Theoretical Computer Science 45 (1986) 87-119
North-Holland

[8]

MANIPULATING DERIVATION FORESTS BY SCHEDULING TECHNIQUES

Jakob GONCZAROWSKI
Department of Computer Science, The Hebrew University of Jerusalem, Jerusalem 91904, Israel

Manfred K. WARMUTH*
Computer Science Department, University of California, Santa Cruz, CA 95064, U.S.A.

Communicated by E. Shamir
Received November 1985

Abstract. This paper continues the investigation into k-parallel rewriting begun in (Gonczarowski/Shamir, 1985) and (Gonczarowski/Warmuth, 1985). This rewriting mechanism is a generalization of context-free rewriting; instead of applying a single production (or, alternatively, arbitrarily many productions) at each derivation step, exactly k productions are applied. In the works mentioned above, polynomial dynamic programming algorithms were presented which require constant k and propagating grammars. We solve the various membership problems left open in (Gonczarowski/Warmuth, 1985), for arbitrary grammars, using Scheduling Theory. We develop various kinds of pumping, obtaining bounds on the sizes of k-derivations and k-derivation forests. A polynomial dynamic programming membership algorithm is presented for arbitrary (i.e., possibly nonpropagating) grammars, for fixed k. If k is a variable of the problem, then membership is in NP and, hence, by (Gonczarowski/Warmuth, 1985), NP-complete. For unary alphabets, the latter problem is polynomial. Similarly, membership is polynomial in the size of k if only k is variable.

Contents

1. Introduction 87
2. Basic notions and ideas 90
3. Combinatorial bounds on k-derivations 94
4. Extended membership complexity 100
5. Membership for nonpropagating grammars 106
6. Summary 118
References 118

1. Introduction

The complexity of context-free languages has been the subject of extensive investigations in the literature (see, e.g., [13, 14]). In particular, the emptiness and

* This research has been done while the second author visited the Hebrew University of Jerusalem. This research was supported by the United States-Israel Binational Science Foundation, Grant No. 2439/82.

the membership problems are solvable in polynomial time for context-free languages [8, 25]. Similar results were shown for E0L languages [18, 23]. The membership problem of ET0L languages was shown to be NP-complete [4, 24]. On the other hand, context-sensitive rewriting is of a much higher degree of complexity. There exist fixed deterministic context-sensitive languages for which the membership problem is PSPACE-complete [2]. Moreover, the emptiness problem for context-sensitive languages is undecidable [14]. We attempt to shed light on some of those characteristics of rewriting which cause the complexity to grow beyond P.

To gain a deeper insight into the nature of rewriting, selective substitution grammars were introduced in [20]. A selective substitution grammar consists of an underlying context-free like rewriting system, and a language, called the *selector.* To allow a sentential form to be rewritten, one looks for a word in the selector that matches the sentential form. In addition, a selector word indicates which symbols in the sentential form are to be rewritten. In [10], it was shown how closure properties of the generated languages depend on closure properties of the selectors. In [16], a classification of the generated languages was given according to certain properties of the selectors. One of these properties is 'symbol-freeness', which means that the selector is closed under symbol substitution. Since the identity of individual symbols is not relevant in symbol-free selectors, the selector is represented by 'masks', which are words over $\{0, 1\}$. A symbol in a sentential form is rewritten if and only if it is matched by a 'one' in the selected mask (see, e.g., [16, 17]). Context-free grammars thus correspond to the selector 0^*10^*, and E0L systems correspond to 1^*.

Another property introduced in [16] is 'interspersion'; the selector is closed under the insertion of symbols which are not to be rewritten. For example, the selector $\{0, 1\}^*$ is the smallest language which contains the E0L selector, and which is interspersed. It is easy to see that also this selector corresponds to the context-free languages; rewriting any number of symbols in a sentential form is equivalent to rewriting exactly one symbol.

It follows from Penttonen Normal Form for context-sensitive grammars [19] that a language is context-sensitive if and only if it can be defined with a selector that is of the form

$$\bigcup_{i=1}^{k} \Sigma^* \bar{A}_i \bar{B}_i \Sigma^*,$$

where Σ is the total alphabet, k is a positive integer, and $A_1, B_1, \ldots, A_k, B_k$ are symbols to be rewritten (a bar over an occurrence indicates that it is to be rewritten). If we weaken such a selector by requiring symbol-freeness, we obtain the selector 0^*110^*. Similarly, scattered context languages [9] correspond to the selector

$$\bigcup_{i=1}^{k} \Sigma^* \bar{A}_i \Sigma^* \bar{B}_i \Sigma^*,$$

which is the interspersed version of the context-sensitive selector (see [16]). (In both these cases, the underlying context-free grammar must be propagating; otherwise, one gets recursively enumerable languages.) Weakening such a selector by requiring symbol-freeness, we obtain $0^*10^*10^*$.

The question that now rises concerns the complexity of the languages defined with these weakened selectors, and with their generalizations $0^*1^k0^*$ and $0^*(10^*)^k$. Intuitively, these selectors imply that each derivation step consists of rewriting k adjacent symbols (any k symbols, respectively). It was proven in [11] that, for propagating grammars and for $k=2$, the membership problem is polynomial in time, and that it is log-space reducible to context-free membership. (Actually, the polynomial time membership algorithms also solve the parsing problem, i.e., they can be used to construct parse trees.) In [12], polynomial parsing algorithms were presented for constant k in the case without the adjacency restriction. These algorithms again assume that the grammar is propagating. In the same paper, two closely related problems were shown to be NP-hard for both the adjacent case and the case without the adjacency restriction; the extended emptiness problem, where k and the grammar are variable, and the extended membership problem, where k and the word are variable.

In this paper we concentrate on the selector $0^*(10^*)^k$, i.e., any k nonterminals must be rewritten at each step. Main results of this paper are the generalizations of the polynomial parsing algorithms from [12] for arbitrary, i.e., possibly nonpropagating, context-free grammars. We obtain a polynomial membership algorithm if k is constant; if, in addition, the grammar is also constant, then the parsing problem can also be solved in polynomial time. For this purpose we develop polynomial bounds on the height of derivation forests and width of derivations. The main difficulty in obtaining these bounds lies in the possibility of erasing. Here it is not even possible to obtain polynomial bounds on the size of derivation trees, as there are derivation trees whose number of nodes is exponential in the size of the grammar.

Using somewhat different techniques, we show that the extended membership problem for unary words can be solved in polynomial time, even if k is part of the input.[1] This result is contrasted with the fact that the same problem is NP-hard if the alphabet consists of at least two symbols (cf. [12]). In fact, we prove in this paper that the extended membership problem for arbitrary size alphabets is in NP and, hence, NP-complete.

As a corollary we also obtain a polynomial bound for another variation of the membership problem, where only k is variable (denoted in unary notation) and both the grammar and the word are fixed (for arbitrary alphabets).

The techniques used for obtaining the above-mentioned bounds and algorithms are nonstandard for Formal Language Theory. We make use of the relationship established in [12] between k-derivations and the k-processor unit-length task scheduling problem of forests (see, e.g., [3, 6, 15]). In particular, the notion of a

[1] As explained in [12], we start out with S^k in this case, because otherwise the problem is trivial; if k were larger than the maximum right-hand side, the language would consist only of those words which can be directly derived from the start symbol.

'median' [6, 7], together with highest-level-first schedules, was applied in [12] to create polynomial dynamic programming algorithms for fixed k. In this paper we use the notion of median to develop nontrivial bounds on the length of k-derivations and of their sentential forms for arbitrary grammars; no propagating restrictions are assumed. In particular, to obtain one of these bounds, a new heuristic for obtaining optimal schedules is introduced; 'highest-lowest' scheduling.

The plan of this paper is as follows. In Section 2, we introduce the needed notions from Formal Language Theory and Scheduling Theory. In Section 3, we develop bounds on derivation lengths and widths. In Section 4, we apply these bounds to obtain the polynomiality of the extended membership problem (where the word and k are variable) for unary words, the NP-completeness of the extended membership problem in the general case, and the polynomiality of the membership problem where only k is variable. In Section 5, we first prove a bound on the height of k-derivation forests that is polynomial in the alphabet size and in the length of the derived word. This bound is used to develop a dynamic programming membership algorithm in the Earley style [8], the running time of which is polynomial if k is constant. In addition, if the grammar is fixed, a variant of this algorithm can be used to extract the corresponding parse tree in polynomial time. The paper is concluded by a table that summarizes the complexity results obtained for all the possible cases of k-parallel rewriting without adjacency restrictions. These results were obtained in this paper and in [12]. In the adjacent case, the complexity for $k \geqslant 3$, as well as for nonpropagating grammars and $k = 2$, is still open. Partial results are presented in [5, 11].

2. Basic notions and definitions

We assume the reader to be familiar with basic Formal Language Theory, as, e.g., in the scope of [13, 21, 22]. 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 L is denoted by the *Kleene Star* and is written as X^*. We shall identify a singleton set $\{a\}$ with its element a whenever this does not cause confusion. The cardinality of a set X is denoted by $\#X$.

A *context-free grammar* (CFG) G is a quadruple $\langle \Sigma, P, S, \Delta \rangle$, where Σ is the *alphabet* of G, Δ is the *terminal alphabet* of G; $\Sigma - \Delta$ is the *nonterminal alphabet* of G and is denoted by $\mathrm{Nt}(G)$, $P \subseteq (\Sigma - \Delta) \times \Sigma^*$ is the set of *productions* of G, $S \in \Sigma - \Delta$ is the *start symbol* of G.

If $w \in \Sigma^*$, then we denote by $\mathrm{Nt}(w)$ the word that is obtained from w by erasing all the terminal symbols.

Using standard Formal Language notation, we will write $A \to x$ if $(A, x) \in P$. The length of the longest right-hand side of a production in G is denoted by $\mathrm{Maxr}(G)$.

Let $u_0, \ldots, u_k \in \Sigma^*$, and let $(A_1, w_1), \ldots, (A_k, w_k) \in P$. We then say that $u_0A_1u_1A_2 \ldots A_ku_k$ *directly derives* $u_0w_1uw_2 \ldots w_ku_k$ *in* $\langle G, k\rangle$, and we write

$$u_0A_1u_1A_2 \ldots A_ku_k \underset{G,k}{\Longrightarrow} u_0w_1u_1w_2 \ldots w_ku_k.$$

(We shall omit k if $k = 1$, and we write then $\Rightarrow_G$; we shall omit G if it is obvious from the context.)

We denote by $\Rightarrow^+_{G,k}$ and $\Rightarrow^*_{G,k}$ the transitive closure and the reflexive and transitive closure of $\Rightarrow_{G,k}$, respectively. If $u \Rightarrow^*_{G,k} v$, then we say that u *derives* v *in* $\langle G, k\rangle$.

A *k-derivation* in G is a sequence of words $x_1, \ldots, x_{l+1}$ such that, for all $1 \leq i \leq l$,

$$x_i \underset{G,k}{\Longrightarrow} x_{i+1},$$

together with a mechanism that keeps track of the individual productions applied at each step. (Such a mechanism is necessary, as there might be more than one way to obtain x_{i+1} from x_i.) We shall not specify this mechanism explicitly, in order to keep the definitions concise. The *length* of a k-derivation $x_1, \ldots, x_{l+1}$ is l, its *width* is $\max\{|x_i| : 1 \leq i \leq l+1\}$, and its *i-th step* is the process of deriving x_{i+1} from x_i.

A *sentential form* (of G) is a word x, such that $S \Rightarrow^*_G x$.

The (*unrestricted*) *k-language of G* is the set

$$U_k(G) = \{w \in \Delta^* : \exists u \in \Sigma^* \text{ such that } S \underset{G}{\Rightarrow} u \text{ and } u \overset{*}{\underset{G,k}{\Rightarrow}} w\}.$$

2.1. Remark. (1) We observe that $U_1(G)$ is the context-free language defined by the grammar G. We thus assume in the sequel that $k \geq 2$. To avoid other trivial cases, we assume also that $\text{Maxr}(G) \geq 2$.

(2) Since rewriting begins with the start symbol, derivations of words in the k-language consist of one 1-derivation step (to rewrite S), followed by a number of k-derivation steps. In particular, all words directly derived from S are in $U_k(G)$, for all k (this implies that the family of k-languages contains all the finite languages).

We shall use trees and forests in the usual manner of Formal Language Theory. The reader is assumed to be familiar with the basic notions, such as root, parent, child, ancestor, descendant, internal node, and leaf.

The *height* of a node is the distance to its furthest descendant. Leaves thus have height zero. The *height* of a forest F is the height of its highest root and is denoted by height(F). The *size* of F, denoted by $|F|$, is the number of nodes in F. The *width* of F, denoted by $\#F$, is the number of trees in F.

2.2. Remark. One has to distinguish the height of a derivation forest from the length of a derivation. The latter is usually much larger.

