Reprinted from JOURNAL OF COMPUTER AND SYSTEM SCIENCES
All Rights Reserved by Academic Press, New York and London

Vol. 33, No. 3, December 1986
Printed in Belgium

Membership for Growing Context-Sensitive Grammars Is Polynomial*

ELIAS DAHLHAUS

Technische Universität Berlin, Fachbereich Mathematik, Strasse des 17. Juni 135, 1000 Berlin, West 12, Germany

AND

MANFRED K. WARMUTH

Department of Computer and Information Sciences, Applied Sciences Building, University of California, Santa Cruz, California 95064

Received July 1, 1985; revised March 10, 1986

A context-sensitive grammar is a *growing* context-sensitive grammar, if the right-hand side of every production is strictly longer than the left-hand side. We show that for any fixed growing context sensitive grammar, the membership problem for the corresponding language is polynomial. © 1986 Academic Press, Inc.

1. INTRODUCTION

Context-sensitive grammars (csgs) are one of the classical grammar families of formal language theory. They were introduced in [Ch59] and have been studied extensively since then (see [Bo73, Ha78] for an overview). Context-sensitive grammars are defined as rewriting systems, where the length of the right-hand side of every production is at least as large as the length of the left-hand side. This restriction on the productions is responsible for the fact that the question of membership for context-sensitive languages (csls) is equivalent to the question of acceptance for nondeterministic linear bounded automata [Ku64]. Therefore membership for csls is PSPACE complete [Ka72] and this is true even for certain fixed grammars. In this paper we show that if we restrict ourselves to "growing" productions, i.e., the right-hand side of every production is strictly longer than the left-hand side, then membership for fixed csls is polynomial.

This may appear surprising in view of the results obtained in [Bo78]. The growing csls are a subclass of $\text{LINEAR}_{\text{CS}}$ as defined in [Gl64, Bo71]. Languages of

* This research was done while the authors were visiting the Hebrew University of Jerusalem. The first author was supported by the Minerva Foundation and the second author by the United States–Israel Binational Foundation Grant 2439-82. A preliminary version of this appeared in the CAAP 86 Proceedings [DW86].

$\text{LINEAR}_{\text{CS}}$ are given by an arbitrary csg which has the property that every word w in the language has a derivation of length at most $c\,|w|$,[1] for some overall constant c, which only depends on the grammar. In [Bo78] it was shown that there are NP-complete languages in $\text{LINEAR}_{\text{CS}}$. Thus our result that the family of growing csls is in P deserves an explanation.

Observe that in $\text{LINEAR}_{\text{CS}}$ "complex derivations" are allowed using nongrowing productions; then the final word may be padded such that the length of the overall derivation is linear in the length of the final word. In fact, for every language in P there is a polynomially padded version of this language which is in $\text{LINEAR}_{\text{CS}}$ [Bo78].

An arbitrary csg may be converted into a growing csg by adding a dummy symbol to the right-hand side of every nongrowing production. The grammar needs to be changed slightly so that the dummy symbols are "ignored." But now padding increases the length of the word exponentially. Each time a "signal" runs from one end of a sentential form to the other, the length increases by a constant factor.

Note that the question of emptiness for csls is undecidable [BPS61]. By padding a csg with dummy symbols a related growing csg is constructed. Clearly, emptiness for the corresponding growing csls is also undecidable. For the question of emptiness, the "exponential padding" is redundant.

The paper is outlined as follows. In Section 2 the basic notations are developed. Given a word w, we want to decide membership for a language defined by some fixed growing csg. A planar directed acyclic graph is associated with every derivation of the grammar. In Section 3 we show that if all productions are growing then there is a path of length $O(\log(|w|)$ from each vertex to some sink (not necessarily the same sink) in the graph. The sinks of the graph are labeled with the word w to be tested for membership. These short paths are then used in Section 4 in a polynomial cut-and-paste algorithm for deciding membership for a growing csl.

In the cut-and-paste algorithm each "piece" of a derivation graph is characterized by a tuple (called a frame) which contains all the essential information about the piece. Because of the short paths there is only a polynomial number of different frames which need to be considered. Frames were used extensively in [GS85, GW85, GW86a] for studying polynomial cases of k-parallel rewriting.

In Section 5 the polynomiality of the membership problem for fixed, growing csls is contrasted with the fact that there are NP-complete languages defined by fixed, growing scattered grammars. In scattered grammars (scgs) [GH68, GW86b] the symbols to be rewritten in parallel are not required to be adjacent. In every production each symbol on the left-hand side is rewritten into a string of length at least one. In growing scgs each symbol must be rewritten into a string of length strictly bigger than one. It is easy to see that in the derivation trees of growing scgs each node has an $O(\log(|w|))$ path to a leaf, where w is the word to be parsed. But since for scattered grammars the rewritten symbols do not need to be adjacent, we cannot cut and paste the derivation trees along the short paths.

[1] $|w|$ denotes the length of w.

The parallel complexity and the space complexity of the membership problem for fixed growing csls is discussed in the conclusion section. The main open problem is to determine the complexity of membership for "variable" growing csls, i.e., not only the word to be tested but also the growing csg is a variable of the input. The question is whether this problem can be solved in polynomial time or whether it is NP-complete.

2. Preliminaries

A *context-sensitive grammar* (csg) G is a quadruple (V, Σ, P, S) where:

(i) V is a finite set of symbols, Σ is the subset of V which contains the terminal symbols, and S is the startsymbol in $V-\Sigma$.

(ii) P is a finite set of productions of the form $\alpha \to \beta$, s.t. $\alpha, \beta \in V^+$ and $|\alpha| \leqslant |\beta|$. For two words u and v in V^*, *u derives v*, denoted $u \Rightarrow v$, if there exist $x, y, \alpha, \beta \in V^*$ s.t. $u = x\alpha y$, $v = x\beta y$ and $\alpha \to \beta \in P$. Let $\overset{*}{\Rightarrow}$ denote the reflexive and transitive closure of $\Rightarrow$. Using this notation we are ready to describe the *context-sensitive language* (csl) defined by the csg G: $L(G) = \{w \mid S \overset{*}{\Rightarrow} w \text{ and } w \in \Sigma^*\}$.

Now the *membership problem* for a csl $L(G)$ is defined as follows:

Input: a word $w \in \Sigma^*$, where Σ is the terminal alphabet of G;
Question: is $w \in L(G)$?

Note that G is fixed, i.e., it is not a variable of the input. There are fixed csgs for which this problem is PSPACE complete [Ku64, Ka72].

