Complexity Aspects of Iterated Rewriting – A Survey –

Peter R.J. Asveld

Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

We present an overview of results on the complexity of the membership problem for families of languages generated by several types of generalized grammars. In particular, we consider generalized grammars based on iterated context-independent rewriting, i.e., grammars consisting of a finite number of (non)deterministic substitutions, and on iterated context-dependent rewriting, i.e., grammars composed of a finite number of transductions. We give some conditions on the classes of these substitutions and transductions that guarantee the solvability of this membership problem within certain time and space bounds. As consequences we obtain additional closure properties of some time- and space-bounded complexity classes.

1. Introduction

The concept of iteration grammar has been introduced as a generalization of a particular kind of parallel rewriting system, viz. ETOL-system [18], in order to extend some ad hoc combinatorial arguments to more general, structural proof techniques. For the origins and the early history of iteration grammars we refer to [2] and the references mentioned there.

In essence an iteration grammar is an ETOL-system in which the finite substitutions or tables have been replaced by arbitrary substitutions. So each table in an iteration grammar contains for each symbol a countable rather than a finite number of productions. Instead of ordinary substitutions one can also use deterministic substitutions which yields a generalization of EDTOL-systems, the so-called deterministic iteration grammars [6,7]. A deterministic substitution differs from an ordinary or nondeterministic substitution in the way it is applied to a string: instead of making a possibly different choice for each occurrence of the same symbol in a string we make a single choice in advance and then substitute this choice consistently for each occurrence of the symbol. Another different way to generalize EDTOL- or ETOL-systems consists of replacing the tables by transductions which yields an abstract context-dependent grammar model [5]. This latter approach extends some earlier generalizations in [9,22,16,4,17]. The major part of the research on these general grammar models is concerned with extending known results from L-system theory in the hope to obtain

general, algebraic or structural arguments rather than the combinatorial proofs usually applied in the original finite case.

In this paper we survey some results that have been obtained in this way with respect to the complexity of the membership problem for these types of grammars, i.e., given such a grammar G and a string x over the terminal alphabet of G, how much time and space does it take to decide whether $x \in L(G)$? As usual in this area, the answer highly depends on the properties of the underlying set of (non)deterministic substitutions and transductions. Or, in other words, making some appropriately chosen assumptions on the set of substitutions or transductions enables us to extend results from L-system theory to these abstract grammars. As a spin-off it becomes clear on which properties of the finite languages the original proof in L-system theory actually depends.

The remaining part of this paper is organized as follows. Some notions of formal language and complexity theory are recalled in Section 2. In Section 3 we consider controlled (non)deterministic iteration grammars and their membership problem; Section 4 is devoted to the complexity of this problem. Context-dependent grammars, i.e., grammars based on transductions are described in Section 5, where we also provide some conditions that imply the decidability of the corresponding membership problem. In Section 6 attention is focused on the complexity of this membership problem. Sections 4 and 6 also provide some interesting closure properties of some complexity classes. Finally, Section 7 contains some open problems and concluding remarks.

2. Preliminaries

We assume the reader to be familiar with the rudiments of formal language theory and of automaton-based complexity theory. For all unexplained concepts and notation we refer to standard texts like [10,11,18]. For each set X, $\mathbf{P}(X)$ denotes the power set of X. The empty word is denoted by λ.

Let K be a family of languages and let V be an alphabet. A (*nondeterministic*) *K-substitution* over V or n*K-substitution* is a mapping $\tau: V \to K \cap \mathbf{P}(V^*)$ extended to words over V by $\tau(\lambda) = \{\lambda\}$, $\tau(xy) = \tau(x)\tau(y)$ for each $x, y \in V^*$, and to languages L over V by $\tau(L) = \bigcup \{\tau(x) \mid x \in L\}$. Similarly, a *deterministic K-substitution* over V or d*K-substitution* is also a mapping $\tau: V \to K \cap \mathbf{P}(V^*)$ but now it is extended to words by $\tau(x) = \{h(x) \mid h$ is a homomorphism such that $h(\alpha) \in \tau(\alpha)$ for each α in $V\}$, and to languages L over V by $\tau(L) = \bigcup \{\tau(x) \mid x \in L\}$. Notice that applying a dK-substitution τ implies that each occurrence of a symbol α in a string ought to be substituted by the very same word from $\tau(\alpha)$. A dK- or nK-substitution τ over V is called λ-*free* if $\lambda \notin \tau(\alpha)$ for each α in V.

An *Abstract Family of Languages* or *AFL* is a family of languages different from $\{\emptyset\}$, which is closed under union, concatenation, Kleene +, λ-free homomorphism, inverse homomorphism and intersection with regular languages.

Frequently, we will tacitly assume that families of languages are closed under isomorphism ("renaming of symbols").

Let $g: \mathbb{N} \to \mathbb{N}$ be a monotonic function, i.e., $m \leqslant n$ implies $g(m) \leqslant g(n)$. For each such g, DSPACE(g) [NSPACE(g)] is the family of languages accepted by [non]deterministic two-way multi-tape Turing machines that use no more than $g(n)$ tape cells on any storage tape during a computation on an input of length n. And DTIME(g) [NTIME(g)] denotes the family of languages accepted by [non]-deterministic two-way multi-tape Turing machines that finish their [accepting] computations within time $g(n)$ on all inputs of length n. We use the following well-known abbreviations:

$$\mathbf{PSPACE} = \bigcup \{\mathrm{DSPACE}(n^d) \mid d \geqslant 1\} = \bigcup \{\mathrm{NSPACE}(n^d) \mid d \geqslant 1\},$$

$$\mathbf{P} = \bigcup \{\mathrm{DTIME}(n^d) \mid d \geqslant 1\}, \qquad \mathbf{NP} = \bigcup \{\mathrm{NTIME}(n^d) \mid d \geqslant 1\}.$$