If F is a forest, then the *bare forest* of F is the forest obtained by deleting all the leaves in F; it is denoted by $\text{Bare}(F)$. Note that, for all forests F, $\text{height}(\text{Bare}(F)) = \text{height}(F) - 1$. A *derivation forest* is a forest where each internal node is labelled with a nonterminal of the grammar. Furthermore, if a node is labelled with the nonterminal A, and its children's labellings (from left to right) form the word x, then $A \to x$ must be a production of the grammar. In particular, a k-derivation forest is a derivation forest that corresponds to a k-derivation. We represent an erasing production by a leaf labelled with λ below the node to which that production is applied (this is to guarantee that the bare forest of a derivation forest captures all those nonterminal symbols which are rewritten).

Some of these notions are illustrated in the following example.

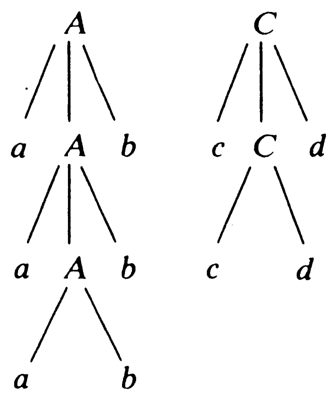
Fig. 1. A derivation forest that is not a 2-derivation forest.

2.3. Example. Let G be the CFG $\langle\{S, A, C, a, b, c, d\}, P, S, \{a, b, c, d\}\rangle$, where P consists of the following productions:

$$S \to AC, \qquad A \to aAb, \qquad A \to ab, \qquad C \to cCd, \qquad C \to cd.$$

Figs. 1 and 2 each show derivation forests. The forest in Fig. 1 is not a 2-derivation forest, whereas that in Fig. 2 is a 2-derivation forest.

Further examples are given in [12].

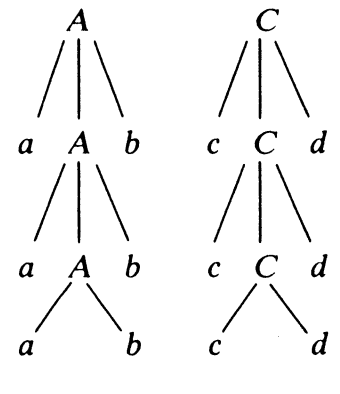
Fig. 2. A 2-derivation forest.

We now proceed to define schedules on forests. We assume that there are k processors (corresponding to rewriting k symbols at each derivation step). Every node in a forest is considered to be a unit-length task, where the parent–child relation in the forest specifies the precedence constraints. A k-schedule is then a sequence of slots, where each slot contains up to k tasks, indicating what tasks are to be scheduled in the corresponding unit of time. This is formalized in the following definition.

2.4. Definition. Let F be a forest and let $k \geqslant 2$. A *k-schedule* of F is a function σ mapping the nodes of F onto the set $\{1, \ldots, l\}$, for some $l \leqslant |F|$, such that

- $1 \leqslant \#\sigma^{-1}(i) \leqslant k$ for all $1 \leqslant i \leqslant l$;
- for each pair of nodes ν_1, ν_2 in F, if ν_2 is a successor of ν_1, then $\sigma(\nu_2) > \sigma(\nu_1)$.

The nodes of F are also called *tasks*. l is called the *length* of σ, and $\sigma^{-1}(i)$ is called the ith *slot* of σ.

The tasks of slot i are scheduled at *time* i (i.e., $\#\sigma^{-1}(i)$ out of the k processors are assigned a task at that time). There are $k - \#\sigma^{-1}(i)$ *idle periods* in slot i.

A schedule σ has $p(\sigma)$ idle periods, where

$$p(\sigma) = \sum_{i=1}^{l} (k - \#\sigma^{-1}(i)) = l \cdot k - |F|.$$

A schedule σ is *optimal* for F if there is no schedule σ' of F with $p(\sigma') \leqslant p(\sigma)$. (Note that optimal schedules have minimal length.) The number of *idle periods* of F, $p(F)$, is the number of idle periods in an optimal schedule for F.

A schedule σ is *perfect* if $p(\sigma) = 0$.

2.5. Example. The following is a perfect 2-schedule for the bare forest of the derivation forest of Fig. 2.

Time slot		
1	2	3
A	A	A
C	C	C

It is easy to see that there is a natural correspondence between k-derivation forests and perfect k-schedules of their bare forests; if $\nu_1, \ldots, \nu_k$ are the nodes labelled with the symbols that are rewritten in step i, then $\nu_1, \ldots, \nu_k$ appear in slot i of the corresponding schedule.

Using the above notions of Scheduling Theory, we can now give alternative definitions of k-derivation forests and k-languages (see [12]).

2.6. Lemma. (1) *A derivation forest is a k-derivation forest if and only if its bare forest has a perfect schedule.*

(2) *Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a CFG. A word w is in $U_k(G)$ if and only if there exists a derivation tree T of w from S in G such that* $\boldsymbol{p}(\text{Bare}(T)) = k - 1$.

A Highest-Level-First (HLF) k-schedule for a forest F is obtained as follows:

(1) if F consists of at least k trees, then $\sigma^{-1}(1)$ contains the roots of the k highest trees (for trees of equal height, the choice is arbitrary);

(2) otherwise, $\sigma^{-1}(1)$ is the set of all the roots;

(3) the tail of the schedule is constructed similarly, with the nodes in $\sigma^{-1}(1)$ deleted from F.

In one of the first papers of Scheduling Theory [15], it was shown that scheduling upside-down forests according to HLF produces optimal schedules. More recently, the same was shown for ordinary forests.

2.7. Theorem ([3, 6]). *Any HLF schedule for a forest is optimal.*

The k-median (see [6, 7]) is a certain height that allows us to distinguish between that part of a forest that is hard to schedule (the 'high forest') and that part which is easy to schedule (the 'low forest'). The *k-median* of a forest F is one plus the height of the kth highest tree of F; it is denoted by $\mu_k(F)$. If F contains less than k trees, then the median is zero. The *k-high forest* of F is the set of all those trees in F which are strictly higher than the median. The *k-low forest* is the set of the remaining trees. These forests are denoted by $\text{High}_k(F)$ and $\text{Low}_k(F)$, respectively. Whenever k is understood from the context, we shall drop k and write high forest, low forest, median, etc. The importance of the median is expressed by the following lemma (cf. [12, Lemma 3.3]).

2.8. Lemma. *Assume that the HLF schedules for* $\text{High}(F)$ *have q idle periods. Then the HLF schedules for F have*

(1) $q - |\text{Low}(F)|$ *idle periods if* $q \geq |\text{Low}(F)|$, *and*

(2) $-|F| \bmod k$ *idle periods otherwise.*

3. Combinatorial bounds on k-derivations

We shall first show that a wide forest of small height has a perfect schedule, as long as its size is a multiple of k, using Lemma 2.8. Using this fact, we shall show that every wide enough k-derivation forest can be reduced in height. As a next step, we shall bound the width of sentential forms in k-derivations by a function of the height of k-derivation forests. Combining the above results, we obtain a bound on the k-derivation width that is polynomial in k, in the length of the word, and in the size of the grammar.

3.1. Lemma. *Let F be a forest such that*

$$|F| = 0 \bmod k \quad \textit{and} \quad \#F \geq (k-1)(\text{height}(F)+2).$$

Then $p(F) = 0$.

Proof. Note that an HLF schedule has at most $k-1$ idle periods for each height, including height zero. It thus follows that

$$p(\text{High}(F)) \leq (k-1)(\text{height}(F)+1).$$

F consists of at least $(k-1)(\text{height}(F)+2)$ trees. Therefore, $\text{Low}(F)$ consists of at least $(k-1)(\text{height}(F)+1)$ trees, i.e., $|\text{Low}(F)| \geq p(\text{High}(F))$. Applying Lemma 2.8(2), we obtain that

$$p(F) = -|F| \bmod k = 0. \qquad \square$$

3.2. Remark. Note that, in Lemma 3.1, the whole forest is scheduled. However, as pointed out in Section 2, when we want to apply Scheduling Theory to derivations, we schedule the bare forest of the derivation forest. Hence, for a derivation forest F, the bound will be $\text{height}(F)+1$ rather than $\text{height}(F)+2$.

The following combinatorial lemma will be needed in the sequel.

3.3. Lemma. *Let k be a positive integer, and let* $l_1, \ldots, l_m$ *be a sequence of positive integers such that*

$$\sum_{j=1}^{m} l_j = i \bmod k$$

for some $i \geq 0$. *Then there are distinct indices* $t_1, \ldots, t_q \in \{1, \ldots, m\}$, *where* $0 \leq q \leq k-1$, *such that*

$$\sum_{j=1}^{q} l_{t_j} = i \bmod k.$$

Proof. If $m \leq k-1$, then the lemma follows. Let thus $m \geq k$. We look at the partial sums

$$s_r = \sum_{j=1}^{r} l_j \bmod k,$$

for all $0 \leq r \leq m$. There must now be indices r_1 and r_2, $0 \leq r_1 < r_2 \leq k$, such that $s_{r_1} = s_{r_2}$, i.e., $l_{r_1+1} + \cdots + l_{r_2} = 0 \bmod k$. We delete $l_{r_1+1}, \ldots, l_{r_2}$ from the given sequence and repeat the above process, until we obtain a sequence of length up to $k-1$, which will be $l_{t_1}, \ldots, l_{t_q}$. Since all the elements removed sum up to 0 mod k, it follows that

$$\sum_{j=1}^{q} l_{t_j} = i \bmod k,$$

and thus, the lemma holds. $\square$

Using the preceding lemmas, we can now show that, for every k-derivation forest of sufficiently large width, there is an equivalent k-derivation forest of bounded height.

3.4. Lemma. *Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a CFG, let $k \geq 2$, and let $x \in \Sigma^*$ and $w \in \Delta^*$ such that $x \Rightarrow^*_{G,k} w$. Let*

$$\bar{h} = (|w| + \#\Sigma + k)\#\Sigma, \quad \textit{and} \quad \bar{b} = (k-1)(\bar{h}+1).$$

If $\mathrm{Nt}(x) \geq \bar{b}$, *then there is a k-derivation forest F' of w from x such that* $\mathrm{height}(F') \leq \bar{h}$.

Proof. Let F be a k-derivation forest of w from x. If $\mathrm{height}(F) \leq \bar{h}$, the lemma trivially holds for F. Otherwise, we construct F' by iteratively 'unpumping'. We replace subtrees of F that correspond to a derivation $A \Rightarrow^+_G A$ by a single node labelled with A. We thus obtain a forest F' with the required height bound that has $0 \bmod k$ internal nodes. It then follows from Lemma 3.1 that $p(\mathrm{Bare}(F')) = 0$, i.e., F' is a k-derivation forest.

To construct F', we use the following concept. A *shrink* φ (of a nonterminal A) is a derivation tree of A from A such that no symbol occurs more than once on any path, except for A, which occurs exactly twice as indicated. Note that $\mathrm{height}(\varphi) \leq \#\Sigma$, and $|\varphi| \leq \mathrm{Maxr}(G)^{\#\Sigma+1}$.

At first, we iteratively replace shrinks by their roots (i.e., a subtree deriving a symbol A from itself is replaced by a single node labelled with A, until no more shrinks are left in the forest). The resulting forest F_1 will be of height $\leq (|w|+1)\#\Sigma$. We proceed to prove this bound.

Assume, on the contrary, that $\mathrm{height}(F_1) > (|w|+1)\#\Sigma$. We call those nodes that derive a nonempty subword of w *propagating*. A *growing* node has at least two propagating children. Obviously, there are at most $|w|-1$ growing nodes in F_1. If there is a path in F_1 that contains $\#\Sigma + 1$ successive propagating nodes, all of which derive the same subword u of w, then at least two of these nodes must be labelled with the same nonterminal, say A. Therefore, $A \Rightarrow^+_G A \Rightarrow^*_G v$. Hence, there exists a shrink in F_1, which is a contradiction to the definition of F_1.

Each growing node is thus preceded by at most $\#\Sigma - 1$ nongrowing (propagating) nodes. Similarly, each symbol in w is also preceded by at most $\#\Sigma - 1$ nongrowing (propagating) nodes. Therefore, each path from a root to a symbol in w contains up to $|w|\#\Sigma$ nodes.

Since $\mathrm{height}(F_1) > (|w|+1)\#\Sigma = |w|\#\Sigma + \#\Sigma$, it follows that there is a path that contains at least $\#\Sigma + 1$ (subsequent) nonpropagating nodes (followed by a λ-leaf). Two of these nodes must be labelled with the same nonterminal, say A. It follows that $A \Rightarrow^+_G A \Rightarrow^*_G \lambda$, i.e., there is a shrink in F_1, and we arrive at a contradiction. Hence, the height bound holds.