We restrict ourselves to a subclass of csgs for which the membership problem is in P (Sect. 4). A csg G is *growing* if for all productions $\alpha \to \beta$ of the grammar, $|\alpha| < |\beta|$.

Following [Lo70] each *derivation* is associated with a planar directed acyclic graph called a (*derivation*) *graph*. The vertices in such a graph will be labeled with the corresponding symbols and productions used in the derivation. Let $\omega(x)$ denote the label of vertex x, where ω is a function from the set of vertices of the derivation graph to $V \cup P$. Vertices labeled with symbols (resp. productions) are called *symbol* (resp. *production*) *vertices*. We inductively define the *derivation graph* $D_k = (V_k, E_k)$ which is associated with the derivation $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_k$:

Case $k = 1$. Let $\alpha_1 = a_1 a_2 \cdots a_p$, where $a_i \in V$. Then $D_1 = (V_1, E_1)$ has the vertices $V_1 = \{x_1, x_2, ..., x_p\}$ s.t. $\omega(x_i) = a_i$ and no edges, i.e., E_1 is empty.

Case $k > 1$. Assume $\alpha_1 \Rightarrow \alpha_2 \cdots \Rightarrow \alpha_{k-1}$ corresponds to the graph $D_{k-1} = (V_{k-1}, E_{k-1})$ and $\alpha_{k-1} = u\alpha v \Rightarrow u\beta v = \alpha_k$. From D_{k-1} and the production $\alpha \to \beta$ the graph D_k is constructed for $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_k$. For the word $\beta = b_1 b_2 \cdots b_q$ create the vertices $V_\beta = \{y_i \mid 1 \leqslant i \leqslant q\}$ and for the production $\alpha \to \beta$ create an additional vertex y. Choose the vertices s.t. V_{k-1}, V_β and $\{y\}$ are distinct. The vertices of V_β

are labeled with the symbols of β, i.e., $\omega(y_i)=b_i$, and y is labeled with the production $\alpha \to \beta$. Let V_α be the sinks (symbol vertices) of D_{k-1} corresponding to α. Now $V_k = V_{k-1} \cup V_\beta \cup \{y\}$ and $E_k = E_{k-1} \cup V_\alpha \times \{y\} \cup \{y\} \times V_\beta$.

An example is given in Fig. 1. The planarity of the derivation graphs follows from the fact that only sinks are connected to the new production vertex. The sources of the graph D_k correspond to α_1 and the sinks to α_k. We say that D_k *derives* α_k. Since the graph is planar there is a natural left to right order amongst the sources: let $\alpha_1 = a_1 a_2 \cdots a_p$, then for $1 \leqslant i < j \leqslant p$ the vertex corresponding to a_i is to the *left* of the vertex of a_j and the vertex of a_j is to the *right* of a_i. Similarly, there is a natural left to right order amongst the sinks of a derivation graph, and amongst the predecessors and successor of every production vertex. Two sources (sinks) are called *adjacent* if they are adjacent in the left to right order of the sources (sinks).

In a derivation graph D a *path* π is defined to be a sequence $x_1, x_2, \dots, x_{e+1}$ of vertices of D. The paths π *starts* at x_1, *finishes* x_{e+1} and has *length* e. Note that a path which contains one vertex has length zero. In this paper we assume that non-empty paths always end at a sink of D. The *distance* $d_x(y)$ of y from x is the length of the shortest paths which start at x and finish at y. Note that $d_x(x)=0$.

In the following lemma we will to show that there exists a shortest paths from each vertex to some sink s.t. no pair of paths is "crossing." To construct such a set of paths we use the following definition of consistency. Two paths are *consistent* if they have no common vertices, or if starting from the first common vertex the paths are identical. A set of paths is consistent if each pair is. Note that since derivation graphs are planar two consistent paths cannot "cross."

LEMMA 1. *For any derivation graph there exists a set of shortest paths from all vertices to sinks such that this set of paths is consistent.*

Proof. Let v_i, for $1 \leqslant i \leqslant m$, be the vertices of a derivation graph D and let π_i be a shortest path starting at v_i (and finishing at a sink of D). We now inductively construct paths π_i' (for $1 \leqslant i \leqslant m$), where π_i' starts at v_i, s.t. $\{\pi_j' \mid 1 \leqslant j \leqslant i\}$ is a consistent set of shortest paths.

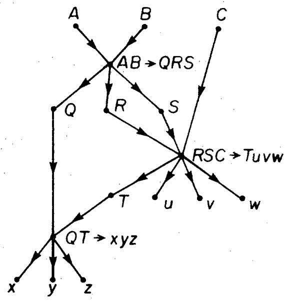


FIG. 1. The derivation graph corresponding to the derivation $\underline{AB}C \Rightarrow Q\underline{RSC} \Rightarrow \underline{QT}uvw \Rightarrow xyzuvw$ (the rewritten symbols are underlined).

Assume the set $\{\pi'_j \mid 1 \leqslant j \leqslant \bar{m} < m\} = \Pi'$ is consistent. If $\pi_{\bar{m}+1}$ is consistent with Π' we set $\pi'_{\bar{m}+1} = \pi_{\bar{m}+1}$ and there is nothing to show. Otherwise, let x be the first common vertex of $\pi_{\bar{m}+1}$ with some path π' of Π'. Since both π' and $\pi_{\bar{m}+1}$ are shortest paths, the suffixes of π' and $\pi_{\bar{m}+1}$ which start with x have the same length. Let $\pi'_{\bar{m}+1}$ be the path which agrees with $\pi_{\bar{m}+1}$ up until x and then follows π' to the sink. Clearly, $\pi'_{\bar{m}+1}$ has the same length as $\pi_{\bar{m}+1}$ and is consistent with Π'. This completes the description of the inductive construction. ∎

3. Short Paths in Derivation Graphs

Consider derivation graphs for a growing csg which derive a word w. In this section we show that in such graphs there is a path of length $O(\log(|w|))$ from each vertex to a sink. We prove this by assigning weights to the vertices, s.t. big weights will correspond to short paths.

Let us first discuss why short paths do not always exist for derivation graphs of grammars which define languages in $\text{LINEAR}_{\text{CS}}$. In [Gl64] it was shown that $L = \{ucu^Rcu: u \in \{a, b\}^*\}$ is not $\text{LINEAR}_{\text{CS}}$.[2] Since growing csls are a subclass of $\text{LINEAR}_{\text{CS}}$ [Bo73] the language L is not a growing csl. Intuitively, only $O(\log(|w|))$ bits can be transmitted across paths of length $O(\log(|w|))$. But in L, $O(|w|)$ bits need to be transmitted to synchronize the production of the words u, u^R and u in $w = ucu^Rcu$.

