Computer Science Journal of Moldova, vol.**30**, no.2(89), 2022

# On Elimination of Erasing Rules from E0S Grammars

Alexander Meduna, Martin Havel

**Abstract**

The present paper describes an alternative algorithm for the removal of erasing rules from E0S grammars. As opposed to the standard way of eliminating erasing rules in most E0S-like grammars, such as context-free grammars, this method requires no predetermination of symbols that derive the empty string. The proposed algorithm is formally verified. In the conclusion of the paper, the applicability of the algorithm to E0S grammars that work in a semi-parallel way is demonstrated. Furthermore, two open problems are formulated.

**Keywords:** Formal languages, E0S grammars, elimination of erasing rules.

**MSC 2020:** 68Q45, 03D05, 68Q70.

## 1 Introduction

In the theory of formal languages, E0S grammars represent, in essence, ordinary context-free grammars generalized so they can rewrite both terminal and nonterminal symbols. Recall that these grammars underlie some important general frameworks of context-free rewriting systems, such as selective substitution grammars (see [2] and [3]).

The standard method of eliminating erasing rules from E0S grammars and their special cases, such as context-free grammars, requires a predetermination of symbols that derive the empty string (see, for instance, Section 5.1.3.2 in [9]). Of course, the theory of formal languages would highly appreciate obtaining an alternative method that performs this elimination without any predetermination like this. The present paper achieves such a method. Indeed, it presents and verifies an alternative algorithm that performs


doi:10.56415/csjm.v30.08

the elimination of erasing rules from E0S grammars without this predetermination. In addition, it is demonstrated that this algorithm is straightforwardly applicable to E0S grammars working in a semi-parallel way, too. Several other derivation modes are also discussed.

The paper is organized as follows. First, Section 2 gives all the necessary terminology. Then, Section 3 presents the main result of the paper—that is, it describes and verifies an alternative algorithm for the elimination of erasing rules from E0S grammars. Finally, Section 4 discusses the applicability of our algorithm to other derivation modes used in E0S grammars, and formulates two open problems.

## 2 Preliminaries and Definitions

This paper assumes that the reader is familiar with formal language theory (see [11]). For an alphabet (finite nonempty set) $V$, $V^*$ represents the free monoid generated by $V$. The unit of $V^*$ is denoted by $\varepsilon$. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, $V^+$ is thus the free semigroup generated by $V$. For $w \in V^*$, $|w|$ denotes the length of $w$, and $\mathrm{alph}(w)$ denotes the set of symbols occurring in $w$. As usual, we consider two languages $L_1$, $L_2$ to be equal if and only if $L_1 - \{\varepsilon\} = L_2 - \{\varepsilon\}$, and we simply write $L_1 = L_2$ in this case. The inclusion relation of languages is interpreted similarly.

An *E0S grammar* (see [2] and [3]) is a quadruple, $G = (V, T, P, S)$, where $V$ is an alphabet, $T \subset V$, $S \in V - T$, and $P \subseteq V \times V^*$ is a finite relation. The components $V$, $T$, $P$, and $S$ are called the *total alphabet*, the alphabet of *terminal symbols*, the set of *rules*, and the *start symbol*, respectively. Each $(X, y) \in P$ is written as $X \to y$ throughout this paper. $G$ is said to be *propagating* if and only if every $X \to y \in P$ satisfies $y \in V^+$. Rules of the form $X \to \varepsilon$ are called *erasing rules*. The *direct derivation relation* over $V^*$, denoted by $\Rightarrow_G$, is defined as follows: $uXv \Rightarrow_G uyv$ if and only if $X \to y \in P$ and $u, v \in V^*$. Let $\Rightarrow_G^n$ and $\Rightarrow_G^*$ denote the $n$th power of $\Rightarrow_G$, for some $n \geq 0$, and the reflexive-transitive closure of $\Rightarrow_G$, respectively. The *language of* $G$ is denoted by $L(G)$ and defined as $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$. Observe that the language family generated by E0S grammars coincides with the family of context-free languages.