F_1 satisfies the required height bound; we must, however, guarantee that $|\mathrm{Bare}(F_1)| = 0 \bmod k$. To achieve this, we re-insert shrinks into F_1. Note that we are not free at this stage to freely choose among all the shrinks that we have eliminated;

possibly, a nonterminal A from the original forest F was deleted altogether in the shrinking process. A shrink of such an A cannot be re-inserted at this point. We must thus make sure that all nonterminals which appeared in the original forest F will also appear now. This is achieved by successively adding shrinks to F_1 such that each shrink introduces at least one new symbol. After adding at most $\#\Sigma - 1$ shrinks, the resulting forest F_2 contains all of the symbols that occurred in F. Moreover,

$$\text{height}(F_2) \leqslant \text{height}(F_1) + (\#\Sigma - 1)\#\Sigma.$$

Let $\varphi_1, \ldots, \varphi_n$ be the sequence of all those shrinks that were deleted from F and not added while constructing F_2 from F_1. Obviously,

$$|\text{Bare}(F_2)| + |\text{Bare}(\varphi_1)| + \cdots + |\text{Bare}(\varphi_m)| = 0 \bmod k$$

since $|\text{Bare}(F)| = 0 \bmod k$. By Lemma 3.3, there are distinct indices $t_1, \ldots, t_q$, $q \leqslant k-1$, such that

$$|\text{Bare}(F_2)| + |\text{Bare}(\varphi_{t_1})| + \cdots + |\text{Bare}(\varphi_{t_q})| = 0 \bmod k.$$

Inserting the shrinks $\varphi_{t_1}, \ldots, \varphi_{t_q}$ into F_2, we obtain a forest F'. F' is a derivation forest of w from x and $|\text{Bare}(F')| = 0 \bmod k$. Moreover,

$$\begin{aligned}\text{height}(F') &\leqslant \text{height}(F_2) + (k-1)\#\Sigma \\ &\leqslant (|w| + 1 + \#\Sigma - 1 + k - 1)\#\Sigma \leqslant (|w| + \#\Sigma + k)\#\Sigma = \bar{h}.\end{aligned}$$

Hence, F' satisfies the height bound, and the lemma follows from Lemma 3.1. □

3.5. Remark. $\bar{b}$ and $\bar{h}$ in Lemma 3.4 are actually (polynomial) functions in $|w|$, k, and $\#\Sigma$. Rather than writing them as such, we prefer, for the sake of conciseness, to omit their arguments. This should not cause confusion.

Lemma 3.4 permits us to decompose each k-derivation into two parts; a 'narrow' part (all sentential forms have at most $\bar{b}$ nonterminals), and a 'wide' part, which begins with the first sentential form that has more than $\bar{b}$ nonterminals. Lemma 3.4 guarantees that there is a k-derivation forest for the wide part of bounded height $\leqslant \bar{h}$. The narrow part of each k-derivation is of width polynomial in $\#\Sigma$, k, and $|w|$, whereas the width of the wide part is polynomial in k and $|w|$, but it might be exponential in $\#\Sigma$. The following lemma guarantees the existence of a k-derivation for the wide part that is of width polynomial in k and $|w|$, as well as in the size of the grammar.

3.6. Lemma. *Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a CFG, and let F be a k-derivation forest in G, for some $k \geqslant 2$. Then there is a k-derivation $x_1, \ldots, x_m$ of F such that*

$$\begin{aligned}Nt(x_i) \leqslant \hat{b} = \max(&\#\text{Bare}(F), (k-1)(\text{height}(F)+1) + k\,\text{Maxr}(G)) \\ &+ k(\text{height}(F) - 1)\,\text{Maxr}(G),\end{aligned}$$

for all $1 \leqslant i \leqslant m$.

Proof. We will construct a perfect schedule σ for Bare(F), that chooses lowest-height roots whenever the forest becomes too wide, according to Lemma 3.1. This prevents the number of roots from growing above $\hat{b}$, as will be shown later.

The formal construction follows. Let F_i denote the subforest of Bare(F) which is obtained by removing all those nodes that were scheduled before slot i. Note that $F_1 = \text{Bare}(F)$. We will prove that, for all i, $\# F_i \leq \hat{b}$. Let $x_1, x_2, \ldots, x_m = w$ be the k-derivation that corresponds to σ, i.e., the nodes rewritten in step i are exactly those nodes that are scheduled in slot i of σ. Then, obviously, $\# F_i = \text{Nt}(x_i)$ for all $1 \leq i \leq m$, and the lemma follows.

We present the schedule σ. Later on we shall prove that $\# F_i \leq \hat{b}$ and that σ is perfect. The schedule σ is constructed by the following algorithm.

```
begin
  for $i := 1$ to $|\text{Bare}(F)| \div k$ do
  begin
    if $\# F_i \leq (k-1)(\text{height}(F)+2)$
      then $\sigma^{-1}(i) := k$ highest roots of $F_i$
      else $\sigma^{-1}(i) := k$ lowest roots of $F_i$;
  end;
  delete the nodes in $\sigma^{-1}(i)$ from $F_i$ to obtain $F_{i+1}$;
end;
```

We show by induction on i that $p(F_i) = 0$. It then follows that σ is a perfect schedule.

Basis step: Since $F_1 = \text{Bare}(F)$ and since F is a k-derivation forest, it follows that $p(F_1) = 0$.

Induction step: Assume that $p(F_j) = 0$ for all $1 \leq j \leq i$. We want to show that $p(F_{i+1}) = 0$. If $\sigma^{-1}(i)$ is a set of highest roots, then we make use of Theorem 2.7. By the induction hypothesis, $p(F_i) = 0$. Hence, every HLF schedule of F_i must be perfect. Thus, in particular, after scheduling k highest nodes, the remaining number of idle periods is still zero, i.e., $p(F_{i+1}) = 0$.

Otherwise, if $\sigma^{-1}(i)$ is a set of lowest roots, then $\# F_i > (k-1)(\text{height}(F)+2)$ and thus, $\# F_{i+1} \geq (k-1)(\text{height}(F)+1)$. Clearly,

$$\text{height}(F_{i+1}) \leq \text{height}(\text{Bare}(F)) = \text{height}(F) - 1.$$

We conclude that

$$\# F_{i+1} \geq (k-1)(\text{height}(F_{i+1})+2).$$

Therefore, by Lemma 3.1, $p(F_i) = 0$. The induction hypothesis thus holds for $i+1$.

We have proven that the above algorithm produces a perfect schedule σ. It remains to show that $\# F_i \leq \hat{b}$ for all $1 \leq i \leq m$. Obviously, this is true for $i = 1$. For those slots i, $2 \leq i \leq m$, for which highest roots were chosen,

$$\# F_i \leq (k-1)(\text{height}(F)+2) \leq \hat{b}.$$

We therefore assume that lowest roots were chosen for slot i. Let j be the minimum slot, such that lowest roots were chosen for the slots $j, j+1, \ldots, i$. We shall see that a consecutive sequence of lowest schedules adds at most $k\mathrm{Maxr}(G)$ roots to F_i for each height between 0 and $\mathrm{height}(F_j)-1$. Note that either $j=1$, or else highest roots were chosen for slot $j-1$.

In the former case, $\# F_j = \mathrm{Bare}(F)$. In the latter case, $\# F_{j-1} \leqslant (k-1)(\mathrm{height}(F)+2)$, and k roots are rewritten to be replaced by at most $k\mathrm{Maxr}(G)$ new ones. Hence,

$$\# F_j \leqslant \max(\mathrm{Bare}(F), (k-1)(\mathrm{height}(F)+1)+k\mathrm{Maxr}(G)).$$

The lemma thus follows for F_i if we show that at most $k\mathrm{Maxr}(G)$ roots are added for each height between 0 and $\mathrm{height}(F_j)-1$.

Let r_q be the number of those roots of F_j which are of height q, for $0 \leqslant q \leqslant \mathrm{height}(F_j)$. We shall see that the number of roots of height q in each of the forests $F_j, \ldots, F_i$ is at most $r_q + k\mathrm{Maxr}(G)$, for $0 \leqslant q \leqslant \mathrm{height}(F_j)-1$. Obviously, this holds for F_j. Assume, on the contrary, that there is an l, $j+1 \leqslant l \leqslant i$, such that F_l has more than $r_q + k\mathrm{Maxr}(G)$ roots of height q for some $0 \leqslant q \leqslant \mathrm{height}(F_j)-1$, whereas F_{l-1} has at most $r_q + k\mathrm{Maxr}(G)$ such roots. F_l has more roots of height q than F_{l-1}. Therefore, some nodes of height at least $q+1$ must have been scheduled in slot $l-1$. Since lowest roots were chosen for that slot, this implies that all roots of F_{l-1} with height $\leqslant q$ must have been scheduled in slot $l-1$. Hence, F_l can have at most $k\mathrm{Maxr}(G)$ roots of height q, which is a contradiction.

We have shown that the number of roots of height q in each of the forests $F_j, \ldots, F_i$ is at most $r_q + k\mathrm{Maxr}(G)$ for $0 \leqslant q \leqslant \mathrm{height}(F_j)-1$. Clearly, $\mathrm{height}(F_j) \leqslant \mathrm{height}(F)-1$.

We now get

$$\begin{aligned}\# F_i &\leqslant \sum_{q=0}^{\mathrm{height}(F_j)-1} (r_q + k\mathrm{Maxr}(G)) + r_{\mathrm{height}(F_j)} = \# F_j + \mathrm{height}(F_j)\cdot k\mathrm{Maxr}(G)\\ &\leqslant \max(\#\mathrm{Bare}(F), (k-1)(\mathrm{height}(F)+1)+k\mathrm{Maxr}(G))\\ &\quad + k(\mathrm{height}(F)-1)\,\mathrm{Maxr}(G) = \hat{b}.\end{aligned}$$

Hence, the lemma holds. $\square$

We can combine the two preceding lemmas to guarantee that, for each k-derivation forest of a word w, there exists a k-derivation of w whose width is polynomial in k, $|w|$, and the size of G.

3.7. Theorem. *Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a CFG, and let $k \geqslant 2$. Let $x \in \Sigma^*$ and $w \in \Delta^*$, such that $x \Rightarrow^*_{G,k} w$, and* $\mathrm{Nt}(x) \leqslant \bar{b}$. *Let $\bar{d} = k\bar{h}(\mathrm{Maxr}(G)+1)$ ($\bar{b}$ and $\bar{h}$ were defined in Lemma* 3.4*). Then there is a k-derivation $x = x_1, x_2, \ldots, x_m = w$ such that $|x_i| \leqslant \bar{d}$, for $1 \leqslant i \leqslant m$.*

Proof. Let $x = y_1, y_2, \ldots, y_{l-1}, y_l = w$ be a k-derivation. If, for all $1 \leqslant j \leqslant l$, $\mathrm{Nt}(y_j) \leqslant \bar{b}$, then the lemma holds for the k-derivation $y_1, \ldots, y_l$, because $\bar{b} + |w| \leqslant \bar{d}$.

Let thus i be the first index such that $\mathrm{Nt}(y_i) > \bar{b}$. By Lemma 3.4, there is a k-derivation forest of F of w from y_i such that $\mathrm{height}(F) \leqslant \bar{h}$. We let $x_j = y_j$ for all $1 \leqslant j \leqslant i$. Applying Lemma 3.6 to F we obtain a k-derivation $y_i = x_i, x_{i+1}, \ldots, x_{m-1}, x_m = w$, such that

$$\begin{aligned}\mathrm{Nt}(x_j) &\leqslant \max(\bar{b} + k\mathrm{Maxr}(G), (k-1)(\bar{h}+1) + k\mathrm{Maxr}(G)) + k(\bar{h}-1)\mathrm{Maxr}(G)\\ &= \bar{b} + k\mathrm{Maxr}(G) + k(\bar{h}-1)\mathrm{Maxr}(G) = \bar{b} + k\bar{h}\mathrm{Maxr}(G),\end{aligned}$$

for all $i \leqslant j \leqslant m$. Hence,

$$\begin{aligned}|x_j| &\leqslant \bar{b} + k\bar{h}\mathrm{Maxr}(G) + |w| = k\bar{h} + (k - \bar{h} - 1 + |w|) + k\bar{h}\mathrm{Maxr}(G)\\ &\leqslant k\bar{h}(\mathrm{Maxr}(G)+1) = \bar{d},\end{aligned}$$

and the theorem holds. □

4. Extended membership complexity

In [12], it was shown that the UXM-problem is NP-hard: Let G be a fixed grammar. Given k and w as input, one wants to determine if $S^k \Rightarrow^*_{G,k} w$. Theorem 3.7 gives us the capability to solve this problem in polynomial time for *unary* words w, i.e., words over a one-letter alphabet. Another consequence of Theorem 3.7 is that one can solve the problem in nondeterministic polynomial time for words w over unrestricted alphabets; thus, the problem is NP-complete.

In the unary case, w is commutative. Rather than operating on words, we shall therefore operate on commutative images of words, in the form of Parikh vectors (see, e.g., [13]). These are defined now.