It is crucial that in the definition of L the word u is over a two symbol alphabet. Just producing three blocks of equal size as in the language $L' = \{a^nb^nc^n: n \geqslant 1\}$ is much easier. One can show that L' is a growing csl. In L' only $O(\log(|w|))$ bits need to be transmitted. It is easy to see that $\hat{L} = \{a^{2n}b^{2n}c^{2n}: n \geqslant 0\}$ is a growing csl. We let a special symbol scan the word. During each complete scan the number of symbols a, b, and c is doubled. From this it is easy to see that L' is also a growing csl. To produce the word $a^nb^nc^n$, $\lfloor \log n \rfloor$ scans are used. Each scan corresponds to a bit in the bit representation of n. Again we double the number of symbols in each scan, but we also add an additional symbol if the corresponding bit is one.

We mentioned already in the introduction that for every language in P there is a padded version [Bo71] which is in $\text{LINEAR}_{\text{CS}}$. Thus even though $\{ucu^Rcu: u \in \{a, b\}^*\}$ is not in $\text{LINEAR}_{\text{CS}}$, the language $\{ucu^Rcud^{(|u|)^2}: u \in \{a, b\}^*\}$ is.

We proceed to prove the existence of short paths in derivation graphs of a growing csg. Let D be such a derivation graph deriving the word w. Each vertex x of D is associated with a subgraph of D. Let D_x be the subgraph induced by all vertices reachable from x.

The length of the paths will depend on the *growth ratios* of the productions in the grammar. The growth ratio of a production $\alpha \rightarrow \beta$ is the ratio $|\beta|/|\alpha|$. The minimum growth ratio of all productions of a grammar is the growth ratio of the grammar.

[2] The word u^R denotes the reverse of the word u.

Throughout the paper this minimum is denoted by g. Note that $g > 1$ for growing csgs.

The growth ratio of a production vertex is the ratio between the number of immediate successors over the number of immediate predecessors. Thus g is a lower bound on the growth ratios of the production vertices of D. Since each production vertex of D_x has at least as many immediate predecessors and the same number of immediate successors in D as in D_x, the growth ratios of the production vertices of D_x are also bounded by g.

We now assign weights $t_x(\cdot)$ to the vertices of D_x according to the following scheme:

(i) $t_x(x) = 1$.

(ii) For a production vertex p, the weight $t_x(p)$ is the sum of all the weights of the immediate predecessors of p.

(iii) If p is a production vertex with k immediate successors, then each of these receives a weight of $t_x(p)/k$.

Note that $\Sigma_{s \text{ is sink of } D_x} t_x(s) = 1$. Since D_x has at most $|w|$ sinks, there is a sink in D_x of weight at least $1/|w|$.

The following lemma shows that big weights correspond to short paths.

LEMMA 2. *Let y be a symbol vertex of D_x and let d be a non-negative integer. If $t_x(y) \geqslant g^{-d}$ then $d_x(y) \leqslant 2d$.*

Proof. We prove this by an induction on d. The base case of $d = 0$ is trivial. Assume the lemma holds for all $d' < d$ and let y be a symbol vertex of D_x s.t. $1 > t_x(y) \geqslant g^{-d}$. Let p be the production vertex which precedes y. Assume p has a immediate predecessors and b immediate successors. Since $t_x(p) = b \cdot t_x(y)$, p must have an immediate predecessor y' with weight at least $(b/a)\, t_x(y)$. By the above remarks $t_x(y') \geqslant g \cdot t_x(y) \geqslant g^{1-d}$. Applying the inductive hypothesis it follows that $d_x(y') \leqslant 2d - 2$ and $d_x(y) \leqslant 2d$. ▮

Since there is a sink of weight at least $1/|w|$ in D_x, the lemma implies the existence of a path of length at most $2\lceil \log_g(|w|) \rceil$ from x to a sink. For fixed growing csgs this bound is $O(\log(|w|))$ since $g > 1$ and since g only depends on the grammar. Paths are called *bounded* if they are of length at most $2\lceil \log_g(|w|) \rceil$. A derivation graph is *bounded* if there is a consistent set of bounded paths from all vertices to sinks. Combining the above remarks with Lemmas 1 and 2, we get the basis for the polynomial algorithm:

THEOREM 1. *Every derivation graph of a growing* csl *which derives the word w is bounded.*

4. Membership of Growing csl Is Polynomial

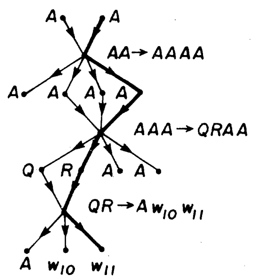
In the previous section we showed there are short paths from all vertices to sinks in derivation graphs. We now use these paths to "cut" derivation graphs into "pieces." Each piece is bordered on the left and on the right by a path of length $O(\log(|w|))$. There is an exponential number of derivation graphs and pieces. We therefore gather the essential information about a piece in a frame. Part of this information will be a description of the left and the right path. There will be only a polynomial number of valid frames, which are all found by the algorithm. The information gathered in the frames will be sufficient to decide membership. The same technique was used extensively in [GS85, GW85, GW86a] to show that membership for various problems of k-parallel rewriting is polynomial. Also the classic Cocke–Kasami–Younger algorithm [Ka65, Yo67, Ha78] for context-free language membership can be described using a simple notion of frames. We first rewrite the Cocke–Kasami–Younger algorithm using all the essential notions of this section: *frames*, *valid* frames, the valid frames of a derivation tree, *instance* of a valid frame, *basic* frames, the set VAL of all valid frames. This will help the reader who is familiar with the Cocke–Kasami–Younger algorithm to understand this section.

Assume that we are given a context-free grammar in Chomsky normal form [Ha78]. Let w be the word to be tested for membership. A frame is a tuple (A, l, r) s.t. A is a symbol of the alphabet and $1 \leqslant l \leqslant r \leqslant |w|$. Note that there is only a polynomial number of frames. A frame (A, l, r) is *valid* if $A \overset{*}{\Rightarrow} w_l w_{l+1} \cdots w_r$ in the context-free grammar. We call the derivation tree that corresponds to such a derivation an instance of (A, l, r). Observe that valid frames parameterize all derivation trees that derive subwords of w. The valid frames of an arbitrary derivation tree D are all frames for which there is a subtree of D that derives a subword of w. It is easy to see that $S \overset{*}{\Rightarrow} w$ iff $(S, 1, |w|)$ is valid. (This corresponds to Theorem 2.)

