A new algorithm for regularizing one-letter context-free grammars

Ștefan Andrei\textsuperscript{a,b,*}, Salvador Valerio Cavadin\textsuperscript{c}, Wei-Ngan Chin\textsuperscript{d}

\textsuperscript{a}CS Programme, Singapore-MIT Alliance, National University of Singapore, Singapore
\textsuperscript{b}Faculty of Computer Science, "Al.I.Cuza" University, Iași, Romania
\textsuperscript{c}Centro de Investigación y Desarrollo de Software, Facultad de Matemática Aplicada, Universidad Católica de Santiago del Estero, Argentina
\textsuperscript{d}School of Computing, National University of Singapore, Singapore

Received 13 December 2002; accepted 6 March 2003
Communicated by A. Salomaa

Abstract

Constructive methods for obtaining regular grammar counterparts for some sub-classes of context-free grammars (CFGs) have been investigated by many researchers. An important class of grammars for which this is always possible is the one-letter CFG. We show in this paper a new constructive method for transforming an arbitrary one-letter CFG to an equivalent regular expression of star-height 0 or 1. Our new result is considerably simpler than a previous construction by Leiss, and we also propose a new normal form for a regular expression with only a single-star occurrence. Through an alphabet factorization theorem, we show how to go beyond the one-letter CFG in a straight-forward way.

© 2003 Elsevier B.V. All rights reserved.

MSC: 68Q42; 68Q45; 68N20

Keywords: Reduction of a context-free grammar; One-letter context-free language; Regular expression

1. Introduction

The subclass of one-letter alphabet languages has been studied for many years. The result “Each context-free one-letter language is regular” was first proven in [13] and

\textsuperscript{*}Corresponding author. CS Programme, Singapore-MIT Alliance, National University of Singapore, Singapore.

E-mail addresses: andrei@comp.nus.edu.sg (S. Andrei), scavadini@ucse.edu.ar (S.V. Cavadini), chinwn@comp.nus.edu.sg (W.-N. Chin).

0304-3975/03/$-$ see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0304-3975(03)00215-9
re-published in [14] using Parikh mappings. A second method based on the “pumping” lemma for context-free languages (CFLs) was presented in [10]. Systems of equations based on $\cup$, $\cdot$, and $*$ operators were used in [15] to prove that the star-height of every one-letter alphabet language is equal to 0 or 1. Later, the first constructive method was proposed in [12] by developing a theory of language equations over an one-letter alphabet. Several key theorems were proven and tied together to provide an algorithm which solves any equation of that type.

In this paper, we shall present a new simpler method using only a single result, called the Regularization Theorem. Like Auteberg et al. [3], Chomsky et al. [7], Ginsburg et al. [10], we will use systems of equations to denote context-free grammars (CFGs). It is known that for a arbitrary CFG, it is undecidable whether its least fixed point can be expressed as a regular expression [4]. We define a new normal form for one-letter equations and a new theorem for solving them. Algorithm A (Section 3) will use this normal form to determine precisely the least fixed point, expressed as a regular expression. By considering the classes of one-letter/one-variable factorizable, we enlarge slightly the class of CFGs for which the construction of a regular expression remains decidable.