Let $G = (V, T, P, S)$ be an E0S grammar. For any $X \Rightarrow_G^* y$, where $X \in V$ and $y \in V^*$, $\Delta(X \Rightarrow_G^* y)$ denotes its corresponding derivation tree. Regarding

derivation trees and related notions, we use the terminology of Section 5.1.1 in [9]. We straightforwardly extend these notions from context-free grammars to E0S grammars. A derivation subtree whose frontier is $\varepsilon$ is called an $\varepsilon$-*subtree*. Let $S \Rightarrow_G^* w$ be of the form $S \Rightarrow_G^* yxz \Rightarrow_G^* w$, where $x, y, z \in V^*$ and $w \in T^*$. We write $S \Rightarrow_G^* y\,{}^{\varepsilon}x\,z \Rightarrow_G^* w$ to express that either (1) $x = \varepsilon$ or (2) all subtrees in $\Delta(S \Rightarrow_G^* w)$ rooted at the symbols in $x$ are $\varepsilon$-subtrees; informally, it means that $x$ is erased in the rest of the derivation.

# 3 Main Result

In this section, we present and verify an alternative algorithm that performs the elimination of erasing rules from E0S grammars. To give a more detailed insight into this algorithm, consider an arbitrary E0S grammar, $G = (V, T, P, S)$, and $Y \in V$. If $Y$ derives $\varepsilon$ in $G$, then a derivation like this can be expressed in the following step-by-step way:

$$Y \Rightarrow_G y_1 \Rightarrow_G y_2 \Rightarrow_G \cdots \Rightarrow_G y_n \Rightarrow_G \varepsilon,$$

where $y_i \in V^*$ for all $i = 1, \ldots, n$, for some $n \geq 1$. If a sentential form contains several occurrences of $Y$, each of them can be erased in this way although there may exist many alternative ways of erasing $Y$. Based upon these observations, the next algorithm introduces compound nonterminals of the form $\langle X, U \rangle$, in which $X$ is a symbol that is not erased during the derivation, and $U$ is a set of symbols that are erased. Within the compound nonterminal, the algorithm simulates the erasure of symbols in $U$ in the way sketched above. Observe that since $U$ is a set, $U$ contains no more than one occurrence of any symbol because there is no need to record several occurrences of the same symbol; indeed, as already pointed out, all these occurrences can be erased in the same way.

**Algorithm 1.** *An alternative elimination of erasing rules in E0S grammars without any predetermination of symbols that derive the empty string.*

**Input:** An E0S grammar, $G = (V, T, P, S)$.

**Output:** A propagating E0S grammar, $H = (V', T, P', S')$, such that $L(G) = L(H)$.

**Method:** Initially, set $V' = T \cup \{\langle X,U\rangle \mid X \in V, U \subseteq V\}$, $S' = \langle S,\emptyset\rangle$, and

$$P' = \{\langle a,\emptyset\rangle \to a \mid a \in T\}.$$

Repeat (1) and (2), given next, until $P'$ cannot be extended.

(1) **If** $Y \to y_0Y_1y_1Y_2y_2\cdots Y_ny_n \in P$, where $y_i \in V^*$, $Y_j \in V$, for all $i$ and $j$, $0 \le i \le n$, $1 \le j \le n$, for some $n \ge 1$,
**then** for every $U \subseteq V$, extend $P'$ by adding

$$\langle Y,U\rangle \to \langle Y_1, U \cup \mathrm{alph}(y_0y_1\cdots y_n)\rangle\langle Y_2,\emptyset\rangle\cdots\langle Y_n,\emptyset\rangle.$$

(2) **If** $\langle X,U\rangle \in V'$ and $Y \to y \in P$, where $Y \in U$ and $y \in V^*$,
**then** extend $P'$ by adding

$$\langle X,U\rangle \to \langle X, (U - \{Y\}) \cup \mathrm{alph}(y)\rangle. \qquad \square$$

Let us point out that the algorithm makes no predetermination of symbols from which $\varepsilon$ can be derived as opposed to most standard methods of removing erasing rules, including the standard removal of erasing rules from context-free grammars (see, for instance, Section 5.1.3.2 in [9]). Indeed, if the output grammar improperly extends the second component of a two-component nonterminal by a symbol that is not erased throughout the rest of the derivation, then this occurrence of the symbol never disappears in this component, so a terminal string cannot be generated under this improper selection. It is also worth pointing out that the cardinality of $V'$ and $P'$ exponentially increases with respect to the cardinality of V and P, respectively.

