Discrete Mathematics 21 (1978) 237–252.

EFFECTIVE CONSTRUCTIONS IN WELL-PARTIALLY-ORDERED FREE MONOIDS

Jan van LEEUWEN

Department of Computer Science, The Pennsylvania State University, University Park, PA 16802, U.S.A.

Received 29 October 1976
Revised 6 June 1977

For strings v and w define $v \leq w$ if and only if v is a scattered substring of w. We give a general solution for a problem of Haines to effectively determine regular expressions for $\tilde{L} = \{w : \exists_{v \in L} v \leq w\}$ and $\underset{\sim}{L} = \{v : \exists_{w \in L} v \leq w\}$ when L denotes an arbitrary context-free language. We show by an inductive argument that one can effectively determine $\tilde{L}$ and $\underset{\sim}{L}$ for each language L in the algebraic extension of some family K if and only if one can do so for each language in K.

1. Introduction

Let Σ be a finite (non-empty) set, and let Σ^* be the free monoid generated by Σ. Arbitrary subsets of Σ^* will be called languages.

For strings x and y in Σ^*, we define $x \leq y$ if and only if there are strings $x_1, \ldots, x_k$ and $y_1, \ldots, y_{k+1}$ (some k) such that $x = x_1 \cdots x_k$ and $y = y_1 x_1 \cdots y_k x_k y_{k+1}$. The structure $[\Sigma^*, \leq]$ is called a well-partially-ordered set, because each subset of Σ^* contains only finitely many minimal elements under this partial-order convention (the "finite basis property", see Kruskal [8]).

Let L be a language over Σ.

Definition 1.1. The *ideal* generated by L is the set $\tilde{L} = \{y \in \Sigma^* : \exists_{x \in L} x \leq y\}$.

Definition 1.2. The *co-ideal* generated by L is the set $\underset{\sim}{L} = \{x \in \Sigma^* : \exists_{y \in L} x \leq y\}$.

It follows from the finite basis property that each ideal in Σ^* is finitely generated. Kruskal [8] observed that this fact has been frequently rediscovered, most recently again by Haines [5] in 1969. Haines [5] seems to have been the first to notice that the result implies that for all L the sets $\tilde{L}$ and $\underset{\sim}{L}$ are regular languages in the sense of formal language theory. Haines' observation has an interesting application in the theory of parallel rewriting systems (Van Leeuwen [11]).

It may very well happen even for recursive L that the regular expressions for $\tilde{L}$ and $\underset{\sim}{L}$ cannot be found effectively, in view of the following observation:

Proposition 1.3. *Let K be a language family. If one can determine regular expressions for $\tilde{L}$ or $\underset{\sim}{L}$ effectively for each $L \in K$, then the emptiness problem for languages in K is decidable.*

The question remained to what extent a converse of Proposition 1.3 might be true. In particular, Haines [5, p. 95] posed as an open problem whether or not one can effectively determine $\tilde{L}$ and $\underset{\sim}{L}$ for arbitrary context-free languages L.

Ullian reportedly settled the case for $\tilde{L}$ in an unpublished note to Haines. Haines' problem was apparently not considered any further until in 1974 Walker [12] presented the outline of a proof that for context-free languages L one can indeed effectively determine $\underset{\sim}{L}$ also. His argument was an almost complete construction based on the (finite) context-free grammar for L, which later appeared to be hard to formalize.

In this paper we shall present a new, inductive proof that is simpler and sufficiently general to carry over to the theory of algebraic extensions (as defined in Van Leeuwen [10]). We shall explore under what assumptions the proof-techniques work, and determine much wider families than just the context-free languages for which both $\tilde{L}$ and $\underset{\sim}{L}$ can be effectively determined.

In Section 3 we show that for families K which are effectively closed under intersection with regular sets one can effectively determine $\tilde{L}$ for each $L \in K$ if and only if the languages in K have a decidable emptiness-problem. Note that in this case we have practically the converse of Proposition 1.3. We conclude further that one can effectively determine $\tilde{L}$ for each L in the algebraic extension of K if and only if one can do so for each L in K.

In Section 4 we consider the more difficult problem to effectively determine the sets $\underset{\sim}{L}$, and obtain the main result of this paper. We show that for all families K one can effectively determine $\underset{\sim}{L}$ for each L in the algebraic extension of K if and only if one can do so for each L in K.

The particularly interesting aspect of our result for $\tilde{L}$ and $\underset{\sim}{L}$ in relation to algebraic extensions is that we need not assume any specific closure-property of K to make it work. It follows that the effective constructibility of $\tilde{L}$ and $\underset{\sim}{L}$ translates under algebraic extension for all interesting families.

2. Preliminaries

For the usual terminology and basic results in formal language theory we refer to standard texts like Hopcroft and Ullman [6] and Salomaa [9].

A (deterministic) finite automaton is a structure $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$ with Q a finite set of states, Σ the input alphabet, δ the transition function mapping $Q \times \Sigma$ to Q, q_0 the initial state ($q_0 \in Q$), and F the set of final states ($F \subseteq Q$). We extend δ to $Q \times \Sigma^*$ by defining: $\delta(p, \varepsilon) = p$, and for $x, y \in \Sigma^* \delta(p, xy) = \delta(\delta(p, x), y)$. The language accepted by $\mathcal{M}$ is $R(\mathcal{M}) = \{x \in \Sigma^* : \delta(q_0, x) \in F\}$.

A nondeterministic finite automaton is a similar structure, except that this time δ maps $Q \times (\Sigma \cup \{\varepsilon\})$ to subsets of Q. We extend δ to $Q \times \Sigma^*$ by defining: for x and $y \in \Sigma^*$, $q \in \delta(p, xy)$ if and only if there is a state s such that $s \in \delta(p, x)$ and

$q \in \delta(s, y)$. The accepted language is now defined to be the set $\{x \in \Sigma^* : \delta(q_0, x) \cap F \neq \emptyset\}$. Each nondeterministic finite automaton can be effectively converted into a deterministic finite automaton which accepts the same language.

