stichting
mathematisch
centrum

MC

AFDELING INFORMATICA
(DEPARTMENT OF COMPUTER SCIENCE)

IW 141/80 JULI

P.R.J. ASVELD

TIME AND SPACE COMPLEXITY OF INSIDE-OUT MACRO LANGUAGES

Preprint

kruislaan 413 1098 SJ amsterdam

Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

1980 Mathematics subject classifications: 68C25, 68F05, 68F10
ACM-Computing Reviews - categories: 5.23, 5.25

Time and Space Complexity of Inside-Out Macro Languages*

by

Peter R.J. Asveld

ABSTRACT

Starting from Fischer's IO Standard Form Theorem we show that for each inside-out (or IO-) macro language L there exists a λ-free IO-macro grammar with the following property: for each x in L there is a derivation of x of length at most linear in the length of x. Then we construct a nondeterministic log-space bounded auxiliary pushdown automaton which accepts L in polynomial time. Therefore the IO-macro languages are (many-one) log-space reducible to the context-free languages. Consequently, the membership problem for IO-macro languages can be solved deterministically in polynomial time and in space $(\log n)^2$.

KEY WORDS & PHRASES: *inside-out macro grammar, complexity of membership problem, (many-one) log-space reducibility, nondeterministic log-space bounded auxiliary pushdown automaton*

---

* This report will be submitted for publication elsewhere.

1. INTRODUCTION

Among the many generalizations of context-free grammars the classes of indexed grammars [1], the outside-in (or OI-) macro grammars, and the inside-out (or IO-) macro grammars [12,13] belong to the most interesting ones. For motivation, additional results and applications of these classes of grammars and some important subclasses and generalizations the reader is also referred to [2,4,5,8,9,19,24].

In [20] Rounds proved that the membership problem for indexed languages is complete for nondeterministic polynomial time. Since the family OI of languages generated by OI-macro grammars coincides with the family of indexed languages [12,13] the same conclusion holds with respect to the membership problem for OI.

Although OI and the family IO of languages generated by IO-macro grammars are incomparable [12], i.e. neither includes the other one, both families are properly included in NSPACE(n), the family of context-sensitive languages [12].

Contrary to the OI-case the membership problem for the family IO is feasible, as Hunt [17] showed that IO is included in the family $\mathcal{P}$ of languages accepted deterministically in polynomial time.

The main object of the present paper is to improve the inclusion $IO \subseteq NSPACE(n)$ [12] to $IO \subseteq DSPACE((\log n)^2)$.

After briefly recalling in Section 2 the definitions of IO-macro grammar and of space-bounded auxiliary pushdown automaton we consider in Section 3 Fischer's IO Standard Form Theorem [12] in somewhat more detail. Using this theorem we effectively construct for each IO-macro language L_0 a λ-free IO-macro grammar G such that (i) $L(G) = L_0 - \{\lambda\}$, and (ii) for each x in L(G) there exists a derivation of x of length at most linear in the length $|x|$ of x.

Using this result we construct in Section 4 for each IO-macro language L, a nondeterministic auxiliary pushdown automaton which accepts each word x in L within polynomial time using at most $\log |x|$ space on its work tapes. By Theorem 1 of [23] this immediately implies that the family of IO-macro languages is (many-one) log-space reducible to the family of context-free languages. As corollaries we obtain the known inclusion $IO \subseteq \mathcal{P}$ [17], as well as the improvement of the space bound already mentioned above.

2. DEFINITIONS

We assume familiarity with standard terminology and basic results in formal language and complexity theory. So for precise definitions of, for instance, the complexity classes P, NSPACE(S(n)), and DSPACE(S(n)) for any space bound S(n) we refer to standard texts like [15,16].

Macro grammars have originally been introduced in [12]. But here we only assume that the reader is familiar with the contents of the extended abstract [13]. We recall the main definitions in order to establish notation.

A *ranked alphabet* Φ is a finite set of symbols, each of which is provided with a unique nonnegative integer called the *rank*. For each $i \geq 0$, Φ_i denotes the subalphabet of Φ consisting of all symbols of rank i (So $\Phi_i \cap \Phi_j = \emptyset$ whenever $i \neq j$). PC denotes the alphabet of punctuation characters, viz. left parenthesis, right parenthesis, and comma symbol. The *set of terms* $T(\Phi)$ *over* Φ is the smallest set of strings over $\Phi \cup PC$ satisfying:

(i) Each element of Φ_0 and the empty word λ are in $T(\Phi)$,

(ii) If t_1 and t_2 are in $T(\Phi)$, then t_1t_2 is in $T(\Phi)$,

(iii) If A is in Φ_m for some $m \geq 1$, and if $t_1,\ldots,t_m$ are in $T(\Phi)$, then $A(t_1,\ldots,t_m)$ is in $T(\Phi)$.