2. Preliminaries

We suppose that the reader is familiar with the basic notions of formal language theory, but some important terminologies are briefly covered here. A CFG is denoted as $G=(V_N, V_T, S, P)$, where $V_N/V_T$ are the alphabets of the variables/terminals, $(V = V_N \cup V_T$ is the alphabet of all symbols of $G)$, $S$ is the start symbol and $P \subseteq V_N \times V^+$ is the set of productions. The productions $X \to x_1, X \to x_2, \ldots, X \to x_k$ will be denoted by $X \Rightarrow x_1 | x_2 | \cdots | x_k$ and the right-hand side of $X$ is denoted by rhs($X$), that is $\{x_1, x_2, \ldots, x_k\}$. A variable $X$ is a self-embedded variable in $G$ if there exists a derivation $X \Rightarrow^* z\alpha\beta$, where $\alpha, \beta \in V^+$ [6]. $G$ is a self-embedded grammar if there exists a self-embedded variable. $G$ is a reduced grammar if $\forall X \in V, S \not\Rightarrow z\alpha\beta$ and $\forall X \in V_N, X \not\Rightarrow u$, with $u \in V_T^*$. The empty word is denoted by $\epsilon$. A CFG is proper if it has no $\epsilon$-productions (i.e. $X \to \epsilon, X \in V_N$) and no chain-productions (i.e. $X \to Y, X, Y \in V_N$). It is known that for every CFG (which does not generate $\epsilon$) there exists an equivalent proper CFG. The set of terminal words attached to the variable $X$ of the grammar $G$ is $L_G(X)=\{w \in V_T^* | X \Rightarrow^*_G w\}$. Note that $\Rightarrow^*_G$ denotes $m$ productions, while $\Rightarrow^*_G$ denotes at least one production have been applied. The set of all sentential forms of $X$ in $G$ is $S_G(X) = \{z \in V^* | X \Rightarrow^*_G z\}$. The set of sentential forms of $G$ is $S(G)=S_G(S)$. The language of $G$ is $L(G)=S_G(S) \cap V_T^* = L_G(S)$. If $G$ is a CFG, then its language is called context-free (denoted by CFL). All the above sets can be easily extended to words, e.g. $L_G(x) = \{z \in V_T^* | x \Rightarrow^*_G w\}$. A permutation with $n$ elements is a one-to-one correspondence from $\{1, \ldots, n\}$ to $\{1, \ldots, n\}$. The set of all
permutations with \( n \) elements is denoted by \( \Pi_n \). \( \mathbb{N} \) denotes the set of natural numbers; \( \mathbb{N}_{\overline{n}} \) denotes the set \( \{1, \ldots, n\} \). \( i, j \in \mathbb{N}_{\overline{n}} \) denotes \( i \in \mathbb{N}_{\overline{n}}, j \in \mathbb{N}_{\overline{n}} \).

We continue by providing some results related to the system of equations [3]. Systems of equations are extremely concise for modeling CFLs [7,10]. The notions of substitution, solution, and equivalence can be found in [3,11].

**Definition 2.1.** Let \( G = (\{X_1, \ldots, X_n\}, V_T, X_1, P) \) be a CFG. A system of \((X_i-)\) equations over \( G \) is a vector \( P = (P_1, \ldots, P_n) \) of subsets of \( V^* \). This is usually written as \( X_i = P_i \forall i \in \mathbb{N}_{\overline{n}} \) with \( P_i = \{x \in V^* \mid X_i \rightarrow x \in P\} \).

The next classical result gives one method for computing the minimal solution of a system of equations by derivations [3].

**Theorem 2.1.** Let \( G = (\{X_1, \ldots, X_n\}, V_T, X_1, P) \) be a CFG. Then the vector \( L_G = (L_G(X_1), \ldots, L_G(X_n)) \) is the least solution of the associated CFG.

The next theorem refers to a well-known transformation which “eliminates” \( X \) from a linear \( X \)-equation [2,15,11]. From now on, unless specified otherwise, we shall use the notations \( x = x_1 + \cdots + x_m, \beta = \beta_1 + \cdots + \beta_n \), where \( m, n \in \mathbb{N} \). We shall use \( X = \beta \) to mean \( X = \beta_j \forall j \in \mathbb{N}_{\overline{n}} \).

**Theorem 2.2.** Let \( X = xX + \beta \) be an \( X \)-equation, where \( X \notin x \), and \( X \notin \beta \). The least solution of the \( X \)-equation is \( X = x^*\beta \), and if \( x \notin \alpha \), then this solution is unique.

### 3. One-letter CFG and its regular construction

In this section, we shall give a new constructive method for regularizing one-letter CFGs that is more concise and general than the method proposed in [12]. Commutativity plays an important role for transforming one-letter CFGs and this is covered in the following lemma.