Before we verify the correctness of the presented algorithm, we illustrate it by an example.

**Example 1.** Consider the E0S grammar

$$G = \big(\{S,a,b\},\{a,b\},\{S \to aSb, S \to \varepsilon\},S\big).$$

Clearly, $L(G) = \{a^nb^n \mid n \ge 0\}$. With $G$ as its input, Algorithm 1 produces a propagating E0S grammar $H$ whose fully detailed definition is left to the reader. We only describe the derivations of *aabb* in $G$ and in $H$. That is,

$$S \Rightarrow_G aSb \Rightarrow_G aaSbb \Rightarrow_G aabb$$

and

$$\langle S,\emptyset\rangle \Rightarrow_H \langle a,\emptyset\rangle\langle S,\emptyset\rangle\langle b,\emptyset\rangle \Rightarrow_H \langle a,\emptyset\rangle\langle a,\{S\}\rangle\langle b,\emptyset\rangle\langle b,\emptyset\rangle$$

$$\Rightarrow_H \langle a,\emptyset\rangle\langle a,\emptyset\rangle\langle b,\emptyset\rangle\langle b,\emptyset\rangle \Rightarrow_H^4 aabb. \qquad \square$$

We proceed to a formal verification of Algorithm 1.

**Theorem 1.** Let $G$ be an E0S grammar. Algorithm 1 halts and correctly converts $G$ to a propagating E0S grammar $H$ satisfying $L(G) = L(H)$.

*Proof.* Clearly, the algorithm always halts. Since $P'$ does not contain any erasing rules, $H$ is propagating. To establish $L(H) = L(G)$, we prove three claims.

The first claim shows how derivations of $G$ are simulated by $H$. It is used to prove that $L(G) \subseteq L(H)$ later in the proof. Recall that the meaning of ${}^{\varepsilon}x$ is defined in Section 2.

**Claim 1.** *If* $S \Rightarrow_G^m {}^{\varepsilon}x_0X_1{}^{\varepsilon}x_1X_2{}^{\varepsilon}x_2\cdots X_h{}^{\varepsilon}x_h \Rightarrow_G^* z$, *where* $z \in L(G)$, $|z| \geq 1$, $x_i \in V^*$, $X_j \in V$, *for all* $i$ *and* $j$, $0 \leq i \leq h$, $1 \leq j \leq h$, *for some* $m \geq 0$ *and* $h \geq 1$, *then* $\langle S,\emptyset\rangle \Rightarrow_H^* \langle X_1,U_1\rangle\langle X_2,U_2\rangle\cdots\langle X_h,U_h\rangle$, *where* $\bigcup_{1\leq i\leq h} U_i \subseteq \bigcup_{0\leq i\leq h} \mathrm{alph}(x_i)$.

*Proof.* This claim is established by induction on $m \geq 0$.

*Basis.* The basis for $m = 0$ is clear.

*Induction Hypothesis.* Suppose that the claim holds for all derivations of length $\ell$, where $0 \leq \ell \leq m$, for some $m \geq 0$.

*Induction Step.* Consider any derivation of the form

$$S \Rightarrow_G^{m+1} w \Rightarrow_G^* z,$$

where $w \in V^+$, $z \in L(G)$, and $|z| \geq 1$. Since $m+1 \geq 1$, this derivation can be expressed as

$$S \Rightarrow_G^m x \Rightarrow_G w \Rightarrow_G^* z,$$

where $x \in V^+$. Let $x = {}^{\varepsilon}x_0X_1{}^{\varepsilon}x_1X_2{}^{\varepsilon}x_2\cdots X_h{}^{\varepsilon}x_h$, where $x_i \in V^*$, $X_j \in V$, for all $i$ and $j$, $0 \leq i \leq h$, $1 \leq j \leq h$, for some $h \geq 1$. Then, by the induction hypothesis,

$$\langle S,\emptyset\rangle \Rightarrow_H^* \langle X_1,U_1\rangle\langle X_2,U_2\rangle\cdots\langle X_h,U_h\rangle,$$

where $\bigcup_{1\leq i\leq h} U_i \subseteq \bigcup_{0\leq i\leq h} \text{alph}(x_i)$, for some $n \geq 0$.

