in a single step by a rule $A \to v$. Then $u\langle v\rangle \in \varphi^1(\varnothing, \ldots, \varnothing)$.

Induction step. Assume that an item $A(u\langle v\rangle)$ is deduced in p steps, and consider the rule used at the last step of deduction:

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd\beta_1 \,\&\, \ldots \,\&\, \lhd\beta_m \,\&\, \lhdeq\gamma_1 \,\&\, \ldots \,\&\, \lhdeq\gamma_{m'}.$$

Then,

- for each conjunct $\alpha_i = X_{i,1} \ldots X_{i,\ell_i}$ in the rule, there is a partition $v = v_1 \ldots v_{\ell_i}$, so that $\vdash_G X_{i,j}(uv_1 \ldots v_{j-1}\langle v_j\rangle)$ for all $j \in \{1, \ldots, \ell_i\}$;
- for each conjunct $\lhd\beta_i = \lhd Y_{i,1} \ldots Y_{i,\ell'_i}$, there is a partition $u = u_1 \ldots u_{\ell'_i}$, for which $\vdash_G Y_{i,j}(u_1 \ldots u_{j-1}\langle u_j\rangle)$ for all $j \in \{1, \ldots, \ell'_i\}$;
- for each conjunct $\lhdeq\gamma_i = \lhdeq Z_{i,1} \ldots Z_{i,\ell''_i}$ there is a partition $uv = w_1 \ldots w_{\ell''_i}$, satisfying $\vdash_G Z_{i,j}(w_1 \ldots w_{j-1}\langle w_j\rangle)$ for all $j \in \{1, \ldots, \ell''_i\}$.

Consider any base conjunct $\alpha_i = X_{i,1} \ldots X_{i,\ell_i}$, with $X_{i,j} \in \Sigma \cup N$, and its j-th symbol $X_{i,j}$. If $X_{i,j} = B \in N$, then the deduction of the item $X_{i,j}(uv_1 \ldots v_{j-1}\langle v_j\rangle)$ takes fewer than p steps, and therefore, by the induction hypothesis, there exist such a number $t_{i,j}$, that $uv_1 \ldots v_{j-1}\langle v_j\rangle \in B(\varphi^{t_{i,j}}(\bot))$. If $X_{i,j} = a \in \Sigma$, then $v_j = a$ and $uv_1 \ldots v_{j-1}\langle v_j\rangle = uv_1 \ldots v_{j-1}\langle a\rangle \in \Sigma^*\langle a\rangle$. The concatenation of these strings with contexts $u\langle v_1\rangle, uv_1\langle v_2\rangle, \ldots$, up to $uv_1 \ldots v_{j-1}\langle v_j\rangle$, is $u\langle v\rangle$, and therefore $u\langle v\rangle \in \alpha_i(\varphi^{t_i}(\bot))$, where $t_i = \max_j t_{i,j}$.

Applying the same procedure to each conjunct $\lhd\beta_i$ or $\lhdeq\gamma_i$ similarly yields $u\langle v\rangle \in \lhd\beta_i(\varphi^{t'_i}(\bot))$ or $u\langle v\rangle \in \lhdeq\gamma_i(\varphi^{t''_i}(\bot))$, for some appropriate numbers t'_i and t''_i. Let t be the maximum of all these numbers. Then,

$$u\langle v\rangle \in \bigcap_{i=1}^{k} \alpha_i\big(\varphi^t(\bot)\big) \cap \bigcap_{i=1}^{m} \lhd\beta_i\big(\varphi^t(\bot)\big) \cap \bigcap_{i=1}^{m'} \lhdeq\gamma_i\big(\varphi^t(\bot)\big) \subseteq \big[\varphi^{t+1}(\bot)\big]_A,$$

as desired. □

4. Normal form

The origin of the normal form for grammars with one-sided contexts developed in this section is the Chomsky normal form for ordinary context-free grammars, in which all rules are of the form $A \to BC$ and $A \to a$. Its extension to conjunctive grammars allows rules $A \to B_1C_1 \,\&\, \ldots \,\&\, B_kC_k$ with multiple conjuncts. The transformation to this normal form is done by first eliminating *empty conjuncts* (also called *null conjuncts*), that is, any rules of the form $A \to \varepsilon \,\&\, \ldots$, and then removing *unit conjuncts*, or rules of the form $A \to B \,\&\, \ldots$.

This normal form and the corresponding transformation shall now be further extended to grammars with contexts.

Definition 5. A grammar with one-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 following two forms.

$$A \to B_1C_1 \,\&\, \ldots \,\&\, B_kC_k \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \lhdeq E_1 \,\&\, \ldots \,\&\, \lhdeq E_n \tag{7a}$$

$$A \to a \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \lhdeq E_1 \,\&\, \ldots \,\&\, \lhdeq E_n \tag{7b}$$

(where $k \geqslant 1$, $m, n \geqslant 0$, $B_i, C_i \in N$, $D_i \in N$, $E_i \in N$, $a \in \Sigma$).

The transformation to this normal form is comprised of three stages. First, there is the *elimination of null conjuncts* (that is, any rules of the form $A \to \varepsilon \,\&\, \ldots$), which is a rather elaborate generalization of the similar procedure for conjunctive grammars. The subsequent *elimination of null contexts* (that is, rules of the form $A \to \lhd\varepsilon \,\&\, \ldots$) removes any checks that the current substring is in the beginning of the entire string. Finally, the elimination of unit conjuncts (any rules $A \to B \,\&\, \ldots$ with $B \in N$) is the same as in the case of conjunctive grammars.

4.1. Null conjuncts

The task of eliminating null conjuncts is formulated in the same way as for conjunctive and ordinary context-free grammars: for any given grammar with contexts, the goal is to construct a grammar without null conjuncts that defines the same language as the original grammar—except for the empty string, which can no longer be generated by the constructed grammar. 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. For an ordinary context-free grammar $G = (\Sigma, N, R, S)$, consider the following sequence of sets of nonterminals, which begins with an empty set, and gradually fills in all symbols generating ε.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

$$\textsc{Nullable}_0(G) = \varnothing$$

$$\textsc{Nullable}_{i+1}(G) = \big\{A \;\big|\; \text{there is a rule } A \to \alpha \in R \text{ that satisfies } \alpha \in \textsc{Nullable}_i^*(G)\big\}$$

Here the Kleene star of $\textsc{Nullable}_i(G)$ gives the set of strings over N that are already known to generate ε. This is an ascending sequence of sets, and its least upper bound, reached in at most $|N|$ steps, is the desired set $\textsc{Nullable}(G) = \{A \in N \mid \varepsilon \in L_G(A)\}$.

For grammars with contexts, as shown in the next example, the same nonterminal symbol may generate the empty string in some contexts, and not generate it in the others.

Example 5. Consider the following grammar, which defines the language $\{a, ac, abc, aabc\}$:

$$S \to aA \mid aaA$$

$$A \to BC$$

$$B \to \varepsilon \,\&\, \lhd D \mid b$$

$$C \to \varepsilon \,\&\, \lhd E \mid c$$

$$D \to a$$

$$E \to a$$

Here B generates ε only in a context described by D, while C generates ε only in a context E (that is, in each case, only in the context a). Hence, A generates ε only in contexts described by both D and E (that is, again, only in the context a). Therefore, the grammar generates the string a by the rule $S \to aA$, but does not generate the string aa, because A no longer generates ε in the left context aa.

Thus, for grammars with contexts, instead of talking about nonterminals generating the empty string, one should consider generation of ε by a nonterminal in a context satisfying a given set of conditions, such as by A in the context $\{D, E\}$, as seen in Example 5.

For simplicity, assume that context operators are applied only to individual nonterminal symbols ($\lhd D$, $\trianglelefteqslant E$); later on, in Lemma 5, it will be shown that there is no loss of generality in this assumption. Then the task is to determine all pairs of the form (U, A), with $A \in N$ and $U \subseteq N$, representing the intuitive idea that A generates ε in a context of the form described by each symbol in U. Returning to Example 5, one should first obtain the pairs $(\{D\}, B)$ and $(\{E\}, C)$ directly from the rules for B and C, then "concatenate" these pairs to find that BC generates ε in the context $\{D, E\}$, and finally infer another pair $(\{D, E\}, A)$ according to the rule $A \to BC$.

This symbolic "concatenation" of pairs is carried out by the following extension of the Kleene star to symbols encumbered with contexts. For any set of pairs $\mathcal{S} \subseteq 2^N \times N$, denote by $\mathcal{S}^\star$ the set of all pairs $(U_1 \cup \dots \cup U_\ell, A_1 \dots A_\ell)$ with $\ell \geqslant 0$ and $(U_i, A_i) \in \mathcal{S}$: in other words, the symbols are concatenated, while their contexts are put together in one set. It is worth noting that for $\ell = 0$, the concatenation of zero pairs yields a pair $(\varnothing, \varepsilon)$, which $\mathcal{S}^\star$ always contains, regardless of $\mathcal{S}$; in particular, $\varnothing^\star = \{(\varnothing, \varepsilon)\}$ (cf. $\varnothing^* = \{\varepsilon\}$ for the Kleene star on languages).

Using this notion, the set of nullable symbols in contexts is obtained as the limit of a sequence of sets, as follows.

Definition 6. Let $G = (\Sigma, N, R, S)$ be a grammar with contexts, in which the context operators are applied only to single nonterminal symbols rather than to concatenations.

Construct the sequence of sets $\textsc{Nullable}_i(G) \subseteq 2^N \times N$, with $i \geqslant 0$, by setting

$$\textsc{Nullable}_0(G) = \varnothing,$$

$$\begin{aligned}\textsc{Nullable}_{i+1}(G) = \big\{\big(\{D_1, \dots, D_m\} \cup \{E_1, \dots, E_n\} \cup U_1 \cup \dots \cup U_k, A\big) \;\big|\; & \text{there exists a rule } A \to \alpha_1 \,\&\, \dots \,\&\, \alpha_k \,\&\, \lhd D_1 \,\&\, \dots \,\&\, \lhd D_m \,\&\, \trianglelefteqslant E_1 \,\&\, \dots \,\&\, \trianglelefteqslant E_n \in R, \\ & \text{and the pairs } (U_1, \alpha_1), \dots, (U_k, \alpha_k) \in \textsc{Nullable}_i^\star(G)\big\}.\end{aligned}$$

Finally, denote its limit by

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

At the first step, in the definition of $\textsc{Nullable}_1(G)$, the pairs $(U_1, \alpha_1), \dots, (U_k, \alpha_k)$ must belong to $\textsc{Nullable}_0^\star(G) = \varnothing^\star = (\varnothing, \varepsilon)$, and accordingly, $\textsc{Nullable}_1(G)$ is the set of all pairs $(\varnothing, A)$ with $A \to \varepsilon \in R$.

In general, the formula defining $\textsc{Nullable}_{i+1}(G)$ expresses the following reasoning. Assume that all base conjuncts $\alpha_1, \dots, \alpha_k$ in some rule are already known to generate the empty string *in some contexts*, and the algorithm should determine a possible set of contexts, in which A generates ε. First, the contexts needed for each conjunct α_i are assembled

together by the star operator, and then are joined as the union $U_1 \cup \cdots \cup U_k$. All context specifications in the chosen rule are also added to the set. Then A generates the empty string in all contexts described by the resulting set.

Example 6. For the grammar in Example 5,

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

and $\textsc{Nullable}(G) = \textsc{Nullable}_2(G)$. The pair $(\{D, E\}, A)$ represents the ability of A to generate the empty string in any context defined by both D and E.

The next lemma states that, indeed, the generation of ε in any context can be determined using the set $\textsc{Nullable}(G)$.

Lemma 4. *Let $G = (\Sigma, N, R, S)$ be a grammar with contexts, let $A \in N$ and $u \in \Sigma^*$. Then, $u\langle\varepsilon\rangle \in L_G(A)$ if and only if the set $\textsc{Nullable}(G)$ contains such a pair $(\{K_1, \ldots, K_t\}, A)$ that $\varepsilon\langle u\rangle \in L_G(K_1), \ldots, \varepsilon\langle u\rangle \in L_G(K_t)$.*

Proof. $\Rightarrow$ Assume that $u\langle\varepsilon\rangle \in L_G(A)$. The existence of a desired pair in $\textsc{Nullable}(G)$ is proved by induction on p, the number of steps used in the deduction of the item $A(u\langle\varepsilon\rangle)$.

Basis. Let $p = 1$. If an item $A(u\langle\varepsilon\rangle)$ is deduced in one step, then it must be obtained by a rule $A \to \varepsilon$. Then the pair $(\varnothing, A)$ is added to $\textsc{Nullable}(G)$ at the first iteration.