**Lemma 3.1.** Let \( G = (V_N, \{a\}, S, P) \) be a one-letter CFG. The set of all commutative grammars of \( G \) is \( \mathcal{G}_{\text{com}}(G) = \{(V_N, \{a\}, S, P_{\text{com}})\} \), where \( P_{\text{com}} = \{X \rightarrow x_{n(1)} \cdots x_{n(k)} \mid X \rightarrow x_1 \cdots x_k \in P, \pi \in \Pi_k\} \). Then for every \( G_{\text{com}} \in \mathcal{G}_{\text{com}}(G) \), it follows \( L(G) = L(G_{\text{com}}) \).

**Proof.** This can be easily proved by induction on \( l, l \geq 1 \). For any \( X \in V_N \), we have: \( X \xrightarrow{G} a^n \) iff \( X \xrightarrow{G_{\text{com}}} a^n \). Complete proof can be found in [1].

Lemma 3.2 allows the symbols of any sentential form of a one-letter CFG to be re-ordered. Its proof is similar to Lemma 3.1.

**Lemma 3.2.** Let \( G = (V_N, \{a\}, S, P) \) be a one-letter CFG and let us consider the derivation \( x_1 \cdots x_k \xrightarrow{G} a^n \). For any \( \pi \in \Pi_k \), we have \( x_{\pi(1)} \cdots x_{\pi(k)} \xrightarrow{G} a^n \).

The next lemma shows how the star-operations can be flattened for one-letter CFGs.
Lemma 3.3. Let $G = (V_N, \{a\}, S, P)$ be a one-letter CFG and $x_1, \ldots, x_n$ be some words over $V_N \cup \{a\}$. The following properties hold:

(i) $L_G((x_1 + \cdots + x_n)^*) = L_G(x_1^* \cdots x_n^*) = L_G((x_1^* + \cdots + x_n^*)^*)$,

(ii) $L_G((x_1 x_2^* \cdots x_n^*)^*) = \varepsilon + L_G(x_1 x_2^* x_3^* \cdots x_n^*)$.

Proof. Focusing to the first equality of (i), we have to prove that: $(x_1 + \cdots + x_n)^* \stackrel{x}{\Rightarrow} a^\infty$ iff $x_1^* \cdots x_n^* \stackrel{x}{\Rightarrow} a^\infty$. Based on Lemma 3.2, the words $x_1, \ldots, x_n$ can be commuted in any order. We proceed by induction on $n$. First, let us suppose that $n = 2$. The inclusion $L_G((x_1 + x_2)^*) \supseteq L_G(x_1^* x_2^*)$ is obvious. For the other inclusion, let us take $\beta = (x_1 + x_2)^n, n \geq 0$. It can be rewritten $\beta = x_1^m x_2^n \cdots x_1^{n_i-1} x_2^{m_i}$, where $n_i \in \mathbb{N}, \forall i \in \overline{1, k}$, and $\sum_{i=1}^k n_i = n$. Applying the commutativity property $x_1 x_2 = x_2 x_1$, several times, we get $\beta = x_1^{n_1} + \cdots + n_i x_2^{n_i}$. Thus, $L_G(\beta) \subseteq L_G(x_1^* x_2^*)$, and $L_G((x_1 + x_2)^n) = L_G(x_1^* x_2^*)$ follows.

In the inductive step, let us assume that the first equality of (i) is true for $n = m$, where $m \geq 2$, and prove that (i) also holds for $n = m + 1$. We have

$$L_G((x_1 + \cdots + x_m + x_{m+1})^*)$$

$$= L_G((x_1 + \cdots + x_m + x_{m+1}^*)^*)$$

$$= L_G((x_1 + \cdots + x_m + x_{m+1})^* x_{m+1}^*) = L_G((x_1 + \cdots + x_m)^*) \cdot L_G(x_{m+1}^*)$$

$$= L_G(x_1^* \cdots x_m^*) \cdot L_G(x_{m+1}^*) = L_G(x_1^* \cdots x_m^* x_{m+1}^*).$$