A [λ-*free*] *nondeterministic generalized sequential machine with accepting states* or *NGSM* [λ*NGSM*] $\tau = (Q, \Delta_1, \Delta_2, \delta, q_0, Q_F)$ consists of a set of states Q with initial state q_0 ($q_0 \in Q$), a set Q_F of final states ($Q_F \subseteq Q$), an input alphabet Δ_1, an output alphabet Δ_2, and a function δ from $Q \times \Delta_1$ into the finite subsets of $Q \times \Delta_2^*$ [$Q \times \Delta_2^+$]. The function δ is extended from $Q \times \Delta_1^*$ into the finite subsets of $Q \times \Delta_2^*$ by

(i) $\delta(q, \lambda) = \{(q, \lambda)\}$,

(ii) $\delta(q, x\alpha) = \{(q', y) \mid y = y_1 y_2$ and for some $p \in Q$, $(p, y_1) \in \delta(q, x)$, and $(q', y_2) \in \delta(p, \alpha)\}$, where $q \in Q$, $\alpha \in \Delta_1$, $x \in \Delta_1^*$.

Each [λ-free] NGSM τ induces a transduction $\tau: \mathbf{P}(\Delta_1^*) \to \mathbf{P}(\Delta_2^*)$, called [$\lambda$-*free*] *NGSM mapping*, defined by $\tau(x) = \{y \mid (q, y) \in \delta(q_0, x)$ for some $q \in Q_F\}$ for each x in Δ_1^*, and $\tau(L) = \bigcup \{\tau(x) \mid x \in L\}$ for each language L over Δ_1.

A NGSM [λNGSM] τ is called *deterministic* or *DGSM* [λ*DGSM*] if δ is a function from $Q \times \Delta_1$ into $Q \times \Delta_2^*$ [$Q \times \Delta_2^+$]. By NGSM [λNGSM] we also denote the family of [λ-free] NGSM mappings, and similarly we use DGSM [λDGSM] in the deterministic case.

3. Iterated Context-Independent Rewriting

Iteration grammars have already been discussed in an informal way in Section 1. Now we recall the formal definition together with the controlled variant which clearly generalizes the concepts of controlled EDTOL- and ETOL-system studied in [1,2,6,7,8,13,14].

Definition 3.1. Let Γ and K be language families. A [*non*]*deterministic* K*-iteration grammar* or dK*-iteration* [nK*-iteration*] grammar

$G = (V,\Sigma,U,S)$ consists of an alphabet V, a terminal alphabet Σ $(\Sigma \subseteq V)$, an initial symbol S $(S \in V)$, and a finite set U of [non]deterministic K-substitutions over V. The language $L(G)$ generated by G is defined by $L(G) = U^*(S) \cap \Sigma^*$. A *$\Gamma$-controlled dK-iteration* [*nK-iteration*] *grammar* $(G;C) = (V,\Sigma,U,S,C)$ consists of a dK-iteration [nK-iteration] grammar (V,Σ,U,S) together with a *control language* $C \subseteq U^*$ with $C \in \Gamma$. The language generated by $(G;C)$ is defined by

$$L(G;C) = C(S) \cap \Sigma^* = (\bigcup \{\tau_p(...(\tau_1(S))...) | \tau_1...\tau_p \in C\}) \cap \Sigma^*.$$

The family of languages generated by dK-iteration [nK-iteration] grammars is denoted by $\eta(K)$ [$H(K)$]. And $\eta(\Gamma,K)$ [$H(\Gamma,K)$] denotes the family of languages generated by Γ-controlled dK-iteration [nK-iteration] grammars. A (Γ-controlled) iteration grammar is called *λ-free* when each of its substitutions is λ-free. □

In derivations of controlled (λ-free) iteration grammars we can use long control words to derive relatively short terminal strings. In this way we are able to simulate an erasing homomorphism. Basicly, this is the core in proving the following characterization of RE, the family of recursively enumerable languages.

Proposition 3.2. [1,2,6]. *If K and Γ are language families such that* $\{L \mid \text{card}(L) = 1\} \subseteq K \subseteq \text{RE}$, *and* $\{h(L) \mid L \in \Gamma$; *h is an arbitrary homomorphism* $\} = \text{RE}$, *then* $\eta(\Gamma,K) = H(\Gamma,K) = \text{RE}$. □

Thus in order to obtain recursive languages (of which we want to determine the complexity) at all, we ought to restrict the families K and Γ in some way. The conditions in the following lemma provide a first step to such a restriction.

Lemma 3.3. [3]. *Let K be a family which contains all alphabets.*
(1) *Let Γ be a family closed under finite substitution and intersection with regular languages, and let $(G;C) = (V,\Sigma,U,S,C)$ be a λ-free Γ-controlled dK-iteration* [*nK-iteration*] *grammar. Then there exists a λ-free Γ-controlled dK-iteration* [*nK-iteration*] *grammar $(G';C') = (V',\Sigma,U',S,C')$ such that $L(G';C') = L(G;C)$, and for each x in $L(G';C')$ there is a control word $\tau_1...\tau_p$ in C' such that $x \in \tau_p...\tau_1(S)$ and $p \leqslant 2|x|$.*
(2) *For each λ-free dK-iteration* [*nK-iteration*] *grammar $G = (V,\Sigma,U,S)$ there exists a λ-free dK-iteration* [*nK-iteration*] *grammar $G' = (V',\Sigma,U',S)$ such that $L(G') = L(G)$, and for each x in $L(G')$ there is a string $\tau_1...\tau_p$ over U' such that $x \in \tau_p...\tau_1(S)$ and $p \leqslant 2|x|$.* □

The proof of this lemma can be found in [3] and will not be repeated here; it extends an earlier result on controlled ETOL-systems from [8]. In Section 5 we meet a similar situation and a corresponding lemma of which the proof will be sketched.