The Cocke–Kasami–Younger algorithm simply computes the set VAL of all valid frames:

```
    0. Initialize VAL with the set of all basic valid frames: $\{(w_i, i, i): 1 \leqslant i \leqslant |w|\}$.
Repeat
    1. Add the frame $(A, i, i)$ to VAL, if the grammar contains the production $A \to b$ and if $(b, i, i)$ is in
       VAL.
    2. Add the frame $(A, l, r)$ to VAL, if the gramar contains the production $A \to BC$ and if $(B, l, m)$ as
       well as $(C, m, r)$ are in VAL.
Until no new frames can be added to VAL.
```

The membership algorithm for growing context-sensitive languages will follow the same outline. The frames will parameterize pieces of the derivation graphs. We need to be able to describe the paths bordering a piece on the left and on the right. One notation for paths of derivation graphs is given in Fig. 2. The productions are the labels of the production vertices on the path and the numbers specify which successors and predecessors are on the path. These numbers are necessary because for



$2/AA \to AAAA/(4,3)/AAA \to QRAA/(2,2)/QR \to Aw_{10}w_{11}/3$

FIG. 2. The description of a path (in boldface).

a given production $\alpha \to \beta$ in the grammar some symbols might have multiple occurrences in α or β. We could present the algorithm using the notation of Fig. 2 which would be more efficient. But for the sake of simplicity of the presentation we assume that the grammar is in a special form.

A grammar is called a *one-grammar* if for each production $\alpha \to \beta$ in the grammar each symbol of the alphabet occurs at most once in α and at most once in β. Using standard methods of Formal Language Theory, it is easy to construct an equivalent one-grammar for a given grammar by increasing the size of the alphabet and by adding chain productions. For *chain* productions $|\alpha| = |\beta| = 1$ must hold.

In the following we outline the construction of an equivalent one-grammar. Details are left to the reader. Assume there is a production $\alpha \to \beta$ in which some symbol A (terminal or nonterminal) appears twice in α. In this case the two occurrences of A in the production are replaced by two new nonterminals A_1 and A_2. Furthermore two new productions are added to the grammar: $A \to A_1$ and $A \to A_2$. By repeatedly applying the above, double occurences of symbols from the left-hand sides of productions are eliminated. With a similar construction we can eliminate double occurrences from the right-hand sides of productions.

Since for the membership problem we assume that the grammar is fixed, the size of the equivalent one-grammar will also be fixed and independent of the input word to be tested for membership. Observe that the original grammar and the equivalent one-grammar define the same language. Furthermore derivation graphs for the original grammar translate into derivation graphs for the corresponding one-grammar and vice versa. A path in the derivation graph of the one-grammar is at most three times as long as the corresponding path in the "equivalent" derivation of the original grammar.

This motivates the following assumptions for the rest of this section. The fixed growing csg of the membership problem is given by its equivalent one-grammar. Paths in derivation graphs of the latter grammar are *bounded* if they are of length $6\lceil \log_g(|w|) \rceil$ instead of $2\lceil \log_g(|w|) \rceil$. Theorem 1 also holds for the equivalent one-grammars with the new bound. (Recall that g is the growth-ratio of the original growing csg.)

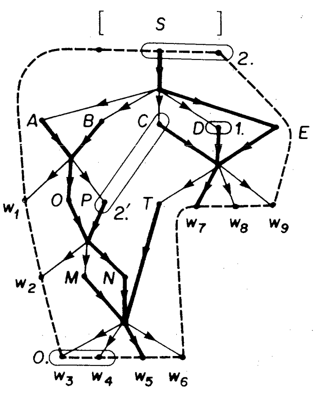
The input word w which is to be tested for membership is denoted as $w_1 w_2 \cdots w_{|w|}$. To get a simple description of the algorithm we add dummy symbols to the beginning and end of w. Let [and] denote two symbols which are not in the alphabet of the grammar. Set $w_0 =$ [and $w_{|w|+1} =$].

To decribe a path $\pi = x_1, x_2, ..., x_e$ in a derivation graph of a one-grammar, it is now sufficient to use the sequence $\omega(x_1)/\omega(x_2)/ \cdots /\omega(x_e)$ which is called the *labeling sequence* of π and is denoted by $\Omega(\pi)$.

Similarly to paths, a labeling sequence is *bounded* if it is of length at most $6\lceil \log_g(|w|) \rceil$. A *frame* is a tuple (t, λ, ρ, l, r) s.t. $t \in V^2 \cup V$, λ and ρ are bounded labeling sequences starting with the first respectively last symbol of t, and $0 \leqslant l \leqslant r \leqslant |w| + 1$.

Intuitively, the above frame specifies a "piece" of a planar derivation graph which might appear in the cut-and-paste process. This piece is bordered on the left (resp. right) by a path labeled with λ (resp. ρ). The piece derives $w_l, w_{l+1}, ..., w_r$, i.e., λ ends at a sink labeled with w_l, ρ ends at a sink labeled with w_r and the sinks in between are labeled accordingly. The word t specifies how the "piece" starts. If the left and right path start at the same vertex (*unary* frame) then t is the label of that vertex. In the case where the paths start at different vertices (*binary* frame), t consists of the labels of both vertices. See Fig. 3 for examples. The polynomial running time of the membership algorithm for fixed growing csls heavily relies on the fact that the number of frames is polynomial in $|w|$. Note that there is only a polynomial number of labeling sequences of bounded paths (length up to $6\lceil \log_g(|w|) \rceil$) since g is a positive constant.

Not every frame corresponds to a piece of a derivation graph, only valid frames do. A frame is *valid* if and only if it is a valid frame w.r.t. a bounded derivation



0. $(w_3 w_4, w_3, w_4, 3, 4)$
1. $(D, D/CDE \rightarrow Tw_7 w_8 w_9/w_7, D/CDE \rightarrow Tw_7 w_8 w_9/w_7, 7, 7)$
2. $(S], S/S \rightarrow ABCDE/E/CDE \rightarrow Tw_7 w_8 w_9/w_7,], 7, 10)$
2'. $(PC, P/OP \rightarrow w_2 MN/N/MNT \rightarrow w_3 w_4 w_5 w_6/w_5,$ $C/CDE \rightarrow Tw_7 w_8 w_9/w_7, 5, 7)$

FIG. 3. Some valid frames with respect to a derivation graph and a set of bounded consistent paths (in boldface); the symbols of the first component of each frame are encircled.

graph D and a consistent set of bounded paths $\Pi = \{\pi_y \mid \pi_y$ starts with the symbol vertex y of $D\}$ from each symbol vertex of D to a sink.[3]

DEFINITION. The valid frames of (D, Π) are given as follows:

(1) The unary frame $(\omega(v), \Omega(\pi_v), \Omega(\pi_v), l, l)$ is valid if

(i) v is a symbol vertex of D;

(ii) π_v ends at a sink labeled with w_l.