A language is called regular if and only if it can be accepted by a (nondeterministic) finite automaton. The family of regular languages will be denoted by REG. A regular language is said to be effectively determined if and only if we have an explicit finite automaton for accepting it.

For $\sigma \in \Sigma$ and language M we denote the result of substituting M for all occurrences of σ in L by $\mathscr{S}^{\sigma}_{M}(L)$. A regular substitution over Σ is a mapping τ from languages to languages (over Σ) defined inductively by:

$$\tau(\varepsilon) = \{\varepsilon\},$$
$$\tau(\sigma) = \text{a regular set over } \Sigma \qquad (\sigma \in \Sigma),$$
$$\tau(x_1 \cdots x_n) = \tau(x_1) \cdots \tau(x_n) \qquad (x_i \in \Sigma \text{ for } 1 \leqslant i \leqslant n),$$
$$\tau(L) = \bigcup_{x \in L} \tau(x) \qquad (L \subseteq \Sigma^*).$$

In particular, if each $\tau(\sigma)$ is a finite set then τ is called a finite substitution.

Definition 2.1. A family of languages K is *closed under intersection with regular sets* ("$\cap R$") if and only if for all $L \in K$ and regular sets R we have $L \cap R \in K$. K is *closed under regular (finite) substitution* if and only if for all $L \in K$ and regular (finite) substitutions τ we have $\tau(L) \in K$.

Warning: We shall assume throughout this paper that closure properties of a family K are listed only if they are effective in terms of the defining or generating mechanism for languages in K.

The theory of context-free grammars (Ginsburg [3]) has in recent years been generalized in the following way (Van Leeuwen [10], also Greibach [4]). Let K be a family of languages.

Definition 2.2. A (context-free) *K-grammar* is a structure $G = \langle V, \Sigma, P, S \rangle$ where V is an alphabet, Σ the set of terminal symbols ($\Sigma \subseteq V$), S the start-symbol ($S \in V - \Sigma$), and P a finite set of production-rules of the form $A \to M$ with $A \in V - \Sigma$ and $M \in K (M \subseteq V^*)$. For $x, y \in V^*$ we write $x \Rightarrow y$ if and only if there are x_1, x_2, A, w, and $A \to M \in P$ such that $x = x_1 A x_2$, $y = x_1 w x_2$ and $w \in M$. We write $x \overset{*}{\Rightarrow} y$ if and only if $x = y$ or $y_1, \ldots, y_n$ exist with $x \Rightarrow y_1 \Rightarrow \cdots \Rightarrow y_n = y$.

The symbols in $V - \Sigma$ are called the variables of G, and they will later be important for an induction argument (as in Van Leeuwen [10]). As for ordinary context-free grammars we can represent derivations with K-grammars by means of derivation trees, except that this time there need not be a bound on the degree of branching at each node. The production rules in a K-grammar allow us to rewrite variables by any string from a possibly infinite language in K, whereas in traditional context-free grammar theory our choice of a right-hand side is always limited to a finite set.

Definition 2.3. A language L is called *algebraic over* K if and only if there is a K-grammar $G=\langle V, \Sigma, P, S\rangle$ such that $L=\{x\in\Sigma^* : S \overset{*}{\Rightarrow} x\}$. The collection of languages algebraic over K is called the *algebraic extension* of K, and will be denoted by K^∇.

We note that K-grammars can be reformulated as systems of equations over K, and the algebraic extension of K becomes the smallest family in which all such equations have a solution. Studying algebraic extensions (rather than just context-free languages) helps to discover deep structural properties and general proof-techniques (instead of ad-hoc combinatorial arguments) for the language-generating mechanism, and similar approaches have led to a better mathematical foundation in various branches of formal language theory (see e.g., Asveld [2] for a survey).

We note (see [10])

Theorem 2.4. *The family of context-free languages is the algebraic extension of the family of regular languages.*

We need two more results for algebraic extensions.

Lemma 2.5. *If K is (effectively) closed under finite substitution and under $\cap R$, then so is K^∇.*

Proof. The argument is rather similar to the known proofs that the family of context-free languages is closed under finite substitution and $\cap R$.

Observe that K must contain all finite languages. Let $L\in K^\nabla$ be generated by a K-grammar $G=\langle V, \Sigma, P, S\rangle$ and let τ be a finite substitution over Σ. One can generate $\tau(L)$ by the K-grammar $G'=\langle V', \Sigma, P', S\rangle$ with

$$V'=V\cup\{\bar\sigma : \sigma\in\Sigma\},$$
$$P'=\{A\to\gamma(M) : A\to M\in P\}\cup\{\bar\sigma\to\tau(\sigma) : \sigma\in\Sigma\},$$

where γ is the finite substitution defined by: $\gamma(A)=A$ for $A\in V-\Sigma$ and $\gamma(\sigma)=\bar\sigma$ for $\sigma\in\Sigma$. Hence $\tau(L)\in K^\nabla$.

Let $L\in K^\nabla$ be as before, and let R be a regular set. Let R be accepted by the finite automaton $\mathcal{M}=\langle Q, \Sigma, \delta, q_0, F\rangle$. One can generate $L\cap R$ by the K-grammar $G'=\langle V', \Sigma, P', S\rangle$ with

$$V'=\{S\}\cup\{[p, \alpha, q] : p, q\in Q \text{ and } \alpha\in V\cup\{\varepsilon\}\}\cup\Sigma,$$
$$P'=\{S\to[q_0, S, q] : q\in F\}$$
$$\cup\{[p, A, q]\to\gamma^A_{pq}(M) : p, q\in Q \text{ and } A\to M\in P\}$$
$$\cup\{[p, \sigma, q]\to R^\sigma_{pq} : p, q\in Q \text{ and } \sigma\in\Sigma\cup\{\varepsilon\}\},$$

where for all $A \in V-\Sigma$, $\sigma \in \Sigma \cup \{\varepsilon\}$ and $p, q \in Q$:

$$\gamma_{pq}^{A}(M) = \{[p, x_1, q_1][q_1, x_2, q_2] \cdots [q_{k-1}, x_k, q] : k \geq 1, x_i \in V \text{ for } 1 \leq i \leq k, \text{ and } x_1 \cdots x_k \in M\} (\text{plus } [p, \varepsilon, q] \text{ if } \varepsilon \text{ in } M)$$

(which must be in K whenever M is), and

$$R_{pq}^{\sigma} = \begin{cases} \{\sigma\} & \text{if } \delta(p, \sigma) = q, \\ \phi & \text{otherwise.} \end{cases}$$

It follows that $L \cap R \in K^{\nabla}$.