Next, we consider all possible forms of $x \Rightarrow_G w$, covered by the next two cases—(i) and (ii).

(i) Let $X_j \to y_0 Y_1 y_1 \cdots Y_q y_q \in P$, where $y_i \in V^*$, for all $i$, $0 \leq i \leq q$, $Y_i \in V$, for all $i$, $1 \leq i \leq q$, for some $j$, $1 \leq j \leq h$, and $q \geq 1$, so

$$\begin{gathered}{}^{\varepsilon}x_0 X_1 {}^{\varepsilon}x_1 \cdots X_j {}^{\varepsilon}x_j \cdots X_h {}^{\varepsilon}x_h \Rightarrow_G \\ {}^{\varepsilon}x_0 X_1 {}^{\varepsilon}x_1 \cdots X_{j-1} {}^{\varepsilon}x_{j-1} {}^{\varepsilon}y_0 Y_1 {}^{\varepsilon}y_1 \cdots Y_q {}^{\varepsilon}y_q {}^{\varepsilon}x_j X_{j+1} {}^{\varepsilon}x_{j+1} \cdots X_h {}^{\varepsilon}x_h.\end{gathered}$$

By (1) in the algorithm,

$$\langle X_j, U_j\rangle \to \langle Y_1, U_j \cup \text{alph}(y_0 y_1 \cdots y_q)\rangle \langle Y_2, \emptyset\rangle \cdots \langle Y_q, \emptyset\rangle \in P'$$

so

$$\begin{gathered}\langle X_1, U_1\rangle \langle X_2, U_2\rangle \cdots \langle X_j, U_j\rangle \cdots \langle X_h, U_h\rangle \Rightarrow_H \\ \langle X_1, U_1\rangle \langle X_2, U_2\rangle \cdots \langle X_{j-1}, U_{j-1}\rangle \langle Y_1, U_j \cup \text{alph}(y_0 y_1 \cdots y_q)\rangle \\ \langle Y_2, \emptyset\rangle \cdots \langle Y_q, \emptyset\rangle \langle X_{j+1}, U_{j+1}\rangle \cdots \langle X_h, U_h\rangle.\end{gathered}$$

Clearly,

$$\begin{gathered}\left(\bigcup_{1\leq i\leq h} U_i\right) \cup \left(\bigcup_{0\leq i\leq q} \text{alph}(y_i)\right) \subseteq \\ \left(\bigcup_{0\leq i\leq h} \text{alph}(x_i)\right) \cup \left(\bigcup_{0\leq i\leq q} \text{alph}(y_i)\right)\end{gathered}$$

which completes the induction step for (i).

(ii) Let $x_j = x'_j Y x''_j$ and $Y \to y \in P$, where $y \in V^*$ and $x'_j, x''_j \in V^*$, so

$${}^{\varepsilon}x_0 X_1 {}^{\varepsilon}x_1 \cdots X_j {}^{\varepsilon}x_j \cdots X_h {}^{\varepsilon}x_h \Rightarrow_G {}^{\varepsilon}x_0 X_1 {}^{\varepsilon}x_1 \cdots X_j {}^{\varepsilon}x'_j {}^{\varepsilon}y {}^{\varepsilon}x''_j \cdots X_h {}^{\varepsilon}x_h.$$