Induction step. Let an item $A(u\langle\varepsilon\rangle)$ be deduced in p steps, with the last step using a rule

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \lhd\!\!\!\!\!= E_1 \,\&\, \ldots \,\&\, \lhd\!\!\!\!\!= E_n. \tag{8}$$

For each base conjunct α_i, let $\alpha_i = X_{i,1} \ldots X_{i,\ell_i}$, where $\ell_i \geqslant 0$ and $X_{i,j} \in N$. Then the string $u\langle\varepsilon\rangle$ is in $L_G(X_{i,j})$ for all $j \in \{1, \ldots, \ell_i\}$, and the last step of the deduction is of the form

$$\big\{X_{i,j}\big(u\langle\varepsilon\rangle\big)\big\}_{i,j}, D_1\big(\varepsilon\langle u\rangle\big), \ldots, D_m\big(\varepsilon\langle u\rangle\big), E_1\big(\varepsilon\langle u\rangle\big), \ldots, E_n\big(\varepsilon\langle u\rangle\big) \vdash A\big(u\langle\varepsilon\rangle\big).$$

For each symbol $X_{i,j}$, by the induction hypothesis, there exists such a set $U_{i,j} \subseteq N$, that $(U_{i,j}, X_{i,j}) \in \textsc{Nullable}(G)$ and $\varepsilon\langle u\rangle \in L_G(K)$ for all $K \in U_{i,j}$. Let $h > 0$ be the least such number that all these pairs, for all i and j, appear in $\textsc{Nullable}_h(G)$. Then the pair $(U_{i,1} \cup \cdots \cup U_{i,\ell_i}, \alpha_i)$ is in $\textsc{Nullable}^\star_h(G)$. Denote $U_i = U_{i,1} \cup \cdots \cup U_{i,\ell_i}$ and $U = U_1 \cup \cdots \cup U_k \cup \{D_1, \ldots, D_m, E_1, \ldots, E_n\}$.

The pair (U, A) belongs to $\textsc{Nullable}_{h+1}(G)$ by construction. The second claim is that $\varepsilon\langle u\rangle \in L_G(K)$ for all $K \in U$. Indeed, for symbols in each $U_{i,j}$ this has been proved above, while the remaining symbols in U are from the contexts in the rule (8), and the items $D_i(\varepsilon\langle u\rangle)$ (for $i \in \{1, \ldots, m\}$) and $E_i(\varepsilon\langle u\rangle)$ (for $i \in \{1, \ldots, n\}$) are known to be true.

$\Leftarrow$ Let $U = \{K_1, \ldots, K_t\}$ and $(U, A) \in \textsc{Nullable}(G)$. Then $(U, A) \in \textsc{Nullable}_h(G)$ for some $h \geqslant 0$. The goal is to prove that $u\langle\varepsilon\rangle \in L_G(A)$, which will be done inductively on h.

Basis. If $h = 0$, then $\textsc{Nullable}_0(G) = \varnothing$ and no symbols $A \in N$ satisfy the assumptions.

Induction step. Let $(U, A) \in \textsc{Nullable}_h(G)$ and $\varepsilon\langle u\rangle \in L_G(K)$ for all $K \in U$. According to Definition 6, the set R contains a rule of the form (8), where, for each base conjunct α_i in this rule, there is a pair (U_i, α_i) in $\textsc{Nullable}^\star_{h-1}(G)$, and $U = U_1 \cup \cdots \cup U_k \cup \{D_1, \ldots, D_m, E_1, \ldots, E_n\}$.

For each conjunct α_i, let $\alpha_i = X_{i,1} \ldots X_{i,\ell_i}$ with $\ell_i \geqslant 0$ and $X_{i,1}, \ldots, X_{i,\ell_i} \in \Sigma \cup N$. Then, by the definition of a "star" of $\textsc{Nullable}_{h-1}(G)$, there exist pairs $(U_{i,1}, X_{i,1}), \ldots, (U_{i,\ell_i}, X_{i,\ell_i})$ in $\textsc{Nullable}_{h-1}(G)$, for which $U_{i,1} \cup \cdots \cup U_{i,\ell_i} = U_i$. Therefore, by the induction hypothesis for each of these pairs, $u\langle\varepsilon\rangle \in L_G(X_{i,j})$ for all j, that is, $\vdash_G X_{i,1}(u\langle\varepsilon\rangle), \ldots, X_{i,\ell_i}(u\langle\varepsilon\rangle)$.

This gives all the remaining premises for deducing the membership of $u\langle\varepsilon\rangle$ in $L_G(A)$.

$$X_{1,1}\big(u\langle\varepsilon\rangle\big), \ldots, X_{k,\ell_k}\big(u\langle\varepsilon\rangle\big), D_1\big(\varepsilon\langle u\rangle\big), \ldots, D_m\big(\varepsilon\langle u\rangle\big), E_1\big(\varepsilon\langle u\rangle\big), \ldots, E_n\big(\varepsilon\langle u\rangle\big) \vdash_G A\big(u\langle\varepsilon\rangle\big)$$

The premises $D_i(\varepsilon\langle u\rangle)$ and $E_i(\varepsilon\langle u\rangle)$ are known to be true, because $\varepsilon\langle u\rangle \in L_G(K)$ for all $K \in U$ by the assumption. □

Now, once all cases of generation of the empty string in a grammar have been enumerated in the set $\textsc{Nullable}(G)$, this information can be used to rebuild the grammar without ever generating ε. For a context-free grammar, this is done by creating variants of the existing rules, in which the nullable symbols are omitted: for instance, if there is a rule $A \to BC$, and B generates ε, then the new grammar gets another rule $A \to C$. For a grammar with contexts, the generation of an empty string in the original grammar may depend on the contexts, which is reflected in the set $\textsc{Nullable}(G)$. Once a nullable symbol is omitted in a variant of an existing rule, the same logical dependence is ensured in the new rule by context operators.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

Example 7. Consider the grammar from Example 5. Once both null rules are removed, the effect of these rules on the generation of longer strings is replicated as follows.

The nonterminals B and C, as shown in Example 6, can generate the empty string only in the left contexts of the form D and E, respectively. For the rule $A \to BC$, when the symbol B is omitted, the resulting extra rule should be $A \to C \,\&\, \lhd D$ (rather than just $A \to C$), which inherits the context dependence from B. Similarly, C can be omitted in $A \to BC$ by introducing a new rule $A \to B \,\&\, \trianglelefteq E$. Note that the latter rule uses an extended context operator, because it refers to the left context of the omitted symbol C, which includes the substring generated by B.

The nonterminal A itself generates the empty string, but only in the contexts described by both D and E. Thus, the generation of ε by A in the rule $S \to aA$ can be handled by adding an extra rule $S \to a \,\&\, \trianglelefteq D \,\&\, \trianglelefteq E$ (rather than $S \to a$), which again preserves the both context dependencies. The rule $S \to aaA$ is processed in the same way.

The resulting set of rules of the new grammar is given below.

$$S \to aA \mid a \,\&\, \trianglelefteq D \,\&\, \trianglelefteq E \mid aaA \mid aa \,\&\, \trianglelefteq D \,\&\, \trianglelefteq E$$

$$A \to BC \mid B \,\&\, \trianglelefteq E \mid C \,\&\, \lhd D$$

$$B \to b$$

$$C \to c$$

$$D \to a$$

$$E \to a$$

Given an arbitrary grammar with contexts, it is convenient to begin the elimination of null conjuncts with simplifying the form of the rules. After the following pre-processing stage, all concatenations are expressed in separate rules $A \to BC$ and all context operators are applied to individual nonterminals.

Lemma 5. *For every grammar with contexts $G_0 = (\Sigma, N_0, R_0, S_0)$, there exists and can be effectively constructed another grammar with contexts $G = (\Sigma, N, R, S)$ generating the same language, in which all rules are of the following form.*

$$A \to BC \qquad (A, B, C \in N) \tag{9a}$$

$$A \to a \qquad (a \in \Sigma) \tag{9b}$$

$$A \to B_1 \,\&\, \ldots \,\&\, B_k \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n \qquad (k \geqslant 1,\ m, n \geqslant 0,\ A, B_i, D_i, E_i \in N) \tag{9c}$$

$$A \to \varepsilon \tag{9d}$$

If the original grammar is unambiguous, then so is the resulting grammar.

Proof. Each conjunct in the given grammar $G_0 = (\Sigma, N_0, R_0, S_0)$ is of the form α, $\lhd\alpha$ or $\trianglelefteq\alpha$, where $\alpha \in (\Sigma \cup N_0)^*$. First, construct a grammar G_1 as follows. Each context operator $\lhd\alpha$ or $\trianglelefteq\alpha$, with $\alpha \in \Sigma$ or $|\alpha| > 1$, is replaced by $\lhd Y_\alpha$ or $\trianglelefteq Y_\alpha$, respectively, where Y_α is a new nonterminal with a unique rule $Y_\alpha \to \alpha$. Then, each conjunct α, with $|\alpha| > 2$, is shortened by introducing new nonterminals: for instance, $\alpha = BaC$ may become BX_{aC}, with a rule $X_{aC} \to aC$. Denote the new grammar by G_2. Finally, every instance of a symbol $a \in \Sigma$ in a conjunct α with $|\alpha| = 2$ is replaced with a new nonterminal X_a with a unique rule $X_a \to a$, resulting in a grammar G of the desired form.

Unambiguity of choice of a rule is preserved at every stage of the transformation. Indeed, every modified rule for an old nonterminal generates the same language as its prototype rule, and the alternatives remain disjoint. Each new nonterminal has a unique rule, which rules out unambiguity of choice.

No new concatenations are introduced in the transformation of G_0 to G_1, and they remain unambiguous. In G_2, each new concatenation is of the form $s_1 \cdot X_{s_2 \ldots s_\ell}$, and if it is ambiguous, then so is the original factorization $s_1 \cdot s_2 \cdot \ldots \cdot s_\ell$, contradicting the assumption. No new concatenations are introduced in G. □

The following transformation eliminates all null conjuncts from a given grammar with contexts.

Construction 1. Let $G = (\Sigma, N, R, S)$ be a grammar with left contexts with all rules of the form (9a)–(9d), as in Lemma 5. Consider the set $\textsc{Nullable}(G)$ and construct a new grammar with contexts $G' = (\Sigma, N \cup \{W\}, R', S)$, where R' contains the following rules.

1. The new nonterminal W generates all non-empty strings in all contexts, using the following rules:

$$W \to a \qquad (a \in \Sigma)$$

$$W \to aW \qquad (a \in \Sigma)$$

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

2. Every rule of the form $A \to a$ in R is preserved in R':

$$A \to a \tag{10}$$

3. For every rule $A \to B_1 \& \ldots \& B_k \& \lhd D_1 \& \ldots \& \lhd D_m \& \lhdeq E_1 \& \ldots \& \lhdeq E_n$ in R, the set R' contains the same rule

$$A \to B_1 \& \ldots \& B_k \& \lhd D_1 \& \ldots \& \lhd D_m \& \lhdeq E_1 \& \ldots \& \lhdeq E_n, \tag{11a}$$

and, possibly, the following extra rule:

$$A \to B_1 \& \ldots \& B_k \& E_1 \& \ldots \& E_n \& \lhd\varepsilon, \quad \text{if } m \geqslant 1 \text{ and } \varepsilon\langle\varepsilon\rangle \in L_G(D_1) \cap \cdots \cap L_G(D_m). \tag{11b}$$

The latter rule handles the case of a null left context, when the substring being defined is a prefix of the whole string. If there is at least one proper left context operator $\lhd D_i$ in the original rule, then the same operator in the new grammar would no longer admit null context, and hence a new rule without these operators has to be introduced. Note that in a null context, an extended left context $\lhdeq E_i$ has the same effect as a base conjunct E_i. Also note that the condition $\varepsilon\langle\varepsilon\rangle \in L_G(D_i)$ can be checked by listing all deductions among the items $X(\varepsilon\langle\varepsilon\rangle)$, which depend only on each other.

4. Every rule of the form $A \to BC$ in R is added to R',

$$A \to BC \tag{12a}$$

along with the following extra rules.

$$A \to B \& \lhdeq K_1 \& \ldots \& \lhdeq K_t \qquad \big(\text{for all } \big(\{K_1, \ldots, K_t\}, C\big) \in \textsc{Nullable}(G)\big) \tag{12b}$$

$$A \to C \& \lhd K_1 \& \ldots \& \lhd K_t \& \lhd W \quad \big(\text{for all } \big(\{K_1, \ldots, K_t\}, B\big) \in \textsc{Nullable}(G)\big) \tag{12c}$$

In each case, a nullable nonterminal is omitted, and the condition of its being nullable is accordingly included in the rule. In the first case, the substring generated by B is a part of the omitted C's context, and hence extended context operators are used ($\lhdeq K_i$). In the second case, once B is omitted, its context is referenced by proper context operators ($\lhd K_i$); one more context operator ($\lhd W$) restricts the applicability of this rule to a non-empty left context. The case when B is omitted in an empty left context is handled in yet another rule

$$A \to C \& \lhd\varepsilon, \tag{12d}$$

which is added to the grammar if there exists such a set $\{K_1, \ldots, K_t\} \subseteq N$, that $(\{K_1, \ldots, K_t\}, B) \in \textsc{Nullable}(G)$ and $\varepsilon\langle\varepsilon\rangle \in L_G(K_1) \cap \cdots \cap L_G(K_t)$.

Thus, among the two rules for a nullable B, the former rule (12c) only generates strings in non-empty left contexts, whereas the latter rule (12d) only applies in the empty left context. This is essential for the construction to preserve unambiguity.

None of the rules (9d) generating the empty string are included in G'.