Lemma 2.6. *Let K be (effectively) closed under $\cap R$. The languages in K^{∇} have a uniformly decidably emptiness-problem if and only if the languages in K have a uniformly decidable emptiness-problem.*

Proof. The "only if" part is immediate, as $K \subseteq K^{\nabla}$.

Suppose that the emptiness-problem is uniformly decidable for languages in K. Consider some $L \in K^{\nabla}$, and let $G = \langle V, \Sigma, P, S \rangle$ be a K-grammar generating L. Assume for the moment that L is not empty, and that $x \in L$. Let D be a derivation tree for x.

Suppose that there is a level in D containing more than one occurrence of some variable A, and let $D_1, D_2, \ldots$ be the subtrees corresponding to each of these A's. If any of these subtrees are different, then one can change D and make another derivation of some terminal string (which, therefore, must be in L also) by demanding that all A's in the given level are expanded in the same way as (say) the first A in this level (Fig. 1).

It follows that we may assume that in all levels of D same variables are rewritten in identical manner. (Thus, in particular, we only need to know how one copy of each of the distinct variables occurring in a particular level is replaced.)

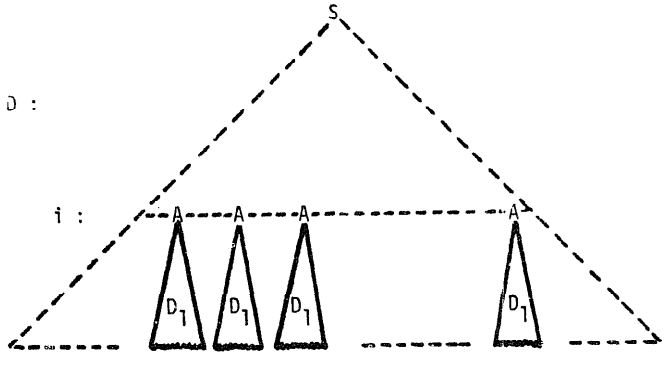

Fig. 1

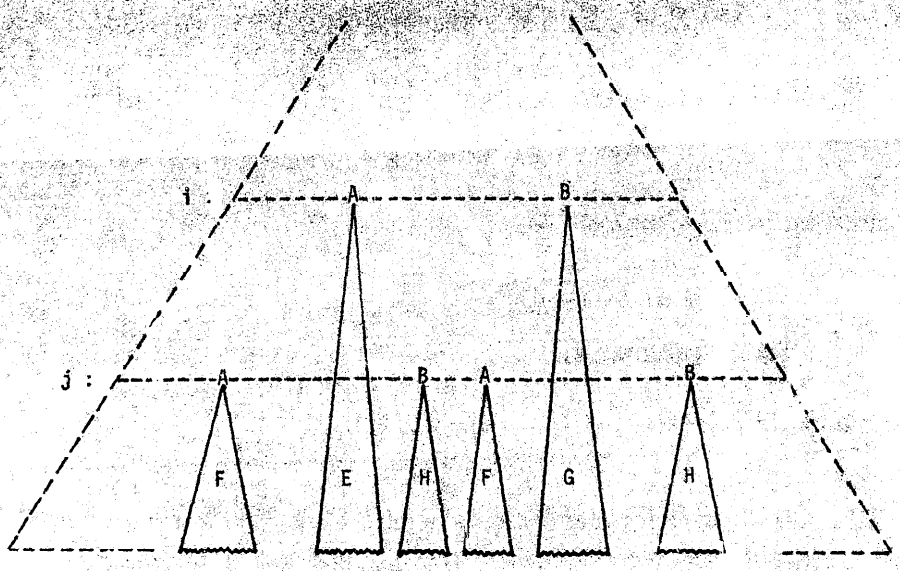

Fig. 2

Suppose next that there are two levels i and j in D where exactly the same sets of distinct variables occur ($i<j$). See Fig. 2. Then one can change D again, cut out the part between levels i and j, and obtain a new derivation of a terminal string by applying the replacements in level j directly to the occurrences of same variables in level i (thus producing a derivation tree for some word in L with fewer levels than D). See Fig. 3. Thus D may be altered again so that a same set of distinct variables never occurs at two different levels (and note that the property of equal replacement in a level after the first alteration is preserved in the process).

We conclude that L is not empty if and only if there is a derivation tree for some terminal string with no more than $2^{\#(V-\Sigma)}$ levels in which per level separate occurrences of same variables are rewritten in an identical manner. One can therefore test L for emptiness by trying to find such a special derivation tree.

