Acta Informatica 27, 81–95 (1989)

Acta Informatica

Scattered Versus Context-Sensitive Rewriting

Jakob Gonczarowski [1] and Manfred K. Warmuth [2,*]

[1] Department of Computer Science, Hebrew University, Jerusalem 91904, Israel

[2] Department of Computer and Information Sciences, University of California at Santa Cruz, Applied Sciences Building, Santa Cruz, CA 95064, USA

Summary. We study the relationship between scattered and context-sensitive rewriting. We prove that an extended version of scattered grammars produces exactly the context-sensitive languages. Also unordered scattered context languages are a proper subset of scattered context languages, and unordered scattered rewriting with erasing does not generate all scattered context (and thus not all context-sensitive) languages.

1. Introduction

One of the major topics of Formal Language Theory is the study of the generative power of rewriting mechanisms. The context-free mechanism, where one rewrites a single symbol at each rewriting step, has been extended in several ways. One such extension rewrites a given consecutive subword into another string; examples are context-sensitive and type-0 grammars. For context-sensitive grammars, one requires the replacement string to be at least as long as the string to be rewritten. An alternate way requires that the word to be rewritten is a scattered subword of the sentential form. Scattered context grammars, introduced in [4], are of this kind. There, one requires each symbol in the scattered word to be rewritten into at least one new symbol. A general study of adjacent versus nonadjacent rewriting was performed in [11]. The generative power and the computational complexity of languages where a fixed (but arbitrary) number of symbols is rewritten at each derivation step, was investigated in [5–7] and [2]. Both the cases have been considered where the symbols to be rewritten are either required to be adjacent or allowed to be scattered. In both cases there is no restriction to the identity of the symbols to be rewritten; there is only a restriction on the number of symbols to be rewritten simultaneously

* Part of this research was done while this author visited the Hebrew University and was supported by the Leibniz Center

Offprint requests to: J. Gonczarowski

(this is termed "symbol-freeness" in [11]). In [3], a particular unordered scattered language was shown to be NP-complete.

It is known that each scattered context language is context-sensitive. The inverse containment is, however, a major open problem. In Sect. 3, we attempt to shed light on this problem by showing that a relaxation of scattered context grammars is equivalent to the context-sensitive ones; we just require the total lengths of the replacement strings to be at least as large as the number of symbols to be rewritten (note that some symbols may be rewritten into the empty word). Other variations of scattered context grammars were already shown to generate all context-sensitive languages, such as scattered context grammars with *appearance checking* [1].

In Sect. 4, we show that the non-uniform membership problem for unordered scattered context languages (see, e.g., [13]) is in NP (we recall that no erasing productions are allowed). It follows that the problem is NP-complete as it was shown in [3] that there exists a particular unordered scattered context language for which non-uniform membership is NP-complete.

We also show that unary unordered scattered context languages are in P by simple dynamic programming. Then we prove the decidability of the membership and emptiness problems for unordered scattered context languages with erasing, via a reduction to vector addition system reachability [9, 12]. As a consequence, we solve two open problems (see, e.g., [13]) for the study of rewriting mechanism: (i) there are context-sensitive languages which are not unordered scattered context with erasing, (ii) there exist scattered context languages which are not unordered scattered with erasing.

The paper is concluded by a summary.

2. Basic Notions and Definitions

We assume the reader to be familiar with basic Formal Language Theory, as, e.g., in the scope of [8] and [13]. Some notions need, perhaps, an additional explanation. An *alphabet* Σ is a finite set of symbols. A *word* w is a finite sequence of symbols, and $|w|$ stands for its length. The empty word is denoted by λ. A *language* is a set of words. The reflexive and transitive closure of a language X is denoted by the *Kleene Star* and is written as X^*; X^+ denotes $X \cdot X^*$. We will identify a singleton set $\{a\}$ with its element a whenever this does not cause confusion. The cardinality of a set X is denoted by $\#X$.

Definition. A *scattered context* (SC) grammar G is a quadruple $\langle \Sigma, P, S, \Delta \rangle$, where

- Σ is its finite *alphabet*,
- $\Delta \subseteq \Sigma$ is its *terminal alphabet*,
- $S \in \Sigma \setminus \Delta$ is its *start symbol*,
- P is a finite set of *productions* of the form

$$(A_1, \ldots, A_k) \to (y_1, \ldots, y_k),$$

where $A_i \in \Sigma \setminus \Delta$ and $y_i \in \Sigma^*$, and $|y_i| \geqq 1$, for $1 \leqq i \leqq k$.