If $Y \notin \bigcup_{1\leq i\leq h} U_i$, then

$$\langle X_1, U_1\rangle \langle X_2, U_2\rangle \cdots \langle X_h, U_h\rangle \Rightarrow_H^0 \langle X_1, U_1\rangle \langle X_2, U_2\rangle \cdots \langle X_h, U_h\rangle$$

and clearly

$$\bigcup_{1\leq i\leq h} U_i \subseteq \left(\bigcup_{0\leq i\leq h, i\neq j} \text{alph}(x_i)\right) \cup \text{alph}\left(x'_j y x''_j\right)$$

so assume that $Y \in \bigcup_{1\leq i\leq h} U_i$. By (2) in the algorithm,

$$\langle X_k, U_k\rangle \to \langle X_k, (U_k - \{Y\}) \cup \mathrm{alph}(y)\rangle \in P',$$

where $U_k = U'_k \cup \{Y\}$, $U'_k \subseteq V$, for some $k$, $1 \leq k \leq h$, so

$$\langle X_1, U_1\rangle\langle X_2, U_2\rangle \cdots \langle X_k, U_k\rangle \cdots \langle X_h, U_h\rangle \Rightarrow_H$$
$$\langle X_1, U_1\rangle\langle X_2, U_2\rangle \cdots \langle X_k, (U_k - \{Y\}) \cup \mathrm{alph}(y)\rangle \cdots \langle X_h, U_h\rangle.$$

Clearly,

$$\left(\bigcup_{1\leq i\leq h, i\neq k} U_i\right) \cup \left(U'_k \cup \mathrm{alph}(y)\right) \subseteq \left(\bigcup_{0\leq i\leq h, i\neq j} \mathrm{alph}(x_i)\right) \cup \mathrm{alph}\left(x'_j y x''_j\right)$$

which completes the induction step for (ii).

Observe that these two cases cover all possible derivations of the form $x \Rightarrow_G w$. Thus, the claim holds. □

The second claim shows that in $H$, every derivation of any $z \in L(H)$ can be expressed as a two-part derivation. In the first part, every occurring symbol is a two-component nonterminal. In the second part, only the rules of the form $\langle a, \emptyset\rangle \to a$, where $a \in T$, are used.

**Claim 2.** *For every $z \in L(H)$, there exists a derivation $\langle S, \emptyset\rangle \Rightarrow_H^* x \Rightarrow_H^* z$, where $x \in V'^+$, and during $x \Rightarrow_H^* z$, only rules of the form $\langle a, \emptyset\rangle \to a$, where $a \in T$, are used.*

*Proof.* Since $H$ is an E0S grammar, we can always rearrange all the applications of rules so the claim holds. □

The third claim shows how derivations of $H$ are simulated by $G$. It is used to prove that $L(H) \subseteq L(G)$ later in the proof.

**Claim 3.** *If $\langle S, \emptyset\rangle \Rightarrow_H^n \langle X_1, U_1\rangle\langle X_2, U_2\rangle \cdots \langle X_h, U_h\rangle$, where $X_i \in V$ and $U_i \subseteq V$, for all $i$, $1 \leq i \leq h$, for some $n \geq 0$ and $h \geq 1$, then $S \Rightarrow_G^* x_0 X_1 x_1 X_2 x_2 \cdots X_h x_h$, where $x_i \in V^*$, for all $i$, $0 \leq i \leq h$, and $\bigcup_{1\leq i\leq h} U_i \subseteq \bigcup_{0\leq i\leq h} \mathrm{alph}(x_i)$.*

*Proof.* This claim is established by induction on $n \geq 0$.

*Basis.* The basis for $n = 0$ is clear.

*Induction Hypothesis.* Suppose that the claim holds for all derivations of length $\ell$, where $0 \leq \ell \leq n$, for some $n \geq 0$.

*Induction Step.* Consider any derivation of the form

$$\langle S, \emptyset \rangle \Rightarrow_H^{n+1} w,$$

where $w \in V'^+$. Since $n+1 \geq 1$, this derivation can be expressed as

$$\langle S, \emptyset \rangle \Rightarrow_H^{n} x \Rightarrow_H w,$$

where $x \in V'^+$. Let $x = \langle X_1, U_1 \rangle \langle X_2, U_2 \rangle \cdots \langle X_h, U_h \rangle$, where $X_i \in V$, $U_i \in V^*$, for all $i$, $1 \leq i \leq h$, for some $h \geq 1$. By the induction hypothesis,

$$S \Rightarrow_G^* x_0 X_1 x_1 X_2 x_2 \cdots X_h x_h,$$

where $x_i \in V^*$, for all $i$, $1 \leq i \leq h$, such that $\bigcup_{1 \leq i \leq h} U_i \subseteq \bigcup_{0 \leq i \leq h} \text{alph}(x_i)$.