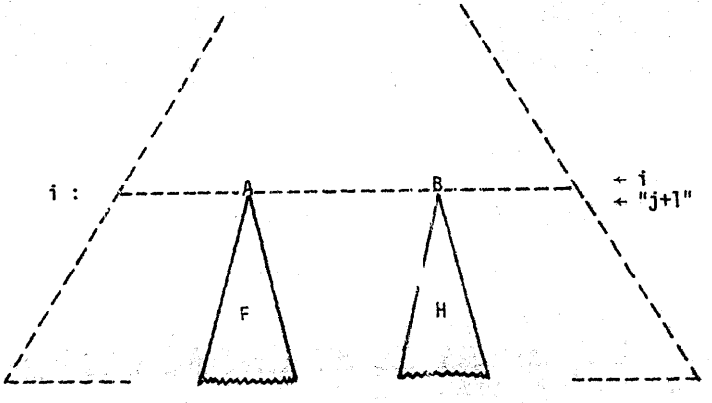

Fig. 3

Let's try to build the tree level after level with the following algorithm. Enumerate all possible sequences $V_0, \ldots, V_k$ with $k \leq 2^{\#(V-\Sigma)}$ (and $V_0 = \{S\}$ and $V_k = \emptyset$) of distinct sets of level-variables in some systematic way. For each sequence $V_0, \ldots, V_k$, suppose that we were able to construct a tree up to the ith level ($0 \leq i < k$) with a string $x \in (V_i \cup \Sigma)^*$ as its current frontier. If $x \in \Sigma^*$, then L is non-empty and we can finish. Otherwise, determine for each $A \in V_i$ occurring in x whether or not there is a production $A \to M \in P$ for which $M \cap (V_{i+1} \cup \Sigma)^*$ is non-empty. If there is an A for which no such production exists, then we cannot proceed with this derivation and have to start all over with the next sequence in the enumeration. Otherwise, rewrite the variables in x by a string over $V_{i+1} \cup \Sigma$ to get to a string in the $(i+1)$st level for which the same tests can be repeated, to see if we can eventually complete the derivation. L is empty if and only if the algorithm does not find a derivation of a terminal string this way.

Note that the algorithm always terminates in finitely many steps, and that all steps are effective. Observe in particular that always $M \cap (V_{i+1} \cup \Sigma)^* \in K$, and that one can effectively test it for emptiness by assumption. Once we know that it is not empty then we can find a $y \in M \cap (V_{i+1} \cup \Sigma)^*$ by testing $M \cap \{y\} (\in K)$ for emptiness while going through a lexicographic enumeration of all y's over $V_{i+1} \cap \Sigma$.

For an overview of the theory of well-partially-ordered sets we refer to Kruskal [8].

3. Effective ideals

There seems to be no accepted terminology for the set $\tilde{L}$. If we interpret $x \leq y$ as x being a "factor" of y, then it is reasonable to call $\tilde{L} = \{y \in \Sigma^* : \exists_{x \in L}\, x \leq y\}$ the *ideal* generated by L. We note that this notion of an ideal substantially differs from a similarly named concept for Σ^* introduced by Jullien [7]. We shall consider the problem of how to determine $\tilde{L}$ for an arbitrary language L.

If L contains just one element, $L = \{x_1 x_2 \cdots x_n\}$, then $\tilde{L} = \Sigma^* x_1 \Sigma^* x_2 \cdots x_n \Sigma^*$. In general, one can always effectively find $\tilde{L}$ in the regular case.

Lemma 3.1. *If L is regular, then $\tilde{L}$ can be effectively determined.*

Proof. Let R be an arbitrary regular language, and let R be accepted by the finite automaton $\mathcal{M} = \langle Q, \Sigma, \delta, q_0, F \rangle$. One can accept $\tilde{R}$ by the nondeterministic finite automaton $\mathcal{M}' = \langle Q, \Sigma, \delta', q_0, F \rangle$ with

$$\delta'(q, a) = \{q\} \cup \{\delta(q, a)\}$$

for all $q \in Q$ and $a \in \Sigma$.

One can prove Lemma 3.1. also by induction based on the regular expression for L (for a definition of regular expressions see [3, 6 or 9]). One only has to observe that $\widetilde{L \cup M} = \tilde{L} \cup \tilde{M}$, $\widetilde{(L \cdot M)} = \tilde{L} \cdot \tilde{M}$, and $\widetilde{(L^*)} = \Sigma^*$ for all languages L, M over Σ.

A string $x \in L$ is minimal (in L) if and only if there is no $y \in L$ with $y \leqslant x$ and $y \neq x$. The set of minimal elements in L is called the *basis* of L. It follows from the theory of well-partially-ordered sets that the basis of each language must be finite. L and $\tilde{L}$ have the same basis, and it easily follows that one can effectively determine $\tilde{L}$ if and only if one can effectively find the basis of L.

Definition 3.2. A *subbasis of* L is any set of minimal elements in L.

Lemma 3.3. *Let B be a subbasis of L. For $x \notin B$, $B \cup \{x\}$ is a subbasis of L if and only if x is a minimal string in $L - \tilde{B} = L \cap (\Sigma^* - \tilde{B})$.*

Proof. If $B \cup \{x\}$ is a subbasis, then x must be minimal in L. Since $x \notin B$ we conclude that $x \notin \tilde{B}$. It follows that x must be minimal in $L - \tilde{B}$.

Conversely, let x be minimal in $L - \tilde{B}$. Clearly $x \notin \tilde{B}$. If x were not minimal in L, then there would be a $y \in L \cap \tilde{B}$ with $y \neq x$ and $y \leqslant x$. As $y \in \tilde{B}$, we conclude that $x \in \tilde{B}$. Contradiction. Thus $B \cup \{x\}$ must be a subbasis of L.

Given a subbasis B, we can extend it to a larger subbasis (as long as it isn't a complete basis) by adjoining a minimal string from $L - \tilde{B} = L \cap (\Sigma^* - \tilde{B})$.

Lemma 3.4. *Any shortest string in L is minimal in L, and any such string must therefore be part of its basis.*

It follows that one can extend a given subbasis B by adjoining a shortest string of $L \cap (\Sigma^* - \tilde{B})$.

Theorem 3.5. *Let K be (effectively) closed under $\cap R$. One can effectively determine $\tilde{L}$ for each $L \in K$ if and only if the languages in K have a uniformly decidable emptiness-problem.*

Proof. Suppose that one can effectively determine $\tilde{L}$ for each $L \in K$. For arbitrary $L \in K$ one can test L for emptiness by determining $\tilde{L}$ and deciding whether or not $\tilde{L}$ (an effectively given regular set) is empty.

Conversely, suppose that the languages in K have a uniformly decidable emptiness-problem. Observe that once we determine that some $L \in K$ is not empty, then we can extract a shortest element from it by waiting for the first hit in the lexicographic enumeration of all strings x over Σ while each time testing $L \cap \{x\} (\in K)$ for emptiness.