G is an *extended scattered context* (ESC) *grammar* if the latter constraint is relaxed to $|y_1 \ldots y_k| \geqq k$. □

Definition. A *context-sensitive* (CS) *grammar* is a quadruple $\langle \Sigma, S, A, \Delta \rangle$, where Σ, S, and Δ are as above, and P is a set of productions of the form $x \to y$, where $x, y \in \Sigma^+$ and $|y| \geqq |x|$. □

We chose this particular way of defining CS grammars to keep our notation as simple as possible. It is easily seen that this definition of CS languages is equivalent to the other standard ones (see, e.g. [13]).

Definition. *Derivations* in a grammar G are sequences of words, such that each word is obtained from the previous one by the application of one production from the grammar:

- A CS derivation may rewrite a word zxz' into a word zyz' if $x \to y$ is a production.
- An SC or ESC derivation may rewrite a word $z_0 A_1 z_1 A_2 z_2 \ldots z_{k-1} A_k z_k$ into $z_0 y_1 z_1 y_2 z_2 \ldots z_{k-1} y_k z_k$ if $(A_1, \ldots, A_k) \to (y_1, \ldots, y_k)$ is a production.

If D is a derivation of a word $y \in \Sigma^*$ from a word $x \in \Sigma^*$ in the grammar G, then we write $x \underset{G}{\overset{*}{\Rightarrow}} y$.

The *language* of a grammar $G = \langle \Sigma, P, S, \Delta \rangle$ is the set

$$\mathbf{L}(G) = \{w \in \Delta^* : S \underset{G}{\overset{*}{\Rightarrow}} w\}.$$

A *sentential form* (of G) is a word y, such that $S \underset{G}{\overset{*}{\Rightarrow}} y$. □

For a given grammar class **G**, the *uniform membership problem* is the following problem:

Input: A word w and a grammar $G \in \mathbf{G}$.

Question: Is $w \in \mathbf{L}(G)$?

The *non-uniform membership problem* is defined similarly. It is, however, required that the grammar G (and thus also $\mathbf{L}(G)$) is fixed (i.e. not part of the input).

Definition. A *vector addition system* (VAS) is a pair $V = \langle Q, \vec{s} \rangle$, where $\vec{s} \in \mathbf{N}_0^n$ for some $n > 0$, and Q is a finite set of vectors in $\mathbf{Z}^n$. A vector $r \in \mathbf{Z}^n$ is *reachable* (*by* V) if there is a sequence of vectors, $\vec{q}_1, \ldots, \vec{q}_m \in Q$, such that

$$\vec{r} = \vec{s} + \vec{q}_1 + \ldots + \vec{q}_m,$$

and

$$\vec{s} + \vec{q}_1 + \ldots + \vec{q}_i \in \mathbf{N}_0^n, \quad \text{for all } 1 \leqq i \leqq m. \quad \square$$

3. The Equality of ESC and CS

In this section we prove the equality of the extended scattered context languages and the context-sensitive languages.

Theorem 1. *A language L is context-sensitive if and only if it is extended scattered context.*

Proof. To prove that every ESC language is also CS, we observe that the lengths of the sentential forms in an ESC derivation are nondecreasing. Hence, the length of the generated word is an upper bound on the work space. It follows that ESC languages can be recognized in linear space, and are thus CS [10]. It remains to show that every CS language is also ESC.

Let $G=\langle \Sigma, P, S, \Delta\rangle$ be a CS grammar. To prove that $\mathbf{L}(G)$ is also ESC, we will simulate a derivation in G as follows. Each step will effect a cyclic shift of the sentential form until the symbol(s) to be rewritten appear at the left end of the shifted sentential form. Then, the symbols are rewritten by erasing them from the left and putting the right-hand sides of the production at the right end of the shifted sentential form. The step is completed by cyclically shifting in the same direction, until the sentential form appears in its natural order.

In the construction we use three markers; the *left marker* L precedes the leftmost position at which a production can be applied; the *middle marker* M follows the rightmost such position, and the *right marker* R marks the right end of the whole sentential form. (Later on, we shall see that we can eliminate the markers). Every sentential form in the ESC derivation will thus be of the form

$$LzMxR,$$

where xz is a sentential form of the CS derivation that we want to simulate. The left marker will guarantee that always the leftmost symbol is rewritten. Otherwise, some symbols will occur to the left of L; but the productions guarantee that those symbols will never be rewritten.