Correctness Statement 1. *Let $G = (\Sigma, N, R, S)$ be a grammar with left contexts and let $G' = (\Sigma, N \cup \{W\}, R', S)$ be the grammar obtained by Construction 1. Then $L_{G'}(A) = L_G(A) \setminus \Sigma^*\langle\varepsilon\rangle$ for every $A \in N$; in particular, $L(G') = L(G) \setminus \{\varepsilon\}$. Furthermore, if G is unambiguous, then so is G'.*

The first claim is that whenever a non-empty string in any context is generated by a symbol in the original grammar, it is also generated by that symbol in the new grammar.

Lemma 6. *Let a grammar $G = (\Sigma, N, R, S)$ be transformed to $G' = (\Sigma, N \cup \{W\}, R', S)$ according to Construction 1. Then, if an item $A(u\langle v\rangle)$, with $A \in N$, $u \in \Sigma^*$ and $v \in \Sigma^+$, is deduced in G, then it is also deduced in G'.*

Proof. The proof is carried out by induction on p, the number of steps in the deduction of $A(u\langle v\rangle)$ in G.

Basis. Let $p = 1$ and consider an item $A(u\langle v\rangle)$ deduced in G by a rule $A \to a$. Then $v = a$ and the last step of the deduction is $a(u\langle a\rangle) \vdash_G A(u\langle a\rangle)$. By the construction, the grammar G' also has the rule $A \to a$, and the same deduction can be carried out in G': $a(u\langle a\rangle) \vdash_{G'} A(u\langle a\rangle)$.

Induction step. Consider the rule in G used at the last step of deduction of $A(u\langle v\rangle)$.

If this is a rule $A \to BC$, then the last step of the deduction takes the form $B(u\langle v_1\rangle), C(uv_1\langle v_2\rangle) \vdash_G A(u\langle v\rangle)$, for some partition $v = v_1v_2$, and each of the premises is deduced in fewer than p steps. If both v_1 and v_2 are non-empty, then, by the induction hypothesis, both items $B(u\langle v_1\rangle)$ and $C(uv_1\langle v_2\rangle)$ are deducible in G', and therefore the item $A(u\langle v\rangle)$ can be obtained in G' by the rule (12a): $B(u\langle v_1\rangle), C(uv_1\langle v_2\rangle) \vdash_{G'} A(u\langle v\rangle)$.

Let now either v_1 or v_2 be empty (both cannot be empty, because $v \neq \varepsilon$).

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

- Let $v_1 = v$ and $v_2 = \varepsilon$. Then, the last step of the deduction of $A(u\langle v\rangle)$ is $B(u\langle v\rangle), C(uv\langle\varepsilon\rangle) \vdash_G A(u\langle v\rangle)$. The item $B(u\langle v\rangle)$ can be deduced in G' by the induction hypothesis. Since $uv\langle\varepsilon\rangle \in L_G(C)$, by Lemma 4, there exist such symbols $K_1, \ldots, K_t \in N$, that $(\{K_1, \ldots, K_t\}, C) \in \text{NULLABLE}(G)$ and all items $K_i(\varepsilon\langle uv\rangle)$ can be deduced in the grammar G. Since $uv \neq \varepsilon$, the induction hypothesis is applicable to these items, and it asserts that each $K_i(\varepsilon\langle uv\rangle)$ can be deduced in the grammar G' as well. Then the item $A(u\langle v\rangle)$ is deduced in G' by the rule (12b), which exists because of the pair $(\{K_1, \ldots, K_t\}, C)$ in $\text{NULLABLE}(G)$; the deduction proceeds as follows: $B(u\langle v\rangle), K_1(\varepsilon\langle uv\rangle), \ldots, K_t(\varepsilon\langle uv\rangle) \vdash_{G'} A(u\langle v\rangle)$.
- If $v_1 = \varepsilon$ and $v_2 = v$, then the last step of deduction of the item $A(u\langle v\rangle)$ in G is $B(u\langle\varepsilon\rangle), C(u\langle v\rangle) \vdash_G A(u\langle v\rangle)$. By the induction hypothesis, the item $C(u\langle v\rangle)$ can be deduced in G'. For $u\langle\varepsilon\rangle \in L_G(B)$, by Lemma 4, there exist $K_1, \ldots, K_t \in N$, such that $(\{K_1, \ldots, K_t\}, B) \in \text{NULLABLE}(G)$ and all items $K_i(\varepsilon\langle u\rangle)$ can be deduced in the grammar G.
  If $u \neq \varepsilon$, then each item $K_i(\varepsilon\langle u\rangle)$ can be deduced in the grammar G' by the induction hypothesis. The string $\varepsilon\langle u\rangle$ is also in $L_{G'}(W)$, and thus the item $A(u\langle v\rangle)$ can be deduced in G' by the rule (12c): $C(u\langle v\rangle), K_1(\varepsilon\langle u\rangle), \ldots, K_t(\varepsilon\langle u\rangle), W(\varepsilon\langle u\rangle) \vdash_{G'} A(u\langle v\rangle)$. Note that the rule (12c) is in G' for the pair $(\{K_1, \ldots, K_t\}, B)$ in $\text{NULLABLE}(G)$.
  Let $u = \varepsilon$. Then $\varepsilon\langle\varepsilon\rangle \in L_G(K_i)$, for all $i \in \{1, \ldots, t\}$, and therefore the grammar G' has a rule (12d). The item $A(\varepsilon\langle v\rangle)$ can be deduced by this rule as $C(\varepsilon\langle v\rangle) \vdash_G A(\varepsilon\langle v\rangle)$.

Consider now the other case, when the last step of the deduction of the item $A(u\langle v\rangle)$ uses the rule $A \to B_1 \& \ldots \& B_k \& \lhd D_1 \& \ldots \& \lhd D_m \& \leqslant E_1 \& \ldots \& \leqslant E_n$. That is, the deduction is $B_1(u\langle v\rangle), \ldots, B_k(u\langle v\rangle), D_1(\varepsilon\langle u\rangle), \ldots, D_m(\varepsilon\langle u\rangle), E_1(\varepsilon\langle uv\rangle), \ldots, E_n(\varepsilon\langle uv\rangle) \vdash_G A(u\langle v\rangle)$.

If $u \neq \varepsilon$, then, by the induction hypothesis, each of the premises can be deduced in the grammar G'. This is sufficient to deduce $A(u\langle v\rangle)$ in the grammar G' by the rule (11a).

Let $u = \varepsilon$. The items $B_i(\varepsilon\langle v\rangle)$ for all $i \in \{1, \ldots, k\}$ and $E_i(\varepsilon\langle v\rangle)$ for all $i \in \{1, \ldots, n\}$ can still be deduced in G' by the induction hypothesis. Since the items $D_1(\varepsilon\langle\varepsilon\rangle), \ldots, D_m(\varepsilon\langle\varepsilon\rangle)$ are deducible in G, the grammar G' has a rule (11b), by which the desired item $A(u\langle v\rangle)$ is deduced in G': $B_1(\varepsilon\langle v\rangle), \ldots, B_k(\varepsilon\langle v\rangle), D_1(\varepsilon\langle\varepsilon\rangle), \ldots, D_m(\varepsilon\langle\varepsilon\rangle), E_1(\varepsilon\langle v\rangle), \ldots, E_n(\varepsilon\langle v\rangle) \vdash_G A(\varepsilon\langle v\rangle)$. □

The converse statement is that whenever a string is generated by some symbol in the new grammar, the same string must be generated in the original grammar.

Lemma 7. *Let $G = (\Sigma, N, R, S)$ be a grammar with left contexts and let $G' = (\Sigma, N \cup \{W\}, R', S)$ be the grammar obtained by Construction 1. Then, if an item $A(u\langle v\rangle)$ can be deduced in the grammar G', it can also be deduced in G, and $v \neq \varepsilon$.*

Moreover, consider the rule used at the last step of deduction of the item $A(u\langle v\rangle)$ in G'. Then,

I. *if $A(u\langle v\rangle)$ is deduced in G' by a rule of the form (10), then it is deduced in G by the (same) rule $A \to a$;*
II. *if $A(u\langle v\rangle)$ is deduced in G' by a rule (11a) or (11b), then it is deduced in G by the rule $A \to B_1 \& \ldots \& B_k \& \lhd D_1 \& \ldots \& \lhd D_m \& \leqslant E_1 \& \ldots \& \leqslant E_n$;*
III. *if $A(u\langle v\rangle)$ is deduced in G' by a rule (12a), (12b), (12c), or (12d), then it is deduced in G by the rule $A \to BC$, and furthermore,*
   (a) *if it is deduced by a rule (12a), and there is such a partition $v = v_1v_2$ with $v_1, v_2 \neq \varepsilon$, that $u\langle v_1\rangle \in L_{G'}(B)$ and $uv_1\langle v_2\rangle \in L_{G'}(C)$, then $u\langle v_1\rangle \in L_G(B)$ and $uv_1\langle v_2\rangle \in L_G(C)$;*
   (b) *if it is deduced by a rule (12b), then $u\langle v\rangle \in L_G(B)$ and $uv\langle\varepsilon\rangle \in L_G(C)$;*
   (c) *if the rule is (12c), then $u \neq \varepsilon$, $u\langle\varepsilon\rangle \in L_G(B)$ and $u\langle v\rangle \in L_G(C)$;*
   (d) *and finally, if the rule is (12d), then $u = \varepsilon$, $u\langle\varepsilon\rangle \in L_G(B)$ and $u\langle v\rangle \in L_G(C)$.*

The latter part of the statement (I–III) refers to the rule used at the last step of deduction of the item $A(u\langle v\rangle)$: if a rule r' is used to deduce it in the new grammar, then this rule was constructed on the basis of some rule r in the original grammar, and the derivation of $A(u\langle v\rangle)$ in the original grammar (constructed in the lemma) uses this rule r.

Proof. The proof is by induction on p, the number of steps used to deduce the item $A(u\langle v\rangle)$ in the grammar G'.

Basis. Let $p = 1$ and let $A(u\langle v\rangle)$ be deduced in G' by the rule $A \to a$. Then $v = a \in \Sigma$ and the desired item is obtained from an axiom: $a(u\langle a\rangle) \vdash_{G'} A(u\langle a\rangle)$. The same deduction can be carried out in G by the same rule $A \to a \in R$.

Induction step. Consider a deduction of an item $A(u\langle v\rangle)$ in G', and let r' be the rule used at the last step of this deduction. Let r be the rule in G, from which r' is obtained in Construction 1. The argument splits into cases depending on the form of r'.

- If the rule r' is of the form (11a), then the last step of deduction of $A(u\langle v\rangle)$ in G' is $B_1(u\langle v\rangle), \ldots, B_k(u\langle v\rangle), D_1(\varepsilon\langle u\rangle), \ldots, D_m(\varepsilon\langle u\rangle), E_1(\varepsilon\langle uv\rangle), \ldots, E_n(\varepsilon\langle uv\rangle) \vdash_{G'} A(u\langle v\rangle)$. By the induction hypothesis, each of the premises can also be deduced in the grammar G, and this allows repeating the last step of deduction in G using the rule r, which is the same as r'.
  The string v is non-empty, because this is ensured by the induction hypothesis for $B_1(u\langle v\rangle)$. The argument for the non-emptiness of v is the same in all subsequent cases.
- Let r' be of the form (11b). Then $u = \varepsilon$ and the last step of the deduction is $B_1(\varepsilon\langle v\rangle), \ldots, B_k(\varepsilon\langle v\rangle), E_1(\varepsilon\langle v\rangle), \ldots, E_n(\varepsilon\langle v\rangle) \vdash_{G'} A(\varepsilon\langle v\rangle)$. By the induction hypothesis, each of the items $B_i(\varepsilon\langle v\rangle)$ (with $i \in \{1, \ldots, k\}$) and $E_i(\varepsilon\langle v\rangle)$ (with

$i \in \{1, \ldots, n\}$) can be deduced in the grammar G. The rule r in G is of the form $A \to B_1 \,\&\, \ldots \,\&\, B_k \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n$, and the condition of existence of r' is that $\varepsilon\langle\varepsilon\rangle \in L_G(D_i)$, for $i \in \{1, \ldots, m\}$. Now the item $A(u\langle v\rangle)$ is deduced in G as follows:

$$\big\{B_i\big(\varepsilon\langle v\rangle\big)\big\}_{i\in\{1,\ldots,k\}}, \big\{D_i\big(\varepsilon\langle\varepsilon\rangle\big)\big\}_{i\in\{1,\ldots,m\}}, \big\{E_i\big(\varepsilon\langle v\rangle\big)\big\}_{i\in\{1,\ldots,n\}} \vdash_G A\big(\varepsilon\langle v\rangle\big).$$

