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

same problem is decidable for the case where there is no ordering imposed on the application of the individual component rules within a production. This will be proven in the next section.

4. Unordered Scattered Context Grammars

An *unordered scattered context* (USC) *grammar* is a quadruple $\langle \Sigma, P, S, \Delta \rangle$, with the same components as an SC grammar. However, a production $(A_1, \ldots, A_k) \rightarrow (y_1, \ldots, y_k)$ is applied by permuting the A_i and the y_i under the same (arbitrary) permutation σ, and rewriting with the production

$$(A_{\sigma(1)}, \ldots, A_{\sigma(k)}) \rightarrow (y_{\sigma(1)}, \ldots, y_{\sigma(k)}).$$

as in the SC case defined above.

Extended (EUSC) and *erasing* (λUSC) unordered scattered grammars are defined similarly.

In [3] it was already shown that there is a particular USC language for which the non-uniform membership problem is NP-complete, We complement this result by showing that the non-uniform USC membership problem is in NP, by establishing polynomial bounds on USC derivation lengths.
We examine the complexity of USC rewriting.

Theorem 2. *Non-uniform* USC *membership is in* NP.

Proof. We establish a polynomial bound on derivation length using the methods from Lemma 4.2 in [7]. Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a USC grammar, and let $w \in \Delta^*$. We will prove that if $w \in \mathbf{L}(G)$, then there is a derivation of w from S in G of length polynomial in $|w|$. The width of the derivation is bounded by $|w|$, since G is nonerasing. The theorem follows then.

Let D be any derivation of w from S. To prove that there is a derivation of w from S of bounded length, we partition D into "blocks", as in [7]. A *block* is a maximal subderivation of D, such that only productions of the form

$$(A_1, \ldots, A_k) \rightarrow (B_1, \ldots, B_k)$$

are used in the block, and $|B_i| = 1$ for $1 \leqq i \leqq k$.

Clearly, there are at most $|w|$ blocks. It suffices, therefore, to construct for each block **B** an equivalent block **B′** (i.e. **B′** starts and ends with the same words as **B**), of polynomial length. Since the productions are unordered, we can apply them to the commutative images of the sentential forms in **B**. However, to be able to reconstruct a derivation on words from a derivation on commutative images, we must also keep track of the respective symbols in the last word of **B** which are derived from the particular occurrences of symbols in **B**. This is achieved by replacing each symbol A in **B** by the *twin* $[A, B]$, where B is the symbol in the last word of **B** derived from that occurrence of A. The symbols in the last word in **B** will thus be replaced by twins of the form $[B, B]$ only.

We obtain a derivation $\bar{\mathbf{B}}$ over the USC grammar $\bar{G}$ with alphabet

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

and productions

$$\bar{P} = \{([A_1, B_1], \ldots, [A_k, B_k]) \to ([C_1, B_1], \ldots, [C_k, B_k]): (A_1, \ldots, A_k) \to (C_1, \ldots, C_k) \in P\}.$$

The *Parikh mapping* ψ associates each word over Φ with its commutative image, its *Parikh vector*. Let $\psi: \Phi^* \to \mathbf{N}_0^{\#\Phi}$ be defined as follows. Let $\{\pi_1, \ldots, \pi_{\#\Phi}\}$ be a fixed but arbitrary ordering of Φ. Then

$$\psi(\pi_i) = (0, \ldots, 0, 1, 0, \ldots, 0),$$

where the only 1 appears in position i, and

$$\psi(xy) = \psi(x) + \psi(y), \quad \text{for all } x, y \in \Phi^*.$$

We can now define a derivation relation on Parikh vectors. Let $\vec{x}, \vec{y} \in \mathbf{N}_0^{\#\Phi}$. Then $\vec{x}$ *directly derives* $\vec{y}$ if there is a production

$$(\pi_{i_1}, \ldots, \pi_{i_k}) \to (\pi_{j_1}, \ldots, \pi_{j_k}) \in \bar{P},$$

such that:
- $\vec{x} - \psi(\pi_{i_1}, \ldots, \pi_{i_k})$ is nonnegative, and
- $\vec{y} = \vec{x} - \psi(\pi_{i_1}, \ldots, \pi_{i_k}) + \psi(\pi_{j_1}, \ldots, \pi_{j_k})$.

The *derivation relation* is defined as the transitive closure of the direct derivation relation.

Let $A_1, \ldots, A_n, B_1, \ldots, B_n \in \Sigma$. Similarly as in Lemma 4.2 from [6] it can be shown that there is a derivation of $B_1 \ldots B_n$ from $A_1 \ldots A_n$ in G if and only if there is a derivation of the Parikh vector $\psi([B_1, B_1] \ldots [B_n, B_n])$ from $\psi([A_1, B_1] \ldots [A_n, B_n])$, of the same length. It remains thus to present an equivalent Parikh vector derivation, of polynomial length, to show that there is a block $\mathbf{B}'$ of polynomial length.