To avoid premature termination of the rewriting process in the middle of a cyclic shift, we use a new alphabet, Σ'; a sential form can only be rewritten into symbols from Σ if there are no symbols between M and R, i.e. if the sentential form is not presently shifted. At this stage, to avoid further applications of "shifting" or "simulating" productions, L is replaced by E. The detailed construction follows.

Let $\Sigma'=\{A': A\in\Sigma\}$ and $\{\hat{S}, E, L, M, R\}$ be alphabets of new symbols. If $x\in\Sigma^*$, then we denote by x' the word in Σ'^* obtained by replacing each $A\in\Sigma$ by A'. Let

$$H=\langle \Sigma\cup\Sigma'\cup\{\hat{S}, L, M, R\}, P', \hat{S}, \Delta\cup\{E, M, R\}\rangle$$

be the ESC grammar where P' consists of the productions described below.

For each production $S \to y$ in G, there is an *initial* production

$$(\hat{S}) \to (Ly'MR).$$

We now simulate the application of a CS production $A_1 \dots A_k \to y$ (where $A_1, \dots, A_k \in \Sigma$) to a sentential form $xA_1 \dots A_k z$ of G. This sentential form is represented as

$$Lx'A_1' \dots A_k' z'MR$$

in H. We first shift the sentential form circularly to its left, using the *shifting* productions

$$(L, A', M, R) \to (\lambda, L, M, A'R), \quad \text{for all } A' \in \Sigma',$$

obtaining the sentential form

$$LA_1' \dots A_k' z'Mx'R.$$

Then we apply the *simulating* production

$$(L, A_1', \dots, A_k', M, R) \to (\lambda, \dots, \lambda, L, M, y'R),$$

obtaining the sentential form

$$Lz'Mx'y'R.$$

Note that $|y'| \geqq k$, because of the constraint on CS productions. It follows that the simulating productions also satisfy the ESC constraints. Then the sentential form is shifted on, using the shifting productions, until we obtain the sentential form

$$LMx'y'z'R.$$

To complete the simulation of the CS derivation step, we "shift" the markers, using the *marker* production

$$(L, M, R) \to (\lambda, L, MR).$$

This produces

$$Lx'y'z'MR,$$

and we can now simulate the next CS production or terminate the simulation of CS derivation steps.

A derivation in the grammar H terminates by replacing each symbol A' by the corresponding symbol A; this is achieved by first entering the termination phase, using the *termination phase* production

$$(L) \to (E),$$

and then the *renaming* productions

$$(E, A', M) \to (\lambda, AE, M) \quad \text{for all } A \in \Sigma.$$

P' will consist of all the productions defined above. It follows easily from the construction of H that for all $w \in \mathbf{L}(G)$, $wEMR \in \mathbf{L}(H)$; observe that there is exactly one simulating production for each original production from the CS grammar.

For the converse containment we shall prove by induction that if $Lz'Mx'R$ is a sentential form of H, then xz is a sentential form of G. This is certainly true after the first derivation step. For the induction we distinguish between the application of shifting, marker, and simulating productions. Assume that we apply a shifting production to the indicated occurrence of A' in the sentential form

$$Lz'_1 A' z'_2 Mx'R.$$

We obtain the sentential form

$$z'_1 Lz'_2 Mx'A'R.$$

If $z'_1 \neq \lambda$, then the sentential form, and in particular z'_1, can never be rewritten to a terminal string, because L (and, later, E) always appears as the leftmost component in a production. Therefore, to successfully derive words in $\mathbf{L}(H)$, shifting productions can only be applied when $z'_1 = \lambda$, i.e. we obtain from

$$LA'z'Mx'R$$

the sentential form

$$Lz'Mx'A'R,$$

and the induction hypothesis holds for shifting productions.

Assume that the marker production is applied to the sentential form

$$Lz'Mx'R.$$

Then we obtain

$$z'LxMR,$$

and z' must be λ for the same reasons as above. The induction hypothesis is thus trivially preserved by an application of the marker production.

Simulating productions have to rewrite a contiguous subword immediately following L, for the same reasons. Moreover, the right hand side of the production is inserted to the left of R. Hence, the induction hypothesis holds also for simulating productions, which completes the induction.

Assume that we have obtained the sentential form

$$Lz'Mx'R.$$

Applying the production $(L)\to(E)$, we obtain

$$Ez'Mx'R.$$

It is easy to see that the renaming productions operate only between E and M; hence, $x'=\lambda$, and the only outcome of the derivation of a word in $\mathbf{L}(H)$ can be the word

$$zEMR.$$

Since $Lz'MR$ is a sentential form in H, z is a sentential form in G, by the said above. Hence, $zEMR\in\mathbf{L}(H)$ implies that $z\in\mathbf{L}(G)$. It follows that

$$\mathbf{L}(H)=\mathbf{L}(G)\cdot\{EMR\}$$

is an ESC language.