Next, we consider all possible forms of $x \Rightarrow_H w$, covered by the next two cases—(i) and (ii).

(i) Let $\langle X_j, U_j \rangle \to \langle Y_1, W \rangle \langle Y_2, \emptyset \rangle \cdots \langle Y_q, \emptyset \rangle \in P'$ be a rule introduced in (1) in the algorithm, where $W \subseteq V$ and $Y_i \in V$, for all $i$, $1 \leq i \leq q$, for some $q \geq 1$, so

$$\langle X_1, U_1 \rangle \langle X_2, U_2 \rangle \cdots \langle X_j, U_j \rangle \cdots \langle X_h, U_h \rangle \Rightarrow_H$$
$$\langle X_1, U_1 \rangle \cdots \langle X_{j-1}, U_{j-1} \rangle \langle Y_1, W \rangle \langle Y_2, \emptyset \rangle \cdots \langle Y_q, \emptyset \rangle \langle X_{j+1}, U_{j+1} \rangle \cdots \langle X_h, U_h \rangle.$$

By (1) in the algorithm, $W$ is of the form $W = U_j \cup \text{alph}(y_0 y_1 \cdots y_q)$, where $y_i \in V^*$, for all $i$, $1 \leq i \leq q$, and $X_j \to y_0 Y_1 y_1 \cdots Y_q y_q \in P$. Therefore,

$$x_0 X_1 x_1 \cdots X_j x_j \cdots X_h x_h \Rightarrow_G$$
$$x_0 X_1 x_1 \cdots X_{j-1} x_{j-1} y_0 Y_1 y_1 \cdots Y_q y_q x_j X_{j+1} x_{j+1} \cdots X_h x_h.$$

Clearly,

$$\left(\bigcup_{1 \leq i \leq h} U_i\right) \cup \left(\bigcup_{0 \leq i \leq q} \text{alph}(y_i)\right) \subseteq$$
$$\left(\bigcup_{0 \leq i \leq h} \text{alph}(x_i)\right) \cup \left(\bigcup_{0 \leq i \leq q} \text{alph}(y_i)\right).$$

(ii) Let $\langle X_j, U_j\rangle \to \langle X_j, W\rangle \in P'$ be a rule introduced in (2) in the algorithm, for some $j$, $1 \le j \le h$, where $W \subseteq V$, so

$$\langle X_1, U_1\rangle\langle X_2, U_2\rangle \cdots \langle X_j, U_j\rangle \cdots \langle X_h, U_h\rangle \Rightarrow_H \langle X_1, U_1\rangle\langle X_2, U_2\rangle \cdots \langle X_j, W\rangle \cdots \langle X_h, U_h\rangle.$$

By (2) in the algorithm, $W$ is of the form $W = \big(U_j - \{Y\}\big) \cup \mathrm{alph}(y)$, where $Y \in V$, $y \in V^*$, and $Y \to y \in P$. Recall that $\bigcup_{1\le i\le h} U_i \subseteq \bigcup_{0\le i\le h} \mathrm{alph}(x_i)$ by the induction hypothesis. Since $Y \in \bigcup_{1\le i\le h} U_i$, some $x_k$ has to be of the form $x_k = x'_k Y x''_k$, where $x'_k, x''_k \in V^*$, so

$$x_0 X_1 x_1 \cdots X_k x_k \cdots X_h x_h \Rightarrow_G x_0 X_1 x_1 \cdots X_k x'_k y x''_k \cdots X_h x_h.$$

Clearly,

$$\big(\bigcup_{1\le i\le h, i\ne j} U_i\big) \cup \big(U_j - \{Y\}\big) \cup \mathrm{alph}(y) \subseteq \big(\bigcup_{1\le i\le h, i\ne k} \mathrm{alph}(x_i)\big) \cup \big(\mathrm{alph}(x_k) - \{Y\}\big) \cup \mathrm{alph}(y).$$

Observe that these two cases cover all possible derivations of the form $x \Rightarrow_H w$. Therefore, the claim holds. □