Proposition 3.4. [5]. *Let Γ and K satisfy the assumptions of Lemma 3.3. If both Γ and K are λ-free subfamilies of the family of recursive languages, then each language in $H(\Gamma,K)$, $H(K)$, $\eta(\Gamma,K)$ and in*

$\eta(K)$ *is recursive.*

```
read x
if x ∉ Σ+ then reject else
        for all u in C' with |u| ≤ 2|x| do
                if x ∈ u(S) then accept fi
        od;
        reject
fi.
```

Figure 1.

Proof: Given a λ-free Γ-controlled [non]deterministic K-iteration grammar $(G;C)$, we first apply Lemma 3.3. Then the algorithm in Figure 1 determines whether a word x belongs to $L(G';C')$. Since after the execution of an **accept**- or **reject**-statement the algorithm is supposed to halt, termination is guaranteed for each input x. Hence $L(G';C')$ is recursive. In the uncontrolled case we replace "**for all** u in C'" by "**for all** u in U'^*" in Figure 1. □

Other conditions that guarantee the decidability of the membership problem for (controlled) λ-free iteration grammars can be found in [1,2].

4. Complexity Aspects of Iterated Context-Independent Rewriting

With respect to the complexity of the membership problem for (controlled) deterministic iteration grammars the following result is of principal interest. Its proof is too long to be given here; it consists of a straightforward generalization of the argument that EDTOL is included in NSPACE $(\log n)$ [12]; cf. [3] for details.

Let 1-NSPACE (g) be the family of languages accepted within space bound $g: \mathbb{N} \to \mathbb{N}$ by nondeterministic multi-tape Turing machines with a one-way read-only input tape.

Theorem 4.1. [3]. *Let* $g(n) \geq \log n$ *for all* $n \in \mathbb{N}$, *let* Γ *and* K *be families satisfying the assumptions of Lemma 3.3, and let* Γ *be closed under reversal.*
(1) *If* Γ *and* K *are included in* 1-NSPACE (g), *with* $g(2n) \leq c.g(n)$ *for some constant* c *and for all* $n \in \mathbb{N}$, *then* $\eta(\Gamma, K) \subseteq$ NSPACE (g).
(2) *If* $K \subseteq$ 1-NSPACE (g), *then* $\eta(K) \subseteq$ NSPACE (g). □

Since for functions $g: \mathbb{N} \to \mathbb{N}$ with $g(n) \geq n$ for all $n \in \mathbb{N}$, the family 1-NSPACE (g) equals NSPACE (g), this implies immediately

Theorem 4.2. [3]. *Let* $g(n) \geq n$ *for all* $n \in \mathbb{N}$, *and let* Γ *and* K *be families which satisfy the assumptions of Lemma 3.3.*
(1) *If* $g(2n) \leq c.g(n)$ *for some constant* c *and for all* $n \in \mathbb{N}$, *and if both* Γ *and* K *are included in* NSPACE (g), *then* $\eta(\Gamma, K) \subseteq$ NSPACE (g).
(2) *If* $K \subseteq$ NSPACE (g), *then* $\eta(K) \subseteq$ NSPACE (g). □

A similar results holds for $H(\Gamma, K)$ and $H(K)$; it has been mentioned implicitly in [20] and formulated in [3] in the following way.

Theorem 4.3. [20,3]. *Let $g(n) \geqslant n$ for all $n \in \mathbb{N}$, and let Γ and K be families which satisfy the conditions of Lemma 3.3.*
(1) *If $g(2n) \leqslant c.g(n)$ for some constant c and for all $n \in \mathbb{N}$, and if both Γ and K are included in* NSPACE(g), *then* $H(\Gamma, K) \subseteq$ NSPACE(g).
(2) $K \subseteq$ NSPACE(g) *implies* $H(K) \subseteq$ NSPACE(g). □

An inclusion analogous to 4.2(2) and 4.3(2) for DSPACE(g) originates from [20]; the proof is based on a divide-and-conquer technique to determine a derivation. The obvious extension to controlled iteration grammars is from [3].

Theorem 4.4. [20,3]. *Let $g(n) \geqslant n \log n$ for all $n \in \mathbb{N}$, and let Γ and K be families which satisfy the assumptions of Lemma 3.3.*
(1) *If $g(2n) \leqslant c.g(n)$ for some constant c and for all $n \in \mathbb{N}$, and if both Γ and K are included in* DSPACE(g), *then* $H(\Gamma, K) \subseteq$ DSPACE(g).
(2) *If* $K \subseteq$ DSPACE(g), *then* $H(K) \subseteq$ DSPACE(g). □

Van Leeuwen's proof of 4.4 can also applied to (controlled) deterministic iteration grammars; as observed in [3] the bound $g(n) \geqslant n \log n$ can in that case be replaced by $g(n) \geqslant n$ for all $n \in \mathbb{N}$.

Theorem 4.5. [3]. *Let $g(n) \geqslant n$ for all $n \in \mathbb{N}$, and let Γ and K be families which satisfy the conditions of Lemma 3.3.*
(1) *If $g(2n) \leqslant c.g(n)$ for some constant c and for all $n \in \mathbb{N}$, and if both Γ and K are included in* DSPACE(g), *then* $\eta(\Gamma, K) \subseteq$ DSPACE(g).
(2) $K \subseteq$ DSPACE(g) *implies* $\eta(K) \subseteq$ DSPACE(g). □

Taking K equal to DSPACE(g) or NSPACE(g) in 4.2(2) - 4.5(2) yields some interesting closure properties for these complexity classes. A language family K is closed under *iterated λ-free [non]deterministic substitution* if for each K-language L over V and each finite set U of λ-free [non]deterministic K-substitutions over V the language $U^*(L)$ belongs to K.