For the other identities (the second equality of (i) and (ii)), let us use the following equations for regular expressions from [15]: $(x^*)^* = x^*$ and $(x y^*)^* = \varepsilon + x(x + y)^*$. We, therefore, have

$$L_G((x_1^* \cdots x_n^*)^*) = L_G(((x_1 + \cdots + x_n)^*)^*)$$

$$= L_G((x_1 + \cdots + x_n)^*) = L_G(x_1^* \cdots x_n^*)$$

and

$$L_G((x_1 x_2^* \cdots x_n^*)^*) = L_G(((x_1 x_2 + \cdots + x_n)^*)^*)$$

$$= L_G((\varepsilon + x_1^{n_1} x_2^{n_i} + \cdots + x_n^*)^*)$$

$$= \varepsilon + L_G(x_1 x_2^* x_3^* \cdots x_n^*).$$

We now define a new normal form for one-letter CFGs, followed by a theorem to normalise each arbitrary one-letter CFG to this form.

Definition 3.1. We say that the equation $X = \mathcal{P}$ is in the one-letter normal form (abbreviated by OLNF) if $\mathcal{P} = aX + \beta$, where $x \notin \beta$.

Theorem 3.1. Let $G = (\{X_1, \ldots, X_n\}, \{a\}, X_1, P)$ be a one-letter reduced CFG. Then every attached $X_i$-equation can be transformed into OLNF.
Proof. Let \( X_i = \alpha X_i + \beta \) be an arbitrary \( X_i \)-equation. Because \( G \) is reduced, it follows that \( \beta \neq 0 \), otherwise there will be no terminal word in \( L_G(X_i) \). Based on Lemma 3.2, it follows that the symbols of \( \alpha \) can be commuted in \( \mathcal{P}_i \) in such a way that \( X_i \) will be at the last position. By distributivity \( (\gamma_1 \cdot X_i + \gamma_2 \cdot X_i = (\gamma_1 + \gamma_2) \cdot X_i) \), it is obvious that every \( X_i \)-equation can be transformed to this form. The only possible term of \( \mathcal{P}_i \) for which \( X_i \) cannot be commuted until the last position is \( \alpha'(\beta' X_i)^\ast \). In this case, \( \alpha'(\beta' X_i)^\ast \) will be rewritten into \( \alpha'(\varepsilon + (\beta' X_i)^\ast(\beta' X_i)) = \alpha' + \alpha'(\beta' X_i)^\ast X_i \). If \( X_i \notin \alpha' \) then the \( X_i \)-equation is in OLNF, otherwise the transformation will continue and stop after a finite number of steps. \( \square \)

By doing this transformation together with a flattening transformation step from Lemma 3.3, we can now formulate Theorem 3.2 as a generalization of Leiss’s results (Theorems 3.1, 4.1, and 4.2 from [12]).

The next theorem is a tool for eliminating the occurrences of the variable \( X \) in an \( rhs \) of its \( X \)-equation. This is a generalization of Theorem 2.2, and is a key ingredient of Algorithm A. Let us denote by \( \alpha[\beta/X] \) the word obtained by replacing every \( X \)-occurrence in \( \alpha \) with \( \beta \). Of course, this substitution is valid only if \( X \) does not occur in \( \beta \).

**Theorem 3.2** (Regularization). Let \( G = (V_N, \{a\}, S, P) \) be a one-letter reduced \( CFG \). Let \( X \in V_N \) and \( X = \alpha X + \beta \) be an OLNF \( X \)-equation. Then, the least solution of the \( X \)-equation is \( X = (\alpha[\beta/X])^\ast \beta \), and if \( G \) is proper, then this solution is unique.