4.1. Definition. Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a CFG such that $\Sigma = \{A_1, \ldots, A_{\#\Sigma}\}$ is ordered under an arbitrary but fixed order. Let $x \in \Sigma^*$. The *Parikh vector* of x, $\psi(x)$, is the vector $(i_1, \ldots, i_{\#\Sigma}) \in \mathbb{N}_0^{\#\Sigma}$, where i_j is the number of occurrences of A_j in x for $1 \leqslant j \leqslant \#\Sigma$.

Since Parikh vectors represent commutative images of words, we can extend notions that were defined for words to Parikh vectors in a natural way. In particular, the *length* of a Parikh vector φ is the sum of its components; it is denoted by $|\varphi|$. Let φ and χ be Parikh vectors. We say that φ *directly derives* χ if and only if there exists words $x, y \in \Sigma^*$ such that $\psi(x) = \varphi$, $\psi(y) = \chi$, and $x \Rightarrow_{G,k} y$; we write $\varphi \Rightarrow_{G,k} \chi$.

A *Parikh k-derivation* is a sequence of Parikh vectors $\chi_1, \ldots, \chi_m$, $m \geqslant 1$, such that $\chi_i \Rightarrow_{G,k} \chi_{i+1}$ for $1 \leqslant i \leqslant m-1$. We then say that χ_1 *k-derives* χ_m and we write $\chi_1 \Rightarrow^*_{G,k} \chi_m$.

The following lemma establishes the relation between ordinary k-derivations and Parikh k-derivations.

4.2. Lemma. *Let $G = \langle \Sigma, P, S, \Delta\rangle$ be a CFG, and let $k \geq 2$. Then there is a k-derivation $x_1, \ldots, x_m$ if and only if there exists a Parikh k-derivation $\chi_1, \ldots, \chi_m$ such that $\psi(x_i) = \chi_i$ for $1 \leq i \leq m$.*

Proof. The 'only if'-direction trivially follows from the definition. We shall show the 'if'-direction by induction on m.

Basis step: Let $m = 1$. Then, obviously, x_1 is a k-derivation for all $x_1 \in \psi^{-1}(\chi_1)$.

Induction step: We assume that the induction hypothesis holds for m. Let $x_1, \ldots, x_m$ be the k-derivation that corresponds to $\chi_1, \ldots, \chi_m$. By the definition of k-derivations, there are y and z, $y \in \psi^{-1}(\chi_m)$ and $z \in \psi^{-1}(\chi_{m+1})$, such that y directly derives z in $\langle G, k\rangle$. We apply the same k productions used in that derivation step (but maybe in a different order) to derive a word x_{m+1} from x_m. Hence, $x_1, \ldots, x_m, x_{m+1}$ is a k-derivation that satisfies the lemma. □

We observe that, for unary words w, ψ is one-to-one. Combining Lemma 4.2 with Theorem 3.7, we obtain the following characterization result.

4.3. Corollary. *Let $G = \langle \Sigma, P, S, \Delta\rangle$ be a CFG and let $k \geq 2$. Let $w \in \Delta^*$ be a unary word. Then $S^k \Rightarrow^*_{G,k} w$ if and only if there is a Parikh k-derivation*

$$\psi(S^k) = \chi_1, \chi_2, \ldots, \chi_{m-1}, \chi_m = \psi(w),$$

such that $|\chi_i| \leq \bar{d}$, for all $1 \leq i \leq m$.

Note that there are at most $(\bar{d}+1)^{\#\Sigma}$ Parikh vectors of length $\leq \bar{d}$ (each component in a vector can take a value between 0 and $\bar{d}$). This quantity is polynomial in k and $|w|$. This is the reason why we get a polynomial algorithm if G is constant.

4.4. Theorem (Unary UXM-Problem). *Let $G = \langle \Sigma, P, S, \Delta\rangle$ be a fixed CFG. It can then be decided in polynomial time, for inputs $k \geq 2$ and unary words $w \in \Delta^*$, whether $S^k \Rightarrow^*_{G,k} w$.*

Proof. We prove that the unary UXM-problem requires time $O(\bar{d}^{\#\Sigma} k^{\#P})$. We construct a directed graph, where the nodes are all the Parikh vectors of length up to $\bar{d}$. Given two vertices φ and χ, we establish an edge between φ and χ if and only if $\varphi \Rightarrow_{G,k} \chi$. Then, by Corollary 4.3, we know that there is a path from $\psi(S^k)$ to $\psi(w)$ if and only if $S^k \Rightarrow^*_{G,k} w$. To determine the existence of such a path, we use a Depth-First Search traversal of the graph, starting at $\psi(S^k)$. This is linear in the size of the graph (see, e.g., [1]). It remains thus to be shown that the graph can be constructed in time $O(\bar{d}^{\#\Sigma} k^{\#P})$.

A *transition* is a pair of Parikh vectors (ζ, η), such that $|\zeta| = k$ and $\zeta \Rightarrow_{G,k} \eta$. It is now easy to see that, for all Parikh vectors φ and χ, $\varphi \Rightarrow_{G,k} \chi$ if and only if there is a transition (ζ, η) such that $\varphi - \zeta$ is a valid Parikh vector (i.e., it has no negative components), and $\varphi - \zeta + \eta = \chi$. The Parikh vector ζ corresponds to the symbols being rewritten, and η corresponds to the right-hand sides of the productions being used. The total number of transitions does not exceed $(k+1)^{\#P}$, because each production can occur between zero and k times. Obviously, it takes no more than $O((k+1)^{\#P})$ time to construct the set of transitions (we recall that the grammar is constant).

Having computed all the transitions we proceed to construct the edges. For each node φ, we try to apply all the transitions (ζ, η). If $\varphi-\zeta$ is a Parikh vector and $|\varphi-\zeta+\eta| \leq \bar{d}$, then we put an edge from φ to $\varphi-\zeta+\eta$. Each such test takes a constant number of steps since $\#\Sigma$ is constant. There are up to $(\bar{d}+1)^{\#\Sigma}$ nodes. Since the number of transitions is at most $(k+1)^{\#P}$, it takes at most $O(\bar{d}^{\#\Sigma}k^{\#P})$ time to construct the desired graph. □

4.5. Remark. Theorem 4.4 also holds if rewriting starts with an arbitrary axiom of length $\leq \bar{b}$; it is obvious that we may permute the axiom in any way, without affecting the outcome of a k-derivation from the axiom. Hence, the Parikh vector technique is also applicable for this case.

We have thus proven that the UXM-problem is in P for unary words w. To show that the same problem is in NP if the alphabet is not restricted, we first prove a bound on the length of k-derivations.

4.6. Lemma. *Let $G=\langle\Sigma, P, S, \Delta\rangle$ be a CFG, and let $k \geq 2$. Let $x \in \Sigma^*$ and $w \in \Delta^*$, such that $|x| \leq \bar{b}$ and $x \Rightarrow^*_{G,k} w$. Let $\bar{l}=(\bar{d}+1)^{\#\Sigma}\cdot(|w|+1)^{\#\Sigma^2}$. Then there is a k-derivation of w from x of length $\leq |w|\,\bar{l}$.*

Proof. Theorem 3.7 allows us to restrict ourselves to k-derivations of width up to $\bar{d}$. It could, however, still happen that the length of a k-derivation is exponential in $\bar{d}$ (and, therefore, in k and $|w|$). To prove the existence of a shorter k-derivation, we partition the original k-derivation into 'blocks' such that in each block, only nongrowing occurrences of symbols are rewritten. (We recall that propagating and growing nodes were defined in Lemma 3.4.)

It will be shown that for each block there is an 'equivalent' block whose length is bounded by $\bar{l}$. Since the number of growing nodes is less than the length of the derived word, the lemma then follows.

To find blocks of bounded length, we shall use Parikh vectors, similarly as in the proof of Theorem 4.4. We observe first that nonpropagating occurrences of symbols can be repositioned within a sentential form without effecting the terminal word derived from it. This permits us to represent the nonpropagating occurrences by Parikh vectors. A similar Parikh-vector representation will be established for propagating symbols. We represent a propagating symbol A by the 'twin' $[A, B]$, where B is the (unique) propagating symbol to be derived from A at the end of the block. As we shall prove later on, these twins are permutable and can thus be kept track of by using Parikh vectors (over the twin alphabet). Therefore, the block length can be bounded by the product of the number of possible Parikh vectors over the ordinary alphabet ($(\bar{d}+1)^{\#\Sigma}$ for the nonpropagating symbols) times the number of possible Parikh vectors over the twin alphabet ($(|w|+1)^{\#\Sigma^2}$ for the propagating symbols). The formal construction and proofs follow.

Let $x = x_1, x_2, \ldots, x_{m-1}, x_m = w$ be a k-derivation. A *block* $\mathcal{B}$ is a maximal consecutive subsequence of $x_1, \ldots, x_m$ (and, therefore, itself a k-derivation) $x_t, \ldots, x_{t+l-1}$ such that only nongrowing symbols are rewritten in steps $t, \ldots, t+l-2$. Note that if $x_{t+l-1} \neq w$, then growing symbols are rewritten in step $t+l-1$.

The k-derivation contains at most $|w|-1$ growing nodes. Hence, the total number of blocks does not exceed $|w|$. It remains, therefore, to be shown that for each block there is an 'equivalent' k-derivation of length less than

$$(\bar{d}+1)^{\#\Sigma}(|w|+1)^{\#\Sigma^2} = \bar{l}.$$

Since each block is followed by at most one 'growing' step, the resulting k-derivation length does not exceed $|w|\,\bar{l}$, and the lemma follows.

We proceed to prove that, for each block $\mathcal{B}$, there exists such a k-derivation $\mathcal{D}$ of length $\leq \bar{l}-1$. As pointed out above, if $y \Rightarrow^*_{G,k} w$, then it follows that $z \Rightarrow^*_{G,k} w$ for all words z which are obtained from y by changing positions of nonpropagating symbols. We thus arrive at the following definition:

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be k-derivations, and let $B_1, \ldots, B_q \in \Sigma$. $\mathcal{D}_1$ and $\mathcal{D}_2$ are $(B_1, \ldots, B_q)$-*equipropagating* if they agree on their first words and on the Parikh vectors of their respective last words (i.e., one last word is a permutation of the other), and if $B_1, \ldots, B_q$ occur in both last words, in this order.

Let $\mathcal{B}$ be a block, let $\mathcal{D}$ be a k-derivation, and let $B_1, \ldots, B_q$ be the sequence of propagating occurrences in the last word of $\mathcal{B}$. Then $\mathcal{B}$ and $\mathcal{D}$ are *equipropagating* if they are $(B_1, \ldots, B_q)$-equipropagating.

We may replace a block $\mathcal{B}$, $x_t, \ldots, x_{t+l-1}$, by an equipropagating k-derivation $\mathcal{D}$, maybe having to change positions of nonpropagating symbols in $x_{t+l}, \ldots, x_{m-1}$. The resulting derivation will still be a k-derivation of w from x, because changing positions of symbols that ultimately produce the empty word do not affect the outcome of the derivation. It thus remains to be shown that for each block $\mathcal{B}$ there is an equipropagating k-derivation $\mathcal{D}$ of length $< \bar{l}$.

Let $\mathcal{B}$ be the block $x_t, \ldots, x_{t+l-1}$. Each x_j, $t \leq j \leq t+l-1$, is represented by two Parikh vectors; an 'ordinary' Parikh vector χ for the nonpropagating symbols and a Parikh vector φ, over an extended alphabet, for the propagating symbols. We use ordinary Parikh vectors for nonpropagating occurrences of symbols in x_j, because, as said before, their positions are irrelevant for the final outcome of the k-derivation since they ultimately yield the empty word. Note that there are at most $(\bar{d}+1)^{\#\Sigma}$ distinct such Parikh vectors, because each symbol can occur in x_j between 0 and $\bar{d}$ times.

A propagating node ν that contributes to x_j (i.e., whose label occurs in x_j) either has exactly one propagating descendent ν' that contributes to x_{t+l-1}, or else ν itself contributes to x_{t+l-1}. In the latter case we let $\nu' = \nu$. The occurrence of the label of

ν in x_j is then represented by a *twin* $[A, B] \in \Sigma \times \Sigma$, where A is the label of ν and B is the label of ν'. Intuitively, B thus specifies the symbol that A needs to derive at the end of the block. Keeping track of this information allows us to permute the symbol pairs. Hence, the propagating nodes of x_j are represented by a Parikh vector φ over $\Sigma \times \Sigma$, which counts the occurrences of twins.

Note that the total number of these Parikh vectors does not exceed $(|w|+1)^{\#\Sigma^2}$, because each twin can occur at most $|w|$ times in a given x_j. The pair (φ, χ) is called the *Parikh vector pair* (PVP) of x_j, and is denoted by $\pi(x_j)$.