- Let r' be of the form (12a), that is, $A \to BC$; the rule r in G is of the same form. The last step of deduction of $A(u\langle v\rangle)$ in the grammar G' is $B(u\langle v_1\rangle), C(uv_1\langle v_2\rangle) \vdash_{G'} A(u\langle v\rangle)$, for some partition $v = v_1v_2$. By the induction hypothesis, both items $B(u\langle v_1\rangle)$ and $C(uv_1\langle v_2\rangle)$ can be deduced in the grammar G. This deduction in G can be continued with the same last step using the rule r: $B(u\langle v_1\rangle), C(uv_1\langle v_2\rangle) \vdash_G A(u\langle v\rangle)$.
  Here the non-emptiness of v is given by the induction hypothesis for $B(u\langle v_1\rangle)$, which asserts that already v_1 is non-empty.
- Assume that r' is of the form (12b), that is, $A \to B \,\&\, \trianglelefteq K_1 \,\&\, \ldots \,\&\, \trianglelefteq K_t$, obtained from a rule $A \to BC$ and a pair $(\{K_1, \ldots, K_t\}, C) \in \text{NULLABLE}(G)$. Then the item $A(u\langle v\rangle)$ is deduced in G' out of the premises $B(u\langle v\rangle)$, $K_1(\varepsilon\langle uv\rangle), \ldots, K_t(\varepsilon\langle uv\rangle)$. By the induction hypothesis, the item $B(u\langle v\rangle)$ can be deduced in the grammar G. The item $C(uv\langle\varepsilon\rangle)$ can be deduced in G by Lemma 4, applied to the pair $(\{K_1, \ldots, K_t\}, C)$. Using these two premises, one can deduce the item $A(u\langle v\rangle)$ in G by the rule $A \to BC$, as follows: $B(u\langle v\rangle), C(uv\langle\varepsilon\rangle) \vdash_G A(u\langle v\rangle)$.
- Let r' be of the form (12c), that is, $A \to C \,\&\, \lhd K_1 \,\&\, \ldots \,\&\, \lhd K_t \,\&\, \lhd W$, obtained from a rule $A \to BC$ in G and a pair $(\{K_1, \ldots, K_t\}, B) \in \text{NULLABLE}(G)$. The last step of deduction of $A(u\langle v\rangle)$ in G' is then $C(u\langle v\rangle), K_1(\varepsilon\langle u\rangle), \ldots, K_t(\varepsilon\langle u\rangle)$, $W(\varepsilon\langle u\rangle) \vdash_{G'} A(u\langle v\rangle)$. By the induction hypothesis, $\vdash_G C(u\langle v\rangle)$. The string u must be non-empty, because $\varepsilon\langle u\rangle \in L_{G'}(W)$ (this fulfils III(c) in the statement of the lemma). By Lemma 4 for the pair $(\{K_1, \ldots, K_t\}, B) \in \text{NULLABLE}(G)$, the string $u\langle\varepsilon\rangle$ is in $L_G(B)$. Then the desired item is deduced in G as $B(u\langle\varepsilon\rangle), C(u\langle v\rangle) \vdash_G A(u\langle v\rangle)$ by the rule $A \to BC$.
- The last case is that the rule r' is (12d), that is, $A \to C \,\&\, \lhd\varepsilon$, obtained from the rule $A \to BC$ in G and a pair $(\{K_1, \ldots, K_t\}, B)$ in $\text{NULLABLE}(G)$, such that $\varepsilon\langle\varepsilon\rangle \in L_G(K_i)$ for all $i \in \{1, \ldots, t\}$. The item $A(u\langle v\rangle)$ is deduced as $C(\varepsilon\langle v\rangle) \vdash_{G'} A(\varepsilon\langle v\rangle)$, and for this deduction to be possible, u must be empty, due to the empty context operator ($\lhd\varepsilon$). By the induction hypothesis, the item $C(\varepsilon\langle v\rangle)$ can be deduced in G. By Lemma 4, applied to the pair $(\{K_1, \ldots, K_t\}, B)$, the item $B(\varepsilon\langle\varepsilon\rangle)$ is also deduced in G. This allows the deduction $B(\varepsilon\langle\varepsilon\rangle), C(\varepsilon\langle v\rangle) \vdash_G A(\varepsilon\langle v\rangle)$ by the rule $A \to BC$. □

An important property of Construction 1 is that it preserves unambiguity of the grammar: if the original grammar is unambiguous, then so is the resulting grammar.

Lemma 8. *Let $G = (\Sigma, N, R, S)$ be an unambiguous grammar with contexts and let $G' = (\Sigma, N \cup \{W\}, R', S)$ be the grammar obtained in Construction 1. Then G' is unambiguous as well.*

Proof. All concatenations in G' are unambiguous, because they are used only in rules of the form $A \to BC$, which are the same as in G. Then, if some string $u\langle v\rangle$ is generated by such a rule in two different ways, as $u\langle v\rangle = u\langle v_1\rangle \cdot uv_1\langle v_2\rangle = u'\langle v'_1\rangle \cdot u'v'_1\langle v'_2\rangle$, then, by Lemma 7 (part III(a)), the same string $u\langle v\rangle$ is generated in two distinct ways in the grammar G as well. This is a contradiction to the unambiguity of G.

To see that the choice of a rule in the new grammar is unambiguous, suppose that some string $u\langle v\rangle$ is generated in the grammar G' by two distinct rules for the same symbol $A \in N$.

First, consider the case when the string $u\langle v\rangle$ is generated in G' by any two rules of the form (10)–(12d), which were created from two different rules in R. Then, by Lemma 7, each of the latter two rules generates the string $u\langle v\rangle$ in G, and this contradicts the assumption that G is unambiguous.

Consider the other possibility, when the string $u\langle v\rangle$ is generated in the grammar G' by two rules, and both of these rules were added to R' by processing a single rule from R. If this single rule is $A \to B_1 \,\&\, \ldots \,\&\, B_k \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \trianglelefteq E_1 \,\&\, \ldots \,\&\, \trianglelefteq E_n$, and it resulted in multiple rules in R', then these must be the two rules (11a) and (11b). By the construction, this can only happen if $m \geqslant 1$. Since G' does not have any null conjuncts, the former rule (11a) may only generate a string with a non-empty context, whereas the latter rule (11b) requires an empty context. Hence, no string can be deduced in G' by both rules (11a), (11b) at the same time.

The only remaining possibility is that $u\langle v\rangle$ is generated in G' by two distinct rules (12a)–(12d) obtained from a single rule $A \to BC$ in G. For each rule of the form (12a)–(12d), consider Lemma 7 and the corresponding condition III(a)–III(d). In all four cases, the string $u\langle v\rangle$ is defined in the grammar G as a concatenation $u\langle v\rangle = u\langle v_1\rangle \cdot uv_1\langle v_2\rangle$, where $u\langle v_1\rangle \in L_G(B)$ and $uv_1\langle v_2\rangle \in L_G(C)$. The four cases are distinguished by the emptiness status of v_1, v_2 and u: both v_1 and v_2 are non-empty in the first case (12a), $v_1 = v$ and $v_2 = \varepsilon$ in the second case (12b), while the remaining two cases both have $v_1 = \varepsilon$ and $v_2 = v$, and differ by having $u \neq \varepsilon$ in the third case (12c) and $u = \varepsilon$ in the fourth case (12d).

Assuming that $u\langle v\rangle$ is generated in G' by two rules of two different types (12a)–(12d), this yields two distinct partitions of $u\langle v\rangle$ into $u\langle v_1\rangle \in L_G(B)$ and $uv_1\langle v_2\rangle \in L_G(C)$. Thus, the concatenation BC is ambiguous in G, witnessed by the string $u\langle v\rangle$, and this contradicts the assumption.

Let now the two rules be of the same type; that is, let the string $u\langle v\rangle$ be generated either by two distinct rules of the form (12b) corresponding to different elements of $\text{NULLABLE}(G)$, or, similarly, by two distinct rules of the form (12c).

Intuitively, any two different elements of $\textsc{Nullable}(G)$ correspond to the generation of ε in two different ways, which should constitute ambiguity. The following claim formalizes this intuition.

Claim 8.1. *Let $G = (\Sigma, N, R, S)$ be a grammar with contexts, let $A \in N$ and $w \in \Sigma^*$, and assume that there exist two distinct sets $U, U' \subseteq N$ with $(U, A), (U', A) \in \textsc{Nullable}(G)$ and $\varepsilon\langle w\rangle \in L_G(K)$ for each $K \in U \cup U'$. Then the grammar is ambiguous.*

Before proving the claim, consider how it can be used to finish the proof of the lemma. There are two cases to consider.

- *The string $u\langle v\rangle$ is generated by two different rules of the form (12b).* These rules were created for two distinct pairs (U, C) and (U', C) in $\textsc{Nullable}(G)$, where $U, U' \subseteq N$ and $U \neq U'$. Since $u\langle v\rangle$ is generated by each of these rules, $\varepsilon\langle uv\rangle \in L_G(K)$ for all $K \in U \cup U'$. Then Claim 8.1 is applicable to the pairs $(U, C), (U', C) \in \textsc{Nullable}(G)$ and the string $w = uv$. This establishes the ambiguity of the grammar G, which contradicts the assumptions.
- *The string $u\langle v\rangle$ is generated by two distinct rules of the form (12c).* Similarly to the previous case, since the rule exists, there are pairs $(U, B), (U', B)$ in $\textsc{Nullable}(G)$, with $U, U' \subseteq N$ and $U \neq U'$, and $\varepsilon\langle u\rangle \in L_G(K)$ for all $K \in U \cup U'$. Thus, Claim 8.1 is applicable, and it contradicts the unambiguity of G.

The above case analysis completes the proof of Lemma 8, and so it remains to prove Claim 8.1.

For the given pairs (U, A), (U', A), let $h \geqslant 1$ be the least number, for which $(U, A), (U', A) \in \textsc{Nullable}_h(G)$. Of all pairs (U, A), (U', A) satisfying the statement of the claim, choose those with the minimal value of h. That is, let (U, A) and (U', A) be two distinct pairs in $\textsc{Nullable}_h(G)$, for some $h \geqslant 1$, and assume that $\varepsilon\langle w\rangle \in L_G(K)$ for all $K \in U \cup U'$. It shall be proved that the choice of a rule for A is ambiguous for the string $w\langle\varepsilon\rangle$.

For the pair (U, A) in $\textsc{Nullable}_h(G)$, by the definition of this set, there exists a rule

$$A \to \alpha_1 \,\&\, \ldots \,\&\, \alpha_k \,\&\, \lhd D_1 \,\&\, \ldots \,\&\, \lhd D_m \,\&\, \trianglelefteqslant E_1 \,\&\, \ldots \,\&\, \trianglelefteqslant E_n, \tag{13}$$

and pairs $(U_1, \alpha_1), \ldots, (U_k, \alpha_k)$ in $\textsc{Nullable}^{\star}_{h-1}(G)$, for which $U = \{D_1, \ldots, D_m, E_1, \ldots, E_n\} \cup U_1 \cup \cdots \cup U_k$. Similarly, for the other pair $(U', A) \in \textsc{Nullable}_h(G)$, there is a rule

$$A \to \alpha'_1 \,\&\, \ldots \,\&\, \alpha'_{k'} \,\&\, \lhd D'_1 \,\&\, \ldots \,\&\, \lhd D'_{m'} \,\&\, \trianglelefteqslant E'_1 \,\&\, \ldots \,\&\, \trianglelefteqslant E'_{n'}, \tag{13′}$$

and pairs $(U'_1, \alpha'_1), \ldots, (U'_{k'}, \alpha'_{k'}) \in \textsc{Nullable}^{\star}_{h-1}(G)$ satisfying $U' = \{D'_1, \ldots, D'_{m'}, E'_1, \ldots, E'_{n'}\} \cup U'_1 \cup \cdots \cup U'_{k'}$.

First, suppose that (13) and (13′) are the same rule. Then, $\{D_1, \ldots, D_m, E_1, \ldots, E_n\} = \{D'_1, \ldots, D'_{m'}, E'_1, \ldots, E'_{n'}\}$, and therefore $U \neq U'$ implies $U_1 \cup \cdots \cup U_k \neq U'_1 \cup \cdots \cup U'_k$. For these unions to be different, they should differ in some position $i \in \{1, \ldots, k\}$, for which $U_i \neq U'_i$, and both pairs (U_i, α_i), (U'_i, α_i), with the same α_i, belong to $\textsc{Nullable}^{\star}_{h-1}(G)$.

Let $\alpha_i = X_{i,1} \ldots X_{i,\ell_i}$, with $\ell_i \geqslant 0$ and $X_{i,1}, \ldots, X_{i,\ell_i} \in N$. By the definition of the "star", since the pair $(U_i, X_{i,1} \ldots X_{i,\ell_i})$ is in $\textsc{Nullable}^{\star}_{h-1}(G)$, it is obtained by combining some pairs $(U_{i,1}, X_{i,1}), \ldots, (U_{i,\ell_i}, X_{i,\ell_i})$ in $\textsc{Nullable}_{h-1}(G)$, which satisfy $U_{i,1} \cup \cdots \cup U_{i,\ell_i} = U_i$. Similarly, for the pair $(U'_i, X_{i,1} \ldots X_{i,\ell_i})$, there are corresponding pairs $(U'_{i,1}, X_{i,1}), \ldots, (U'_{i,\ell_i}, X_{i,\ell_i}) \in \textsc{Nullable}_{h-1}(G)$, and $U'_{i,1} \cup \cdots \cup U'_{i,\ell_i} = U'_i$. Since $U_i \neq U'_i$, these two unions should again differ in some j-th position, that is, there are two distinct pairs $(U_{i,j}, X_{i,j})$ and $(U'_{i,j}, X_{i,j})$ in $\textsc{Nullable}_{h-1}(G)$. Furthermore, as $U_{i,j} \subseteq U$ and $U'_{i,j} \subseteq U'$, it is known that $\varepsilon\langle w\rangle \in L_G(K)$ for all $K \in U_{i,j} \cup U'_{i,j}$. Thus, these two new pairs satisfy all conditions of this claim, for a smaller number of iterations than h. This contradicts the assumption made in the beginning of the proof, and thus demonstrates that the rules (13) and (13′) are distinct.