**Proof.** Before starting the proof, let us refer to the uniqueness of the solution. Because \( G \) is proper, it follows that \( G \) has no \( e \)-productions and chain-productions, so \( e \notin \alpha \), and \( e \notin \beta \). Following Theorem 2.2, we can show that the solution obtained for this \( X \)-equation is unique. By applying Lemmas 3.2 and 3.3 finitely many times, we can assume without loss of generality, that \( \alpha \) is equivalent to a regular expression over \( V_N \cup \{a\} \) of star-height 0 or 1. The general form of \( \alpha \) is \( \alpha = \sum_{i=1}^t \alpha_{0,i}(\alpha_{1,i}X^{k_{1,i}})^\ast \cdots (\alpha_{m,i}X^{k_{m,i}})^\ast \). For simplicity, let us focus on \( (\alpha_{1,i}X^{k_{1,i}})^\ast \). Using commutativity, \( (\alpha_{1,i}X^{k_{1,i}})^\ast = \{(\alpha_{1,i}X^{k_{1,i}})^{n_{1,i}} | n_{1,i} \geq 0\} = \{\alpha_{1,i}X^{k_{1,i}}^{n_{1,i}} | n_{1,i} \geq 0\} \). Hence, \( \alpha = \sum_{i=1}^t \alpha_{0,i}(\alpha_{1,i}^{n_{1,i}}X^{k_{1,i}+\cdots+k_{m,i}})^\ast \). This result can be denoted by \( \alpha = \sum_{i=1}^t \alpha'_iX^{Q_i} \), where \( \alpha'_i \) are words over \( V_N \cup \{a\} \) and \( Q_i \) are (linear) polynomials in the variables in \( n_{j,i} \in \mathbb{N}, (k_{j,i} \in \mathbb{N} \text{ are constants}) \). Therefore, the initial \( X \)-equation becomes \( X = (\sum_{i=1}^t \alpha'_iX^{Q_i})X + \beta \), which corresponds to the following \( X \)-productions in \( G : X \rightarrow \alpha'_iX^{Q_i} | X \cdots | \alpha'_iX^{Q_i}X | \beta_1 \cdots | \beta_n \). Because \( X \notin \alpha'_i \), \( \forall i \in I, T \), and \( X \notin \beta_j \), \( \forall j \in \overline{1,n} \), it follows that \( S_G(X) \) can be generated by applying productions of the form \( X \rightarrow \alpha'_iX^{Q_i}X, i \in \overline{1,t} \), several times (say \( s \)-times), followed by productions of the form \( X \rightarrow \beta_j \), \( j \in \overline{1,n} \), in order to remove all the occurrences of \( X \). According to Lemma 3.2, we can re-order the symbols in any sentential form, and thus apply the current \( X \)-production to the last occurrence of the variable \( X \). With this, we obtain a set of \( X \)-derivations: \( X \xrightarrow{G} \alpha'_i \cdots \alpha'_iX^{Q_i}X \cdots X^{Q_i}X \), where \( i_1, \ldots, i_s \in \overline{1,t} \). After applying \( Q_1 + \cdots + Q_n + 1 \) productions of the type \( X \rightarrow \beta_j, j \in \overline{1,n} \), we obtain the words...
\[ \alpha_i \cdots \alpha_i \beta_{j_i} \cdots \beta_{j_i, \sigma_i} \cdots \beta_{j, \sigma_j, \beta_j}. \]

Applying Lemma 3.2, we have
\[ L_G(\alpha_i \cdots \alpha_i \beta_{j_i} \cdots \beta_{j_i, \sigma_i} \cdots \beta_{j, \sigma_j, \beta_j}) = L_G(\alpha_i \beta_{j_1} \cdots \beta_{j_i, \sigma_i} \cdots \alpha_i \beta_{j_1} \cdots \beta_{j, \sigma_j, \beta_j}). \]

But the words \( \alpha_i \beta_{j_1} \cdots \beta_{j_i, \sigma_i} \cdots \alpha_i \beta_{j_1} \cdots \beta_{j, \sigma_j, \beta_j} \) correspond to \( (x[\beta/X])^* \beta \), so it follows that the solution of the \( X \)-equation is \( X = (x[\beta/X])^* \beta. \)

We shall now present a constructive algorithm, named \( \mathbf{A} \), to regularise an arbitrary one-letter \( \mathsf{CFG} \) represented using systems of equations. Solving each system of equations by our method yields an equivalent regular expression. As we assume reduced \( \mathsf{CFG} \), each recursive \( X \)-equation must have at least one term without any occurrence of \( X \).