The total number of distinct PVPs is thus bounded by

$$\bar{l} = (\bar{d}+1)^{\#\Sigma} \cdot (|w|+1)^{\#\Sigma^2}.$$

We can now define k-derivations between PVPs. Then we shall show that one can go from the block $\mathcal{B}$ to a PVP k-derivation $\mathcal{P}$. Eliminating repetitions from $\mathcal{P}$ one obtains a PVP derivation $\mathcal{P}'$ of length $\leq \bar{l}-1$. From there, one can return to a k-derivation $\mathcal{D}$ that is of the same length as $\mathcal{P}'$ and that is equipropagating to the original block $\mathcal{B}$. A PVP k-derivation step will consist of rewriting k 'symbols'; j propagating ones (i.e., twins) and $k-j$ nonpropagating ones, for some $0 \leq j \leq k$.

Let (φ, χ) be a PVP. The application of a production $C \to u$ to a nonpropagating occurrence of C consists of replacing χ by $\chi - \psi(C) + \psi(u)$. A production $A \to yDz$ is applied to a propagating occurrence of A (represented by a twin $[A, B]$) by replacing φ with $\varphi - \psi([A, B]) + \psi([D, B])$ and by replacing χ with $\chi + \psi(yz)$. The vector φ thus keeps track of all the propagating occurrences, whereas χ keeps track of all nonpropagating occurrences.

Formally, let (φ, χ) be a PVP. Let $[A_1, B_1], \ldots, [A_j, B_j] \in \Sigma \times \Sigma$ for some $0 \leq j \leq k$ such that $\psi([A_1, B_1] \ldots [A_j, B_j]) \leq \varphi$, and let $C_{j+1}, \ldots, C_k \in \Sigma$ such that $\psi(C_{j+1} \ldots C_k) \leq \chi$. Let $A_i \to y_i D_i z_i \in P$, where $D_i \in \Sigma$ and $y_i, z_i \in \Sigma^*$, for all $1 \leq i \leq j$, and let $C_i \to u_i \in P$ for $j+1 \leq i \leq k$. Then (φ, χ) *directly derives* (ξ, η), where

$$(\xi, \eta) = (\varphi - \psi([A_1, B_1] \ldots [A_j, B_j]) + \psi([D_1, B_1] \ldots [D_j, B_j]),$$
$$\chi - \psi(C_{j+1} \ldots C_k) + \psi(y_1 \ldots y_j z_1 \ldots z_j u_{j+1} \ldots u_k));$$

this is denoted by $(\varphi, \chi) \Rightarrow_{G,k} (\xi, \eta)$. The reflexive and transitive closure $\Rightarrow^*_{G,k}$ and PVP k-derivations are defined accordingly.

Given the k-derivation $x_t, \ldots, x_{t+l-1}$, it is easy to see that $\pi(x_t), \ldots, \pi(x_{t+l-1})$ is a PVP k-derivation. We then eliminate duplicates; if $\pi(x_{l_1}) = \pi(x_{l_2})$ for some $t \leq l_1 < l_2 \leq t+l-1$, then we delete $\pi(x_{l_1+1}), \ldots, \pi(x_{l_2})$. This is done repeatedly until there are no more duplicate PVPs. We obtain the PVP k-derivation $\mathcal{P}'$

$$\pi(x_t) = (\xi_1, \eta_1), (\xi_2, \eta_2), \ldots, (\xi_{s-1}, \eta_{s-1}), (\xi_s, \eta_s) = \pi(x_{t+l-1})$$

of length $\leq \bar{l}-1$.

We now proceed to return from PVP k-derivations to k-derivations (of words). This is done by constructing the k-derivation such that (i) the order of the derived propagating symbols in the last word of the new k-derivation is the same as that in the original k-derivation, and (ii) the root words of the original and the new k-derivation are identical.

Let $(\xi_1, \eta_1), \ldots, (\xi_s, \eta_s)$ be a PVP k-derivation as above. We inductively construct a k-derivation $x_t = y_1, \ldots, y_s$ such that

(i) if $B_1, \ldots, B_j$ is the sequence of the propagating symbols in x_{t+l-1}, then there are exactly j marked symbols in y_i such that the mth marked symbol A_m is marked with B_m (from left to right) for $1 \leq m \leq j$;

(ii) $\psi([A_1, B_1] \ldots [A_m, B_m]) = \xi_i$;

(iii) if $E_1, \ldots, E_r$ are all the unmarked occurrences in y_i, then $\psi(E_1 \ldots E_r) = \eta_i$.

Basis step: Let $y_1 = x_t$ and mark the mth propagating symbol with B_m, the mth propagating symbol from x_{t+l-1}. It is easy to see that conditions (i)–(iii) hold for $i = 1$.

Induction step: We construct y_{i+1} from y_i and from the productions applied while deriving (ξ_{i+1}, η_{i+1}) from (ξ_i, η_i). If a production was applied to a symbol E in η_i (i.e., to an unmarked symbol), then we apply that production to some unmarked occurrence of E in y_i.

Each occurrence of a twin $[A, B]$ in ξ_i corresponds to an occurrence of a symbol A in y_i that is marked with B, by (ii). Let $A \to uCv$ be the production that is applied to $[A, B]$ such that $[C, B]$ is the twin that is contributed to ξ_{i+1}. Then we apply the production $A \to uCv$ to the above occurrence of A in y_i and we mark the derived occurrence of C with B. Again it is easy to see that conditions (i)–(iii) hold.

Observe that all twins in ξ_s are of the form $[B, B]$. It thus follows from (i) that the sequence of marked symbols in y_s is exactly $B_1 \ldots B_j$ (which is the sequence of propagating symbols in x_{t+l-1}). We obtain from (iii) that the Parikh vector of the unmarked symbols in y_s is η_{s+1}. Since $y_1 = x_t$, the k-derivation $y_1, \ldots, y_s$ is equipropagating to $\mathscr{B}$, and it is thus the required k-derivation $\mathscr{D}$. Since $\mathscr{D}$ is of length $\leq \bar{l} - 1$, the lemma follows. $\square$

We obtain from Lemma 4.6 the NP-completeness of UXM.

4.7. Theorem (UXM-problem). *Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a fixed CFG. It is then* NP-*complete for inputs $k \geq 2$ and $w \in \Delta^*$ to decide whether $S^k \Rightarrow^*_{G,k} w$.*

Proof. It was shown in [12] that the UXM-problem is NP-hard. On the other hand, by Lemma 4.6 and Theorem 3.7, there is a k-derivation of length $|w|\bar{l}$ of w from S^k such that each sentential form in that k-derivation is of length $\leq \bar{d}$. Since $\bar{l}$ and $\bar{d}$ are both polynomial in k and $|w|$, it follows that the UXM-problem is in NP. $\square$

We conclude this section by proving that the membership problem is polynomial in the value of k if G and w are fixed and only k is variable. Observe that now w may be arbitrary but fixed. We do not know whether this problem is polynomial in the size of k since k can be encoded in binary notation. This result contrasts Theorem 4.7, where it was proven that membership is NP-complete for arbitrary words w if both k and w are variables of the problem.

4.8. Theorem. *Let G be a fixed CFG, and let w be a fixed word. Let $k \geq 2$. Then it is decidable in time polynomial in k whether $S \Rightarrow_{G,k} w$.*

Proof. Let $G=\langle\Sigma, P, S, \Delta\rangle$, and let w in Δ^*, $w=a_1\dots a_{|w|}$. We shall construct from G and w a constant grammar $\hat{G}=\langle\hat{\Sigma}, \hat{P}, \hat{S}, \phi\rangle$. $\hat{G}$ is obtained from G as follows. We perform the standard regular intersection construction (see, e.g., [13]) with the set of states $\{0,\dots,|w|\}$, where 0 is the initial state and $|w|$ is the final state; a transition is made from $i-1$ to i if the ith symbol of w was encountered. We encode the states in the nonterminal symbols of the grammar; an occurrence of a symbol $[i, A, j]$ indicates that it intends to derive the $(i+1)$st through the jth symbols of w.

It thus follows that $S^k \Rightarrow_{G,k} w$ if and only if there are indices $i_0, i_1, \dots, i_k$, $0=i_0\leqslant i_1\leqslant\cdots\leqslant i_{k-1}\leqslant i_k=|w|$, such that

$$[i_0, S, i_1]\dots[i_{k-1}, S, i_k]\underset{\hat{G},k}{\Longrightarrow}[0, a_1, 1]\dots[|w|-1, a_{|w|}, |w|].$$

Let H be the CFG obtained from $\hat{G}$ by replacing every $[i-1, a_i, i]$ in the right-hand side of a production by the empty word. Then, obviously,

$$S^k\underset{G,k}{\Longrightarrow} w \text{ if and only if } [i_0, S, i_1]\dots[i_{k-1}, S, i_k]\underset{H,k}{\Longrightarrow}\lambda.$$

Note that the set of choices for the indices $i_0, \dots, i_k$ is of size polynomial in k; each position of w (between 1 and $|w|$) gets assigned to one of the start symbols $[i_{j-1}, S, i_j]$, indicating that the symbol in this position of w is derived from that start symbol. Since there are k start symbols, we get that the total number of possibilities to assign the positions of w to start symbols is bounded by $k^{|w|}$.

It thus remains to be shown that it can be determined in time polynomial in the value of k whether

$$[i_0, S, i_1]\dots[i_{k-1}, S, i_k]\underset{H,k}{\Longrightarrow}\lambda.$$

This follows, however, from Theorem 4.4 (see Remark 4.5). □

5. Membership for nonpropagating grammars

In this section we shall develop a polynomial dynamic programming algorithm that solves the membership problem for k-languages, where k is fixed. We shall also see that the parsing problem is polynomial if, in addition, also the grammar is fixed. This algorithm generalizes results from [12], where parsing algorithms were presented which heavily depend on the assumption that the grammar is propagating.

The main combinatorial step is to show that, for each k-derivation of a word w from a word x, there is a k-derivation forest, the height of which is polynomially bounded in the size of the grammar and in the length of the word. A difficulty is the lack of propagating normal forms for k-parallel rewriting and of normal forms which bound Maxr(G). This is solved by Earley-style algorithms [8, 11]. Our algorithm will be similar to the Earley-style algorithm outlined in [12]. The algorithm of [12] requires no bound on Maxr(G). It requires, however, that the grammar is propagating; one of the dynamic programming parameters is the number of nodes in a derivation subforest; this parameter may grow to exponential size if the grammar has erasing rules. As will be seen later on, this parameter can be replaced by the number of nodes modulo k, whenever the number of nodes exceeds a certain polynomial bound.

The polynomial k-derivation tree-height bounds from [11, 12] hold only for propagating grammars. We proceed to develop a polynomial height bound for the nonpropagating case applying the $(k+1)$-median to k-schedules. (The algorithm, however, will use the k-median.) Theorem 5.1 is the restriction to forests of [7, Theorem 3.1]. Corollary 5.2 gives the $k+1$ version, which follows readily.

5.1. Theorem ([7]). *Let F be a forest, and let σ be a k-schedule for* $\mathrm{High}_k(F)$. *Then there is a k-schedule σ' for the whole forest F such that*

(1) *if $p(\sigma) \geqslant |\mathrm{Low}_k(F)|$, then σ' is at most as long as σ;*

(2) *if $p(\sigma) \leqslant |\mathrm{Low}_k(F)|$, then σ' may have idle periods only in its last slot. Hence,* $p(\sigma) = -|F| \bmod k$.

5.2. Corollary. *Let F be a forest and let σ be a k-schedule for* $\mathrm{High}_{k+1}(F)$. *Then there exists a k-schedule σ' for the whole forest F such that*

(1) *if $p(\sigma) \geqslant |\mathrm{Low}_{k+1}(F)|$, then σ' is at most as long as σ;*

(2) *if $p(\sigma) \leqslant |\mathrm{Low}_{k+1}(F)|$, then σ' may have idle periods only in its last slot. Hence,* $p(\sigma) = -|F| \bmod k$.

Proof. We first construct a k-schedule $\tilde{\sigma}$ for $\mathrm{High}_k(F)$ from σ. For this purpose, we remove all nodes from σ which are in $\mathrm{High}_{k+1}(F)$ but not in $\mathrm{High}_k(F)$. Since our definition of schedules requires that there are no empty slots, we have yet to 'squeeze' the resulting sequence of slots. To obtain the valid schedule $\tilde{\sigma}$, we repeatedly find the first empty slot; then, all the nodes appearing after that slot are scheduled one slot earlier. Let r be the number of nodes which were removed from σ to obtain $\tilde{\sigma}$. Obviously,

$$r = |\mathrm{High}_{k+1}(F)| - |\mathrm{High}_k(F)| = |\mathrm{Low}_k(F)| - |\mathrm{Low}_{k+1}(F)|,$$

and $p(\tilde{\sigma}) \leqslant p(\sigma) + r$.

Clearly, $\tilde{\sigma}$ is at most as long as σ, and it is a schedule of $\mathrm{High}_k(F)$. Now, by Theorem 5.1, there exists a k-schedule σ' of F such that, if σ' is longer than $\tilde{\sigma}$, then σ' has idle periods only in its last slot. We proceed to prove that σ' satisfies the corollary, by performing a case analysis.