Theorem 4.6. [3]. *Let for all $n \in \mathbb{N}$, $g(n) \geqslant n$ and $g(2n) \leqslant c.g(n)$ for some constant c. Then* NSPACE(g) *and* DSPACE(g) *are AFL's closed under intersection and iterated λ-free deterministic substitution. Moreover,* NSPACE(g) *is closed under iterated λ-free nondeterministic substitution; this also applies to* DSPACE(g) *provided that $g(n) \geqslant n \log n$ for all $n \in \mathbb{N}$.* □

Corollary 4.7. [3]. *The following language families are AFL's closed under intersection and under iterated λ-free deterministic substitution:*

(1) **PSPACE**,

(2) NSPACE(n), *the family of nondeterministic context-sensitive languages,*

(3) DSPACE(n), *the family of deterministic context-sensitive languages,*

(4) NSPACE(n^2), *the family of two-way nondeterministic nonerasing stack automaton languages,*

(5) DSPACE($n \log n$), *the family of two-way deterministic nonerasing stack automaton languages.*

Moreover, the families under (1), (2), (4) *and* (5) *are also closed under iterated* λ*-free nondeterministic substitution.* □

We call a language family K closed under *controlled iterated* λ*-free* [*non*]*deterministic substitution* if $\eta(K,K) \subseteq K$ [$H(K,K) \subseteq K$]. To obtain in a similar way closure under controlled iterated λ-free substitution for these complexity classes fails; cf. Proposition 4.8, the proof of which is based on Proposition 3.2 and the fact that the Dyck set is in DSPACE ($\log n$).

A language family K is closed under *removal of right endmarker* if for each language Lc in K where $L \subseteq \Sigma^*$ for some alphabet Σ with $c \notin \Sigma$, the language L is in K too.

Proposition 4.8. [3]. *Let K be a family closed under removal of right endmarker. If* DSPACE ($\log n$) $\subseteq K \subset$ RE, *then K is not closed under controlled iterated λ-free (non)deterministic substitution. In particular this applies to each complexity class which includes* DSPACE ($\log n$). □

For the time complexity classes **P** and **NP** we have the familiar situation; viz. **NP** has "strong" closure properties, whereas **P** shares these properties if and only if **P** = **NP**.

Theorem 4.9. [3].
(1) **NP** *is an AFL closed under intersection and iterated* λ*-free (non)-deterministic substitution.*
(2) *Let C be either* DSPACE ($\log n$), NSPACE ($\log n$), *or* **P**. *Then the following propositions are equivalent.*

(a) $C = \mathbf{P} = \mathbf{NP}$.

(b) *C is closed under iterated λ-free nondeterministic substitution.*

(c) *C is closed under iterated λ-free deterministic substitution.*

(d) *C is closed under λ-free homomorphism.* □

5. Iterated Context-Dependent Rewriting

Central in our approach (cf. [5]) to introduce an abstract context-dependent grammar model is the notion of transduction.

Definition 5.1. Let V be an alphabet. A *transduction* τ over V is a function $\tau: V^* \to \mathbf{P}(V^*)$ extended to languages by $\tau: \mathbf{P}(V^*) \to \mathbf{P}(V^*)$ with $\tau(L) = \bigcup \{\tau(x) \mid x \in L\}$ for each language L over V.
Let f be an n-ary operation on languages. A family **T** of transductions is *closed* under (composition to the left with) f, if for all **T**-transductions $\tau_1, \ldots, \tau_n$ over some alphabet V, there exists a **T**-transduction τ over V such that $\tau(x) = f(\tau_1(x), \ldots, \tau_n(x))$ for all x

in V^*. □

In many proofs one wants to construct a new grammar G_N from an old one G_O by attaching a finite amount of information to the symbols of G_O. Then the transductions in G_N over this extended alphabet will be defined in terms of the old transductions of G_O using closure under isomorphism. Finally, we strip this additional information by applying an isomorphism in order to obtain words over the original alphabet. Therefore we make the following basic assumption.

Assumption 5.2. Henceforth **T** is a family of transductions that
(1) is closed under (composition to the left with) isomorphisms; cf. Definition 5.1,
(2) is closed under composition to the right with isomorphisms, i.e., for each **T**-transduction τ_1 over V_1 and each isomorphism $i: V \to V_1$ there exists a **T**-transduction τ over V such that $\tau(x) = \tau_1(i(x))$ for each x in V^*, and
(3) contains for each V the identity mapping over V. □

From 5.2 it follows that **T** also contains all isomorphisms. We are now ready for the main formal definition.

Definition 5.3. Let **T** be a family of transductions. A **T**-*grammar* $G = (V,\Sigma,U,S)$ consists of an alphabet V, a terminal alphabet Σ $(\Sigma \subseteq V)$, an initial symbol S $(S \in V)$, and a finite set U of T-transductions over V. The language $L(G)$ generated by G is defined by

$$L(G) = U^*(S) \cap \Sigma^* =$$

$$= (\cup \{\tau_p(...(\tau_1(S))...) | p \geqslant 0;\ \tau_i \in U,\ 1 \leqslant i \leqslant p\}) \cap \Sigma^*.$$

Let Γ be a family of languages. A Γ-*controlled* **T**-*grammar* $(G;C) = (V,\Sigma,U,S,C)$ is a **T**-grammar $(V,\Sigma,U,S,)$ provided with a control language $C \subseteq U^*$ from Γ. The language $L(G;C)$ generated by $(G;C)$ is defined by

$$L(G;C) = C(S) \cap \Sigma^* = (\cup \{\tau_p(...(\tau_1(S))...) | \tau_1 \cdots \tau_p \in C\}) \cap \Sigma^*.$$

L(**T**) [**L**(**T**;Γ), respectively] is the family of languages generated by [Γ-controlled] **T**-grammars, and **L**(**T**;m) [respectively **L**(**T**;Γ;m)] is the subfamily of languages generated by [Γ-controlled] T-grammars that possess at most m $(m \geqslant 1)$ T-transductions. □

Example 5.4. (1) Let HOM and FINSUB be the families of all homomorphisms and of all finite substitutions, respectively. Then L(HOM) = EDTOL and L(FINSUB) = ETOL. For Γ-controlled variations, see e.g. [1,2,6,8,13,14].
(2) Let dK-SUB [nK-SUB] denote the family of all [non]deterministic K-substitutions. Then L(dK-SUB) = $\eta(K)$ [respectively, L(nK-SUB) = $H(K)$] i.e., the family of languages generated by [non]deterministic K-iteration grammars.
(3) The language families **L**(**T**;Γ) and **L**(**T**) with **T** equal to λNGSM,

λDGSM, NGSM, and DGSM have been investigated in [9,22,16,4,17]; see [17] in particular, where e.g. the family of context-free languages is characterized by $\mathbf{L}(\mathbf{T})$ for a family $\mathbf{T}$ of restricted NGSM mappings. □

All the examples in 5.4 are transductions in the sense of Definition 5.1, and they all satisfy Assumption 5.2.