**Algorithm A.**

**Input:** \( G = (\{X_1, \ldots, X_n\}, \{a\}, X_1, P) \) a reduced and proper one-letter \( \mathsf{CFG} \)

**Output:** \( L_G(X_i) \) is regular, \( \forall i \in [1, n] \)

**Method:**
1. Construct \( X_i = P_i, \forall i \in [1, n] \) as in Definition 2.1;
2. for \( i := 1 \) to \( n \) do begin
3. Transform the \( X_i \)-equation into \( \mathsf{OLNF} \)
4. \( P_i = (x[\beta/X_i])^* \beta \);
5. Apply Lemma 3.3 to obtain the star-height 0 or 1 for \( P_i \)
6. for \( j := i + 1 \) to \( n \) do \( P_j = P_j[P_i/X_i] \);
7. for \( i := n - 1 \) downto 1 do
8. for \( j := n \) downto \( i + 1 \) do begin
9. \( P_i = P_i[P_j/X_j] \);
10. Apply Lemma 3.3 to obtain the star-height 0 or 1 for \( P_i \)
11. \( L_G = (X_1, \ldots, X_n) \)

**Theorem 3.3.** Algorithm \( \mathbf{A} \) is correct and completes within a finite number of steps.

**Proof.** The lines 1, 11 are due to Definition 2.1 and Theorem 2.1, respectively. The instructions between lines 3 and 5 are based on Theorem 3.2 and Lemma 3.3 and imply that \( \forall i \in [1, n], P_i \) does not contain \( X_i \). Line 6 ensures that \( \forall i \in [1, n], P_i \) does not contain any \( X_j \) with \( j < i \). The occurrences of \( X_j \) from \( P_i \), where \( j > i \) are replaced with terminal words at lines 7–10. After the execution of Algorithm \( \mathbf{A}, P_i \) is a regular expression over \( \{a\} \) of star-height 0 or 1. Thus, \( L_G(X_i) \) is regular \( \forall i \in [1, n] \). By induction on \( i \), it can be easily proved using Lemma 3.3 that \( P_i \) has the star-height 0 or 1. \( \Box \)

As a side remark, if we assume that the steps 3–5 and 9 and 10 require constant time, we can state that the time-complexity of Algorithm \( \mathbf{A} \) is \( O(n^2) \).

**Example 3.1.** Let us consider \( G = (\{X_1, X_2\}, \{a\}, X_1, P) \) with \( P \) given by the following productions: \( X_1 \rightarrow aX_1X_2 | a, X_2 \rightarrow X_1X_2 | aa \). Line 1 of Algorithm \( \mathbf{A} \) will construct the...
system: \( X_1 = aX_1X_2 + a, \) \( X_2 = X_1X_2 + a^2. \) After executing line 4, we get \( X_1 = (aX_2)^*a, \) and after line 6, we obtain \( X_2 = a(aX_2)^*X_2 + a^2. \) At the next iteration, we get \( X_2 = (a(a^4)^*)^*a^2, \) and after line 5, \( X_2 = a^2 + a^3 \cdot (a^3)^*. \) At line 9, we get \( X_1 = a(a^3 + a^4 \cdot a^*(a^3)^*)^*, \) and after line 10, \( X_1 = (a^3)^* \cdot (a + a^3 \cdot a^* \cdot (a^3)^* \cdot (a^4)^*). \)

As a further remark, the order of eliminating \( X_i \) in Algorithm A can be arbitrary. For instance, by eliminating \( X_2, \) followed by \( X_1, \) we obtain a pair of (equivalent) simpler expressions: \( X_1 = a + a^4 \cdot a^* \) and \( X_2 = a^2 \cdot a^*. \) We shall next show that every one-letter regular expression can be reduced to only one occurrence of \(*. \)

**Definition 3.2.** We say that \( e = e_1 + \cdots + e_n \) (where each \( e_i \) contains only \( \cdot \) and \( * \) operators) is in single-star normal form if \( \forall i \in \{1, n\}, e_i \) has at most one occurrence of \(*. \)

This normalization is captured in the following theorem. The conclusion of the next theorem is simple from the point of view of finite automata. The language is accepted by a deterministic finite automaton, which always gives a single-star form. The minimal normal form that is generated here is considered in detail in [15].