Let $L \in K$. The method for effectively determining $\tilde{L}$ consists in effectively determining the finite basis B of L by the technique suggested by Lemmas 3.3 and 3.4. Start from an initially empty subbasis B, and extend it by each time adjoining a shortest element of $L \cap (\Sigma^* - \tilde{B})(\in K)$ until this set is empty and B has become a complete basis.

Algorithm 3.6.

Step 1. [set B empty]
$B \leftarrow \emptyset$

Step 2. [if B basis, then stop]
if $L \cap (\Sigma^* - \tilde{B}) = \emptyset$ *then go to* 6

Step 3. [extract a shortest string]
$x \leftarrow a$ shortest string in $L \cap (\Sigma^* - \tilde{B})$

Step 4. [extend B]
$B \leftarrow B \cup \{x\}$

Step 5. [and continue]
go to 2

Step 6. [stop and output result; $\tilde{B}$ is effective by Lemma 3.1]
output $\tilde{B}$

Step 7. HALT

All steps are effective, and the finite basis property guarantees that the algorithm terminates in finitely many steps. Note that $\tilde{L} = \tilde{B}$.

Corollary 3.7. *For context-free languages L one can effectively determine $\tilde{L}$.*

Proof. The family of context-free languages is (effectively) closed under $\cap R$, and the emptiness-problem for context-free languages is decidable (see Ginsburg [3]). Apply Theorem 3.5.

From 3.5 one may conclude that $\tilde{L}$ can be effectively determined for much wider language-families also, like for the family of indexed languages (Aho [1], see also Salomaa [9]).

From a mathematical viewpoint there is an elegant generalization of Corollary 3.7. We prove a weaker version first.

Lemma 3.8. *Let K be (effectively) closed under finite substitution and under $\cap R$. One can effectively determine $\tilde{L}$ for each $L \in K^\nabla$ if and only if one can do so for each $L \in K$.*

Proof. The "only if" part is easy, as $K \subseteq K^\nabla$.

Suppose one can effectively determine $\tilde{L}$ for each $L \in K$. It follows that languages in K have a uniformly decidable emptiness-problem. From Lemma 2.5

we conclude that K^{∇} is closed under $\cap R$, and from Lemma 2.6 we conclude that languages in K^{∇} have a uniformly decidable emptiness-problem. It follows from Theorem 3.5 that $\tilde{L}$ can be effectively determined for each $L \in K^{\nabla}$.

We can now show (i.e., with the help of Lemma 3.8 itself) that 3.8 is even true without the assumptions on K. Our main result is

Theorem 3.9. *For all K, one can effectively determine $\tilde{L}$ for each L in the algebraic extension of K if and only if one can do so for each L in K.*

Proof. The "only if" part is easy again, since $K \subseteq K^{\nabla}$.

Suppose that one can effectively determine $\tilde{L}$ for each $L \in K$. Let L be an arbitrary language in K^{∇}, and suppose that L is generated by a K-grammar $G = \langle V, \Sigma, P S\rangle$.

One can generate $\tilde{L}$ by the effectively determined REG-grammar $G' = \langle V, \Sigma, P', S\rangle$ where $P' = \{A \to \tilde{M}: A \to M \in P\}$ (which indeed is effectively determined by assumption on K). It follows that $\tilde{L}$ is an effectively given member of the algebraic extension of the family of regular languages. We conclude from Lemma 3.8 (with $K = \mathrm{REG}$, or Corollary 3.7) that the ideal generated by $\tilde{L}$, thus $\tilde{\tilde{L}} = \tilde{L}$ as a regular set itself, can be effectively determined.

4. Effective co-ideals

It appears to be harder to determine the sets $\underset{\sim}{L}$ for a given language L. In anticipation to Proposition 4.1. it is reasonable to call $\underset{\sim}{L} = \{x \in \Sigma^*: \exists_{y \in L} x \leqslant y\}$ the *co-ideal* generated by L. We shall examine the structure of co-ideals of algebraic languages, and establish a general result answering Haines' original question (Haines [5], see also Walker [12]) as a special instance.

The reason why $\underset{\sim}{L}$ is harder to determine than $\tilde{L}$ seems to be that to some extent we need information about its complement (and complementation is not a favorite closure-property for most traditional language-families). Haines [5] showed:

Proposition 4.1. *The complement of a co-ideal is an ideal (and conversely).*

Proof. For some co-ideal J, consider $\Sigma^* - J = I$. Let $y \in \tilde{I}$, and let $x \leqslant y$ for some $x \in I$. Suppose $y \in J$. It follows that $x \in \underset{\sim}{J} = J$, thus $x \notin I$. Contradiction. It follows that $\tilde{I} = I$ and I is an ideal.

The converse is shown by a similar argument.

Thus, if B is the (finite) basis for $\Sigma^* - \underset{\sim}{L}$ then we have $\underset{\sim}{L} = \Sigma^* - \tilde{B}$. However, there seems to be no algorithm to extract the basis of $\Sigma^* - \underset{\sim}{L}$ without knowing $\underset{\sim}{L}$

beforehand, and a different approach is needed. We shall develop such an approach in the main result of this section.

If L contains just one (non-ε) element, $L=\{x_1 \cdots x_n\}$, then we have $\underset{\sim}{L}=\{x_{i_1} \cdots x_{i_m}: 0 \leq m \leq n \text{ and } 1 \leq i_1 < \cdots < i_m \leq n\}$. In general, one can always effectively determine $\underset{\sim}{L}$ in the regular case by a direct construction.

Lemma 4.2. *If L is regular, then L can be effectively determined.*

Proof. Let R be an arbitrary regular set, and let R be accepted by the finite automaton $\mathcal{M}=\langle Q, \Sigma, \delta, q_0, F\rangle$. $\underset{\sim}{R}$ is accepted by the nondeterministic finite automaton $\mathcal{M}'=\langle Q, \Sigma, \delta', q_0, F\rangle$ with

$$\delta'(q, a)=\{\delta(q, a)\}$$

$$\delta'(q, \varepsilon)=\{\delta(q, a): a \in \Sigma\}$$

for all $q \in Q$ and $a \in \Sigma$.