Case A: Let $p(\sigma) \leqslant |\mathrm{Low}_{k+1}(F)|$. It follows that $p(\sigma) + r \leqslant |\mathrm{Low}_{k+1}(F)| + r$, and, therefore, $p(\tilde{\sigma}) \leqslant |\mathrm{Low}_k(F)|$. Now, Theorem 5.1(2) with respect to $\tilde{\sigma}$ and σ' guarantees that σ' may have idle periods only in its last slot. Hence, Corollary 5.2(2) holds for Case A.

Case B: Let $p(\sigma) \geqslant |\mathrm{Low}_{k+1}(F)|$ and assume on the contrary that σ' is longer than σ. Since $\tilde{\sigma}$ is at most as long as σ, σ' is also longer than $\tilde{\sigma}$. It follows from Theorem 5.1(2) that σ' has idle periods only in its last slot. The total number of nodes in F is thus at least $k(l'-1)+1$, where l' is the length of σ'. On the other hand, σ contains all nodes of $\mathrm{High}_{k+1}(F)$. From this we obtain that the total number of nodes is at most

$$kl-p(\sigma)+|\mathrm{Low}_{k+1}(F)|\leq kl,$$

where l is the length of σ. Since $l\leq l'-1$, we arrive at a contradiction. Hence, σ' is at most as long as σ, and the corollary holds. □

The following theorem is the prerequisite for a polynomial dynamic programming membership algorithm; it guarantees the existence of a k-derivation forest of bounded height.

5.3. Theorem. *Let $G=\langle\Sigma, P, S, \Delta\rangle$ be a CFG, and let $k\geq 2$. Let $w\in U_k(G)$. Then there is a derivation tree T of w from S with $p(\mathrm{Bare}(T))=k-1$ such that*

$$\mathrm{height}(T)\leq \hat{h}=k|w|^{2k+2}\#\Sigma^{k+2}((|w|+1)\#\Sigma+1)(k+|w|+\#\Sigma).$$

Proof. Let T' be a derivation tree of w from S with $p(\mathrm{Bare}(S))=k-1$. Such a derivation tree exists by Lemma 2.6(2). If $\mathrm{height}(T')\leq\hat{h}$, then the theorem holds for $T=T'$. Otherwise, we shall obtain from T' the required derivation tree T. We look at an optimal schedule σ' of $\mathrm{Bare}(T')$. The construction is performed in two stages. A coarse outline of the proof follows.

First, we look at all those initial slots of σ' for which the median and the number of nodes in the low forest do not exceed certain polynomial bounds. It will be shown that each such 'constrained' derivation step (i.e., slot) can be characterized by a triplet $\langle v, m, c\rangle$, where v is the root word of the $(k+1)$-high forest, m is the $(k+1)$-median, and c is the size of the $(k+1)$-low forest. There is at most a polynomial number of these triplets, because all components of a triplet are polynomially bounded. Given the sequence of triplets that corresponds to the constrained derivation steps of the derivation of T', we can eliminate sequences of triplets which begin and end at the same triplet. Returning from triplets to derivations, we obtain a derivation that is equivalent to T and which has the same initial triplet and thus the same number of idle periods. Moreover, the constrained initial part of the derivation is of polynomial length.

The second stage (when either the median or the low forest size becomes too large) uses the shrinking technique of Lemma 3.4 to reduce the size of derivation forests. If the $(k+1)$-median is bounded and the size of the $(k+1)$-forest is large (Case A), then first shrinking and then unshrinking to the proper modulo class of the number of nodes yields the height bound. In Case B, where the $(k+1)$-median is large, the shrinking technique is used to produce k trees of polynomial heights such that the heights are almost equal. The resulting forest will be shown to have the same number of idle periods as the original one, i.e., zero, which will conclude the proof of the theorem.

As in Lemma 3.4, we distinguish between nonpropagating, propagating, and growing nodes. We extend the alphabet to the alphabet Ξ; each nonpropagating symbol A is represented by A itself, whereas each propagating symbol is represented by a compound symbol $[i, A, j]$; this indicates that A derives the subword of w from the ith through the jth symbol.

Let H be the grammar obtained from G using the alphabet Ξ. Note that H can be easily obtained from G using the standard regular intersection technique (see, e.g., [14]). Let U' be the derivation tree obtained from T' by replacing every propagating symbol with the corresponding compound symbol. Obviously,

$$p(\text{Bare}(U')) = p(\text{Bare}(T')) = k-1.$$

We say that two derivation forests F and F' are *equivalent* if their roots are labelled identically, if they derive the same word, and if $p(\text{Bare}(F)) = p(\text{Bare}(F'))$.

We will construct a derivation tree U in H that is equivalent to U', of height $\leqslant \hat{h}$. Upon replacing in U each compound symbol by the original nonterminal symbol, the required derivation tree T is obtained.

A more detailed outline of the construction of U follows. The construction of U will consist of two stages; in Stage I, we will construct an equivalent derivation tree $\hat{U}$ with the following property. Let $\hat{\sigma}$ be a highest-level-first schedule for Bare$(\hat{U})$. Let $\hat{U}_i$ be the subforest of $\hat{U}$ which contains the nodes scheduled in $\hat{\sigma}$ at slot i or later, as well as the leaves of $\hat{U}$. We call $\hat{U}_i$ the *i-th subforest* of $\hat{U}$. Let f be the first slot of $\hat{\sigma}$ such that

$$|\text{Low}_{k+1}(\text{Bare}(\hat{U}_i))| \geqslant (k-1)(k+\#\Sigma+|w|)\#\Sigma \quad \text{or} \quad \mu_{k+1}(\text{Bare}(\hat{U}_i)) > (|w|+1)\#\Sigma.$$

Then the forest $\hat{E}$, obtained by deleting from $\hat{U}$ all the non-root nodes of $\hat{U}_f$, will satisfy

$$\text{height}(\hat{E}) = h_{\hat{E}} \leqslant ((|w|^2\#\Sigma)^{k+1}-1)((|w|+1)\#\Sigma+1)(k-1)(k+|w|+\#\Sigma)\#\Sigma.$$

In Stage II, we shall show that for $\hat{U}_f$ there is an equivalent derivation forest F such that

$$\text{height}(F) \leqslant h_F = \max(k(k+\#\Sigma)\#\Sigma, (|w|+k+\#\Sigma)\#\Sigma).$$

Let U be the result of replacing $\hat{U}_f$ by F in $\hat{U}$. Then, U is equivalent to $\hat{U}$, and its height is bounded by $h_{\hat{E}} + h_F$. The theorem then follows from the inequality

$$h_{\hat{E}} + h_F \leqslant (|w|^2\#\Sigma)^{k+1}((|w|+1)\#\Sigma+1)k(k+|w|+\#\Sigma)\#\Sigma = \hat{h}.$$

We now present the detailed proof. To show the height bound for Stage I, let σ' be an HLF schedule for Bare(U'), and let U'_i denote the ith subforest of U' with respect to σ'. Let f' be the first slot of σ' such that

$$|\text{Low}_{k+1}(\text{Bare}(U'_i))| \geqslant (k-1)(k+\#\Sigma+|w|)\#\Sigma \quad \text{or}$$
$$\mu_{k+1}(\text{Bare}(U'_i)) > (|w|+1)\#\Sigma.$$

We associate with each U'_i, $1 \leqslant i \leqslant f'$, a triplet

$$\langle \text{word}(\text{High}_{k+1}(\text{Bare}(U'_i))),\ \mu_{k+1}(\text{Bare}(U'_i)),\ |\text{Low}_{k+1}(\text{Bare}(U'_i))|\rangle,$$

where word(U'_i) designates the word that is formed by the labellings of the roots of U'_i. The number of triplets is bounded by $h_{\hat{E}}$, because (i) there are at most $(|w|^2\#\Sigma)^{k+1}-1$ possible words over Ξ of length up to k, and, by the definition of f', (ii) the median is bounded by $(|w|+1)\#\Sigma$, and (iii) the low-forest size is bounded by $(k-1)(k+\#\Sigma+|w|)\#\Sigma-1$.

We construct an equivalent derivation tree $\hat{U}$ for U' with the property that all triplets are distinct. Assume that there is, in U', a triplet that occurs twice, say for U'_{i_1} and U'_{i_2}, $i_1 < i_2$. Note that $i_1 \neq 1$, because there is only a single node in the first slot of σ', whereas all other slots contain k nodes. The high forests $\text{High}_{k+1}(\text{Bare}(U'_{i_1}))$ and $\text{High}_{k+1}(\text{Bare}(U'_{i_2}))$ have the same root word. Moreover, the jth component trees of $\text{High}_{k+1}(U'_{i_1})$ and of $\text{High}_{k+1}(U'_{i_2})$ yield the same subword of w. (This is guaranteed by the extended alphabet Ξ.) The low forests, however, do not necessarily have identical root words. Let thus $\hat{U}_{i_1}$ be the forest obtained from U'_{i_1} (and from U'_{i_2}) as follows.

Let $C_{j,1}$ be the component tree of U'_{i_1} whose bare tree is the jth component of $\text{High}_{k+1}(\text{Bare}(U'_{i_1}))$, and let $C_{j,2}$ be defined similarly with respect to $\text{High}_{k+1}(\text{Bare}(U'_{i_2}))$. Then $\hat{U}_{i_1}$ is the result of replacing $C_{j,1}$ by $C_{j,2}$ in U'_{i_1} for all $1 \leqslant j \leqslant \#\text{High}_{k+1}(\text{Bare}(U'_{i_1}))$.

Since $C_{j,1}$ and $C_{j,2}$ have the same root labellings for all j, it follows that $\hat{U}_{i_1}$ has the same root word as U'_{i_1}. Moreover, the use of triplets guarantees that $C_{j,1}$ and $C_{j,2}$ derive the same words for all j. To prove that $\hat{U}_{i_1}$ is equivalent to U'_{i_1} it remains, therefore, to be shown that $p(\hat{U}_{i_1})=0$.

We observe that

$$p(\text{High}_{k+1}(\text{Bare}(U'_{i_2}))) \leqslant |\text{Low}_{k+1}(\text{Bare}(U'_{i_2}))|, \tag{1}$$

because $p(\text{Bare}(U'_{i_2}))=0$. On the other hand, the equality of the triplets for U'_{i_1} and U'_{i_2} implies that

$$|\text{Low}_{k+1}(\text{Bare}(U'_{i_1}))| = |\text{Low}_{k+1}(\text{Bare}(U'_{i_2}))| \quad \text{and}$$
$$\mu_{k+1}(\text{Bare}(U'_{i_1})) = \mu_{k+1}(\text{Bare}(U'_{i_2})). \tag{2, 3}$$

It follows from (3) that

$$\text{High}_{k+1}(\text{Bare}(\hat{U}_{i_1})) = \text{High}_{k+1}(\text{Bare}(U'_{i_2})) \quad \text{and}$$
$$\text{Low}_{k+1}(\text{Bare}(\hat{U}_{i_1})) = \text{Low}_{k+1}(\text{Bare}(U'_{i_1})). \tag{4}$$

Combining (1), (2), and (4), we obtain

$$p(\text{High}_{k+1}(\text{Bare}(\hat{U}_{i_1}))) \leqslant |\text{Low}_{k+1}(\text{Bare}(\hat{U}_{i_1}))|.$$

We can now apply Corollary 5.2(2), stating that

$$p(\text{Bare}(\hat{U}_{i_1})) = -|\text{Bare}(\hat{U}_{i_1})| \bmod k.$$

But, by (2) and (4),

$$|\text{Bare}(\hat{U}_{i_1})| = |\text{Bare}(U'_{i_2})| = 0 \bmod k,$$

and, thus, $p(\mathrm{Bare}(\hat{U}_{i_1}))=0$. It follows that $\hat{U}_{i_1}$ is a k-derivation forest which is equivalent to U'_{i_1}.

We can thus 'cut and paste' U', replacing the forest U'_{i_1} by $\hat{U}_{i_1}$, obtaining a derivation forest of w from S with $k-1$ idle periods. We repeat this process, until there are no more repeating triplets. Let $\hat{U}$ be the resulting forest. Obviously, $\hat{U}$ is equivalent to U'. Let f be defined for $\hat{U}$ analogously to the definition of f' for U'. Let $\hat{E}$ be the tree obtained from $\hat{U}$ by deleting all the non-root nodes of $\hat{U}_f$. Then $\mathrm{height}(\hat{E})\leqslant h_{\hat{E}}$.

Stage II is to show that for $\hat{F}=\hat{U}_f$ there is an equivalent derivation forest F whose height is bounded by h_F. We distinguish between two cases.