**Theorem 3.4.** Every regular expression over an one-letter alphabet can be transformed into an equivalent single-star normal form.

**Proof.** If \( e \) is a regular expression of star-height 1 (the case 0 is trivial) then it can be written as \( e = e_1 + \cdots + e_n, \) where \( \forall i \in \{1, n\}, e_i = a^{m_{0,i}} \cdot (a^{m_{1,i}})^* \cdots (a^{m_{k,i}})^*, \) where \( m_{1,i} < \cdots < m_{k,i}. \) We suppose, without loss of generality, that the cases \( m_{k,i} = m_{k+1,i} \) are excluded based on the property \( x^*x^* = x^*. \) Let \( G(a_1, \ldots, a_k) \) be the greatest number \( b \) such that the Diophantine equation \( a_1x_1 + \cdots + a_kx_k = b \) has no solution in \( \mathbb{N}, \) where the greatest common divisor of \( a_1, \ldots, a_k \) is 1 (notation \( gcd(a_1, \ldots, a_k) = 1)). \) This means that for any \( b > G(a_1, \ldots, a_k) \) the equation \( a_1x_1 + \cdots + a_kx_k = b \) has always a solution in \( \mathbb{N}. \) Let us denote by \( F(a_1, \ldots, a_k) \) the set of all natural numbers less than \( G(a_1, \ldots, a_k) \) such that the above equation has solution in \( \mathbb{N}. \) According to Chrobak [8], if \( a_1 < \cdots < a_k \) and \( gcd(a_1, \ldots, a_k) = 1, \) then \( G(a_1, \ldots, a_k) \leq (a_k - 1)(a_1 - 1). \) Using \( d = gcd(m_1, \ldots, m_{k,i}), \) and the above Diophantine equation, it follows that \( e_i \) can be equivalently transformed to \( a^{m_{0,i}} \cdot (e + a^d)^{m_{1,i}} + \cdots + a^d \cdot n_s \cdot ((a^d)^{(m_{k,i}/d)} - 1)((m_{k,i}/d) - 1) + 1(a^{d^k})^*, \) where \( n_1, \ldots, n_s \in F(m_{1,i}/d, \ldots, m_{k,i}/d). \) In this way, each factor \( e_i \) of \( e \) has at most one star, so \( e \) is in single-star normal form. \( \square \)

**Example 3.2.** The following regular expressions of star-height 1 are reduced to the single-star normal form: \((a^3)^* (a^3)^* = e + a^3 \cdot a^*, (a^4)^* (a^8)^* = e + a^4 (a^2)^* \) and \((a^3)^* (a^9)^* (a^9)^* = e + a^4 + a^6 + a^8 + a^9 + a^{10} + a^{12} \cdot a^*.

A particular case of the above theorem is to reduce the expression \((a^m)^* \cdot (a^n)^*\) for which \( m \equiv 0 \) (mod \( n)). \) So, \( gcd(m, n) = m, \) and by Theorem 3.4, it follows that \((a^m)^* \cdot (a^n)^* = e + (a^m)^* \cdot (a^n)^* = (a^m)^*). \) To illustrate this idea in more detail, we present the following example.
Example 3.3. Let us consider the CFG from Example 3.1. Using Theorem 3.4, we can reduce to single-star form: $X_1 = a \cdot (a^3)^+ + a^5 \cdot a^*$ and $X_2 = a^2 + a^3 \cdot a^*$.

We shall now explore a straightforward way to beyond one-letter CFGs through the use of alphabet factorisation.