One could prove Lemma 4.2. again by induction on the regular expression for L, by observing that for all languages L, $M \subseteq \Sigma^*$: $\underset{\sim}{(L \cup M)}=\underset{\sim}{L} \cup \underset{\sim}{M}$, $\underset{\sim}{(L \cdot M)}=\underset{\sim}{L} \cdot \underset{\sim}{M}$, and $\underset{\sim}{(L^*)}=(\underset{\sim}{L})^*$.

We shall demonstrate that effectively determining $\underset{\sim}{L}$ always translates under algebraic extension. The main construction is developed in the next theorem.

Theorem 4.3. *Let K be (effectively) closed under regular substitution and under $\cap R$. One can effectively determine $\underset{\sim}{L}$ for each $L \in K^{\nabla}$ if and only if one can do so for each $L \in K$.*

Proof. The "only if" part is immediate, as $K \subseteq K^{\nabla}$.

Suppose that one can effectively determine $\underset{\sim}{L}$ for each $L \in K$. It follows that languages in K have a uniformly decidable emptiness-problem. We shall prove that L can be effectively determined for each L in K^{∇} by induction on the number of variables in the K-grammar for L. (This technique was used before in Van Leeuwen [10].)

Let us do the induction-step first, and assume that $\underset{\sim}{L}$ can be effectively determined for each L definable by a K-grammar with $\leq n$ variables ($n \geq 1$). Suppose $L \in K^{\nabla}$ can be generated by some K-grammar $G=\langle V, \Sigma, P, S\rangle$ with $\#(V-\Sigma)=n+1$.

Choose an $A \in V-\Sigma (A \neq S)$, and define the K-grammar $G_A=\langle V, V-\{A\}, P_A, A\rangle$ with $P_A=\{A \to M: A \to M \in P\}$. G_A precisely contains all information needed to specify A in terms of the remaining variables and terminals of G. Let the language of G_A be L_A. By the induction hypothesis (for $n=1$) one can effectively determine $\underset{\sim}{L_A}$ (a regular set). Now consider the K-grammar $G'=\langle V-\{A\}, \Sigma, P', S\rangle$ with

$$P'=\{B \to \mathcal{S}^A_{\underset{\sim}{L_A}}(M): B \neq A \text{ and } B \to M \in P\}.$$

Let the generated language be L'. The effect of choosing P' is that one can immediately go from any decendent of A to an arbitrary substring of such a decendent, as follows from inspecting an arbitrary derivation tree. Clearly $\underline{L} = \underline{L'}$, but L' is defined by a K-grammar with only n variables. By induction we can effectively determine $\underline{L'}$, and thus $\underline{L}$, and the step is complete.

It follows from this argument that the induction-basis (the case $n=1$) is the crucial part of the construction.

Consider $L \in K^{\vee}$, and suppose that L is generated by a K-grammar $G = \langle \{S\} \cup \Sigma, \Sigma, P, S \rangle$ which has only one variable. Distinguish the following two possibilities:

Case 1. For each production $S \to M \in P$ and $x \in M$ there is at most one occurrence of S in x.

Case 2. There is a production $S \to M \in P$ and an $x \in M$ such that x contains at least two occurrences of S.

One can effectively determine which case applies by testing whether or not there is an $S \to M \in P$ for which $M \cap (\Sigma \cup S)^* \cdot S \cdot \Sigma^* \cdot S \cdot (\Sigma \cup S)^* (\in K)$ is not empty. The construction of $\underline{L}$ will differ for each case.

Construction for Case 1. Change P into an equivalent set of productions by splitting each rule $S \to M \in P$ in two rules

$$S \to M \cap \Sigma^* \quad \text{(type 1)},$$

$$S \to M \cap \Sigma^* S \Sigma^* \quad \text{(type 2)},$$

while discarding those with an empty righthandside. Let the productions of type 1 be

$$S \to M_1, \ldots, S \to M_k,$$

let the productions of type 2 be

$$S \to N_1, \ldots, S \to N_l,$$

and assume that $k > 0$ (otherwise $L = \emptyset$).

If $l = 0$ then it easily follows that $L = \bigcup_1^k M_i$, and L is obtained effectively by assumption on K.

If $l > 0$ then we first determine the following quotients of each $N_i (1 \leq i \leq l)$:

$$P_i = \{x : \exists_y xSy \in N_i\},$$

$$Q_i = \{y : \exists_x xSy \in N_i\}.$$

The closure-properties of K guarantee that P_i, $Q_i \in K$. To show this we first observe that K must contain all finite languages. Define a finite substitution τ_1 by $\tau_1(S) = \{S\}$ and $\tau_1(a) = \{a, \bar{a}\}$ for $a \in \Sigma$, and another finite substitution τ_2 by $\tau_2(S) = \{\varepsilon\}$, $\tau_2(a) = \{\varepsilon\}$ and $\tau_2(\bar{a}) = \{a\}$ for $a \in \Sigma$. It follows that

$$P_i = \tau_2(\tau_1(N_i) \cap \bar{\Sigma}^* \cdot S \cdot \Sigma^*),$$

$$Q_i = \tau_2(\tau_1(N_i) \cap \Sigma^* \cdot S \cdot \bar{\Sigma}^*).$$

Let

$$C = \left(\bigcup_1^l \underline{P}_i\right)^* \left(\bigcup_1^k \underline{M}_i\right) \left(\bigcup_1^l \underline{Q}_i\right)^*.$$

We claim that $\underline{L} = C$. Obviously $\underline{L} \subseteq C$.

To show the converse we shall consider some arbitrary element u of C, which without loss of generality can be written as

$$p_{i_1} p_{i_2} \cdots p_{i_s} m_i q_{j_s} \cdots q_{j_2} q_{j_1} (p_{i_\alpha} \in P_{i_\alpha},\ m_i \in M_i,\ q_{j_\beta} \in Q_{j_\beta})$$

with $s,\ t > 0$.