For these context-dependent abstract grammars we also need a decidable membership problem; cf. Proposition 5.7. Therefore we restrict our attention to so-called locally context-independent transductions [20,5] which enables us to establish an analogue of Lemma 3.3, viz. Lemma 5.6.

Definition 5.5. A transduction τ over some alphabet V is called *locally context-independent* if

(1) τ is *monotonic*, i.e., for each y in V^*, $y \in \tau(x)$ implies $|y| \geqslant |x|$.

(2) τ is context-independent in length-preserving applications, i.e., for all x_i, y_i in V^* with $|x_i| = |y_i|$ $(i = 1,2,3)$, $y_1y_2y_3 \in \tau(x_1x_2x_3)$ implies $y_1y_3y_2 \in \tau(x_1x_3x_2)$. □

Lemma 5.6. [5]. *Let* $\mathbf{T}$ *be a family of locally context-independent transductions, and let* $\mathbf{T}$ *contain for all alphabets* V *the length-preserving finite substitutions*

$$\tau(\alpha) = W \qquad \alpha \text{ in } V, \quad W \subseteq V.$$

(1) *Let* Γ *be a family closed under finite substitutions and under intersection with regular languages, and let* $(G;C) = (V,\Sigma,U,S,C)$ *be a* Γ*-controlled* $\mathbf{T}$*-grammar. Then we can effectively construct a* Γ*-controlled* $\mathbf{T}$*-grammar* $(G';C') = (V,\Sigma,U',S,C')$ *such that* $L(G';C') = L(G;C)$, *and for each* x *in* $L(G';C')$, *there is a control word* $\tau_1 \ldots \tau_p$ *in* C' *such that* $x \in \tau_p \ldots \tau_1(S)$ *and* $p \leqslant 2|x|$.
(2) *For each* $\mathbf{T}$*-grammar* $G = (V,\Sigma,U,S)$, *we can effectively construct a* $\mathbf{T}$*-grammar* $G' = (V,\Sigma,U',S)$ *such that* $L(G') = L(G)$, *and for each* x *in* $L(G')$, *there exists a word* $\tau_1 \ldots \tau_p$ *in* U'^* *such that* $x \in \tau_p \ldots \tau_1(S)$, *and* $p \leqslant 2|x|$.

Proof: (1) We add new control words to C such that the corresponding derivations possess the property that each length-preserving step in such a derivation is immediately followed by a length-increasing step.

If $V = \{\alpha_1, \ldots, \alpha_k\}$ for some $k \geqslant 1$, then we define $U' = U \cup \{[\tau,q] \mid \tau \in U, q \in Q\}$ with $Q = \{<X_1, \ldots, X_k> \mid X_i \subseteq V,\ 1 \leqslant i \leqslant k\}$, and $C' = \sigma(C)$ where $\sigma = (Q,U,U',\delta,q_0,Q_F)$ is an NGSM with $q_0 = <\{\alpha_1\}, \ldots, \{\alpha_k\}>$, $Q_F = \{q_0\}$, while δ is defined by

$$\delta(<X_1, \ldots, X_k>, \tau) = \{(q_0,\tau) \mid <X_1, \ldots, X_k> = q_0\} \cup$$

$$\cup \{(<\tau(X_1) \cap V, \ldots, \tau(X_k) \cap V>, \lambda), (q_0, [\tau, <X_1, \ldots, X_k>])\}$$

(Notice that $C \subseteq C'$).

Next we indicate the way in which the new additional control words are obtained by means of σ from C, together with the effect of

these new control words. Consider an arbitrary derivation D according to $(G;C)$. At each step in D, determined by the application of some T-transduction τ, one of the following three cases applies (cf. the definition of δ):

Case (a): This application of τ is length-increasing.
The corresponding transition in σ is the identity transition: $(q_0,\tau)\in \delta(q_0,\tau)$. This case does not give rise to the addition of new control words.

Case (b): This application of τ is length-preserving and the next step in D will also be length-preserving.
The corresponding occurrence of τ in the control word is erased, and the length-preserving context-independent effect (cf. Definition 5.5) of τ is stored by means of changing the state of σ from $<X_1,...,X_k>$ to $<\tau(X_1)\cap V,...,\tau(X_k)\cap V>$.

Case (c): This application of τ is length-preserving but either the next step in D will be length-increasing, or this application of τ is the last step in D.
In the old control word we replace the corresponding occurrence of τ by $[\tau,<X_1,...,X_k>]$ where $<X_1,...,X_k>$ is the current state of σ in which the ultimate length-preserving effect of a consecutive sequence of erased transductions (cf. Case (b)) has been stored. This new transduction $[\tau,<X_1,...,X_k>]$ is a length-preserving finite substitution defined by

$$[\tau,<X_1,...,X_k>](\alpha_i)=\tau(X_i)\cap V \qquad \text{for each } i\ (1\leqslant i\leqslant k).$$

(2) Define $U'=U\cup\{\tau_u \mid u\in U^+\}$ with for each u in U^+, τ_u is the length-preserving substitution defined by

$$\tau_u(\alpha)=u(\alpha)\cap V \qquad \alpha \text{ in } V.$$

Then U' is finite, because there are only a finite number of length-preserving substitutions over V. □

The proof of the following result is almost identical to the one of Proposition 3.4.