4. Beyond one-letter CFGs

As is well-known, non-self-embedded variables/CFGs are easily converted to the regular sublanguages. Theorem 4.1 (proven in [1]) shows that any CFG, $G$, generates a regular language if all its self-embedded variables can be shown to generate regular languages.

**Theorem 4.1.** Let $G$ be an arbitrary reduced and proper CFG. If for all self-embedded variables $X$ the language $L_G(X)$ is regular, then $L(G)$ is regular.

In the following, we shall combine the property of an one-letter alphabet, together with self-embeddedness, in order to obtain a more powerful class of CFGs which generates regular languages.

**Definition 4.1.** A CFG $G = (V_N, V_T, S, P)$ is called one-letter factorizable if for every self-embedded variable $X$, $L_G(X) \subseteq \{a\}^*$, where $a \in V_T$.

In other words, if $G$ is one-letter factorizable, then every self-embedded variable has the corresponding language defined over an one-letter alphabet.

The notion of one-variable factorizable is introduced next. This topic is dual to the notion of one-letter factorizable, by considering at most one occurrence of a variable $A_i$ in $\text{rhs}(X_i)$.

**Definition 4.2.** We say that $G = (V_N^1 \cup V_N^2, V_T, X_1, P)$, where $V_N^1 = \{X_1, \ldots, X_n\}$, and $V_N^2 = \{A_1, \ldots, A_n\}$, and $V_N^1 \cap V_N^2 = \emptyset$, is one-variable factorizable if for every self-embedded variable $X_i$, we have $\text{rhs}(X_i) \subseteq \{X_i, A_i\}^*$ and $\text{rhs}(A_i) \subseteq V_T^*$.

**Theorem 4.1 (Factorization).** The following facts hold:

(a) a one-letter factorizable CFG generates a regular language,
(b) an one-variable factorizable CFG generates a regular language.

**Proof.** (a) Let $G = (V_N, V_T, S, P)$ be a one-letter factorizable CFG. For every self-embedded variable $X \in V_N$, we know that $L_G(X) \subseteq \{a\}^*$. So due to Theorem 3.3, it follows that $L_G(X)$ is regular. Applying Theorem 4.1, it follows that $L(G)$ is regular.

(b) Let $G = (V_N^1 \cup V_N^2, V_T, X_1, P)$ be a one-variable factorizable CFG, where $V_N^1 = \{X_1, \ldots, X_n\}$, $V_N^2 = \{A_1, \ldots, A_n\}$ ($V_N^1 \cap V_N^2 = \emptyset$) and for every self-embedded variable $X_i$, we have $\text{rhs}(X_i) \subseteq \{X_i, A_i\}^*$ and $\text{rhs}(A_i) \subseteq V_T^*$. Let us construct the
Following Theorem 4.1, we get the equation that could be used to enlarge the class of results of this paper is part of a larger effort to provide efficient constructive methods. Result that enabled us to go beyond one-letter languages in a straightforward way. The regular expressions using the one-letter normal form. We also introduced a factorization into regular automata (with counters) for handling one-letter sublanguages. Later Chrobak [8] of the earliest work in this area is the work of [5] which investigated efficient pushdown automata (with counters) for handling one-letter sublanguages. Later Chrobak [8] showed that the problem of converting from non-deterministic to deterministic finite-state automata remains a hard problem even for one-letter languages.

More recently, Domaratzki et al. [9] even investigated efficient methods for the converse problem of converting from finite-state automaton over an one-letter alphabet to its equivalent context-free grammar in the Chomsky normal form. One-letter languages have also been used recently in [16] to aid in the decomposition of finite languages.

Our work has advanced the frontier of research on regularizing one-letter CFGs. We provided a much simpler constructive method for transforming one-letter CFGs into regular expressions using the one-letter normal form. We also introduced a factorization result that enabled us to go beyond one-letter languages in a straightforward way. The results of this paper is part of a larger effort to provide efficient constructive methods that could be used to enlarge the class of CFGs that can be regularized.
Acknowledgements

We thank to anonymous TCS referees for helpful comments and suggestions which improved the paper.

References