To show that $\mathbf{L}(G)$ (i.e. without the markers) is also an ESC language, we use Lemma 1.2 from [4]:

Lemma. *If $L\subseteq\Xi^+$, C is a symbol not in Ξ, and H is an* SC *grammar with $\mathbf{L}(H)=L\cdot C$, then there is an* SC *grammar $\bar{H}$ with $\mathbf{L}(\bar{H})=L$.*

Proof Outline. Let $H=\langle\Sigma, P, S, \Xi\rangle$. Let $\bar{H}=\langle\Sigma\cup\bar{\Sigma}\cup\Phi, \bar{P}, \bar{S}, \Delta\setminus\{C\}\rangle$, where

$$\bar{\Sigma}=\{\bar{A}\colon A\in\Sigma\}$$

$$\Phi=\{[A, B]\colon A, B\in\Sigma\}$$

are sets of new symbols.

We split each derivation into two parts. In the first part, arbitrary component rules can be applied to the rightmost symbol in a sentential form. In the second part, only chain rules can be applied to the rightmost symbol. Throughout the first part, the rightmost symbol in each sentential form occurs barred. This is achieved by adding, for each production

$$(A_1, A_2, \ldots, A_n)\to(w_1, w_2, \ldots, w_n)$$

the production

$$(A_1, A_2, \ldots, \overline{A_n})\to(w_1, w_2, \ldots, w\bar{A})$$

where $w_n=wA$.

In the second part of a derivation, we use a symbol pair at the end of the sentential form; the pair shows the two symbols at the end of the original sentential form in H. The production switching from the first part to the second part of a derivation is given below. Let $A, B, D\in\Sigma$. Let

$$(A_1, A_2, \ldots, A)\to(w_1, w_2, \ldots, w_{n-1}, wDB)\in P.$$

Then we add the production

$$(A_1, A_2, \ldots, \bar{A})\to(w_1, w_2, \ldots, w_{n-1}, w[D, B]).$$

From this point on the sentential form is terminated by a symbol pair, as desired. To simulate rewriting the second symbol to the right, but not the rightmost symbol, we add, for each production

$$(A_1, A_2, \ldots, A_{n-1}, A_n) \to (w_1, w_2, \ldots, w_{n-1}, wD) \in P,$$

the production

$$(A_1, A_2, \ldots, A_{n-1}, [A_n, B]) \to (w_1, w_2, \ldots, w_{n-1}, w[D, B]).$$

To rewrite only the rightmost symbol, we add for the production

$$(A_1, A_2, \ldots, A_{n-1}, A_n) \to (w_1, w_2, \ldots, w_{n-1}, B)$$

the new production

$$(A_1, A_2, \ldots, A_{n-1}, [D, A_n]) \to (w_1, w_2, \ldots, w_{n-1}, [D, B]).$$

To rewrite both symbols, we add for the production

$$(A_1, A_2, \ldots, A_{n-1}, A_n) \to (w_1, w_2, \ldots, w_{n-1}, B)$$

the production

$$(A_1, A_2, \ldots, [A_{n-1}, A_n]) \to (w_1, w_2, \ldots, w[D, B]),$$

where $w_{n-1} = wD$. Finally, the rightmost pair in the derivation is resolved by a production of the form

$$([D, C]) \to (D), \quad \text{for all } D \text{ in } \Delta \setminus \{C\}.$$

The detailed correctness proof of this construction is left to the reader.

We return to the proof of Theorem 1. It is easy to see that this lemma holds also for ESC grammars. Applying it three times, for R, M, and E, we obtain an ESC grammar $\bar{H}$ with

$$\mathbf{L}(\bar{H}) = \mathbf{L}(G).$$

This completes the proof of the theorem. □

Omitting altogether the restriction on the number of symbols in the right hand side of an ESC production (i.e. for the production

$$(A_1, , \ldots, A_k) \to (x_1, \ldots, x_k),$$

the length, $|x_1 \ldots x_k|$, can be arbitrary), we obtain *erasing scattered* (λSC) *grammars*. With those grammars, one can simulate the application of erasing homomorphisms. It follows that the resulting family of languages is RE (the set of recursively enumerable languages), because every RE language is the image of a CS language under some erasing homomorphism. Hence, the membership problem is undecidable for these languages. It turns out, however, that the