Proposition 5.7. [5]. *Let* Γ *and* T *satisfy the assumptions of Lemma 5.6. If* Γ *is a subfamily of the family of recursive languages and if* T *is a subfamily of the family of recursive transductions, then each language in* $\mathbf{L}(\mathbf{T};\Gamma)$ *and in* $\mathbf{L}(\mathbf{T})$ *is recursive.* □

6. Complexity Aspects of Iterated Context-Dependent Rewriting

In this section we determine upper bounds for the space and time complexity of languages generated by (controlled) T-grammars; viz. Theorems 6.2, 6.5 and 6.8.

Throughout this section "function" means a monotone increasing function g over the natural numbers satisfying $g(n)\geqslant n$ for each

$n \in \mathbb{N}$.

Definition 6.1. Let for each function $g: \mathbb{N} \to \mathbb{N}$, DSPACETR($g$) [NSPACETR($g$), respectively] be the family of those transductions τ that satisfy
(1) τ is locally context-independent, and
(2) there exists a [non]deterministic algorithm that can decide a query "$y \in \tau(x)$?" for each x and y within space $g(|y|)$. □

Theorem 6.2. [5]. *Let g be a function.*
(1) *If* $\mathbf{T} \subseteq$ NSPACETR(g), *then* $\mathbf{L}(\mathbf{T}) \subseteq$ NSPACE(g).
(2) $\mathbf{L}$(NSPACETR(g)) = NSPACE(g).
(3) *Let Γ be a family closed under finite substitution and under intersection with regular languages. If* $\Gamma \subseteq$ NSPACE(g), $\mathbf{T} \subseteq$ NSPACETR(g), *and* $g(2n) \leqslant c.g(n)$ *for some constant c, then* $\mathbf{L}(\mathbf{T};\Gamma) \subseteq$ NSPACE(g).

Proof: (1) Consider the algorithm in Figure 2; remove the assignments in which the variable control is involved, and replace the last statement by **accept**.

```
read x;
control := λ;
if x ∉ Σ+ then reject else
      y := S;
      while y ≠ x and |y| ⩽ |x| do
            guess τ ∈ U;
            guess z ∈ V+ with |y| ⩽ |z| ⩽ |x|;
            if z ∈ τ(y) then control := control.τ;
                        y := z
                        else reject
            fi
      od
fi;
if control ∈ C then accept else reject fi.
```

Figure 2.

Then each step in this modified algorithm requires at most linear space, except the test "$z \in \tau(y)$" for which we need $g(|z|) \leqslant g(|x|)$ space. Thus for $|x| = n$, the total amount of space is $O(n + g(n)) = O(g(n))$.

(2) The inclusion $\mathbf{L}$(NSPACETR(g)) $\subseteq$ NSPACE(g) follows immediately from (1) by taking $\mathbf{T}$ = NSPACETR(g).

Conversely, let $L_0 \subseteq \Sigma^*$ be a language in NSPACE(g). Define $G = (V, \Sigma, \{\tau\}, S)$ with $V = \Sigma \cup \{S\}$, and τ is defined by

$$\tau(S) = L_0$$
$$\tau(w) = \{w\} \qquad \text{for each } w \text{ in } \Sigma^*.$$

Then $\tau \in$ NSPACETR(g), G is an NSPACETR(g)-grammar, $L(G) =$

L_0, and hence NSPACE $(g) \subseteq$ L (NSPACETR (g)).

(3) Consider the algorithm of Figure 2. By Lemma 5.6 the last statement requires space $O(g(2n))$ which is $O(g(n))$ due to the assumption on g. So the total space needed to execute the algorithm is $O(n+g(n))+O(g(n)) = O(g(n))$; cf. the proof of (1). □

Corollary 6.3. [5]. L (NSPACETR (n)) = NSPACE (n). □

NSPACE (n) or, equivalently, the family of λ-free context-sensitive languages can be characterized by much simpler transductions than those used in 6.3; in 6.4 we combine results from [9,22,16,4,17] together with some simple properties.

Theorem 6.4.

L (λNGSM;REG) = L (λNGSM) = L (λNGSM ;1) =
L (λDGSM;REG) = L (λDGSM) = L (λDGSM ;2) = NSPACE (n). □

Although this solves partially an open problem from [22], viz. L (λDGSM ;2) = NSPACE (n), the precise nature of L (λDGSM ;1) and an analogous characterization of DSPACE (n) are still unknown. However, it is easy to show that L (λDGSM ;1) $\subseteq$ DSPACE (n).

For a deterministic counterpart of Theorem 6.2 we can generalize the proof of Theorem 5.2 in [21] straightforwardly. The details are left to the reader.

Theorem 6.5. *Let g be a function with $g(n) \geqslant n \log n$ for each $n \in \mathbb{N}$, and there exists a constant c such that $g(2n) \leqslant c.g(n)$ for each $n \in \mathbb{N}$. Let Γ be a family of languages closed under finite substitution and under intersection with regular languages.*
(1) *If* T $\subseteq$ DSPACETR (g), *then* L (T) $\subseteq$ DSPACE (g).
(2) L (DSPACETR (g)) = DSPACE (g).
(3) *If* $\Gamma \subseteq$ DSPACE (g) *and* T $\subseteq$ DSPACETR (g), *then* L (T ;Γ) $\subseteq$ DSPACE (g). □

Next we turn to time-bounded transductions and time-bounded complexity classes. Instead of a single bounding function we now consider a class of functions that is closed under certain operations. The following definition is a slight modification of a concept from [20].

Definition 6.6. A class C of functions is called *natural* if
(1) C contains the identity function $\lambda x.\ x$,
(2) for each f and g in C, there is a monotone increasing function in C that majorizes $\lambda x.(f(x)+g(x))$,
(3) for each f and g in C, there is a monotone increasing function in C that majorizes $\lambda x.(f(x)g(x))$, and
(4) for each f in C, there is a monotone increasing function in C that majorizes $\lambda x.f(2x)$. □

Definition 6.7. Let for each class C of functions, NTIMETR (C) be the family of those transductions τ that satisfy
(1) τ is locally context-independent, and

(2) there exists a nondeterministic algorithm that can decide a query "$y \in \tau(x)$?" within time $g_\tau(|y|)$ for some g_τ in **C**. □

For a class **C** of functions NTIME(**C**) is defined by NTIME(**C**) = $\bigcup \{\text{NTIME}(g) \mid g \in \mathbf{C}\}$. Let *poly* be the class of all polynomials over the natural numbers. Obviously, poly is a natural class.