Now the goal is to show that both rules (13) and (13′) can be used to deduce the item $A(w\langle\varepsilon\rangle)$, which will form the desired ambiguity. Consider, for instance, the first rule (13). For each conjunct α_i in this rule, let $\alpha_i = X_{i,1} \ldots X_{i,\ell_i}$. Since $(U_i, \alpha_i) \in \textsc{Nullable}^{\star}_{h-1}(G)$, by definition of the "star", there are pairs $(U_{i,1}, X_{i,1}), \ldots, (U_{i,\ell_i}, X_{i,\ell_i})$ in $\textsc{Nullable}_{h-1}(G)$, which satisfy $U_{i,1} \cup \cdots \cup U_{i,\ell_i} = U_i$. It is known that $\varepsilon\langle w\rangle$ is in $L_G(K)$ for all $K \in U_{i,j}$. Applying Lemma 4 to every pair $(U_{i,j}, X_{i,j})$ with $j \in \{1, \ldots, \ell_i\}$ gives that the string $w\langle\varepsilon\rangle$ is in $L_G(X_{i,j})$. It is also known that $\varepsilon\langle w\rangle \in L_G(D_i)$ for all i, and $\varepsilon\langle w\rangle \in L_G(E_i)$ for all i, because $D_i, E_i \in U$. Then the desired item $A(w\langle\varepsilon\rangle)$ can be deduced by the rule (13) as follows.

$$\big\{X_{i,j}\big(w\langle\varepsilon\rangle\big)\big\}_{i,j}, \big\{D_i\big(\varepsilon\langle w\rangle\big)\big\}_i, \big\{E_i\big(\varepsilon\langle w\rangle\big)\big\}_i \vdash_G A\big(w\langle\varepsilon\rangle\big)$$

By the same argument as above, the string $w\langle\varepsilon\rangle$ can be generated using the rule (13′). This constitutes ambiguity of the choice of a rule, and completes the proof of Claim 8.1, as well as of the entire lemma. □

4.2. Null contexts

The second stage of transforming a grammar to the binary normal form is the elimination of *null context specifications* of the form $\lhd\varepsilon$, which assert that the substring being defined is a prefix of the entire string.

For example, a rule $A \to a \,\&\, \lhd\varepsilon$, where A generates a only in the empty left context, can be replaced with a rule $A \to a \,\&\, \trianglelefteqslant a$. In the general case, the condition $\lhd\varepsilon$ can be simulated by a conjunct defining all non-empty strings in an empty left context.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

Lemma 9. *Let $G = (\Sigma, N, R, S)$ be grammar with one-sided contexts, and assume that no symbol $A \in N$ generates the empty string in any context, that is, $L_G(A) \subseteq \Sigma^* \langle \Sigma^+ \rangle$. Construct another grammar $G' = (\Sigma, N \cup \{W, X\}, R', S)$, where every rule with a null context*

$$A \to \Phi \,\&\, {\lhd}\varepsilon \tag{14}$$

in R, with Φ being a conjunction of any conjuncts of the form α, ${\lhd}\beta$ and ${\trianglelefteq}\gamma$, is replaced by a rule

$$A \to \Phi \,\&\, W. \tag{15}$$

All rules in R without a null context are kept in R' as they are. The symbol W generates all non-empty strings in an empty left context by the following rules.

$$W \to Wa \qquad (\text{for all } a \in \Sigma) \tag{16a}$$

$$W \to a \,\&\, {\trianglelefteq}X \qquad (\text{for all } a \in \Sigma) \tag{16b}$$

$$X \to a \qquad (\text{for all } a \in \Sigma) \tag{16c}$$

Then $L_G(A) = L_{G'}(A)$, for all $A \in N$. In particular, $L(G) = L(G')$. Furthermore, if G is unambiguous, then so is G'.

In the rules for W, the length of the current substring is reduced to a single symbol, and the extended context operator is then applied to that symbol. It ensures that the symbol is the prefix of the entire string, that is, its left context is empty.

Proof. $\Rightarrow$ It has to be shown that whenever an item $A(u\langle v\rangle)$ is deducible in G, it can also be deduced in G'. The proof is by induction on p, the number of steps used in deduction of $A(u\langle v\rangle)$ in G.

Let the item $A(u\langle v\rangle)$ be deduced in p steps in G, and assume that the last step of deduction uses a rule of the form (14), which has an empty left context. The conjunct ${\lhd}\varepsilon$ requires that $u = \varepsilon$ and thus the deduction takes the form

$$\mathcal{X} \vdash_G A\big(\varepsilon\langle v\rangle\big),$$

where $\mathcal{X}$ is the set of premises corresponding to the conjuncts Φ in the rule. Also note that $v \neq \varepsilon$, because no symbol in G generates the empty string in any context.

Each of the premises in $\mathcal{X}$ is deduced in fewer than p steps, and hence, by the induction hypothesis, is deducible in G'. Using the rules (16), one can deduce the item $W(\varepsilon\langle v\rangle)$. Now the desired item $A(\varepsilon\langle v\rangle)$ can be deduced in G' out of the premises $\mathcal{X}$ and $W(\varepsilon\langle v\rangle)$ by the rule (15), which corresponds to the rule (14) in the original grammar.

Consider the case when $A(u\langle v\rangle)$ is deduced in G, and the last step uses some rule r, which does not have null contexts. By the induction hypothesis, each premise of such deduction can be deduced in the new grammar G' as well. By the construction, the same rule r also is in G', and the desired deduction $\vdash_{G'} A(u\langle v\rangle)$ can be repeated.

$\Leftarrow$ The converse statement is that $u\langle v\rangle \in L_{G'}(A)$ implies $u\langle v\rangle \in L_G(A)$. It is proved by induction on the number of steps used in the deduction of the item $A(u\langle v\rangle)$ in G'.

Let $A(u\langle v\rangle)$ be deduced in G', and let the last step of its deduction use a rule of the form (15), which was created from a rule (14) in G. Then this last step is

$$\mathcal{X}, W\big(u\langle v\rangle\big) \vdash_{G'} A\big(u\langle v\rangle\big),$$

where $\mathcal{X}$ is the set of premises corresponding to the conjuncts Φ in the rule. By the induction hypothesis, all the premises $\mathcal{X}$ can be deduced in G. Since all the strings generated by W are in an empty left context, $u\langle v\rangle \in L_{G'}(W)$ implies that $u = \varepsilon$. Then the desired item $A(\varepsilon\langle v\rangle)$ is deduced in G by the rule (14) from the premises $\mathcal{X}$.

Let the item $A(u\langle v\rangle)$ be deduced in G' and let the last step of the deduction use some rule r, which is not of the form (15). By the induction hypothesis, each premise used in such deduction is deducible in G'. Using these premises, the deduction of $A(u\langle v\rangle)$ can be carried out in G using the same rule r. This proves that the new grammar generates the same language as the original grammar G.

It remains to show that the transformation preserves unambiguity of a grammar. Consider first the unambiguity of choice of a rule. The rules for nonterminal W define disjoint languages and so do the rules for X. For each nonterminal $A \in N$, the rules for A in G and in G' are in one-to-one correspondence, and the corresponding rules generate the same languages in both grammars. Thus, if these languages were disjoint in G, they will remain disjoint in G.

Turning to the ambiguity of concatenation, the only new introduced concatenation in G' is Wa, which is unambiguous. Thus, if G is unambiguous, then so is the new grammar G'. □

4.3. Unit conjuncts

The last stage of the transformation to the normal form is removing *unit conjuncts* in rules of the form $A \to B \,\&\, \ldots$, where B is a nonterminal symbol. Already for conjunctive grammars, the only known transformation involves substituting

all rules for B into all rules containing a unit conjunct B; in the worst case, this results in an exponential blowup [19]. The same construction applies verbatim to grammars with contexts.

The following lemma shows that a single substitution of rules into a unit conjunct does not affect the language defined by the grammar.

Lemma 10. *Let $G = (\Sigma, N, R, S)$ be a grammar with left contexts. Let*

$$A \to B \,\&\, \Phi \tag{17}$$

be any rule with a unit conjunct $B \in N$, where Φ is a conjunction of any remaining conjuncts, which may contain any context operators.

I. *If $A \neq B$, construct a new grammar $G' = (\Sigma, N, R', S)$, in which the rule (17) is replaced by the following rules. Let*

$$B \to \Psi_i \tag{18}$$

with $1 \leqslant i \leqslant s$, be all rules for the nonterminal B, where each Ψ_i is a conjunction of any conjuncts. Then the rule (17) is replaced with s rules

$$A \to \Psi_i \,\&\, \Phi \tag{19}$$

for $1 \leqslant i \leqslant s$. The rest of the rules in G' are the same as in G. Then $L_{G'}(C) = L_G(C)$ for all $C \in N$.

II. *If $A = B$, then the grammar $G' = (\Sigma, N, R \setminus \{A \to A \,\&\, \Phi\}, S)$, with the rule (17) removed and with no rules added, satisfies $L_{G'}(C) = L_G(C)$ for all $C \in N$.*

In both cases, if G is unambiguous, then so is G'.

Proof. Part I ($A \neq B$). It has to be shown that an item $A(u\langle v\rangle)$ can be deduced in G if and only if it also can be deduced in G'.

$\Rightarrow$ The proof is by induction on p, the number of steps used in the deduction of the item $A(u\langle v\rangle)$ in G.

Let $A(u\langle v\rangle)$ be deduced in p steps in G, and first assume that the pth step uses a rule of the form (17), which has a unit conjunct $B \neq A$. Then the last step of deduction is

$$B\big(u\langle v\rangle\big), \mathcal{X} \vdash_G A\big(u\langle v\rangle\big),$$

where $\mathcal{X}$ denotes the premises corresponding to the conjuncts in Φ. Then the item $B(u\langle v\rangle)$ has been deduced earlier in this deduction. Let (18) be the rule used for deducing $B(u\langle v\rangle)$, and let $\mathcal{Y}$ be all the premises.

$$\mathcal{Y} \vdash_G B\big(u\langle v\rangle\big)$$

All the items in the sets $\mathcal{X}$ and $\mathcal{Y}$ are deduced in fewer than p steps, and hence, by the induction hypothesis, they are deducible in G'. Now the new rule (19) can be used to deduce the desired item in G'.

$$\mathcal{Y}, \mathcal{X} \vdash_{G'} A\big(u\langle v\rangle\big)$$

Consider the case when the item $A(u\langle v\rangle)$ is deduced in G, and the last step of its deduction uses a rule without any unit conjuncts. Then there is the same rule in the new grammar G'. By the induction hypothesis, every premise of deduction of $A(u\langle v\rangle)$ in G can be deduced in the new grammar. Then, the deduction of $A(u\langle v\rangle)$ can be repeated in G'.

$\Leftarrow$ The proof of the converse implication is again carried out by an induction on p, the number of steps used in the deduction of the item $A(u\langle v\rangle)$ in G'.

Let $A(u\langle v\rangle)$ be deduced in G', and let the last step of this deduction use some rule $r \in R$.

Assume that r is a new rule of the form (19), obtained from some rules (17) and (18). Denote the premises of the deduction by r as follows: let $\mathcal{X}$ be the set of premises corresponding to the conjuncts Φ, and let $\mathcal{Y}$ be the premises corresponding to the conjuncts Ψ_i. By the induction hypothesis, all the items in $\mathcal{X} \cup \mathcal{Y}$ can be deduced in the grammar G. Using these premises, one can deduce the desired item $A(u\langle v\rangle)$ in two steps as follows.

$$\mathcal{Y} \vdash_G B\big(u\langle v\rangle\big)$$
$$B\big(u\langle v\rangle\big), \mathcal{X} \vdash_G A\big(u\langle v\rangle\big)$$

In case the rule r used in the deduction of $A(u\langle v\rangle)$ in G' is not of the form (19) defined in the construction, this rule is also contained in G. All the premises of the last step of the deduction of $A(u\langle v\rangle)$ in G' are deducible in G by the induction hypothesis, and hence $A(u\langle v\rangle)$ can be deduced in G.

This proves that the new grammar generates the same language as the original grammar G. Turning to the subject of ambiguity, the construction preserves the unambiguity of concatenation, because no new concatenations are introduced. Assuming that the choice of a rule is unambiguous in G, the proof that it remains unambiguous in G' is carried out by a contradiction; there are the following three cases of a supposed ambiguity in G'.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

- Let some item $A(u\langle v\rangle)$ be deduced in G' by two distinct rules of the form (19). Then, by the construction, G has two different rules for B of the form (18), and each of them can be used to deduce the item $B(u\langle v\rangle)$. This contradicts the assumed unambiguity of G.
- Let now $A(u\langle v\rangle)$ be deduced in G' by one rule of the form (19) and by another rule r of a different form. By construction, G contains r, as well as a rule of the form (17), and the deduction $\vdash_G A(u\langle v\rangle)$ can be made using both these rules. This contradicts the assumption.
- Finally, the case when $A(u\langle v\rangle)$ has been deduced in G' by two distinct rules, both of which are copied to G' from G, again forms a contradiction, because then the same two rules would produce $A(u\langle v\rangle)$ in G.