Case A: Let

$$\mu_{k+1}(\mathrm{Bare}(\hat{F}))\leqslant(|w+1|)\#\Sigma \quad\text{and}\quad |\mathrm{Low}_{k+1}(\mathrm{Bare}(\hat{F}))|\geqslant(k-1)(k+\#\Sigma+|w|)\#\Sigma.$$

To construct F, we eliminate shrinks from every component tree $\hat{T}_i$ of $\hat{F}$ whose bare tree is in $\mathrm{High}_{k+1}(\mathrm{Bare}(\hat{F}))$ such that the height of each resulting tree is at least $(|w|+1)\#\Sigma\geqslant\mu_{k+1}(\mathrm{Bare}(\hat{F}))$, but at most $(|w|+2)\#\Sigma-1$. A similar shrinking process is shown in detail in Lemma 3.4. Call the resulting trees $\tilde{T}_i$. We re-insert shrinks into each tree $\tilde{T}_i$, obtaining a new tree T_i such that $|\mathrm{Bare}(T_i)|=|\mathrm{Bare}(\hat{T}_i)| \bmod k$. (As in Lemma 3.4, this is done in two phases: first, we guarantee that each symbol which occurred in $\hat{T}_i$ occurs also in T_i; then, we insert up to $k-1$ shrinks, until the bare tree is in the correct modulo class.) For all those trees $\hat{T}_i$ whose bare trees are in $\mathrm{Low}_{k+1}(\mathrm{Bare}(\hat{F}))$ we let $T_i=\hat{T}_i$.

It follows that

$$\mathrm{height}(T_i)\leqslant(|w|+2)\#\Sigma-1+(\#\Sigma-1)\#\Sigma+(k-1)\#\Sigma\leqslant(\#\Sigma+k+|w|)\#\Sigma.$$

Replacing each tree $\hat{T}_i$ in $\hat{F}$ by T_i we obtain a forest F such that

$$\mathrm{height}(F)\leqslant(\#\Sigma+k+|w|)\#\Sigma\leqslant h_F.$$

Obviously, eliminating and re-inserting shrinks preserves the roots, as well as those leaves which are not labelled with λ. Hence, F and $\hat{F}$ are forests of the same derivation.

To complete Case A, it thus remains to be proven that $p(F)=p(\hat{F})=0$, i.e., that F is also a k-derivation forest. We first note that an HLF schedule contributes at most $k-1$ idle periods for each height (including height zero). Since

$$\mathrm{height}(\mathrm{Bare}(F))\leqslant(\#\Sigma+k+|w|)\#\Sigma-1,$$

it follows that

$$p(\mathrm{High}_{k+1}(\mathrm{Bare}(F)))\leqslant(k-1)(\#\Sigma+k+|w|)\#\Sigma.$$

On the other hand, by the assumption of Case A,

$$|\mathrm{Low}_{k+1}(\mathrm{Bare}(F))|\geqslant(k-1)(\#\Sigma+k+|w|)\#\Sigma.$$

Hence, $p(\mathrm{High}_{k+1}(\mathrm{Bare}(F)))\leqslant|\mathrm{Low}_{k+1}(\mathrm{Bare}(F))|$.

Corollary 5.2(2) now yields the existence of a k-schedule of Bare(F) that may have idle periods only in its last slot. Since $|\text{Bare}(\hat{F})| = 0 \bmod k$ and our construction has preserved the modulo class of each bare tree, it follows that $|\text{Bare}(F)| = 0 \bmod k$. Hence, there are no idle periods in the last slot of the schedule of Bare(F), and we conclude that F is a k-derivation forest that is equivalent to $\hat{F}$.

Case B: Let $\mu_{k+1}(\text{Bare}(F)) > (|w|+1)\#\Sigma$. This is the part of the proof that needs the $(k+1)$-median rather than the k-median; the construction relies on the fact that there are $k+1$ 'high' component trees. We first shrink every tree $\hat{T}_i$ of $\hat{F}$ the height of which exceeds $(|w|+2)\#\Sigma$, until its height is between $(|w|+1)\#\Sigma+1$ and $(|w|+2)\#\Sigma$. Then we re-insert shrinks into exactly $k+1$ of those trees, until the height of each new tree is at least $(k-1)(k+\#\Sigma)\#\Sigma$ and at most $\max((k-1)(k+\#\Sigma)+1, |w|+2)\#\Sigma$. This can be done because every tree of height $>(|w|+1)\#\Sigma$ contains at least one shrink (see the proof of Lemma 3.4).

Finally, we insert shrinks to create the tree T_i such that each bare tree is of the same modulo class as the original $\hat{T}_i$; in addition, we also require the set of symbols occurring in T_i to be the same as in $\hat{T}_i$. At most $\#\Sigma - 1 + k - 1$ shrinks have to be inserted for this purpose (see Lemma 3.4). Therefore,

$$\text{height}(T_i) \leqslant \max(k(k+\#\Sigma), |w|+k+\#\Sigma)\#\Sigma = h_F.$$

Moreover, the $k+1$ highest trees differ in height by at most $(k+\#\Sigma)\#\Sigma$ and they all are above $\mu_{k+1}(\text{Bare}(\hat{F}))$.

Let F be the forest obtained replacing each $\hat{T}_i$ by T_i. Since height(F) $\leqslant h_F$, it remains to show that F is a k-derivation forest, i.e., that $p(F)=0$. We use the following two claims. The first one is a consequence of [7, Lemma 3.1].

Claim 1. *Let B be a forest. If $|B| = 0 \bmod k$ and B has at least k trees of height* $\geqslant$height(B) -1, *then* $p(B)=0$.

Proof of Claim. The proof works by induction on the size of $|B|$. The claim trivially holds if height(B) $=0$. Otherwise, let B' be a subforest obtained from B by removing the k highest roots. Obviously, $p(B') = p(B)$. Then it is easy to see that either height(B') $=0$, in which case we are done, or else B' contains at least k trees of height $\geqslant$height(B') -1. Since $|B'| < |B|$, the claim follows from the induction hypothesis. □

Claim 2. *Let B be a forest. If $|B| = 0 \bmod k$, and if*

$$\sum_{T \text{ in } B} \text{height}(T) \geqslant k\ \text{height}(B),$$

then $p(B)=0$.

Proof of Claim. We operate again by induction on $|B|$. We first note that B has at least k roots; otherwise, the sum of the component tree heights would be less than k fixed height(B). If B has at least k trees of height height(B), then $p(B)=0$. This follows from Claim 1. Let thus B' be a subforest obtained from B by removing k highest roots. Clearly, height(B') $=$ height(B) -1. The sum $\sum_{T \text{ in } B}$ height(T)

decreases at most by k when the k roots are removed from B (note that some trees might split into several trees). Hence,

$$\sum_{T' \text{ in } B'} \text{height}(T') \geqslant k \text{ height}(B) - k = k(\text{height}(B) - 1) = k \text{ height}(B').$$

The claim thus follows from the induction hypothesis since $|B'| < |B|$. □

We use Claim 2 to show that $p(\text{Bare}(F)) = 0$. Obviously, the highest tree of Bare(F) has height height(Bare(F)). From the construction of F we know that the next $k-1$ highest trees have height $\geqslant \text{height}(F) - (k + \#\Sigma)\#\Sigma$ and that tree $k+1$ has height $\geqslant (k-1)(k + \#\Sigma)\#\Sigma$. Hence,

$$\begin{aligned}\sum_{T \text{ in Bare}(F)} \text{height}(T) &\geqslant \text{height}(\text{Bare}(F)) + (k-1)(\text{height}(\text{Bare}(F)) - (k + \#\Sigma)\#\Sigma) \\ &\quad + (k + \#\Sigma)(k-1)\#\Sigma \\ &\geqslant k \text{ height}(\text{Bare}(F)),\end{aligned}$$

and, by Claim 2, $p(\text{Bare}(F)) = 0$, which completes also Case B. □

Having obtained a polynomial bound on the height of a derivation forest we can develop a dynamic programming membership algorithm for arbitrary grammars. In the algorithm we will again use k-medians, k-high forests, and k-low forests.

We define frames, as introduced in [12].

5.4. Definition. Let G be the CFG $\langle \Sigma, P, S, \Delta \rangle$, and let $w = a_1 \ldots a_n$, where $a_1, \ldots, a_n \in \Delta$.

A *frame* R (of w) is quintuple $\langle A, l, r, h, c \rangle$ such that

- $A \in \Sigma$ is the *root* of R;
- $1 \leqslant l$ and $l - 1 \leqslant r \leqslant n$;
- there is a derivation tree T of $a_l \ldots a_r$ from A in G such that its bare tree has height h and c nodes. If the derivation tree is of height zero, i.e., A is a terminal symbol, then $c = 0$ and $h = -1$.

A tree T as above is called a *frame tree* for R.

The *height* of R is $\text{height}(R) = h$; the *size* $|R|$ of R is c. An ordered set $\mathcal{R}$ of frames is called a *frame collection*. height($\mathcal{R}$), the *height* of $\mathcal{R}$, is the maximum of the frame heights in $\mathcal{R}$. The *size* $|\mathcal{R}|$ of $\mathcal{R}$ is the sum of the sizes of the frames in $\mathcal{R}$.

If F is a forest such that the ith tree in F is a frame tree for the ith frame in $\mathcal{R}$ for $1 \leqslant i \leqslant \#F = \#\mathcal{R}$, then F is called a *frame forest* of $\mathcal{R}$.

5.5. Example. The forest of Fig. 1 is a frame forest for the frame collection $\{\langle A, 1, 6, 2, 3 \rangle, \langle C, 7, 10, 1, 2 \rangle\}$.

The notions of k-median, k-high collection (k-high forest) and k-low collection (k-low forest) carry over from forests to frame collections in the obvious way. In particular,

$$p(\mathcal{R}) = \min\{p(\text{Bare}(F)) : F \text{ is a frame forest of } \mathcal{R}\}.$$

To recur from a frame to its child frames (i.e., from a forest to the subtrees at the children of its roots) we need the following definition.

5.6. Definition. Let $R = \langle A, l, r, h, c\rangle$ be a frame of a word w and let $\mathcal{R} = \{R_1, \ldots, R_j\}$ be a frame collection of w, where $R_i = \langle A_i, l_i, r_i, h_i, c_i\rangle$ for all $1 < i \leqslant j$. We say that $\mathcal{R}$ is a *child collection* of R if

- $A \to A_1 \ldots A_j \in P$;
- $l = l_1$, $r_j = r$, and $l_i = r_{i-1} + 1$ for all $2 \leqslant i \leqslant j$;
- $h = 1 + \max\{h_1, \ldots, h_j\}$;
- $c = 1 + \sum_{i=1}^{j} c_i$.

A *child collection* of a frame collection $\mathcal{R}$ is obtained by choosing a child collection for each of the frames in $\mathcal{R}$ and by taking their union.

The following lemmas from [12] provide the recurrence rules for the dynamic programming algorithm presented there.

5.7. Lemma ([12, Corollary 3.10]). *Let $\mathcal{R}$ be a frame collection. Then*

$$p(\mathcal{R}) = \begin{cases} p(\mathrm{High}_k(\mathcal{R})) - |\mathrm{Low}_k(\mathcal{R})| & \textit{if } p(\mathrm{High}_k(\mathcal{R})) \geqslant |\mathrm{Low}_k(\mathcal{R})|, \\ -|\mathcal{R}| \bmod k & \textit{otherwise.} \end{cases}$$

5.8. Lemma ([12, Lemma 3.11]). *Let j be the number of frames in a frame collection $\mathcal{Q}$, where $\mathcal{Q} = \mathrm{High}_k(\mathcal{Q})$. Then*

$$p(\mathcal{Q}) = \begin{cases} 0 & \textit{if } \mathcal{Q} \textit{ is empty}, \\ k - j + \min\{p(\mathcal{R}') : \mathcal{R}' \textit{ is a child collection of } \mathcal{Q}\} \\ \quad \textit{otherwise.} \end{cases}$$

Note that the possible number of symbols in a frame is bounded by the size of the alphabet, the positions are bounded by the length of the word, and, by Theorem 5.3, the height is bounded by $\hat{h}$. The problem is that the number of nodes may get exponentially large. This can, however, be overcome. To determine the number of idle periods of a frame collection, we only need the number of idle periods of its k-high collection and the number of nodes in its k-low collection (Lemma 5.7). Moreover, since the number of idle periods for an HLF schedule is at most $k-1$ times the height, it follows that we have to keep track only of the modulo class of the number of nodes if it exceeds $\hat{h}(k-1)$.

We thus modify the size component c of a frame to be either an integer in the range $0 \leqslant c \leqslant \hat{h}(k-1)$ or a modulo class; we represent the class $i \bmod k$ by $[i]$. The size of a frame collection is defined accordingly. It is easy to see that Lemmas 5.7 and 5.8 hold for this modified definition of frames.