For each p_{i_α} there must be a p'_{i_α} such that $p_{i_\alpha} S p'_{i_\alpha} \in N_{i_\alpha}$, and for each q_{j_β} there must be a q'_{j_β} such that $q'_{j_\beta} S q_{j_\beta} \in N_{j_\beta}$.

Now consider the derivation

$$\begin{aligned}
S &\Rightarrow p_{i_1} S p'_{i_1} \\
&\Rightarrow p_{i_1} p_{i_2} S p'_{i_2} p'_{i_1} \\
&\ \ \vdots \\
&\Rightarrow p_{i_1} p_{i_2} \cdots p_{i_t} S p'_{i_t} \cdots p'_{i_2} p'_{i_1} \\
&\Rightarrow p_{i_1} p_{i_2} \cdots p_{i_t} q'_{j_1} S q_{j_1} p'_{i_t} \cdots p'_{i_2} p'_{i_1} \\
&\ \ \vdots \\
&\Rightarrow p_{i_1} p_{i_2} \cdots p_{i_t} q'_{j_1} \cdots q'_{j_s} S q_{j_s} \cdots q_{j_1} p'_{i_t} \cdots p'_{i_2} p'_{i_1} \\
&\Rightarrow p_{i_1} p_{i_2} \cdots p_{i_t} q'_{j_1} \cdots q'_{j_s} m_i q_{j_s} \cdots q_{j_1} p'_{i_t} \cdots p'_{i_2} p'_{i_1},
\end{aligned}$$

showing that the latter string is in L.

It follows that $p_{i_1} p_{i_2} \cdots p_{i_t} m_i q_{j_s} \cdots q_{j_2} q_{j_1} \in \underline{L}$ and we conclude that $u \in \underline{L}$, and thus $C \subseteq \underline{L}$.

Hence $\underline{L} = C$. Note that C is effectively constructable.

Construction for Case 2. Change P again into an equivalent set of productions, this time splitting each rule $S \to M$ into 2 rules

$$\begin{aligned}
&S \to M \cap \Sigma^* && \text{(type 1)}, \\
&S \to M \cap (\Sigma \cup S)^* S (\Sigma \cup S)^* && \text{(type 2)},
\end{aligned}$$

while again discarding those with empty righthandside. Let the productions of type 1 be

$$S \to M_1, \ldots, S \to M_k,$$

let the productions of type 2 be

$$S \to N_1, \ldots, S \to N_l,$$

and assume without loss of generality that $k > 0$ and $l > 0$.

For each i, $1 \leq i \leq l$, determine the following quotients of N_i:

$$P_i = \{x \in \Sigma^* : \exists_y xSy \in N_i\},$$

$$Q_i = \{x \in \Sigma^* : \exists_{y_1,y_2} y_1 Sxy_2 \in N_i\},$$

$$R_i = \{x \in \Sigma^* : \exists_y ySx \in N_i\}.$$

The closure properties of K imply that P_i, Q_i, $R_i \in K$ by means of a similar argument as in Case 1.

By assumption (for Case 2) we know that there is at least one non-empty set Q_i. Let $w = \hat{w}_1 S \hat{w}_2 S \hat{w}_3$ be an element of one such Q_i.

Let

$$C = \left(\left(\bigcup_i M_i \right) \cup \bigcup_i (P_i \cup Q_i \cup R_i) \right)^*.$$

We claim that $L = C$. Inspecting derivation-trees it is obvious that $L \subseteq C$. To show the converse, consider some arbitrary string u in C. Because C is a co-ideal it is no restriction to consider a string $v \in C$ with $u \leq v$ instead, which can be written as

$$m_{i_1} p_{i_1} q_{i_1} r_{i_1} m_{i_2} p_{i_2} q_{i_2} r_{i_2} \cdots m_{i_t} p_{i_t} q_{i_t} r_{i_t}$$

with $m_{i_\alpha} \in M_{i_\alpha}$, $p_{i_\alpha} \in P_{i_\alpha}$, $q_{i_\alpha} \in Q_{i_\alpha}$, and $r_{i_\alpha} \in R_{i_\alpha}$ (as each u in C will be "under" a string of this form).

For each p_{i_α} there must be a $\hat{p}_{i_\alpha}$ such that $p_{i_\alpha} S \hat{p}_{i_\alpha} \in N_{i_\alpha}$, for each q_{i_α} there must be $\hat{q}_{i_\alpha}$ and $\hat{q}'_{i_\alpha}$ such that $\hat{q}_{i_\alpha} S q_{i_\alpha} S \hat{q}'_{i_\alpha} \in N_{i_\alpha}$, and for each r_{i_α} there must be an $\hat{r}_{i_\alpha}$ such that $\hat{r}_{i_\alpha} S r_{i_\alpha} \in N_{i_\alpha}$ (where we used $\hat{}$ to denote that a string may still contain further occurrences of S). Recall the choice of the string $w = \hat{w}_1 S \hat{w}_2 S \hat{w}_3$.

Now consider the following derivation

$$S \Rightarrow \hat{w}_1 S \hat{w}_2 S \hat{w}_3$$

$$\Rightarrow \hat{w}_1 \hat{w}_1 S \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \hat{w}_3$$

$$\Rightarrow \hat{w}_1 \hat{w}_1 \hat{w}_1 S \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \hat{w}_3$$

$$\vdots$$

$$\Rightarrow \hat{w}_1^{4t-1} S \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \cdots \hat{w}_3 \hat{w}_2 S \hat{w}_3$$

