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

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 $k \geq 3$, 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 $\overset{+}{\Rightarrow}_{G,k}$ and $\overset{*}{\Rightarrow}_{G,k}$ the transitive closure and the reflexive and transitive closure of $\Rightarrow_{G,k}$, respectively. If $u \overset{*}{\Rightarrow}_{G,k} v$, 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 $1 \leqslant i \leqslant l$,

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

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 $\max\{|x_i| : 1 \leqslant i \leqslant l+1\}$, 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 $S \overset{*}{\Rightarrow}_G x$.

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 $k \geqslant 2$. To avoid other trivial cases, we assume also that $\mathrm{Maxr}(G) \geqslant 2$.

(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 $\mathrm{height}(F)$. 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 Bare(F). Note that, for all forests F, height(Bare(F)) = 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.

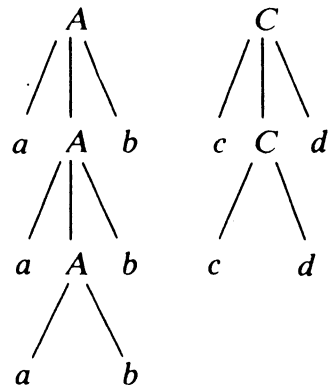

Fig. 1. A derivation forest that is not a 2-derivation forest.

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].

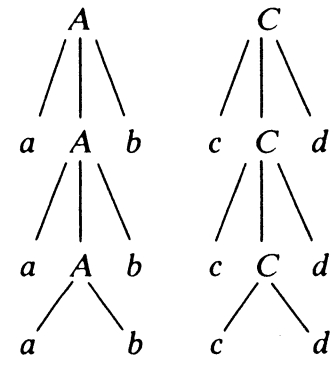

Fig. 2. A 2-derivation forest.

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 $k \geq 2$. A *k-schedule* of F is a function σ mapping the nodes of F onto the set $\{1, \ldots, l\}$, for some $l \leq |F|$, such that

- $1 \leq \#\sigma^{-1}(i) \leq k$ for all $1 \leq i \leq l$;
- 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 $p(\sigma') \leq p(\sigma)$. (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* $p(\text{Bare}(T)) = k - 1$.

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*

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

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;