An *inside-out* (or *IO-*) *macro grammar* $G = (\Phi,\Sigma,X,P,S)$ consists of

- an *initial symbol* S (S has rank zero: $S \in \Phi_0$),
- a ranked alphabet Φ of *nonterminals*,
- a *terminal alphabet* Σ,
- a finite set X of *variables* (The elements in both Σ and X have rank zero. Φ,Σ and X are assumed to be mutually disjoint. X contains at least as many variables as the highest rank of any symbol present in Φ),
- a finite set P of *productions* each of the form $A(x_1,\ldots,x_m) \rightarrow t$ with $A \in \Phi_m$, $x_1,\ldots,x_m$ are mutually distinct elements from X, and t is a term from $T(\Phi \cup \{x_1,\ldots,x_m\} \cup \Sigma)$.

The sentential forms obtainable by G are elements in $T(\Phi \cup \Sigma)$. More precisely, for σ and τ in $T(\Phi\cup\Sigma)$, $\sigma \underset{IO}{\Rightarrow} \tau$ holds if

- σ contains a substring over $\Phi \cup \Sigma \cup PC$ of the form $A(w_1,\ldots,w_m)$ for some A in Φ_m ($m \geq 0$) and $w_1,\ldots,w_m \in \Sigma^*$ (i.e. this occurrence of A in σ is

innermost),

- $A(x_1,\ldots,x_n) \to t$ is a production in P,
- τ is the result of replacing that occurrence of $A(w_1,\ldots,w_m)$ in σ by t', where t' is obtained by substituting the word w_i for the variable x_i $(1 \le i \le m)$ in t.

As usual $\underset{IO}{\Rightarrow}^+$ is the transitive closure of the relation $\underset{IO}{\Rightarrow}$. The language L(G) generated by G is defined by $L(G) = \{w \in \Sigma^* \mid S \underset{IO}{\Rightarrow}^+ w\}$. The family of languages generated by IO-macro grammars is denoted by IO.

An IO-macro grammar G is called λ-*free* if the right-hand side of each production does not contain any occurrence of λ.

EXAMPLE 2.1. Consider $G = (\Phi,\Sigma,X,P,S)$ with $\Phi = \Phi_0 \cup \Phi_1$, $\Phi_0 = \{S\}$, $\Phi_1 = \{A,B\}$, $\Sigma = \{a,b\}$ and $X = \{x\}$. P consists of the following productions:

$S \to A(a)$

$A(x) \to B(A(xa))$

$A(x) \to B(x)$

$B(x) \to xbx$

Then $L(G) = \{a^m(ba^m)^n \mid m \ge 1;\ n = 2^m-1\}$ is in IO. In [12,13] it was shown that L(G) is not an indexed (or, equivalently, an OI-macro) language. □

Next we turn to nondeterministic auxiliary pushdown automata and related concepts. Formal definitions are given in [6,10,11,23,24], but for our purpose an intuitive description suffices.

Let $f:\Sigma_1^* \to \Sigma_2^*$ be a mapping, where Σ_1 and Σ_2 are alphabets. Then f is *log-space computable* if there exists a deterministic Turing machine with a two-way read-only input tape, a one-way write-only output tape, and a two-way read/write work tape, which when started with $x \in \Sigma_1^*$ on its input tape halts with $f(x) \in \Sigma_2^*$ on its output tape, and uses at most $\log|x|$ tape squares on its work tape during the computation of f(x).

A language L_1 over Σ_1 is (many-one) *log-space reducible* to a language L_2 over Σ_2, denoted by $L_1 \le_{\log} L_2$, if there exists a log-space computable function $f:\Sigma_1^* \to \Sigma_2^*$ such that, for all $x \in \Sigma_1^*$, x is in L_1 if and only if f(x) is in L_2.