Next, we establish the identity $L(H) = L(G)$. Consider a special case of Claim 1 when $x_i = \varepsilon$, $X_j \in T$, for all $i$ and $j$, $0 \le i \le h$, $1 \le j \le h$, for some $h \ge 1$. Then, $S \Rightarrow_G^* X_1 X_2 \cdots X_h$ implies that $\langle S, \emptyset\rangle \Rightarrow_H^* \langle X_1, \emptyset\rangle\langle X_2, \emptyset\rangle \cdots \langle X_h, \emptyset\rangle$. By the initialization part of the algorithm, $\langle X_j, \emptyset\rangle \to X_j \in P'$, for all $j$, $1 \le j \le h$, so

$$\begin{aligned} \langle S, \emptyset\rangle &\Rightarrow_H X_1\langle X_2, \emptyset\rangle \cdots \langle X_h, \emptyset\rangle \\ &\Rightarrow_H X_1 X_2 \cdots \langle X_h, \emptyset\rangle \\ &\ \ \vdots \\ &\Rightarrow_H X_1 X_2 \cdots X_h. \end{aligned}$$

Hence, $L(G) \subseteq L(H)$. Let $z \in L(H)$. By Claim 2, $\langle S, \emptyset\rangle \Rightarrow_H^* x \Rightarrow_H^* z$, where $x \in V'^+$. By Claim 3, $S \Rightarrow_G^* z$. Therefore, $L(H) \subseteq L(G)$, and the theorem holds. □

# 4 Concluding Remarks

In this final section, we discuss the applicability of our algorithm to other derivation modes in E0S grammars. Then, we propose two open problems to consider in the future investigation related to the subject of this paper.

## 4.1 Semi-Parallel and Parallel Derivation Modes

During every derivation step in an E0S grammar, a single symbol is rewritten. Hence, these grammars work under a *sequential derivation mode*. We next define a generalization of this mode, where some symbols are simultaneously rewritten while others remain unrewritten (like in scattered context grammars, see [1]).

Let $G = (V, T, P, S)$ be an E0S grammar. $G$ makes a *semi-parallel derivation step* from $u_0v_1u_1 \cdots v_nu_n$ to $u_0w_1u_1 \cdots w_nu_n$, denoted by

$$u_0v_1u_1 \cdots v_nu_n \text{ s-par}\Rightarrow_G u_0w_1u_1 \cdots w_nu_n$$

if and only if $u_i \in V^*$ for all $i = 1, \ldots, n$, $v_j, w_j \in V^*$, and $v_j \Rightarrow_G w_j$ for all $j = 1, \ldots, n$, for some $n \geq 1$. Let $\text{s-par}\Rightarrow_G^*$ denote the reflexive-transitive closure of $\text{s-par}\Rightarrow_G$. The *language generated by G under the semi-parallel mode* is denoted by $L(G, \text{s-par}\Rightarrow_G)$ and defined as

$$L\big(G, \text{s-par}\Rightarrow_G\big) = \big\{w \in T^* \mid S \,\text{s-par}\Rightarrow_G^* w\big\}.$$

The following theorem says that Algorithm 1 is applicable to E0S grammars working under this mode. Let us note that the standard algorithm also works for this mode.

**Theorem 2.** Let $G$ be an E0S grammar. Algorithm 1 halts and correctly converts $G$ to a propagating E0S grammar $H$ satisfying $L(G, \text{s-par}\Rightarrow_G) = L(H, \text{s-par}\Rightarrow_H)$.

*Proof.* This theorem can be established by analogy with the proof of Theorem 1, so we leave its proof to the reader. □

By analogy with the definition of the semi-parallel mode, we may define a *parallel mode*, where during a single derivation step, all occurrences

of symbols in the current sentential form have to be rewritten. However, observe that neither Algorithm 1 nor the standard algorithm are applicable to E0S grammars working under this mode. To remove erasing rules from E0S grammars working under the parallel mode, we may use the algorithm for the elimination of erasing rules in E0L systems (see pages 63–65 in [10]).

## 4.2 Open problems

We close this paper by proposing two open problems.