Part II ($A = B$). Since the set of rules of G' is a proper subset of that of G, every deduction in G' is a deduction in G, and hence $L_{G'}(A) \subseteq L_G(A)$. It has to be shown that, conversely, whenever an item $A(u\langle v\rangle)$ is deduced in G, it can also be deduced in G'. The proof is an induction on the number of steps in the shortest deduction of $A(u\langle v\rangle)$ in G.

If the last step of the deduction of $A(u\langle v\rangle)$ in G uses a rule without unit conjuncts, then, by construction, such a rule is also in G'. By the induction hypothesis, each of the premises used in this deduction can be deduced in G' as well. Then the desired item $A(u\langle v\rangle)$ is deduced in G from the same premises.

Suppose that the last step of the deduction of $A(u\langle v\rangle)$ in G uses a rule of the form (17). One of the premises of this deduction is the same item $A(u\langle v\rangle)$, which must have already been deduced earlier in this deduction. Therefore, there is a shorter deduction of $A(u\langle v\rangle)$ in G, which contradicts the assumption. □

In the theorem below, Lemma 10 is used multiple times to eliminate all unit conjuncts. The theorem summarizes all transformations towards the normal form developed in this section, and asserts that an arbitrary grammar with left contexts can be transformed to the binary normal form. Furthermore, the theorem estimates the worst-case size of the resulting grammar, where the size is measured by the total number of symbols used in the description of the grammar.

Theorem 2. *For each grammar with left contexts $G = (\Sigma, N, R, S)$ there exists and can be effectively constructed a grammar with left contexts $G' = (\Sigma, N', R', S)$ in the binary normal form, such that $L(G) = L(G')$. The size of G' is at most exponential in the size of G. The construction preserves unambiguity of the grammar.*

Proof. First, the rules of the grammar G are pre-processed according to Lemma 5, resulting in a new grammar G_1, which generates the same language. If G is unambiguous, then so is G_1.

Given a grammar G_1, null conjuncts of the form $A \to \varepsilon \,\&\, \ldots$ are eliminated by Construction 1, leading to another grammar G_2, which, by Lemmata 6 and 7, defines the same language without the empty string (see Correctness Statement 1). If G_1 is unambiguous, then so is G_2, by virtue of Lemma 8.

The grammar G_2 may still have null contexts ($\lhd\varepsilon$), which are eliminated by Lemma 9. The new grammar G_3 is proved to generate the same language and the unambiguity of the grammar is preserved.

Unit conjuncts of the form $A \to B \,\&\, \ldots$ are removed from G_3 by applying the transformation in Lemma 10 multiple times, as in the case of conjunctive grammars [19, Lem. 2]. Again, the unambiguity of the grammar is preserved.

Finally, if $\varepsilon \in L(G)$, then a new initial symbol S' is introduced, each rule $S \to \Phi$ in R' is copied to a rule $S' \to \Phi$, and a new rule $S' \to \varepsilon$ is added.

Consider the size of the resulting grammar. The preprocessing of the grammar according to Lemma 5 may increase the size of the grammar by a constant factor. The elimination of null conjuncts in Construction 1 leads to at most exponential blowup in the size of the grammar, because the size of the resulting grammar depends on the number of pairs in the set $\textsc{Nullable}(G)$, which is at most exponential. Applying Lemma 9 for eliminating null contexts increases the size by an additive constant. Finally, the elimination of unit conjuncts according to Lemma 10 may lead to another exponential blowup [19]. This leads to a rough double exponential upper bound on the size of the resulting grammar in the normal form.

A more careful analysis shows that, in fact, the blowup is bounded by a single exponential function. Consider the set of conjuncts occurring in the rules of the grammar G_1 obtained after the first pre-processing stage. Let all nonterminal symbols of G_1 be included in this set as well; the number of elements in the resulting set is linear in the size of G. Neither Construction 1, nor Lemma 10 add any new conjuncts to the grammar, whereas Lemma 9 adds a bounded number of conjuncts. Thus, the number of different conjuncts in the ultimate resulting grammar is still linear in the size of G, and one can construct at most exponentially many different rules from these conjuncts. □

5. Parsing algorithms

A *parsing algorithm* determines whether a given string is generated by a grammar (given or fixed), and if it does, constructs a parse tree of that string. The existence of parsing algorithms of low complexity is a crucial property for any family of formal grammars. The basic parsing algorithm for ordinary context-free grammars is the Cocke–Kasami–Younger algorithm, which determines the membership of all substrings of the input string in the languages generated by all nonterminals of the grammar, and stores the results in a table; a parse tree can then be constructed on the basis of the table. The algorithm works in time $\mathcal{O}(n^3)$, where n is the length of the input string.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

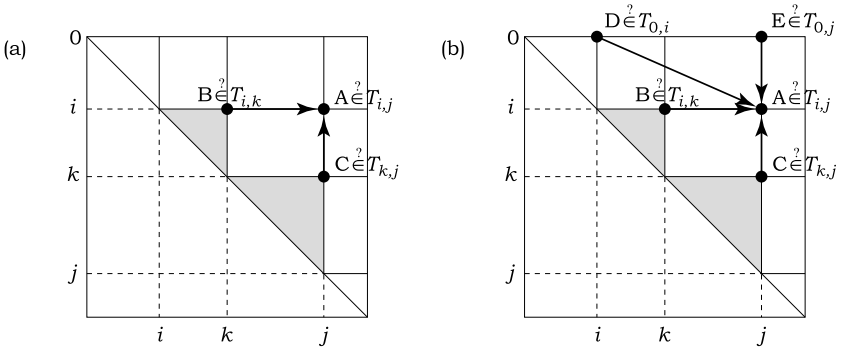

Fig. 3. How the membership of A in $T_{i,j}$ depends on other data, for rules (a) $A \to BC$ and (b) $A \to BC \,\&\, \lhd D \,\&\, \leqslant E$.

The Cocke–Kasami–Younger algorithm has earlier been extended to conjunctive grammars [19], with very few differences to the ordinary case. This section presents a further extension of this algorithm, applicable to grammars with one-sided contexts. Though its differences from the original Cocke–Kasami–Younger are more substantial this time, the running time of the new algorithm is still bounded by $\mathcal{O}(n^3)$. Another parsing algorithm, presented later on in Section 5.2, works in time $\mathcal{O}(n^2)$ on unambiguous grammars.

5.1. The cubic-time algorithm

Let $G = (\Sigma, N, R, S)$ be a grammar with 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 two positions i, j, with $0 \leqslant i < j \leqslant n$, let

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

be the set of nonterminals generating the substring of w delimited by these two positions. In particular, the string w is in $L(G)$ if and only if $S \in T_{0,n}$.

For context-free grammars, including their conjunctive variant, each entry $T_{i,j}$ of this table logically depends upon all entries $T_{k,\ell}$ with $i \leqslant k < \ell \leqslant j$ and $\ell - k < j - i$. A direct dependence by a rule $A \to BC$, where the membership of B in $T_{i,k}$ and the membership of C in $T_{k,j}$ together imply that $A \in T_{i,j}$, is illustrated in Fig. 3(a). Accordingly, all sets $T_{i,j}$ can be constructed inductively on the length $j - i$ of the corresponding substrings, that is, beginning with sets corresponding to shorter substrings and continuing with longer substrings.

For grammars with left contexts, each set $T_{i,j}$ additionally depends upon the sets $T_{0,i}$ and $T_{0,j}$, which are needed to evaluate left contexts $\lhd D$ and extended left contexts $\leqslant E$, respectively. The dependencies induced by a rule $A \to BC \,\&\, \lhd D \,\&\, \leqslant E$ are depicted in Fig. 3(b). While the dependence on $T_{0,i}$ can be handled within the existing inductive approach, by constructing all elements $T_{0,j}, T_{1,j}, \dots, T_{j-1,j}$ before proceeding to construct any $T_{k,\ell}$ with $\ell > i$, the dependence of $T_{i,j}$ on $T_{0,j}$ may form cyclic dependencies, where the membership of a symbol in $T_{i,j}$ depends on the membership of another symbol in $T_{0,j}$, which in turn depends on whether some other symbol is in $T_{i,j}$. Consider the following example.

Example 8. The following grammar generates a single string $w = ab$.

$$S \to aC$$
$$C \to b \,\&\, \leqslant A$$
$$A \to aB$$
$$B \to b$$

This string is generated as follows.

$$\begin{aligned}
\vdash a\big(\varepsilon\langle a\rangle\big) &\quad \text{(axiom)}\\
\vdash b\big(a\langle b\rangle\big) &\quad \text{(axiom)}\\
b\big(a\langle b\rangle\big) \vdash B\big(a\langle b\rangle\big) &\quad (B \to b)\\
a\big(\varepsilon\langle a\rangle\big), B\big(a\langle b\rangle\big) \vdash A\big(\varepsilon\langle ab\rangle\big) &\quad (A \to aB)\\
b\big(a\langle b\rangle\big), A\big(\varepsilon\langle ab\rangle\big) \vdash C\big(a\langle b\rangle\big) &\quad (C \to b \,\&\, \leqslant A)\\
a\big(\varepsilon\langle a\rangle\big), C\big(a\langle b\rangle\big) \vdash S\big(\varepsilon\langle ab\rangle\big) &\quad (S \to aC)
\end{aligned}$$

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

In terms of the table $T_{i,j}$, for the grammar in this example, $B \in T_{1,2}$ implies $A \in T_{0,2}$, which in turn implies $C \in T_{1,2}$, and from the latter fact, one can infer that $S \in T_{0,2}$. Thus, an algorithm constructing the sets $T_{i,j}$ for this grammar has to alternate between the entries $T_{0,2}$ and $T_{1,2}$ until all their elements are gradually determined.

In general, while calculating each j-th column of the table T, one has to re-calculate all entries $T_{j-1,j}, T_{j-2,j}, \ldots, T_{1,j}, T_{0,j}$ after adding any symbols to the set $T_{0,j}$. Such an iterative calculation is carried out in Algorithm 1.