Theorem 6.8. [5]. *Let* **C** *be a natural class of functions, and let* Γ *be a family of languages closed under finite substitution and under intersection with regular languages.*
(1) *If* $\mathbf{T} \subseteq \text{NTIMETR}(\mathbf{C})$, *then* $\mathbf{L}(\mathbf{T}) \subseteq \text{NTIME}(\mathbf{C})$.
(2) $\mathbf{L}(\text{NTIMETR}(\mathbf{C})) = \text{NTIME}(\mathbf{C})$.
(3) *If* $\Gamma \subseteq \text{NTIME}(\mathbf{C})$ *and* $\mathbf{T} \subseteq \text{NTIMETR}(\mathbf{C})$, *then* $\mathbf{L}(\mathbf{T};\Gamma) \subseteq \text{NTIME}(\mathbf{C})$.

Proof: The proof is similar to the one of Theorem 6.2. As an example we show (3). Let $(G;C) = (V,\Sigma,U,S,C)$ be a Γ-controlled **T**-grammar. Assume $U = \{\tau_1,...,\tau_m\}$, and for each i $(1 \leqslant i \leqslant m)$ a query "$z \in \tau(y)$?" can be resolved within time $g_i(|z|)$ for some g_i in **C**. Since **C** is natural there exists a function g in **C** that majorizes $\lambda x.(g_1(x)+...+g_m(x))$ and hence $g(x) \geqslant g_i(x)$ for each x and each i $(1 \leqslant i \leqslant m)$.

Consider the algorithm of Figure 2. By Lemma 5.6 it suffices to execute the body of the **while**-loop at most $2n$ times where n is the length of the input. All statements in this body require time $O(n)$ only, except the test "$z \in \tau(y)$?" which is $O(g(n))$. Therefore this **while**-loop can be executed in time at most $O(n.(n+g(n)))$. The preceding statements consume $O(n)$ time, while the last statement of the algorithm needs time $h_1(2n)$ for some h_1 in **C** (assuming that $C \in \text{NTIME}(h_1)$). As **C** is natural, $\lambda n.h_1(2n)$ is majorized by some h in **C**. Thus the total time to execute the algorithm is $O(n+n(n+g(n))+h(n))$. Since **C** is natural this is majorized by some function in **C**. Hence $L(G;C) \in \text{NTIME}(\mathbf{C})$. □

Corollary 6.9. L(NTIMETR(poly)) = NP. □

Theorem 6.8(2) and Corollary 6.9 are variations of results established by Van Leeuwen [20] for another rather abstract grammatical model.

In addition to Theorem 6.8 we remark that from the main result in [19] it follows that if **T** contains all (λ-free) finite substitutions, then the membership problem for $\mathbf{L}(\mathbf{T})$ is NP-hard.

We conclude this section with a counterpart of Theorem 4.6 with respect to closure under iterated locally context-independent time- or space-bounded transductions. We call a family K of languages closed under *iterated* **T**-*transductions* if for each language L in K with $L \subseteq V^*$ for some alphabet V, and each finite set U of **T**-transductions over V, the language $U^*(L)$ belongs to K.

Theorem 6.10. [5]. *Let g be a function such that there exists a constant c with $g(2n) \leqslant c.g(n)$ for each $n \in \mathbb{N}$.*
(1) *If $g(n) \geqslant n$ for each $n \in \mathbb{N}$, then* NSPACE(g) *is the smallest AFL closed under iterated locally context-independent nondeterministic g-space-bounded transductions. In particular this applies to*
– NSPACE(n), *the family of context-sensitive languages;*
– NSPACE(n^2), *the family of two-way nondeterministic nonerasing stack automaton languages;*
– **PSPACE**.
(2) *If $g(n) \geqslant n \log n$ for each $n \in \mathbb{N}$, then* DSPACE(g) *is the smallest AFL closed under iterated locally context-independent deterministic g-space-bounded transductions. In particular this applies to*
– DSPACE$(n \log n)$, *the family of two-way deterministic nonerasing stack automaton languages.*
(3) *If* **C** *is a natural class of functions, then* NTIME(**C**) *is the smallest AFL closed under iterated locally context-independent nondeterministic* **C***-time-bounded transductions. In particular this applies to* **NP**.

Proof: From 6.2(2), 6.5(2) and 6.8(2) closure under iterated **T**-transductions easily follows for **T** equal to NSPACETR(g), DSPACETR(g), and NTIMETR(**C**) respectively. Closure under iterated **T**-transductions implies closure under union, concatenation, Kleene + and λ-free homomorphism. The remaining two AFL-properties (closure under inverse homomorphism and intersection with regular languages) can be proved by standard automaton-theoretic constructions. Since each AFL closed under iterated **T**-transductions includes **L**(**T**), it is easy to see that **L**(**T**) is the smallest AFL closed under iterated **T**-transductions.

For the characterization of two-way nonerasing stack automaton languages in terms of complexity classes we refer to [11]. □

It is an open problem whether a similar proposition holds for DSPACE(n), the family of deterministic context-sensitive languages.

7. Concluding Remarks

We summarized some results on the complexity of the membership problem for (controlled) iteration grammars (Section 4) and for (controlled) grammars based on transductions (Section 6). In the former case these results are rather satisfactory; in the latter one we could extend the results of Section 4 only at the price of requiring local context-independency. Whether this is a serious restriction is still open. However, when we drop this condition we will probably need a rather different approach to solving the membership problem in the context-dependent case.