Let thus $\vec{\mathbf{B}}$ be the Parikh vector derivation obtained from $\mathbf{B}$ by applying ψ to each sentential from in $\mathbf{B}$. If n is the width of $\mathbf{B}$, then $n \leqq |w|$, and the total number of Parikh vectors over twins is at most $(n+1)^{\#\Sigma^2}$, which is polynomial in $|w|$. Eliminating duplicates in $\vec{\mathbf{B}}$, we obtain an equivalent Parikh vector derivation $\vec{\mathbf{B}}'$ of length $\leqq (n+1)^{\#\Sigma^2}$, and thus also a derivation $\mathbf{B}'$ of $B_1 \ldots B_n$ from $A_1 \ldots A_n$ of the same length. Hence, the theorem follows. □

We contrast Theorem 2 with the observation that EUSC membership is polynomial for unary languages. This is a direct corollary from the following simple lemma for vector addition systems; we note that in derivations of unary words, sentential forms can be permuted freely, and derivations can thus be represented by Parikh vector derivations.

Lemma 3. *Let $V = \langle Q, \vec{s} \rangle$ be a fixed* VAS, *such that, for all vectors in Q, the sum of their components is nonnegative, and let $\vec{v}$ be an input vector the components*

of wich appear in unary notation. Then the question whether $\vec{v}$ is reachable by V can be decided in polynomial time.

Proof. We construct the set of all those non-negative vectors, reachable from $\vec{s}$, whose sum of components is at most $|\vec{v}|$ ($\|$ denotes the sum of components).

Let $R=\{\vec{s}\}$. Then, for each $\vec{q}\in Q$, we iterate on all the vectors $\vec{r}\in R$; if $\vec{q}+\vec{r}$ is a non-negative vector and is not yet in R, and if $|\vec{q}+\vec{r}|\leqq|\vec{v}|$, then we add $\vec{q}+\vec{r}$ to R. This process is repeated until R does not change any more. The above process is polynomial in $|\vec{v}|$: Let $\vec{v}=(i_1, \ldots, i_n)$. (Recall that the dimension, n, is fixed.) Then there are at most $(|\vec{v}|+1)^n$ choices for vectors in R, which is polynomial. □

Lemma 3 yields the following result.

Corollary 4. EUSC *membership is polynomial for unary languages if the grammar is fixed.* □

Theorem 5. *Membership is decidable for* λUSC.

Proof. Let $G=\langle\Sigma, P, S, \Delta\rangle$ be a λUSC grammar, and let $w\in\Delta^*$. To prove that it is decidable whether $w\in\mathbf{L}(G)$, we reduce λUSC membership to VAS reachability. Since the latter problem is decidable [9, 12], the theorem holds.

The vectors of the VAS should represent the Parikh vectors of sentential forms, and a derivation would correspond to a "legal" sequence of the Parikh vectors of the sentential forms in the derivation (i.e. each vector would be obtained from its predecessor by one addition of a vector in the VAS). A Parikh vector characterizes the communtative image of a word. Unfortunately, if $x\overset{*}{\Rightarrow}w$ in a λUSC, then permutations of x do not necessarily derive w as well. Abstracting to Parikh vectors will thus cause loss of crucial information. To remedy this deficiency, we encode the final word w into the grammar, and we derive the empty word λ instead of deriving w. Note that now

$$x\overset{*}{\Rightarrow}w \quad \text{if and only if} \quad x'\overset{*}{\Rightarrow}\lambda$$

for all permutations x' of x, since unordered grammar productions can be applied without respect to the order of occurrence of symbols in sentential forms.

In detail, we replace each symbol $A\in\Sigma$ by a triple $[i, A, j]$. These components indicate that $[i, A, j]$ is to derive the (possibly empty) subword of w from position i to position $j-1$. Performing the standard regular-intersection construction (see, e.g., [8]) we obtain from G the λUSC grammar

$$\bar{G}=\langle\bar{\Sigma}, \bar{P}, [1, S, |w|+1], \{\}\rangle,$$

where

$$\bar{\Sigma}=\{[i, A, j]: 1\leqq i\leqq j\leqq|w|+1, \text{ and } A\in\Sigma\},$$

and, for each production $(A_1, \ldots, A_k)\to(x_1, \ldots, x_k)\in P$, $\bar{P}$ contains all the productions obtained by replacing each component rule $A_l\to x_l$, as follows:

If $x_l = \lambda$, then we replace it by a rule from the set

$$\{[i, A_l, i] \to \lambda : 1 \leqq i \leqq |w|+1\}.$$

If $x_l = a$ and $a \in \Delta$, and a occurs in w in positions $i_1, \ldots, i_m$, then replace the component rule $A_l \to x_l$ by a rule from the set

$$\{[i, A_l, i+1] \to \lambda : i \in \{i_1, \ldots, i_m\}\}.$$

Otherwise, if $x_l = B_1 \ldots B_m$, where $m \geqq 1$ and $B_1, \ldots, B_m \in \Sigma$, we replace $A_l \to x_l$ by a rule from the set

$$\{[i_0, A_l, i_m] \to [i_0, B_1, i_1][i_1, B_2, i_2] \ldots [i_{m-1}, B_m i_m] : 1 \leqq i_0 \leqq i_1 \leqq \ldots \leqq i_m \leqq |w|+1\}.$$

It is straightforward to see that $w \in \mathbf{L}(G)$ if and only if $\lambda \in \mathbf{L}(\bar{G})$.

We now construct from $\bar{G}$ a VAS $V = \langle Q, \vec{v} \rangle$ and a homomorphism ϕ, such that $\phi(x)$ is reachable by V if and only if x is a sentential form of $\bar{G}$. The VAS analog of a production involves two steps: First, the symbols in the left hand side of the production must be "subtracted" from the vector, and then the symbols in the right hand side must be added. These steps must be synchronized such that no other addition or subtraction step can be applied in between.

To achieve the desired synchronization, we proceed as follows. Let ψ be a Parikh mapping for $\bar{\Sigma}$, let χ be a Parikh mapping for $\bar{P}$. The VAS V has dimension $\#\bar{\Sigma} + \#\bar{P} + 1$. The first $\#\bar{\Sigma}$ components of a vector correspond to the letters of $\bar{\Sigma}$. The next $\#\bar{P}$ components correspond to the productions of $\bar{P}$; a nonzero component will indicate the production the left hand side of which was just applied. This component will be zeroed after the right hand side of that production was applied as well. The last component is either 1 or 0. A zero indicates that we have applied a left hand side and we are to apply a right hand side.

We denote vector concatenation by $\times$. Then the start vector is

$$\vec{v} = \psi([1, S, |w|+1]) \times (0, 0, \ldots, 0, 1).$$

Q contains two vectors for each production

$$\pi = (C_1, \ldots, C_k) \to (y_1, \ldots, y_k)$$

in $\bar{P}$: The *left hand* vector is

$$(-\psi(C_1 \ldots C_k)) \times \chi(\pi) \times (-1).$$

We first note that no two left hand vectors can be added in succession because the last component becomes zero after applying one such vector. The occurrence of 1 in the position of π in $\chi(\pi)$ guarantees that, as the next step, the corresponding *right hand* vector must be applied:

$$\psi(y_1 \ldots y_k) \times (-\chi(\pi)) \times (1).$$

Let ϕ be the homomorphism defined by

$$\phi(x)=\psi(x)\times(0, 0, \ldots, 0, 1).$$

It is straightforward that ϕ satisfies the requirements, i.e., for all $x\in\bar{\Sigma}$, x is a sentential form of $\bar{G}$ if and only if $\phi(x)$ is reachable by V. It follows that $\lambda\in\mathbf{L}(\bar{G})$ if and only if $\phi(\lambda)$ is reachable by V. The theorem follows from the decidability for reachability in VAS. □

Note that the above proof method cannot be generalized to λSC grammars; there, the order of occurrence of the symbols to which a production is applied is relevant. Parikh vectors can thus not be used, as they do not keep track of that order.

Corollary 6. *Emptiness is decidable for* λUSC *grammars.*

Proof. Let G be a λUSC grammar. We replace every terminal symbol in G by the empty word, obtaining a new λUSC grammar G_λ. However, $\mathbf{L}(G)\neq\varnothing$ if and only if $\lambda\in\mathbf{L}(G_\lambda)$. By Theorem 5, this is decidable, and the corollary follows. □

We define a new decision problem that will be used to distinguish ordered scattered rewriting from erasing unordered scattered rewriting.

Definition. Let G be a fixed grammar with terminal alphabet Δ, and let $a\in\Delta$. The *padded non-uniform membership* problem for G (and a) is defined as follows:

Input: A word $w\in(\Delta\setminus\{a\})^*$.

Question: Is there a word

$$a^{i_0}w_1a^{i_1}w_2\ldots w_na^{i_n}$$

in $\mathbf{L}(G)$, where $w=w_1w_2\ldots w_n$. □

Theorem 7. *The family of* λUSC *languages does not contain all* SC *languages.*

Proof. We shall construct an SC grammar which generates the padded version L_a of an undecidable language, L. The proof will proceed by contradiction, assuming that L_a is a λUSC language. It will be proven that L must then also be λUSC, which is a contradiction to the decidability of λUSC membership, as shown in Theorem 5.