The family of languages log-space reducible to context-free languages is denoted by LOG(CF), i.e., $LOG(CF) = \{L \mid L \le_{\log} L' \text{ for some } L' \text{ in CF}\}$. Many examples of language families which are included in LOG(CF) can be

found in [10,11,22,23].

Finally, a *nondeterministic log(n)-space bounded auxiliary pushdown automaton* is a nondeterministic Turing acceptor with a two-way read-only input tape, a finite number of log(n)-space bounded read/write work tapes (where n is the length of the input), and an unbounded pushdown store. Adding a polynomial time restriction on these devices yields a characterization of the class LOG(CF), as it was established by Sudborough in [23]:

THEOREM 2.2 [23].

(1) *A language* L *is in* LOG(CF) *if and only if it is accepted by some nondeterministic log*(n)*-space bounded auxiliary pushdown automaton in polynomial time.*

(2) LOG(CF) $\subseteq$ $P \cap$ DSPACE($(\log n)^2$). □

3. A LINEAR BOUND ON THE DERIVATION LENGTH

In this section we first recall Fischer's IO Standard Form Theorem [12]. Then we transform each λ-free IO-macro grammar (in IO Standard Form) into an equivalent λ-free IO-macro grammar allowing a linear bound on the derivation length for all words in the corresponding language. In general, however, the new grammar need not be in IO standard form.

DEFINITION 3.1. A λ-free IO-macro grammar $G = (\Phi,\Sigma,X,P,S)$ is in *IO standard form* if

(1) each of its productions is *argument preserving*, i.e. if $A(x_1,\ldots,x_m) \to t$ is in P, then $x_1,\ldots,x_m$ do occur in t,

(2) each of its productions is of one of the following two forms:

(2.1) $A(x_1,\ldots,x_m) \to B(C(y_1,\ldots,y_k),z_2,\ldots,z_\ell)$ $(m \geq 0,\ \ell \geq 1)$ where $A \in \Phi_m$, $B \in \Phi_\ell$, $C \in \Phi_k$ and $y_1,\ldots,y_k,\ z_2,\ldots,z_\ell \in \{x_1,\ldots,x_m\}$ (Note that by (1), for each x_p $(1 \leq p \leq m)$, there is either a y_i $(1 \leq i \leq k)$ with $y_i = x_p$, or a z_j $(2 \leq j \leq \ell)$ with $z_j = x_p$).

(2.2) $A(x_1,\ldots,x_m) \to w$ with w in $(\Sigma \cup \{x_1,\ldots,x_m\})^+$ and $m \geq 0$. (By (1) each x_p $(1 \leq p \leq m)$ occurs at least once in w). □

From this definition and a straightforward induction over the length of

derivation we obtain the following consequences ($|w|$ denotes the length of the word w).

COROLLARY 3.2. *Let* $G = (\Phi,\Sigma,X,P,S)$ *be a* λ*-free IO-macro grammar in IO standard form. Then for each* σ *in* $T(\Phi \cup \Sigma)$ *with* $S \overset{*}{\underset{IO}{\Rightarrow}} \sigma \overset{*}{\underset{IO}{\Rightarrow}} x$ *for some* x *in* Σ^+ *we have*

$$\sigma = A_p(\ldots A_2(A_1(w_{11},\ldots,w_{1n_1}),w_{22},\ldots,w_{2n_2}),\ldots w_{p2},\ldots,w_{pn_p})$$

for some $A_i \in \Phi_{n_i}$ $(n_i \geq 0; 1 \leq i \leq p)$, *and some* w_{11}, w_{ij} *in* Σ^+ $(1 \leq i \leq p;\ 2 \leq j \leq n_i)$ *where* w_{11} *and* w_{ij} *are nonempty subwords of* x *satisfying*

$$0 \leq |w_{11}| + \sum_{i,j} |w_{ij}| \leq |x|.$$

Moreover, the number of times we apply a rule of the form 3.1(2.2) in a derivation $S \overset{+}{\underset{IO}{\Rightarrow}} x$ (x *in* Σ^+) *equals one plus the number of times that we apply a production of the form 3.1(2.1) in that derivation.* □

THEOREM 3.3 (*IO Standard Form Theorem* [12, Theorem 3.3.4]). *For each IO-macro grammar* G, *we can effectively construct a* λ*-free IO-macro grammar* G' *in IO standard form such that* $L(G') = L(G) - \{\lambda\}$. □