(2) The binary frame $(\omega(u)\,\omega(v), \Omega(\pi_u), \Omega(\pi_v), l, r)$ is valid if

(i) u and v are symbol vertices of D s.t. adding the edge (u, v) to D does not violate the planarity of D;

(ii) the edge (v, w) does not leave the planar circle which encloses all edges of D and is defined by the edges between adjacent sources, the edge between the rightmost source and the rightmost sink, the edges between adjacent sinks, and the edge between the leftmost sink and the leftmost source (see broken circle of Fig. 3);

(iii) there is no path from u to v and vice versa;

(iv) π_u ends at some sink s;

(v) the $r-l+1$ sinks starting from s going to the right are labeled with $w_l, w_{l+1}, \ldots, w_r$;

(vi) the $(r-l+1)$th such sink is the one at which π_v ends.

There are many valid frames beloning to (D, Π). For a particular frame we want to specify the subgraphs of derivation graphs which correspond to that frame. Let $F = (t, \lambda, \rho, l, r)$ be a valid frame of some tuple (D, Π). If F is unary then $\rho = \lambda$ and the path of vertices in D which corresponds to λ is an *instance* of F. In the case where F is binary then the subgraph I induced by the vertices v of D for which the following conditions hold is called an *instance* of the frame F:

(i) v has a predecessor amongst the two vertices corresponding to t;

(ii) π_v ends at a sink corresponding to w_m, where $l \leqslant m \leqslant r$;

(iii) if v is not on the path corresponding to λ but π_v and λ have some vertex x as their first common vertex, then the predecessor of x on λ is he left of the predecessor of x on π_v;

(iv) if v is not on the path ρ but π_v and ρ have some vertex x as their first common vertex, then the predecessor of x on ρ is to the right of the predecessor of x on π_v.

Intuitively I consists of all vertices of D "below" T, to the "right" of λ and to the "left" of ρ. Applying the above definition of valid frames gives the following equivalence.

[3] Note that D does not necessarily derive the whole word w, but in the case where w and the word derived by D have no subword in common then no valid frames belong to D.

THEOREM 2. *$S \overset{*}{\Rightarrow} w$ if and only if there are two valid frames* $([S, [, \mu, 0, m)$ *and* $(S], \mu,], m, |w|+1)$, *for some bounded path μ and $0 \leqslant m \leqslant |w|+1$.*

Proof. Assume $S \overset{*}{\Rightarrow} w$. Theorem 1 implies the existence of a bounded derivation graph for $S \overset{*}{\Rightarrow} w$. By adding two vertices corresponding to the dummy symbols [and] one gets a bounded derivation graph D for $[S] \overset{*}{\Rightarrow} [w]$. Let Π be some bounded set of consistent paths of D as defined above. Furthermore let v be the source of D which is labeled with S and assume π_v ends at the mth symbol of w. From the above definition it follows that $([S, [, \Omega(\pi_v), 0, m)$ and $(S], \Omega(\pi_v),], m, |w|+1)$ are valid frames of (D, Π).

To prove the converse let I be an instance of $([S, [, \mu, 0, m)$ and J be an instance of $(S], \mu,], m, |w|+1)$. Assume that I and J have distinct sets of vertices. By identifying the vertices on the rightmost path of I with the vertices on the leftmost path of J one can build a derivation graph for $[S] \overset{*}{\Rightarrow} [w]$. Finally removing the two nodes labeled with the dummy symbols [and] leads to a derivation graph for $S \overset{*}{\Rightarrow} w$. ∎

ALGORITHM

```
(*Constructs the set VAL of all valid frames.*):
(*We assume that $|w| \geqslant 3$.*)
0.   Initialize VAL with the sets of basic unary and binary frames
     $\{(w_i, w_i, w_i, i, i): 0 \leqslant i \leqslant |w|+1\} \cup \{(w_i w_{i+1}, w_i, w_{i+1}, i, i+1): 0 \leqslant i \leqslant |w|\}$
Repeat
1.   (*Generating a new valid unary frame from a valid unary frame.*)
     Add the frame $(A_i, A_i/P/\lambda, A_i/P/\lambda, l, l)$ to VAL, if $P = A_1 A_2 \cdots A_k \to B_1 B_2 \cdots B_{k'}$ is a production of
     the grammar and if $(B_j, \lambda, \lambda, l, l)$ is in VAL.
2.1. (*Generating a valid binary frame from a valid unary frame; the two symbol vertices that corre-
     spond to the top of the new frame have the same production vertex as their immediate successor.*)
     Add the frame $(A_i A_{i+1}, A_i/P/\lambda, A_{i+1}/P/\lambda, l, l)$ to VAL, if $P = A_1 A_2 \cdots A_k \to B_1 B_2 \cdots B_{k'}$ is a
     production of the grammar and if $(B_j, \lambda, \lambda, l, l)$ is in VAL.
2.2. (*A new valid binary frame is generated from valid binary frames.*)
     Add the frame $(XA_1, \lambda, A_1/P/\rho_j, l, r_j)$ to VAL, if $P = A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_{k'}$ is a production of
     the grammar and if $(XB_1, \lambda, \rho_1, l, r_1)$ as well as $(B_i B_{i+1}, \rho_i, \rho_{i+1}, r_i, r_{i+1})$, for $1 \leqslant i < j$, are in
     VAL.
2.3. (*Symmetric to Step 2.2.*)
     Add the frame $(A_k Y, A_k/P/\lambda_j, \rho, l_j, r)$ to VAL, if $P = A_1 A_2 \cdots A_k \to B_1 B_2 \cdots B_{k'}$ is a production of
     the grammar and if $(B_i B_{i+1}, \lambda_i, \lambda_{i+1}, l_i, l_{i+1})$, for $j \leqslant i < k'$, as well as $(B_{k'} Y, \lambda_{k'}, \rho, l_{k'}, r)$ are in
     VAL.
Until no new frame can be added to VAL.
```

THEOREM 3. *The algorithm finds exactly all valid frames and can be implemented in polynomial time.*

Proof. The first part of the theorem is proved in two inductions. In Induction 1 we show that all frames in the set VAL of the algorithm are valid according to the above definition. Induction 2 shows that all valid frames are in the set VAL created