[the intermediate string now shows at least $4t$ explicit S variables]

$$\Rightarrow \hat{w}_1^{4t-1} m_{i_1} \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \cdots \hat{w}_3 \hat{w}_2 S \hat{w}_3$$

$$\Rightarrow \hat{w}_1^{4t-1} m_{i_1} \hat{w}_2 p_{i_1} S \hat{p}_{i_1} \hat{w}_3 \hat{w}_2 S \hat{w}_3 \hat{w}_2 S \cdots \hat{w}_3 \hat{w}_2 S \hat{w}_3$$

$$\Rightarrow \hat{w}_1^{4t-1} m_{i_1} \hat{w}_2 p_{i_1} S \hat{p}_{i_1} \hat{w}_3 \hat{w}_2 \hat{q}_{i_1} S q_{i_1} S \hat{q}'_{i_1} \hat{w}_3 \hat{w}_2 \cdots \hat{w}_3 \hat{w}_2 S \hat{w}_3$$

$$\vdots$$

$$\Rightarrow \hat{w}_1^{4t-1} m_{i_1} \hat{w}_2 p_{i_1} S \hat{p}_{i_1} \hat{w}_3 \cdots m_{i_t} \hat{w}_3 \hat{w}_2 p_{i_t} S \hat{p}_{i_t} \hat{w}_3 \hat{w}_2 \hat{q}_{i_t} S q_{i_t} S \hat{q}'_{i_t} \hat{w}_3 \hat{w}_2 r_{i_t} S r_{i_t} \hat{w}_3$$

and continue to replace all occurrences of S which still remain by (say) $S \to m_{i_1}$ to obtain an element of L.

It follows that

$$m_{i_1} p_{i_1} q_{i_1} r_{i_1} \cdots m_{i_k} p_{i_k} q_{i_k} r_{i_k} \in \underset{\sim}{L},$$

and $v \in \underset{\sim}{L}$, thus $u \in \underset{\sim}{L}$. We conclude that $C \supseteq \underset{\sim}{L}$.

Hence $\underset{\sim}{L} = C$. Note that C is effectively constructable.

The result of Theorem 4.3 immediately enables us to solve the original problem of Haines [5].

Corollary 4.4. *For context-free languages L one can effectively determine $\underset{\sim}{L}$.*

Proof. In Lemma 4.2. we showed that $\underset{\sim}{L}$ can be effectively determined for each $L \in \mathrm{REG}$. In view of Theorem 2.4 we can now apply Theorem 4.3 with $K = \mathrm{REG}$ to obtain the desired conclusion.

Again Theorem 4.3 and Corollary 4.4 are not the best possible results one can obtain along these lines, and there is an elegant generalization. The interesting fact is that by a further argument (using Theorem 4.3 itself) one can prove the conclusion of Theorem 4.3 without the need for assuming any closure-properties of K.

Our main result is

Theorem 4.5. *For all K, one can effectively determine $\underset{\sim}{L}$ for each L in the algebraic extension of K if and only if one can do so for each L in K.*

Proof. The "only if" part is easy, as $K \subseteq K^{\nabla}$.

Suppose that one can effectively determine the regular set $\underset{\sim}{L}$ for each $L \in K$. Let L be an arbitrary language in K^{∇}, and suppose that L is generated by the K-grammar $G = \langle V, \Sigma, P, S \rangle$. Inspecting derivation-trees one can see that $\underset{\sim}{L}$ can be generated by the REG-grammar $G' = \langle V, \Sigma, P', S \rangle$ with $P' = \{A \to \underset{\sim}{M} : A \to M \in P\}$ which is effectively determined by assumption on K.

It follows that $\underset{\sim}{L}$ is effectively given, in fact, as a member of the algebraic extension of the family of regular languages.

It follows from Theorem 4.3 (with $K = \mathrm{REG}$, or Corollary 4.4) that the co-ideal generated by $\underset{\sim}{L}$, thus $\underset{\sim}{\underset{\sim}{L}} = \underset{\sim}{L}$ itself, can be effectively determined.

Both Theorems 3.9 and 4.5 show that, regardless of whether K has certain closure properties or not, the ability to effectively determine $\tilde{L}$ and $\underset{\sim}{L}$ for each $L \in K$ translates upwards undertaking algebraic extensions in all cases. This seems to be the most general form in which we can presently settle Haines' question.

It follows now that one can effectively determine $\bar{L}$ and $\underline{L}$ for languages L in much wider families than just the context-free languages. Denoting the family of context-free languages by CF, it follows for instance that one can effectively determine $\bar{L}$ and $\underline{L}$ for all languages L in the algebraic extension of $CF \cup \{a^n b^n c^n : n \geq 1\}$.

Acknowledgement

I thank the referee for various helpful comments.

References

[1] A.V. Aho, Indexed grammars—an extension of contextfree grammars, J. Assoc. Comput. Mach. 15 (1968) 647–671.
[2] P.R.J. Asveld, Rational, algebraic, and hyper-algebraic extensions of families of languages, TW Memo 90, Dept. of Applied Math. Twente University of Technology, Enschede (1975).
[3] S. Ginsburg, The Mathematical Theory of Contextfree Languages (McGraw-Hill, New York. 1966).
[4] S. Greibach, Full AFLs and nested iterated substitution, Information and Control 16 (1970) 7–35.
[5] L.H. Haines, On free monoids partially ordered by embedding, J. Combinatorial Theory 6 (1969) 94–98.
[6] J.E. Hopcroft and J.D. Ullman, Formal Languages and Their Relation to Automata (Addison-Wesley, Reading, MA, 1969).
[7] P. Jullien, (Analyse Combinatoire-) Sur un théorème d'extension dans la theory des mots, C. R. Acad. Sci. Paris Sér. A 266 (1968) 851–854.
[8] J.B. Kruskal, The theory of well-quasi-ordering: a frequently rediscovered concept, J. Combinatorial Theory 13 (A) (1974) 297–305.
[9] A. Salomaa, Formal Languages (Academic Press, New York, 1973).
[10] J. van Leeuwen, A generalization of Parikh's theorem in formal language theory, in: J. Loeckx, ed., Automata, Languages, and Programming (Springer, LCS 14, 1974) 17–26.
[11] J. van Leeuwen, A regularity condition for parallel rewriting systems, Sigact News 8:4 (1976) 24–27 (see also Sigact News 9:1 (1977) 9).
[12] A. Walker, Dynamically stable strings in developmental systems, self-repair, and the Chomsky hierarchy, Proc. 1974 Conf. on Biologically Motivated Automata Theory, McLean, VA (1974) 46–49.