Let U be a universal Turing machine. Since λSC is exactly the family of all recursively enumerable languages (see Sect. 3), computations of U can be simulated by some fixed λSC grammar H, such that a word w is accepted by U if and only if $w\in\mathbf{L}(H)$. Obviously, the membership problem for $\mathbf{L}(H)$ is undecidable.

Pad the λSC grammar H into an SC grammar H_a by replacing each component erasing production $B\to\lambda$ by the production $B\to a$, where a is a new terminal symbol. Then w is in $\mathbf{L}(H)$ if and only if there is a word obtained from w by padding with occurrences of a, which is in $\mathbf{L}(H_a)$.

Let us assume on the contrary that there is a λUSC grammar G_a such that $\mathbf{L}(G_a)=\mathbf{L}(H_a)$. Replacing each occurrence of a in a production by the empty word, we obtain the λUSC grammar G. Obviously, $w\in\mathbf{L}(G)$ if and only if w is accepted by the Turing Machine U, and thus membership for $\mathbf{L}(G)$ is undecidable. This is a contradiction to Theorem 5.

It follows that the language $\mathbf{L}(H_a)$ is an SC language but not a λUSC language. □

The following open problems from [13] are implied by this theorem:

Corollary 8. *The family of* USC *languages is a proper subset of the* SC *languages, and* λUSC *is a proper subset of* λSC (=RE).

Proof. It can easily be seen that the families of USC and λUSC languages are subsets of the families of SC and λSC languages, respectively. Given a (erasing or non-erasing) USC grammar, G, one can construct, for every scattered production, all the possible permutations of component rules. The resulting SC grammar, H, clearly satisfies that $\mathbf{L}(H)=\mathbf{L}(G)$. The properness follows from Theorem 7 and its proof. □

Corollary 8. *The family of* λUSC *languages does not contain all* CS *languages.*

Proof. Immediate from Theorem 7, as all scattered context languages are context-sensitive. □

5. Summary

We have carried on the investigation into the generative power of scattered context grammars, and have obtained the following classification results (here, $\mathbf{L}(X)$ denotes the family of X languages):

- $\mathbf{L}$(ESC)=$\mathbf{L}$(CS),
- $\mathbf{L}$(USC) is a proper subset of $\mathbf{L}$(SC) and $\mathbf{L}(\lambda\text{USC})$ is a proper subset of $\mathbf{L}(\lambda\text{SC})$
- $\mathbf{L}$(SC) (and thus also $\mathbf{L}$(CS)) is not a subset of $\mathbf{L}(\lambda\text{USC})$
- $\mathbf{L}$(USC) is in NP (and, by [DW], contains NP-complete languages).

The two problems that remain open are

- the equality of $\mathbf{L}$(CS) and $\mathbf{L}$(SC), and
- the inclusion of $\mathbf{L}(\lambda\text{USC})$ in $\mathbf{L}$(CS).

References

1. Cremers, A.B.: Normal Forms for Context-Sensitive Grammars. Acta Inf. **3**, 59–73 (1973)
2. Dahlhaus, E., Gaifman, H.: Concerning 2-Adjacent Context-Free Languages. Theor. Comput. Sci. **41**, 169–184 (1985)
3. Dahlhaus, E., Warmuth, M.K.: Membership for Growing Context-Sensitive Languages is Polynomial. J. Comput. Syst. Sci. **33**, 456–472 (1986)
4. Greibach, S., Hopcroft, J.: Scattered Context Grammars. J. Comput. Syst. Sci. **3**, 233–247 (1969)
5. Gonczarowski, J., Shamir, E.: Pattern Selector Grammars and Several Parsing Algorithms in the Context-Free Style. J. Comput. Syst. Sci. **30**, 249–273 (1985)

6. Gonczarowski, J., Warmuth, M.K.: Applications of Scheduling Theory to Formal Language Theory. Theor. Comput. Sci. **37**, 217–243 (1985)
7. Gonczarowski, J., Warmuth, M.K.: Manipulating Derivation Forests by Scheduling Techniques. Theor. Comput. Sci. **45**, 87–119 (1986)
8. Harrison, M.A.: Introduction to Formal Language Theory. Reading, MA: Addison-Wesley 1978
9. Kosaraju, S.R.: Decidability of Reachability in Vector Addition Systems. Proc. 14th ACM Symp. on Theory of Computing (1982), pp. 267–281
10. Kuroda, S.-Y.: Classes of Languages and Linear Bounded Automata. Inf. Control **6**, 131–136 (1963)
11. Kleijn, H.C.M., Rozenberg, G.: Context-Free Like Restrictions on Selective Rewriting. Theor. Comput. Sci. **16**, 237–269 (1981)
12. Mayr, E.W.: An Algorithm for the General Petri Net Reachability Problem. SIAM J. Comput. **13**, 441–460 (1984)
13. Salomaa, A.: Formal Languages. New York: Academic Press 1973

Received December 15, 1987 / June 13, 1989