I. Observe that there exist other derivation modes under which the algorithm achieved in the previous section does not work properly. For instance, the algorithm is inapplicable to the Indian derivation mode (see [7], [8], or Section 2.4 in [12]). Recall that an Indian derivation step is performed so a rule is selected, and all symbols coinciding with the left-hand side of this rule are rewritten by it in the current sentential form. Consider the Indian parallel grammar having the three rules $S \to SS$, $S \to a$, and $S \to \varepsilon$, where $S$ is a nonterminal and $a$ is a terminal. Obviously, its language equals

$$\left\{ a^{2^n} \mid n \geq 0 \right\} \cup \{\varepsilon\}.$$

If we convert this grammar to another Indian parallel grammar by Algorithm 1, the resulting Indian grammar generates $\{a^n \mid n \geq 1\}$, which differs from $\{a^{2^n} \mid n \geq 0\}$. As a result, the algorithm is inapplicable to derivation modes that involve the Indian derivation mode, such as the mode of Russian parallel grammars (see [4] or Section 2.4 in [12]). Also, note that our algorithm is not applicable to the mode of $k$-grammars, where at every derivation step, exactly $k$ occurrences of symbols have to be simultaneously rewritten (see [5], [6] and page 126 in the second volume of [11]). Can we modify Algorithm 1 so it works under these derivation modes as well? Recall that it is an open problem whether we can always eliminate all erasing rules from any Russian parallel grammar or from any $k$-grammar.

II. Compared to its input grammar, the output grammar produced by Algorithm 1 has many more symbols and rules. To be precise, the cardinality of symbols and rules of output grammar has an exponential

increase from the cardinality of symbols and rules of input grammar. Can we improve this algorithm so it works in a more economical way?

## 5 Funding

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II), from the project IT4Innovations excellence in science—LQ1602.

## 6 Acknowledgement

We are grateful for comments made by Petr Zemek and the anonymous referee of this paper.

## References

[1] S. A. Greibach and J. E. Hopcroft, "Scattered context grammars," *Journal of Computer and System Sciences*, vol. 3, no. 3, pp. 233–224, August, 1969.

[2] H. C. M. Kleijn, "Basic ideas of selective substitution grammars," *Lecture Notes in Computer Science*, vol. 281, pp. 75–95, 1987.

[3] H. C. M. Kleijn and G. Rozenberg, "Context-free like restrictions on selective rewriting," *Theoretical Computer Science*, vol. 16, no. 3, pp. 237–269, 1981.

[4] M. K. Levitina, "On some grammars with global productions," *NTI*, vol. 2, no. 3, pp. 32–36, 1972. (in Russian)

[5] K. Salomaa, "Hierarchy of k-context-free languages, part 1," *International Journal of Computer Mathematics*, vol. 26, no. 2, pp. 69–90, Mar, 1989.

[6] K. Salomaa, "Hierarchy of k-context-free languages, part 2," *International Journal of Computer Mathematics*, vol. 26, no. 3, pp. 193–205, Mar, 1989.

[7] R. Siromoney and K. Krithivasan, "Parallel context-free languages," *Information and Control*, vol. 24, no. 2, pp. 155–162, 1974.

[8] S. Skyum, "Parallel context-free languages," *Information and Control*, vol. 26, no. 3, pp. 280–285, 1974.

[9] A. Meduna, *Automata and Languages: Theory and Applications*, London, Springer, 2000.

[10] G. Rozenberg and A. Salomaa, *Mathematical Theory of L Systems*, Orlando, Academic Press, 1980.

[11] *Handbook of Formal Languages, Volumes I through III*, G. Rozenberg, A. Salomaa, Eds. New York, Springer, 1997.

[12] J. Dassow and G. Paun, *Regulated Rewriting in Formal Language Theory*, New York, Springer, 1989.

Alexander Meduna, Martin Havel

Received February 2, 2022
Accepted March 17, 2022

Alexander Meduna
Faculty of Information Technology, Brno University of Technology
Božetěchova 1/2, 612 66 Brno, Czech Republic
E–mail: meduna@fit.vutbr.cz

Martin Havel
Faculty of Information Technology, Brno University of Technology
Božetěchova 1/2, 612 66 Brno, Czech Republic
E–mail: xhavel44@stud.fit.vutbr.cz