Algorithm 1. Let $G = (\Sigma, N, R, S)$ be a grammar with left contexts in the binary normal form. Let $w = a_1 \ldots a_n \in \Sigma^+$ (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:  for j = 1, ..., n do
2:      while T_{0,j} changes do
3:          for all A → a & ◁D_1 & ... & ◁D_{m'} & ⋖E_1 & ... & ⋖E_{m''} ∈ R do
4:              if a_j = a and D_1, ..., D_{m'} ∈ T_{0,j-1} and E_1, ..., E_{m''} ∈ T_{0,j} then
5:                  T_{j-1,j} = T_{j-1,j} ∪ {A}
6:          for i = j − 2 to 0 do
7:              let P = ∅ (P ⊆ N × N)
8:              for k = i + 1 to j − 1 do
9:                  P = P ∪ (T_{i,k} × T_{k,j})
10:             for all A → B_1C_1 & ... & B_mC_m & ◁D_1 & ... & ◁D_{m'} & ⋖E_1 & ... & ⋖E_{m''} ∈ R do
11:                 if (B_1, C_1), ..., (B_m, C_m) ∈ P, D_1, ..., D_{m'} ∈ T_{0,i}, and E_1, ..., E_{m''} ∈ T_{0,j} then
12:                     T_{i,j} = T_{i,j} ∪ {A}
13: accept if and only if S ∈ T_{0,n}
```

Theorem 3. *For every grammar with left contexts G in the binary normal form, Algorithm 1, given an input string $w = a_1 \ldots a_n$, constructs the sets $T_{i,j}$ and determines the membership of w in $L(G)$, and does so in time $\mathcal{O}(|G|^2 \cdot n^3)$, using space $\mathcal{O}(|G| \cdot n^2)$.*

Proof. The proof of the correctness of the algorithm is comprised of a series of claims that at a certain point in the computation, certain sets $T_{i,j}$ contain certain elements. The top-level statement of this kind is the following invariant of the outer loop: every iteration of this loop completely calculates another column of the table.

Claim 3.1. *After each j-th iteration of the loop in line 1, with $j \in \{1, \ldots, n\}$, all sets $T_{i,\ell}$ with $0 \leqslant i < \ell \leqslant j$ are correctly computed, that is, the value of each variable $T_{i,\ell}$ will be $\{A \in N \mid a_1 \ldots a_i \langle a_{i+1} \ldots a_\ell \rangle \in L_G(A)\}$.*

The proof is by induction on j. Assuming that Claim 3.1 holds for the iteration $j-1$, consider the next iteration j. The nested loop in line 2 executes while any new symbols can be added to the set $T_{0,j}$. Its first iteration adds all symbols $A \in N$, for which the item $A(\varepsilon \langle a_1 \ldots a_j \rangle)$ can be deduced without using any extended left contexts. Then the second iteration may use any items $E(\varepsilon \langle a_1 \ldots a_j \rangle)$ obtained in the first iteration, in order to resolve extended left contexts $\lessdot E$, which may eventually lead to other symbols $A' \in N$ being added to $T_{0,j}$, etc.

These dependencies of symbols in $T_{0,j}$ on other symbols in $T_{0,j}$ form a tree of dependencies, and every symbol in $T_{0,j}$ has its height in this tree. For every symbol $A \in N$ that generates the prefix $\varepsilon \langle a_1 \ldots a_j \rangle$, define its *height* with respect to this prefix, denoted by $h_j(A)$, as follows. Consider the shortest deduction of $A(\varepsilon \langle a_1 \ldots a_j \rangle)$. If no items of the form $E(\varepsilon \langle a_1 \ldots a_j \rangle)$, with $E \in N$, are ever used as an intermediate proposition in this deduction, then let $h_j(A) = 1$. Otherwise, define $h_j(A) = \max_E h_j(E) + 1$, where the maximum is taken over all nonterminals E, for which $E(\varepsilon \langle a_1 \ldots a_j \rangle)$ is used in the deduction. The height is undefined for any nonterminals A with $\varepsilon \langle a_1 \ldots a_j \rangle \notin L_G(A)$. If the height of a nonterminal is defined, then it is at most $|N|$.

Iterations of the loop in line 2 correspond to the height of the symbols added to $T_{0,j}$. The following correctness statement ensures that at each iteration, the algorithm determines all symbols A of height greater by 1.

Claim 3.2. *Consider the j-th iteration of the loop in line 1. After each h-th iteration of the* while *loop in line 2, each variable $T_{i,j}$ with $0 \leqslant i < j$ is guaranteed to contain all symbols $A \in N$, for which it is possible to deduce the item $A(a_1 \ldots a_i \langle a_{i+1} \ldots a_j \rangle)$ without using any intermediate statements $E(\varepsilon \langle a_1 \ldots a_j \rangle)$ with $h_j(E) \geqslant h$.*

In particular, $T_{0,j}$ will contain all symbols A with $h_j(A) \leqslant h$.

This claim occurs within the inductive proof of Claim 3.1 and is proved by a nested induction on h. Every h-th iteration begins with a loop in lines 3–5, which is given a completely constructed set $T_{0,j-1}$ and a partially constructed set $T_{0,j}$, where the latter is guaranteed to contain all nonterminals A with $h_j(A) < h$; in particular, in the beginning of the first iteration, the set $T_{0,j}$ is empty. This loop deduces all true items $A(a_1 \ldots a_{j-1} \langle a_j \rangle)$, whose deductions do not use any items $E(\varepsilon \langle a_1 \ldots a_j \rangle)$ with $h_j(E) \geqslant h$.

The next loop in lines 6–12 similarly handles the deductions of all correct items $A(a_1 \ldots a_i \langle a_{i+1} \ldots a_j \rangle)$, with $0 \leqslant i < j-1$, which can be deduced without referring to any items $E(\varepsilon \langle a_1 \ldots a_j \rangle)$ with $h_j(E) \geqslant h$.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

Claim 3.3. *Consider the j-th iteration of the loop in line 1 and the h-th iteration of the nested* while *loop in line 2. Then, after the iteration i of the loop in line 6, the variable $T_{i,j}$ contains all symbols $A \in N$, for which one can deduce $A(a_1 \dots a_i \langle a_{i+1} \dots a_j \rangle)$ without using any intermediate statements $E(\varepsilon\langle a_1 \dots a_j \rangle)$ with $h_j(E) \geqslant h$.*

Claim 3.3 is proved by an induction on i, and its proof is nested within the inductions for Claim 3.2 and Claim 3.1.

The loop in lines 6–12 begins its execution by computing the set $P = \bigcup_{k=i+1}^{j-1} T_{i,k} \times T_{k,j}$. By Claim 3.1, each set $T_{i,k}$ was completely constructed at the k-th iteration of the outer loop (Claim 3.1 is used here as the induction hypothesis, because $k < j$). Each set $T_{k,j}$ has been updated at the earlier k-th iteration of the loop in line 6, and, by Claim 3.3 (used as the induction hypothesis), it should reflect all deductions that avoid using $E(\varepsilon\langle a_1 \dots a_j \rangle)$ with $h_j(E) \geqslant h$. Thereby, the set P will contain all pairs (B, C), for which there exists some position k, with $a_1 \dots a_i \langle a_{i+1} \dots a_k \rangle \in L_G(B)$ and $a_1 \dots a_k \langle a_{k+1} \dots a_j \rangle \in L_G(C)$, where $C(a_1 \dots a_k \langle a_{k+1} \dots a_j \rangle)$ is deduced without using any intermediate statements $E(\varepsilon\langle a_1 \dots a_j \rangle)$ with $h_j(E) \geqslant h$.

The subsequent inner loop in lines 10–12 uses this set P to determine all nonterminals A, for which $A(a_1 \dots a_i \langle a_{i+1} \dots a_j \rangle)$ is deduced without using any intermediate statements $E(\varepsilon\langle a_1 \dots a_j \rangle)$ with $h_j(E) \geqslant h$. By Claims 3.1 and 3.2 (both used as induction hypotheses), each lookup to $T_{0,i}$ correctly determines the nonterminals generating $\varepsilon\langle a_1 \dots a_i \rangle$, and each lookup to $T_{0,j}$ returns whether $\varepsilon\langle a_1 \dots a_i \rangle$ is in $L_G(E)$, as long as $h_j(E) < h$. This proves Claim 3.3 (for the values of j and h in the ongoing proof).

In order to conclude the proof of Claim 3.2, consider the state of the computation after the last iteration of the loop in lines 6–12, which is given by Claim 3.3 for $i = 0$. It asserts that $T_{0,j}$ contains all such $A \in N$, that $A(\varepsilon\langle a_1 \dots a_j \rangle)$ is deduced without using any intermediate statements $E(\varepsilon\langle a_1 \dots a_j \rangle)$ with $h_j(E) \geqslant h$. Therefore, all A with $h_j(A) \leqslant h$ are in $T_{0,j}$, and Claim 3.2 is proved (for the value of j in the current proof).

Once the *while* loop in line 2 terminates after h iterations, by Claim 3.2, this means that there exist no such nonterminals A, that the shortest deduction of $A(\varepsilon\langle a_1 \dots a_j \rangle)$ uses any intermediate items $E(\varepsilon\langle a_1 \dots a_j \rangle)$ with $h_j(E) = h - 1$, and therefore there are no nonterminals of height h. Since there cannot be any gaps in the height, this indicates that the set $T_{0,j}$ has been constructed completely, and thus proves Claim 3.1.

Finally, Claim 3.1 asserts that when line 13 is executed, the symbol S is in $T_{0,n}$ if and only if $\varepsilon\langle a_1 \dots a_n \rangle$ is in $L_G(S)$. Therefore, the algorithm accepts exactly the strings in $L(G)$.

It remains to estimate the complexity of the algorithm. From the loops in lines 1, 2, 6 and 8, it is evident that the innermost statement in line 9 is executed $\mathcal{O}(|G| \cdot n^3)$ times. In order to do the calculations in line 9 in time proportional to $|G|$, one can replace the Cartesian product by considering only the conjuncts occurring somewhere in the grammar. The memory requirements of the algorithm are given by the size of the sets $T_{i,j}$. □

Once the table $T_{i,j}$ is calculated for a given input string, and it is verified that this string belongs to the language, its parse tree can be constructed by a recursive procedure $parse(A, i, j)$. Its arguments are a symbol $A \in N$ and two positions i and j, and, assuming that $A \in T_{i,j}$, the procedure returns a pointer to the parse tree of $a_1 \dots a_i \langle a_{i+1} \dots a_j \rangle$ from A. In order to avoid re-calculating the same data and constructing identical subtrees, the procedure should employ memoization, returning a pointer to an existing subtree whenever applicable. Thus, the procedure is generally the same as in the case of conjunctive grammars [27, Sect. 6].

5.2. Square-time parsing for unambiguous grammars

For ordinary context-free grammars, besides the cubic-time Cocke–Kasami–Younger algorithm, there also exists a slightly more complicated algorithm by Kasami and Torii [16], which works in time $\mathcal{O}(n^3)$ on every grammar, and in time $\mathcal{O}(n^2)$ for unambiguous grammars. This algorithm was generalized for unambiguous conjunctive grammars [25], and this section presents a further generalization for unambiguous grammars with contexts.

Let $G = (\Sigma, N, R, S)$ be a grammar with left contexts in the binary normal form, let $w = a_1 \dots a_n$ with $n \geqslant 1$ and $a_i \in \Sigma$ be an input string. The main idea is to construct the same sets $T_{i,j} = \{A \mid A \in N,\ \vdash_G A(a_1 \dots a_i \langle a_{i+1} \dots a_j \rangle)\}$ as in Algorithm 1, representing them in a different data structure: namely, as sets of positions

$$T_j[A] = \left\{ i \mid \vdash_G A\left(a_1 \dots a_i \langle a_{i+1} \dots a_j \rangle\right) \right\}, \quad \text{for all } j \text{ with } 1 \leqslant j \leqslant n \text{ and for all } A \in N,$$

each stored as a sorted list. Note that $i \in T_j[A]$ is equivalent to $A \in T_{i,j}$, and for each j, the lists $T_j[A]$, for all $A \in N$ contain exactly the same information as the j-th column of the table $T_{i,j}$. The input string $a_1 \dots a_n$ is in $L(G)$ if and only if the position 0 is in $T_n[S]$.

These lists are computed by Algorithm 2, which first constructs $T_1[A]$ for all $A \in N$, then all $T_2[A]$, and so on until $T_n[A]$. For each j, the algorithm carries out an iterative calculation while the lists $T_j[A]$ can be modified. However, the order of computation inside each iteration is not the same as in Algorithm 1, and this difference accounts for faster operation on unambiguous grammars.

Theorem 4. *For every grammar with one-sided contexts G in the binary normal form, Algorithm 2 constructs the lists $T_j[A]$ and 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 $\mathcal{O}(|G| \cdot n^2)$. If all concatenations in the grammar are unambiguous, then its running time is bounded by $\mathcal{O}(|G|^2 \cdot n^2)$.*

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

Algorithm 2. Let $G = (\Sigma, N, R, S)$ be a grammar with one-sided contexts in the binary normal form.
Denote by $\mathcal{P} = \{(B, C) \mid A \to BC \,\&\, \ldots \in R\}$ the set of all concatenations occurring in the grammar. Let $w = a_1 \ldots a_n \in \Sigma^+$ $(n \geqslant 1)$ be an input string. For each $j \in \{1, \ldots, n\}$, let $T_j[A]$ be a variable ranging over the subsets of $\{0, \ldots, j-1\}$. Let P_k range over the subsets of $\mathcal{P}$, for each $k \in \{0, \ldots, n-1\}$.