In the sequel we have to add integers to integers, integers to modulo classes, or modulo classes to modulo classes. For this purpose, we define the operation $\oplus$, as follows:

$$i \oplus j = \begin{cases} i+j & \text{if } i+j \leq \hat{h}(k-1), \\ [i+j] & \text{otherwise;} \end{cases}$$

$$[i] \oplus j = i \oplus [j] = [i+j] \quad \text{and} \quad [i] \oplus [j] = [i+j] \qquad \text{for all } 0 \leq i, j \leq k-1.$$

If Maxr(G) were constant (which is clearly the case for a fixed G), we could, at this point, have formulated a polynomial dynamic programming membership algorithm which first finds all the frames and then computes the number of idle periods by growing height. This was done in [12] for propagating grammars. Such an algorithm can also be used to extract the parse tree in polynomial time.

Since it is not known whether there is a normal form, where Maxr(G) is bounded, we introduce structures called 'items', which are the generalization of Earley's 'dotted items' to frames. An item represents a 'partially complete frame', as follows.

An *item* R is an octuple of the form $[A \to B_1 \ldots B_m, i, l, r, h, c, \mathscr{H}, \mu]$, where

- $A \to B_1 \ldots B_m$ is the production used at the 'root' of the frame;
- i represents the dot in Earley's dotted items [8]; it indicates that we have already processed frames for $B_1, \ldots, B_i$, i.e., there exists a frame collection $\mathscr{R}$ with $\#\mathscr{R} = i$, the jth frame in $\mathscr{R}$ has B_j as its root for $1 \leq j \leq i$. $\mathscr{R}$ covers $A_{p_l} \ldots A_{p_r}$, the height of $\mathscr{R}$ is h, c is the size of the k-low collection of $\mathscr{R}$, $\text{High}_k(\mathscr{R}) = \mathscr{H}$, and $\mu_k(\mathscr{R}) = \mu$.

If $i = m$, then the item is called *complete*; it then corresponds to the frame $Q = \langle A, l, r, h+1, c' \rangle$, where $c' = 1 + c + |\mathscr{H}|$ (counting 1 for the node at the root, c for the nodes in the k-low collection, and $|\mathscr{H}|$ for the nodes in the k-high collection). We denote this Q by frame(R), where R is the complete item. In consistency with frames we denote the height of an item R by height(R) and its size by $|R|$.

We observe that, similarly as for frames, the total number of items is also polynomial in the size of G and in $|w|$. Based on the recurrences of Lemmas 5.7 and 5.8, we can now formulate the algorithm. Its acceptance condition is expressed by the following lemma (this is the frame version of Lemma 2.6(2); see [12, Lemma 3.7]).

5.9. Lemma ([12, Lemma 3.7]). *Let $G = \langle \Sigma, P, S, \Delta \rangle$ be a CFG. A word w is in $U_k(G)$ if and only if there exists a frame $R = \langle S, 1, |w|, h, c \rangle$, for some h and c such that $p(R) = k-1$.*

The algorithm follows.

Algorithm UM;
Given: an integer k.
Input: a CFG $G = \langle \Sigma, P, A_1, \Delta \rangle$, where $\Sigma = \{A_1, \ldots, A_{\#\Sigma}\}$, and a word $w \in \Delta^*$, $w = A_{p_1} \ldots A_{p_{|w|}}$.
Output: ACCEPT if $w \in U_k(G)$, otherwise REJECT.

```
begin
  comment test whether $w$ is directly derived from $S$;
  if $A_1 \to w$ then ACCEPT;
  comment construct all the frames for the leaves and erasing nodes;
  for $i := 1$ to $|w|$ do
  begin
    $\langle A_{p_i}, i, i, -1, 0\rangle$ is a frame and $p(\langle A_{p_i}, i, i, -1, 0\rangle) = 0$;
    for all $A \to \lambda$ in $P$ do $\langle A, i, i-1, 0, 1\rangle$ is a frame and $p(\langle A, i, i-1, 0, 1\rangle) = k-1$;
  end;
  comment construct all the items of height $h$;
  for $h := 0$ to $\hat{h}$ do
  for all $A \to B_1 \dots B_m$ in $P$ do
  begin
    comment construct all the items of height $h$, with $i = 1$;
    for $l := 1$ to $|w|$ do
    for all frames $Q := \langle B_1, l, r, h-1, c'\rangle$ do
    begin
      if $h = 1$ then begin
        $\mathcal{H} := \{\ \}$; $c := c'$;
      end else begin
        $\mathcal{H} := \{Q\}$; $c := 0$;
      end;
    $R := [A \to B_1 \dots B_m, 1, l, r, h, c, \mathcal{H}, 0]$ is an item;
    if $m = 1$ then frame($R$) is a frame;
    end;
  end;
  comment we construct all the items of height $h$ with $i \geq 2$;
  for $i := 2$ to $m$ do
  for all items $[A \to B_1 \dots B_m, i-1, l, r', h', c', \mathcal{H}', j']$ with $h' \leq h$ do
  for all frames $Q := \langle B_i, r'+1, r, h'', c'\rangle$ with $h = \max(h', h'')$ do
  begin
    $j := \max(\mu_k(\mathcal{H}' \cup \{Q\}), j')$;
    $\mathcal{H} := \mathrm{High}_k(\mathcal{H}' \cup \{Q\})$;
    $c := c' + |(\mathcal{H}' \cup \{Q\}) - \mathcal{H}|$;
    $R := [A \to B_1 \dots B_m, i, l, r, h, c, \mathcal{H}, j]$ is an item;
    if $i = m$ then frame($R$) is a frame;
  end;
  comment compute the number of idle periods for all collections of up to $k-1$
          frames, by increasing height;
  $p(\{\ \}) := 0$;
  for $h := 1$ to $\hat{h}$ do
  for all frame collections $\mathcal{F} = \{F_1, \dots, F_r\}$ of height $h$ with $r \leq k-1$ do
  begin
```

```
        comment we find the number of idle periods for a frame collection $\mathcal{F}$ consisting
                of $r \leq k-1$ frames, by computing the number of idle periods for all
                the collections of $r$ items which correspond to the frames, and by
                taking the minimum;
        $q := \infty$;
        for all collections of complete items $\mathcal{R} = \{R_1, \ldots, R_r\}$ such that $F_i = \text{frame}(R_i)$
            do
        begin
          let $R_i = [C_i \to v_i, |v_i|, l_i, r_i, h_i, c_i, \mathcal{Q}_i, j_i]$;
          $\mathcal{Q} := \text{High}_k(\bigcup_{i=1}^{r} \mathcal{Q}_i)$;
          $\mathcal{L} := \text{Low}_k(\bigcup_{i=1}^{r} \mathcal{Q}_i)$;
          comment since height$(\mathcal{Q}) \leq h-1$, we can recur on $p(\mathcal{Q})$;
          if $p(\mathcal{Q}) \leq (k-1)\hat{h}$ then begin
            comment lowc is the size of the low collection of the child collection of $\mathcal{R}$;
            lowc$:= |\text{Low}_k(R_1)| \oplus \cdots \oplus |\text{Low}_k(R_r)| \oplus |\mathcal{L}|$;
            if lowc is not a modulo class and $p(\mathcal{Q}) \geq$ lowc
              then $q := \min(q, p(\mathcal{Q}) - \text{lowc})$
              else $q := \min(q, -\text{lowc} \bmod k)$;
          end;
        end;
        comment fix the number of idle periods for $\mathcal{F}$;
        $p(\mathcal{F}) := k - r + q$;
      end;
      comment this is the membership test;
      for $h := 0$ to $\hat{h}$ do
      for all $c \in \{1, \ldots, (k-1)\hat{h}, [0], \ldots, [k-1]\}$ do
        if $Q := \langle A_1, 1, |w|, h, c\rangle$ is a frame and $p(Q) = k-1$ then ACCEPT;
      REJECT;
    end;
```

5.10. Theorem. *The membership problem for* $U_k(G)$ *is in* P, *for constant* k, *even if the grammar is variable.*

Proof. Theorem 5.3 allows us to restrict ourselves to a polynomial height. As pointed out before, the number of frames and items is polynomial in the size of the grammar and of the word to be tested. Hence, Algorithm UM runs in polynomial time. □

Algorithm UM can easily be turned into a parsing algorithm if the grammar is fixed. On the one hand, we have to be able to determine the set of child collections for any frame. This can obviously be done in polynomial time (see [12, Theorem 3.12]). On the other hand, we can show by using the shrinking technique that for each parse tree there is an equivalent parse tree of identical height and modulo class whose number of nodes is polynomial in $|w|$. We thus arrive at the following theorem.

5.11. Theorem. *The parsing problem is polynomial for fixed k and fixed arbitrary grammars G.*

6. Summary

In this paper we have continued and completed the investigation of the complexity of various open membership problems for unrestricted (i.e., not necessarily adjacent) k-parallel rewriting that was begun in [12]. Using Scheduling Theory arguments as well as combinatorial properties of derivations, we have obtained polynomial bounds on derivations as prerequisites to the complexity results. Table 1 summarizes the bounds and the complexity results obtained in this paper, as well as earlier results from [12] (these are marked by asterisks). As defined in Section 3,

$$\bar{d} = k(|w| + \#\Sigma + k)\#\Sigma(\mathrm{Maxr}(G)+1).$$

Table 1

Fixed	Variable	Complexity	Bound
k	G, w	P	Forest height: $k\|w\|^{2k+2}\#\Sigma^{k+2}((\|w\|+1)\#\Sigma+1)(k+\|w\|+\#\Sigma)$
G	k, w	NP-complete (*NP-hard)	Derivation length: $\|w\|(\bar{d}+1)^{\#\Sigma}(\|w\|+1)^{\#\Sigma^2}$; Derivation width: $\bar{d}$
G	k, unary w	P	Derivation length: $(\bar{d}+1)^{\#\Sigma}$; width: $\bar{d}$
w	k, G	*NP-hard	-
G, w	k	P	Reduces to variable (k, unary w)

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).

[2] R.V. Book, On the complexity of formal grammars, *Acta Inform.* **9** (1978) 171-182.

[3] J. Bruno, Deterministic and stochastic scheduling problems with treelike precedence constraints, *Proc. NATO Conference*, Durham, England 1981.

[4] C. Beeri and E. Shamir, Checking stacks and context-free programmed grammars accept P-complete languages, *Proc. 2nd Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science **14** (Springer, Berlin, 1974) 27-33.

[5] E. Dahlhaus and H. Gaifman, Concerning two-adjacent context-free languages, *Theoret. Comput. Sci.* **41** (1985) 169-184.

[6] D. Dolev and M.K. Warmuth, Scheduling flat graphs, *SIAM J. Comput.* **14** (1985) 638-657.

[7] D. Dolev and M.K. Warmuth, Profile scheduling of opposing forests and level orders, *SIAM J. Algebraic Discrete Methods* **6** (1985) 665-687.

[8] J. Earley, An efficient context-free parsing algorithm, *Comm. ACM* **13** (1970) 94-102.

[9] S.A. Greibach and J.E. Hopcroft, Scattered context grammars, *J. Comput. System Sci.* **3** (1969) 233-247.

[10] J. Gonczarowski, H.C.M. Kleijn and G. Rozenberg, Closure properties of selective substitution grammars, Part I and II, *Internat. J. Comput. Math.* **14** (1983) 19-42 and 109-135.

[11] 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.

[12] J. Gonczarowski and M.K. Warmuth, Applications of scheduling theory to formal language theory, *Theoret. Comput. Sci.* **37** (1985) 217-243.
[13] M.A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, Reading, MA, 1978).
[14] J.E. Hopcroft and J.D. Ullman, *Formal Languages and Their Relation to Automata* (Addison-Wesley, Reading, MA, 1969).
[15] N.C. Hu, Parallel sequencing and assembly line problems, *Oper. Res.* **9**(6) (1961) 841-848.
[16] H.C.M. Kleijn and G. Rozenberg, Context-free like restrictions on selective rewriting, *Theoret. Comput. Sci.* **16** (1981) 237-269.
[17] H.C.M. Kleijn and G. Rozenberg, On the generative power of regular pattern grammars, *Acta Inform.* **20** (1983) 391-411.
[18] J. Opatrny and K. Culik, II, Time complexity of recognition and parsing of E0L Languages, in: A. Lindenmayer and G. Rozenberg, eds., *Automata, Languages, Development* (North-Holland, Amsterdam, 1976) 243-250.
[19] M. Penttonen, One-sided and two-sided context in formal grammars, *Inform. and Control* **25** (1974) 371-392.
[20] G. Rozenberg, Selective substitution grammars, Part I, *Elektron. Informationsverarb. Kybernet.* **13** (1977) 455-463.
[21] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L-Systems* (Academic Press, New York, 1980).
[22] A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).
[23] I.H. Sudborough, The time and tape complexity of developmental languages, *Proc. 4th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science **52** (Springer, Berlin, 1977) 509-523.
[24] J. van Leeuwen, The membership question for ET0L languages is polynomially complete, *Inform. Process. Lett.* **3** (1975) 138-143.
[25] D.H. Younger, Recognition and parsing of context-free languages in time n^3, *Inform. and Control* **10** (1967) 189-208.