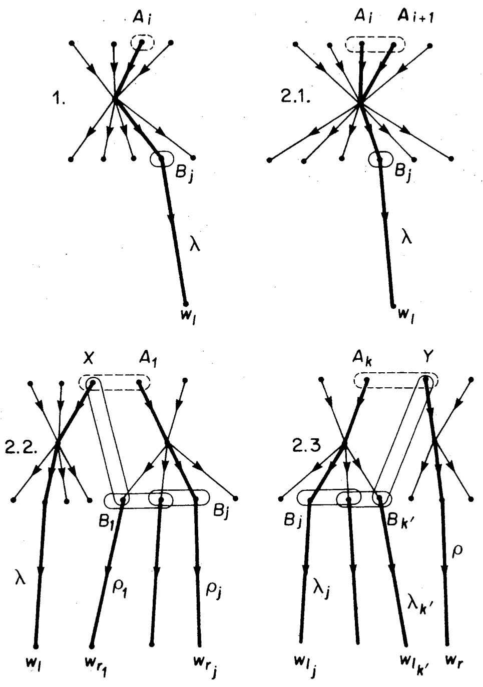


FIG. 4. Schematic description of the Cases 1–2.3 of the algorithm; labeled paths are indicated in boldface and the symbols of the first component of each frame are encircled.

by the algorithm. Note that the above Definition and the Algorithm are outlined in the same way. A schematic description of the algorithm is given in Fig. 4.

Induction 1. Let F be the first frame added to VAL by the algorithm which is not valid according to the above definition. Let R be the set of frames of VAL which caused the algorithm to add F to VAL. Clearly the frames of R are valid. By combining instances for the frames of R one can build an instance for F (see proof of Theorem 2) and this is a contradiction to the assumption that F is not valid. For a complete proof we need to distinguish in which step F was added to VAL and reason in each case that F is valid. We only show this for Step 2.3. The remaining cases are similar.

Let I_i, for $j \leqslant i < k'$, be an instance of the frame $(B_i B_{i+1}, \lambda_i, \lambda_{i+1}, l_i, l_{i+1})$ (see Step 2.3) and $I_{k'}$ be an instance of $(B_{k'} Y, \lambda_{k'}, \rho, l_{k'}, r)$. Since these frames are valid the instances exist. Assume that the vertex sets of the instances are disjoint. Let a and p be two new vertices s.t. $\omega(a) = A_k$ and $\omega(p) = A_1 \cdots A_k \rightarrow B_1 B_2 \cdots B_{k'}$. To build the instance for F we combine the instances by identifying the vertices on the rightmost path of I_i with the vertices on the leftmost path of I_{i+1}, for $j \leqslant i < k'$. Furthermore, we add the edges (a, p) and the edges (p, v_i), for $j \leqslant i \leqslant k'$, where v_i is the vertex corresponding to B_i. Since there is an instance for F this frame must be valid and we get a contradiction.

Similarly one can prove in a second induction that all valid frames are in the set

VAL created by the algorithm. Assume $F=(t, \lambda, \rho, l, v)$ is a valid frame of (D, Π) (see the above Definition) which is not in VAL. Let T be the vertices of D which correspond to t. We choose F so that the number of vertices which have a predecessor in T is minimum. In Step 0 all valid frames are added to VAL for which the length of both λ and ρ is 0. Thus in the frame F either λ or ρ is of positive length. We distinguish the following cases.

(1) $|T|=1$, $\lambda=\rho$ and λ has positive length;

(2.1) $|T|=2$, the vertices of T have a common successor;

(2.2) $|T|=2$, the vertices of T do not have a common successor, ρ has positive length;

(2.3) $|T|=2$, the vertices of T do not have a common successor, λ has positive length.

We still need to show that in each case F is added to VAL by the algorithm which is a contradiction. We only show this for Case 2.2. The remaining cases are similar.

Let $T=\{u, v\}$, $\omega(u)=X$, $\omega(v)=A_1$, let the successor of v be labeled with the production $A_1A_2\cdots A_k \rightarrow B_1B_2\cdots B_{k'}$, and let v_i be the vertex of D corresponding to B_i. Because of the minimality of F the set VAL contains the frame $(XB_1, \Omega(\pi_u), \Omega(\pi_{v_1}), l, r_1)$ and the frames $(B_iB_{i+1}, \Omega(\pi_{v_i}), \Omega(\pi_{v_{i+1}}), r_i, r_{i+1})$, for $1 \leqslant i < k'$, where w_l corresponds to the sink where π_u ends and w_{r_i} to the sink at which π_i ends, for $1 \leqslant i \leqslant k'$. We conclude that F would have been added to VAL in Step 2.2 which is a contradiction.

The polynomiality of the algorithm follows from the fact that the number of different frames is polynomial and from the fact that only a constant number of different frames need to be considered to create a new valid frame. ∎

Combining Theorems 2 and 3 gives us the main result of this paper.

THEOREM 4. *The membership problem for fixed growing* csgs *is polynomial.*

Proof. Given an input word w we compute the set VAL of all valid frames w.r.t. w using the above polynomial algorithm. We then check whether there are two frames $([S, [, \mu, 0, m)$ and $(S], \mu,], m, |w|+1)$ in VAL. Theorem 2 guarantees that the answer is yes iff the word w is in the language. Since |VAL| is polynomial the test requires only polynomial time. ∎

In the conclusion section we discuss parallel algorithms for this problem.

5. *NP*-COMPLETE GROWING SCATTERED LANGUAGES

We will exhibit a fixed growing scattered language which is *NP*-complete. A *scattered grammar* (scg) G is a quadruple (V, Σ, P, S) where the components have the same meaning as for a csg except that the productions of P are defined differently [GH68]. The productions have the form $(A_1, A_2, \ldots, A_k) \rightarrow (\alpha_1, \alpha_2, \ldots, \alpha_k)$, s.t.

$A_i \in V-\Sigma$, $\alpha_i \in V^*$ and $|\alpha_i| \geqslant 1$. In a *growing* scg the last condition is replaced by $|\alpha_i| > 1$.

As in a derivation step for csgs, the left-hand side of a production is replaced by the right-hand side, but in the case of scgs the symbols A_i need not be adjacent. For to words u and v of V^*, $u \Rightarrow v$ if $u = u_1 A_1 u_2 \cdots A_k u_{k+1}$, $v = u_1 \alpha_1 u_2 \cdots \alpha_k u_{k+1}$ and $(A_1,..., A_k) \rightarrow (\alpha_1,..., \alpha_k)$ is in P. The (*growing*) *scattered lnguage* defined by the (growing) scg G is the set $L(G) = \{w \mid S \overset{*}{\Rightarrow} w \text{ and } w \in \Sigma^*\}$.