```
1:  let T_j[A] = ∅ for all j = 1, ..., n, A ∈ N
2:  for j = 1 to n do
3:      while 0 can be added to T_j[A] for any A ∈ N do
4:          for all A ∈ N do
5:              if A → a_j & ◁D_1 & ... & ◁D_m' & ◁E_1 & ... & ◁E_m'' ∈ R then
6:                  if 0 ∈ T_{j-1}[D_1] ∩ ··· ∩ T_{j-1}[D_m'] and 0 ∈ T_j[E_1] ∩ ··· ∩ T_j[E_m''] then
7:                      T_j[A] = {j − 1}
8:              else
9:                  T_j[A] = ∅
10:         let P_i = ∅ for all i ∈ {0, ..., j − 1}
11:         for k = j − 1 to 1 do
12:             for all (B, C) ∈ P do
13:                 if k ∈ T_j[C] then
14:                     for all i ∈ T_k[B] do
15:                         P_i = P_i ∪ {(B, C)}
16:             for all A ∈ N do
17:                 for all A → B_1C_1 & ... B_mC_m & ◁D_1 & ... & ◁D_m' & ◁E_1 & ... & ◁E_m'' ∈ R do
18:                     if (B_1, C_1), ..., (B_m, C_m) ∈ P_{k−1} and 0 ∈ T_{k−2}[D_1], ..., T_{k−2}[D_m'] and 0 ∈ T_j[E_1], ..., T_j[E_m''] then
19:                         T_j[A] = T_j[A] ∪ {k − 1}
20: accept if and only if 0 ∈ T_n[S]
```

Sketch of a proof. Consider lines 4–19 of Algorithm 2, which, mathematically, have exactly the same effect as lines 3–12 of Algorithm 1. Each variable $P_i \subseteq N$, with $i \in \{0, \ldots, j-1\}$, is used to gather exactly the same set of nonterminals as the variable P in Algorithm 1 at the i-th iteration of its inner loop. There is one difference: Algorithm 2 iterates over intermediate positions k in partitions of substrings $a_{i+1} \ldots a_j$ into BC, and then follows the links given in $T_j[C]$ and $T_k[B]$ in order to determine the initial position i, while the above Algorithm 1 had a more obvious iteration over the initial positions i, and then a nested loop by k.

The correctness statement of lines 4–19 of Algorithm 2 reads as follows.

Claim 4.1. *(Cf. Claim 3.3.) Consider the j-th iteration of the loop in line 2 and the h-th iteration of the nested* while *loop in line 3. Then, after the iteration k of the loop in line 11,*

- *each list $T_j[A]$ contains all such positions $i \in \{k-1, k, \ldots, j-1\}$, that one can deduce $A(a_1 \ldots a_i \langle a_{i+1} \ldots a_j \rangle)$ without using any intermediate statements $E(\varepsilon \langle a_1 \ldots a_j \rangle)$ with $h_j(E) \geqslant h$;*
- *each set P_i with $0 \leqslant i < j$ contains all pairs $(B, C) \in \mathcal{P}$, for which there exists such a position $\ell \in \{k, k+1, \ldots, j-1\}$, that $a_1 \ldots a_i \langle a_{i+1} \ldots a_\ell \rangle \in L_G(B)$ and there is a deduction of $C(a_1 \ldots a_\ell \langle a_{\ell+1} \ldots a_j \rangle)$ that does not use any intermediate statements $E(\varepsilon \langle a_1 \ldots a_j \rangle)$ with $h_j(E) \geqslant h$.*

From this claim, one can infer the correctness of the entire algorithm, that is, that it determines the sets $T_j[A]$ correctly. It remains to estimate its running time.

The following properties of all operations on the lists $T_j[A]$ carried out in the course of a computation shows that they can be implemented efficiently.

Claim 4.2. *(See [25, Lem. 2].) Each list $T_j[A]$ always remains sorted. Each time the algorithm checks the condition in line 13, the set $T_j[A]$ does not contain any elements less than k. Each time the algorithm is about to execute the line 19, the set $T_j[A]$ does not contain any elements less than k.*

By Claim 4.2, checking the membership of an element in $T_j[A]$ in line 13 can be done by examining the top element of the list, while the modifications done in line 19 append an element on its top. Hence, all these operations can be done in constant time, and the overall running time of the algorithm is estimated as $\mathcal{O}(|G|^2 \cdot n^3)$ in the same way as in Theorem 3.

The quadratic upper bound for the unambiguous case relies on the following property.

Claim 4.3. *Assume that all concatenations in G are unambiguous, as in Definition 3. Then, for every j, for every pair $(B, C) \in \mathcal{P}$ and for every i, there exists at most one number k, such that (B, C) can be added to P_i at the iteration k of the loop in line 11 within the i-th iteration of the loop in line 2.*

Indeed, the existence of two such numbers would imply two different partitions of the substring $a_{i+1} \ldots a_j$ into BC.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

It follows from Claim 4.3 that the assignment statement $P_i = P_i \cup \{(B, C)\}$ in the inner loop is executed at most $|\mathcal{P}| \cdot |N| \cdot n^2$ times, which gives the desired upper bound on the running time. □

6. Conclusion and future work

Thus, the formalism of context-free grammars has been extended to allow context specifications, and the basic properties of the resulting model have been explored. Most importantly, the new grammars manage to maintain the principle of defining the properties of strings inductively from the properties of their substrings; introducing the properties of left contexts into this scheme did not break it. Such inductive definitions form parse trees, which can be determined by efficient parsing algorithms, etc. Overall, the model seems to provide meaningful syntactic definitions. An extensive example of using a grammar with contexts to define the syntax of a programming language was recently given by the first author [4].

The new model leaves many theoretical questions to ponder. The most crucial open problem is whether grammars with contexts can define any languages that cannot be generated by a conjunctive grammar [19,26], that is, without ever using the context operators. The language of declarations before or after use, presented in Example 3, might separate these two families, as it is not known how to construct a conjunctive grammar for that language. Unfortunately, there are no known non-trivial methods for proving that a given language is not defined by any conjunctive grammar [26]. Thus, showing that conjunctive grammars are weaker in power than grammars with contexts requires further understanding of conjunctive grammars.

Establishing any limitations of the expressive power of grammars with contexts appears to be an even more challenging task. A possible language that might be not representable by these grammars is the language $\{ww \mid w \in \{a, b\}^*\}$, which, on the other hand, can be defined both by Boolean grammars [21] and by tree-adjoining grammars.

For the unambiguous subclass of grammars with contexts, it would be interesting to know whether it is weaker in power than the full family of grammars with contexts. In other words, is there any inherent ambiguity in the world of grammars with contexts? This question is also beyond the current state of knowledge on formal grammars: already for conjunctive grammars and their unambiguous subclass, no separation is known [25]. Perhaps for another subclass of *linear grammars with contexts*, recently studied by the authors [5], there might be a better chance of separating them from the whole class of grammars with contexts.

Consider another natural theoretical question: is there a parsing algorithm for grammars with one-sided contexts working in substantially less than cubic time? For ordinary context-free grammars, Valiant [35] discovered an algorithm that offloads the most intensive computations into calls to a Boolean matrix multiplication procedure, and thus can work in time $\mathcal{O}(n^\omega)$, with $\omega < 3$; according to the current knowledge on matrix multiplication, ω can be reduced to 2.373. The main idea of Valiant's algorithm equally applies to conjunctive and Boolean grammars, which can be parsed in time $\mathcal{O}(n^\omega)$ as well [27]. 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 $\mathcal{O}(n^\alpha)$-time parsing strategy for these grammars, with $\alpha < 3$, which would be interesting to discover.

Another direction is to develop practical parsing algorithms for grammars with one-sided contexts. One of the methods to try is the recursive descent parsing, where *ad hoc* restrictions resembling contexts of the form $\trianglerighteq D\Sigma^*$ have long been used to guide deterministic computation; an extension of the recursive descent to grammars with right contexts is under development by the first author [3]. The Lang–Tomita Generalized LR parsing is worth being investigated as well.

Yet another direction for further research is to consider *grammars with two-sided contexts*, which would allow rules of the form $A \to BC \,\&\, \triangleleft D \,\&\, \trianglelefteq E \,\&\, \trianglerighteq F \,\&\, \triangleright H$. In this more general case, one may expect definitions by deduction and by language equations, some kind of normal form and polynomial-time parsing, though the degree of the polynomial will most likely be higher than 3. One could also investigate *Boolean grammars with contexts* by adapting the definitions of Boolean grammars [17,21] to strings with contexts.

Acknowledgments

We are grateful to an anonymous referee for bringing numerous small mistakes in the earlier version of this paper to our attention.

Research supported by the Academy of Finland under grants 134860 and 257857.

References

[1] T. Aizikowitz, M. Kaminski, LR(0) conjunctive grammars and deterministic synchronized alternating pushdown automata, in: Computer Science in Russia, CSR 2011, St. Petersburg, Russia, 14–18 June 2011, in: Lect. Notes Comput. Sci., vol. 6651, 2011, pp. 345–358.

[2] J. Autebert, J. Berstel, L. Boasson, Context-free languages and pushdown automata, in: Rozenberg, Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer-Verlag, 1997, pp. 111–174.

[3] M. Barash, Recursive descent parsing for grammars with contexts, in: SOFSEM 2013 Student Research Forum, Local Proceedings II, Špindlerův Mlýn, Czech Republic, 26–31 January, 2013, Institute of Computer Science AS CR, 2013, pp. 10–21.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003

[4] 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, Österreichische Computer Gesellschaft, 2013, pp. 51–67, books@ocg.at 294, http://users.utu.fi/mikbar/kieli/.
[5] M. Barash, A. Okhotin, Linear grammars with one-sided contexts and their automaton representation, in: LATIN 2014: Theoretical Informatics, Montevideo, Uruguay, 31 March–4 April 2014, in: Lect. Notes Comput. Sci., vol. 8392, pp. 190–201.
[6] N. Chomsky, On certain formal properties of grammars, Inf. Control 2 (2) (1959) 137–167, http://dx.doi.org/10.1016/S0019-9958(59)90362-6.
[7] K. Čulík II, R. Cohen, LR-regular grammars—an extension of LR(k) grammars, J. Comput. Syst. Sci. 7 (1) (1973) 66–96, http://dx.doi.org/10.1016/S0022-0000(73)80050-9.
[8] Z. Ésik, W. Kuich, Boolean fuzzy sets, Int. J. Found. Comput. Sci. 18 (6) (2007) 1197–1207, http://dx.doi.org/10.1142/S0129054107005248.
[9] B. Ford, Parsing expression grammars: a recognition-based syntactic foundation, in: Proceedings of POPL 2004, Venice, Italy, January 14–16, 2004, 2004, pp. 111–122, http://doi.acm.org/10.1145/964001.964011.
[10] S. Ginsburg, H.G. Rice, Two families of languages related to ALGOL, J. ACM 9 (1962) 350–371, http://dx.doi.org/10.1145/321127.321132.
[11] S. Jarząbek, T. Krawczyk, LL-regular grammars, Inf. Process. Lett. 4 (1975) 31–37, http://dx.doi.org/10.1016/0020-0190(75)90009-5.
[12] A. Jeż, Conjunctive grammars can generate non-regular unary languages, Int. J. Found. Comput. Sci. 19 (3) (2008) 597–615, http://dx.doi.org/10.1142/S012905410800584X.
[13] A. Jeż, A. Okhotin, Conjunctive grammars over a unary alphabet: undecidability and unbounded growth, Theory Comput. Syst. 46 (1) (2010) 27–58, http://dx.doi.org/10.1007/s00224-008-9139-5.
[14] A.K. Joshi, L.S. Levy, M. Takahashi, Tree adjunct grammars, J. Comput. Syst. Sci. 10 (1) (1975) 136–163, http://dx.doi.org/10.1016/S0022-0000(75)80019-5.
[15] A.K. Joshi, K. Vijay-Shanker, D.J. Weir, The convergence of mildly context-sensitive grammatical formalisms, in: Sells, Shieber, Wasow (Eds.), Foundational Issues in Natural Language Processing, 1991.
[16] T. Kasami, K. Torii, A syntax-analysis procedure for unambiguous context-free grammars, J. ACM 16 (3) (1969) 423–431, http://dx.doi.org/10.1145/321526.321531.
[17] V. Kountouriotis, Ch. Nomikos, P. Rondogiannis, Well-founded semantics for Boolean grammars, Inf. Comput. 207 (9) (2009) 945–967, http://dx.doi.org/10.1016/j.ic.2009.05.002.
[18] R. Kowalski, Logic for Problem Solving, North-Holland, Amsterdam, 1979.
[19] A. Okhotin, Conjunctive grammars, J. Autom. Lang. Comb. 6 (4) (2001) 519–535.
[20] A. Okhotin, Conjunctive grammars and systems of language equations, Program. Comput. Softw. 28 (5) (2002) 243–249, http://dx.doi.org/10.1023/A:1020213411126.
[21] A. Okhotin, Boolean grammars, Inf. Comput. 194 (1) (2004) 19–48, http://dx.doi.org/10.1016/j.ic.2004.03.006.
[22] A. Okhotin, The dual of concatenation, Theor. Comput. Sci. 345 (2–3) (2005) 425–447, http://dx.doi.org/10.1016/j.tcs.2005.07.019.
[23] A. Okhotin, Generalized LR parsing algorithm for Boolean grammars, Int. J. Found. Comput. Sci. 17 (3) (2006) 629–664, http://dx.doi.org/10.1142/S0129054106004029.
[24] A. Okhotin, Recursive descent parsing for Boolean grammars, Acta Inform. 44 (3–4) (2007) 167–189, http://dx.doi.org/10.1007/s00236-007-0045-0.
[25] A. Okhotin, Unambiguous Boolean grammars, Inf. Comput. 206 (2008) 1234–1247, http://dx.doi.org/10.1016/j.ic.2008.03.023.
[26] A. Okhotin, Conjunctive and Boolean grammars: the true general case of the context-free grammars, Comput. Sci. Rev. 9 (2013) 27–59, http://dx.doi.org/10.1016/j.cosrev.2013.06.001.
[27] A. Okhotin, Parsing by matrix multiplication generalized to Boolean grammars, Theor. Comput. Sci. 516 (2014) 101–120, http://dx.doi.org/10.1016/j.tcs.2013.09.011.
[28] A. Okhotin, C. Reitwießner, Conjunctive grammars with restricted disjunction, Theor. Comput. Sci. 411 (26–28) (2010) 2559–2571, http://dx.doi.org/10.1016/j.tcs.2010.03.015.
[29] A. Okhotin, P. Rondogiannis, On the expressive power of univariate equations over sets of natural numbers, Inf. Comput. 212 (2012) 1–14, http://dx.doi.org/10.1016/j.ic.2012.01.004.
[30] T. Parr, K. Fisher, LL(∗): the foundation of the ANTLR parser generator, in: Programming Language Design and Implementation, PLDI 2011, San Jose, USA, 4–8 June 2011, 2011, pp. 425–436.
[31] F.C.N. Pereira, D.H.D. Warren, Parsing as deduction, in: 21st Annual Meeting of the Association for Computational Linguistics, ACL 1983, Cambridge, Massachusetts, USA, 15–17 June 1983, pp. 137–144.
[32] W.C. Rounds, LFP: A logic for linguistic descriptions and an analysis of its complexity, Comput. Linguist. 14 (4) (1988) 1–9.
[33] H. Seki, T. Matsumura, M. Fujii, T. Kasami, On multiple context-free grammars, Theor. Comput. Sci. 88 (2) (1991) 191–229, http://dx.doi.org/10.1016/0304-3975(91)90374-B.
[34] K. Sikkel, Parsing Schemata, Springer-Verlag, 1997.
[35] L.G. Valiant, General context-free recognition in less than cubic time, J. Comput. Syst. Sci. 10 (2) (1975) 308–314, http://dx.doi.org/10.1016/S0022-0000(75)80046-8.
[36] K. Vijay-Shanker, D.J. Weir, A.K. Joshi, Characterizing structural descriptions produced by various grammatical formalisms, in: ACL 1987, pp. 104–111.

Please cite this article in press as: M. Barash, A. Okhotin, An extension of context-free grammars with one-sided context specifications, Inform. and Comput. (2014), http://dx.doi.org/10.1016/j.ic.2014.03.003