Apart from this general problem, a few more concrete open questions have already been mentioned; cf. the remarks after Theorems 6.4 and 6.10. Another long standing open problem in this area, is the

question whether DSPACE(n) is a hyper-AFL, i.e., whether it is an AFL closed under iterated λ-free nondeterministic substitution [21,3]. It is even unknown whether this question is equivalent to the classic LBA-problem, i.e., is DSPACE(n) = NSPACE(n)?. Thus, our knowledge in this respect is more restricted in case of linear space than it is in case of polynomial time; cf. Theorem 4.9.

Finally, we mention a different approach to the subject of this survey based on the notion of nondeterministic log-space-bounded reducibility; cf. [15]. Let for each language family K, NLOG(K) be the family of languages that are many-one reducible to a language in K by a reduction function computable nondeterministically in space $\log n$ by a Turing machine of which each computation is of polynomial length. As usual, LOG(K) is the class of languages many-one log-space reducible to languages in K. Then the following holds.

Proposition 7.1. [15].
(a) LOG(EDTOL) = NSPACE($\log n$),
(b) NLOG(EDTOL) = **NP**,
(c) LOG(ETOL) = **NP**,
(d) NLOG(ETOL) = **NP**. □

Let ONE be the language family of singleton sets, i.e., ONE = $\{L \mid \text{card}(L) = 1\}$. Then, e.g., η(ONE) = EDTOL.

Theorem 7.2. [15]. *Let Γ be a language family closed under reversal, finite substitution and intersection with regular languages. Then* $\eta(\Gamma, \text{ONE}) \subseteq \text{NLOG}(\Gamma)$. □

As corollaries one obtains the restricted version of Theorem 4.1 with K = ONE, the main result of [12], and implications of the form: if Γ satisfies the conditions of Theorem 7.2 and $\Gamma \subseteq$ LOG(CF), then $\eta(\Gamma, \text{ONE}) \subseteq$ LOG(CF) where CF is the family of context-free languages; cf. [15]. On the other hand it is still open whether η(EDTOL,ONE) $\subseteq$ **P**.

References

1. P.R.J. Asveld: Controlled iteration grammars and full hyper-AFL's, *Inform. and Control* **34** (1977) 248-269.
2. P.R.J. Asveld: Iterated Context-Independent Rewriting – An Algebraic Approach to Families of Languages, Doctoral Dissertation (1978), Twente University of Technology, Enschede, The Netherlands.
3. P.R.J. Asveld: Space-bounded complexity classes and iterated deterministic substitution, *Inform. and Control* **44** (1980) 282-299.
4. P.R.J. Asveld: On controlled iterated GSM mappings and related operations, *Rev. Roumaine Math. Pures Appl.* **XXV** (1980) 139-145.

5. P.R.J. Asveld: Abstract grammars based on transductions, Memorandum INF-85-13 (1985), Twente University of Technology, Enschede, The Netherlands.

6. P.R.J. Asveld & J. Engelfriet: Iterated deterministic substitution, *Acta Inform.* **8** (1977) 285-302.

7. P.R.J. Asveld & J. Engelfriet: Extended linear macro grammars, iteration grammars, and register programs, *Acta Inform.* **11** (1979) 259-285.

8. P.R.J. Asveld & J. van Leeuwen: Infinite chains of hyper-AFL's, TW-memorandum No. 99 (1975), Twente University of Technology, Enschede, The Netherlands.

9. A.C. Fleck: Formal languages and iterated functions with an application to pattern representations, Report No. 75-03 (1975), Department of Computer Science, The University of Iowa, Iowa City.

10. M.A. Harrison: *Introduction to Formal Language Theory* (1978), Addison-Wesley, Reading, Mass.

11. J.E. Hopcroft & J.D. Ullman: *Introduction to Automata Theory, Languages, and Computation* (1979), Addison-Wesley, Reading, Mass.

12. N.D. Jones & S. Skyum: Recognition of deterministic ETOL languages in logarithmic space, *Inform. and Control* **35** (1977) 177-181.

13. K.-J. Lange: Context-free controlled ETOL systems, *in:* J. Díaz (Ed.): *Automata, Languages and Programming — 10th Colloquium*, Lect. Notes in Comp. Sci. **154** (1983) 723-733, Springer-Verlag, Berlin, Heidelberg, New York.

14. K.-J. Lange: Kontextfrei Kontrolierte ETOL-Systeme, Doctoral Dissertation (1983), University of Hamburg, F.R.G.

15. K.-J. Lange: L systems and NLOG-reductions, *in:* G. Rozenberg & A. Salomaa (Eds.): *The Book of L* (1985), Springer-Verlag, Berlin, Heidelberg, New York, pp. 245-252.

16. G. Paun: On the iteration of GSM mappings, *Rev. Roumaine Math. Pures Appl.* **XXIII** (1978) 921-937.

17. B. Rovan: A framework for studying grammars, *in:* J. Gruska & M. Chytil (Eds.): *Mathematical Foundation of Computer Science* 1981, Lect. Notes Comp. Sci. **118** (1981) 473-482, Springer-Verlag, Berlin, Heidelberg, New York.

18. G. Rozenberg & A. Salomaa: *The Mathematical Theory of L Systems* (1980), Academic Press, New York.

19. J. van Leeuwen: The membership question for ETOL languages is polynomially complete, *Inform. Process. Lett.* **3** (1975) 138-143.

20. J. van Leeuwen: Extremal properties of non-deterministic time-complexity classes, *in:* E. Gelenbe & D. Potier (Eds.): *International Computing Symposium* (1975), pp. 61-64, North-Holland, Amsterdam.

21. J. van Leeuwen: A study of complexity in hyper-algebraic families, *in:* A. Lindenmayer & G. Rozenberg (Eds.): *Automata, Languages, Development* (1976), pp. 323-333, North-Holland, Amsterdam.

22. D. Wood: Iterated a-NGSM maps and Γ systems, *Inform. and Control* **32** (1976) 1-26.