The main open problem concerning scattered languages (scls) is the question whether every csl is also a scl. This is rather unlikely, but it holds if productions of the type $(A_1,..., A_k) \rightarrow (\alpha_1,..., \alpha_k)$, s.t. $|\alpha_1 \cdots \alpha_k| \geqslant k$ are allowed [GW86b].

It is easy to see that $L = \{ucu^R cu : u \in \{a, b\}^*\}$ (see introduction of Sect. 3) is a growing scl. Since the symbols to be rewritten are not required to be adjacent, the derivations in different parts of the word can be synchronized.

In a growing scg it takes at most $|w|$ steps to derive a word w. Thus the growing scls are a subclass of NP. We will present a polynomial reduction of 3-Partition to a growing scg.

3-*Partition.*

Instance. $3k$ numbers n_i and a bound B.

Question. Can the numbers be partitioned with k 3-element subsets each of which sums to B.

3-Partition was the first problem to be shown strongly *NP*-complete, i.e., it remains *NP*-complete even if the n_i are encoded in unary [GJ78]. The language C for which we will provide a growing scg has the property that $\langle n_1,..., n_{3k}, B\rangle$ is an instance of 3-Partition if and only if the word $xa^{n_1}xa^{n_2} \cdots xa^{n_{3k}}(yb^B)^k$ is in C.

Note that the word describes the instance of 3-Partition and that its length is polynomial in the length of the unary encoding of the instance. Thus the above equivalence implies that C is *NP*-complete.

THEOREM 5. *There are fixed, groing* scls[4] *which are NP-complete.*

Proof. We will construct a growing scl $C = L(G)$ which fulfills the above equivalence. To simplify the construction, we assume that the numbers n_i, $1 \leqslant i \leqslant 3k$, are all at least three and $n > 1$,

$$G = (\{a, b, x, y, X, \bar{X}, Y, \bar{Y}, \hat{Y}\}, \{a, b, x, y\}, P, S),$$

where

$$\begin{aligned} P = \{&(S) \rightarrow (XY\hat{Y}\hat{Y}), (Y) \rightarrow (Y\hat{Y}\hat{Y}Y) \\ &(X, Y) \rightarrow (xa\bar{X}, yb\bar{Y}), (X, \hat{Y}) \rightarrow (xa\bar{X}, b\bar{Y}), \\ &(\bar{X}, \bar{Y}) \rightarrow (a\bar{X}, b\bar{Y}) \mid (aaX, bb) \mid (aa, bb)\}. \end{aligned}$$

[4] The theorem also holds for the case were the language is *unordered* [Sa73] in addition to being growing and scattered. Note that the grammar used in the reduction is a growing unordered scattered grammar.

Outline of the proof: To show the above equivalence observe that the grammar produces a sequence of blocks of a's followed by a sequence of blocks of b's. The sizes of the blocks of a's correspond to the numbers n_i. While X is deriving $xa^{n_i}X$ either some Y derives yb^{n_i} or some $\hat{Y}$ derives b^{n_i}. There is a block of b's for each n_i but the blocks of b's are permuted and grouped in threes. Each group of three sums to B.

A more detailed proof: Suppose we are given a solution of the instance of 3-Partition, i.e., disjoint sets A_q, $1 \leqslant q \leqslant k$, each of which contains 3 n_i's that add to B. We will show that the word $w = xa^{n_1}xa^{n_2} \cdots xa^{n_{3k}}(yb^B)^k$ that describes the instance of 3-Partition is in $L(G)$. Clearly $S \overset{*}{\Rightarrow} X(Y\hat{Y}\hat{Y})^k$. Associate the set A_q with the qth group $Y\hat{Y}\hat{Y}$ and associate each of the 3 elements of the set with one of the 3 symbols Y, $\hat{Y}$, and $\hat{Y}$, respectively, in the group. The association within each group is arbitrary. The derivation $X(Y\hat{Y}\hat{Y})^k \overset{*}{\Rightarrow} w$ is organized in $3k$ phases. In the jth phase, for $1 \leqslant j < 3k$, X is rewritten to $xa^{n_j}X$ and in parallel the Y-symbol (resp. $\hat{Y}$-symbol) that is associated with n_j is rewritten to yb^{n_j} (resp. b^{n_j}). In the $3k$th phase X is rewritten to $xa^{n_{3k}}$ and in parallel the Y-symbol (resp. $\hat{Y}$-symbol) that is associated with n_{3k} is rewritten to $yb^{n_{3k}}$ (resp. $b^{n_{3k}}$). Since the numbers of A_q add to B each group $Y\hat{Y}\hat{Y}$ derives yb^B.

For the opposite directions assume that $S \overset{*}{\Rightarrow} w$, where $w = xa^{n_1}xa^{n_2}x \cdots xa^{n_{3k}}(yb^B)^k$. Normalize the derivation by applying the production steps $(Y) \rightarrow (Y\hat{Y}\hat{Y}Y)$ as early as possible within the derivation of w. The normalized derivation has the form:

$$S \overset{*}{\Rightarrow} X(Y\hat{Y}\hat{Y})^k \overset{*}{\Rightarrow} w$$

The symbol X derives $\bar{X}$ and after a number of steps X again. More exactly X produces $xa^{n_j}X$ at the jth phase, for $1 \leqslant j < 3k$, and $xa^{n_{3k}}$ in the last phase. Furthermore in the ith phase, for $1 \leqslant i \leqslant 3k$, a particular Y (resp. $\hat{Y}$) derives yb^{n_i} (resp. b^{n_i}). Observe that each non-terminal Y is responsible for a terminal y in w and the Y's are 3 apart. The terminal y-s function as separators and each group of $Y\hat{Y}\hat{Y}$ must produce exactly B b's. Each group thus corresponds to a different set of 3 numbers that adds to B and there are k such sets. ▌

6. Conclusions

We can express the membership problem for fixed growing csls as a membership problem for a variable context-free language. Given input word w and a fixed growing csg G, then we construct a context-free grammer G'_w from the frames (Sect. 4) of w and G. The frames form the nonterminals of G'_w. Note that the number of nonterminals is polynomial in w. The derivations of G'_w are defined using the recursions of Steps 1–2.3. of the Algorithm of Sect. 4. The basic frames of Step 0 all derive the empty word ε. We still need to add a special start symbol S' which derives all combinations of two frames $([S, [, \mu, 0, m)$ and $(S], \mu,], m, |w|+1)$ (see also Theorem 2).

It is easy to see that $w \in L(G)$ iff $\varepsilon \in L(G'_w)$. Also the derivation trees for ε in G'_w have ony $O(|w|)$ nodes since the original grammar G is growing. We now sketch that growing csls are in the LOG(CFL), the family of languages that are log-tape reducible to context-free languages [Su78]. LOG(CFL) is exactly the family of languages recognized by a nondeterministic $\log(n)$ tape bounded auxiliary pushdown automata within polynomial time [Su78]. n denotes the length of the input. To see that $L(G)$ is accepted by the latter type of automata, we simulate derivations of G'_w with a pushdown automata. The additional $\log(|w|)$ tape is needed to store the nonterminals (frames) involved in the current production. Note that a frame requires at most $\log(|w|)$ space.

It was further shown in [Ru80] that LOG(CFL) are those languages accepted by an Alternating Turing Machine in $\log(n)$ space and polynomial tree size. The space complexity and parallel time complexity of LOG(CFL) has been studied. Every language of LOG(CFL) can be recognized in $(\log(n))^2$ space by a deterministic Turing machine [Co70]. Thus growing csls can be recognized within the same space complexity as the lowest space complexity found for context-free languages [LSH65].

As for the parallel complexity [Ru80], LOG(CFL) is contained in NC^2, the class of problems solved by uniform circuits of depth $O((\log(n)^2)$ using a polynomial number of bounded fan-in gates. For example NC^2 circuits for the membership problem in a fixed context-free language are described in [UV85] and these circuits also solve the question whether $\varepsilon \in G'_w$. Actually [UV85] gives PRAM algorithms for the same problems and these algorithms translate into circuits. The PRAM algorithms run in $O((\log(|w|)^2$ parallel time and require a polynomial number of processors.

We showed that the membership problem for fixed growing csls can be solved in polynomial time and has reasonable space complexity and parallel time complexity. The main open problem is to determine the complexity of membership for "variable" growing csls, i.e., not only the word to be tested but also the growing csg is a variable of the input. The question is whether this problem can be solved in polynomial time or whether it is NP-complete.

ACKNOWLEDGMENTS

We would like to thank Allen Goldberg and Habib Krit for helping to simplify the presentation of the results. Furthermore we are thankful to an anonymous referee who pointed out that growig csls are accepted by Alternating Turing Machines in $\log(n)$ space with polynomial tree size and are thus contained in LOG(CFL) (see conclusions section.)

REFERENCES

[BPS61] Y. M. BAR-HILLEL, M. PERLES, AND E. SHAMIR, On formal properties of simple phase structure grammars, *Z. Phonetik Sprachwiss. Kommunikationsforschung*, **14** (1961), 143–172.

[Bo71] R. V. BOOK, Time bounded grammars and their Languages, *J. Comput. Systems Sci.* **5** (1971), 397–429.

[Bo73] R. V. BOOK, On the structure of context-sensitive Grammar, *Inernat. J. Comput. Inform. Sci.* **2** (2) (1973), 129–139.

[Bo78] R. V. BOOK, On the complexity of formal grammars, *Acta Inform.* **9** (1978), 171–182.

[Ch59] N. CHOMSKY, A note on phase-structure Grammars, *Inform. and Control* **2** (1959), 137–167.

[Co70] S. A. COOK, Path systems and language recognition, *in* "Proc. Second Annual ACM Symposium on Theory of Computing," pp. 70–72, 1970.

[DW86] E. DAHLHAUS AND M. K. WARMUTH, Membership for growing context-sensitive grammars is polynomial, *in* "Proceedings of the Eleventh Colloquium on Trees in Algebra and Programming," Lecture Notes in Computer Sience, Vol. 214, Springer-Verlag, Berlin, March 24–26, 1986.

[GJ78] M. R. GAREY AND D. S. JOHNSON, "Computers and intractability," Freeman, San Francisco 78.

[Gl64] A GLADKII, On the complexity of derivations in phase-structure grammars, *Algebri i Logika, Sem.* **3** (5–6), (1964), 26–44. [Russian]

[GS85] J. GONCZAROWSKI AND E. SHAMIR, Pattern selector grammars and several parsing algorithms in the context-free style, *J. Comput. System Sci.* **30** (1985), 249–273.

[GW85] J. GONCZAROWSKI AND M. K. WARMUTH, "Applications of Scheduling Theory to Formal Language Theory," Fundamental Studies Issue of Theoretical Computer Science, Vol. 37, No. 2, pp. 217–243, 1985.

[GW86a] J. GONCZAROWSKI AND M. K. WARMUTH, "Manipulating Derivation Forests by Scheduling Techniques," to appear in Fundamental Studies Issue of Theoretical Computer Science, 1986.

[GW86b] J. GONCZAROWSKI AND M. K. WARMUTH, "Scattered versus Context-Sensitive Rewriting," Technical Report CS-86-11, Department of Computer Science, Hebrew University, Jerusalem, July 1986.

[GH68] S. GREIBACH AND J. HOPCROFT, Scattered context grammars, *J. Comput. Systems Sci.* **3** (1969), 233–249.

[Ha78] M. A. HARRISON, "Introduction to Formal Language Theory," Addison–Wesley, Reading, Mass., 1978.

[Ka72] R. M. KARP, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations", (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–103, Plenum, New York, 1972.

[Ka65] T. KASAMI, "An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages," Science Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Mass., 1965.

[Ku64] S. Y. KURODA, Classes of languages and linear bounded automata, *Inform. and Control,* **7** (1964), 207–223.

[Lo70] J. LOECKS, The parsing of general phase-structure grammars, *Inform. and Control.* **16** (1970), 443–464.

[Ru80] W. L. RUZZO, Tree-size bounded alternation, *J. Comput. System Sci.* **21** (1980), 218–235.

[Sa73] A. SALOMAA, "Formal Languages," Academic Press, New York, 1973.

[LSH65] P. M. LEWIS, R. E. STEARNS, AND J. HARMANIS, Memory bounds for recognition of context-free and context-sensitive languages, *in* "Proc. Sixth Annual IEEE Symposium Switching Circuit Theory and Logical Design," pp. 191–212, 1965.

[Su78] H. SUDBOROUGH, On the tape complexity of deterministic context-free languages, *J. Assoc. Comput. Mach.* (1978), 405–414.

[UV85] J. D. ULLMAN AND A. VAN GELDER, "Parallel Complexity of Logical Query Programs," Technical Report STAN-CS-85-1089, Department of Computer Science, Stanford Univ., 1985, to appear in 27th Symposium on Foundations of Computer Science, Toronto, Canada, 1986.

[Yo67] D. H. YOUNGER, Recognition and parsing of context-free languages in time n^3, *Inform. and Control* **10** (1967), 189–208.

Printed by the St. Catherine Press Ltd., Tempelhof 41, Bruges, Belgium