EXAMPLE 3.4. Let $G' = (\Phi',\Sigma,X,P',S)$ with $\Phi' = \Phi'_0 \cup \Phi'_1$, $\Phi'_0 = \{S,C\}$, $\Phi'_1 = \{A,B,D,E,F\}$, $\Sigma = \{a,b\}$, $X = \{x\}$, and P' contains the productions

S → A(C)

C → a

A(x) → B(D(x))

D(x) → A(E(x))

E(x) → xa

B(x) → xbx

A(x) → B(F(x))

F(x) → x

Then G' is in IO standard form and $L(G') = L(G)$ where G is the λ-free IO-macro grammar of Example 2.1. □

With each term t in $T(\Phi \cup \Sigma)$ we associate a natural number f(t) which

equals the length of the string in Σ^* obtained by erasing all symbols from $\Phi \cup PC$ in $t \in T(\Phi \cup \Sigma) \subseteq (\Phi \cup PC \cup \Sigma)^*$. Thus $0 \leq f(t) \leq |t|$ for each $t \in T(\Phi \cup \Sigma)$. Moreover $f(t) = 0$ iff t is in $T(\Phi)$, and $f(t) = |t|$ iff t is in Σ^*. And if t equals the sentential form σ mentioned in Corollary 3.2, then

$$f(\sigma) = |w_{11}| + \sum_{i,j} |w_{ij}| \qquad 1 \leq i \leq p, \quad 2 \leq j \leq n_i.$$

We now turn to the main result of this section.

THEOREM 3.5. *For each IO-macro grammar* G *we can effectively construct a* λ*-free IO-macro grammar* G' *such that*

(i) $L(G') = L(G) - \{\lambda\}$.

(ii) *for each* x *in* L(G') *there exists a derivation of* x *according to* G' *of length at most* $c \cdot |x|$ *where* c *is a constant depending on* G *only.*

PROOF. By Theorem 3.3 there exists a λ-free IO-macro grammar $G_0 = (\Phi, \Sigma, X, P, S)$ in IO standard form such that $L(G_0) = L(G) - \{\lambda\}$.

We call a derivation step $\sigma \underset{IO}{\Rightarrow} \sigma'$ *length preserving* [*length-increasing*] if $f(\sigma) = f(\sigma')$ [$f(\sigma) < f(\sigma')$ respectively]; since G_0 is λ-free the case $f(\sigma) > f(\sigma')$ never occurs.

In a derivation $S \underset{IO}{\Rightarrow}^+ x$ the number of length-increasing steps is at most $|x|$. But in general there is no bound on the number of length-preserving steps: certain subderivations may leave a sentential form unchanged (viz. $\sigma \underset{IO}{\Rightarrow}^+ \sigma$) and so these subderivations can be repeated any number of times (cf. Lemma 2.3 in [3] for an analogous situation).

A production is called *length-preserving* [*length-increasing*] if, when applied to a sentential form, it gives rise to a length-preserving [length-increasing] derivation step. Since G_0 is argument-preserving, a production of the form

3.1(2.1) $\quad A(x_1,\ldots,x_m) \rightarrow B(C(y_1,\ldots,y_k),z_2,\ldots,z_\ell) \qquad (m \geq 0,\ \ell \geq 1)$

is length-preserving if and only if $k + \ell = m + 1$. Similarly, a production of the form

3.1(2.2) $\quad A(x_1,\ldots,x_m) \rightarrow w \qquad (m \geq 0)$

is length-preserving if and only if $m \geq 1$ and $w = x_{\pi(1)} \ldots x_{\pi(m)}$ for some permutation π of $\{1,\ldots,m\}$.

The grammar G' is obtained by removing from G_0 all length-preserving productions of the form 3.1(2.2) with m = 1, i.e., of the form $Z(x_1) \to x_1$, and by adding new rules in the following way. For each production in G_0 of the form 3.1(2.1) in which Z occurs innermost, viz.

(1) $\quad A(x_1,\ldots,x_m) \to B(Z(y_1),z_2,\ldots,z_\ell)$

we add a rule

(1') $\quad A(x_1,\ldots,x_m) \to B(y_1,z_2,\ldots,z_\ell) \qquad 1 \le m \le \ell$

and similarly, for each rule of the form 3.1(2.1) in which Z occurs outermost, viz.

(2) $\quad A(x_1,\ldots,x_m) \to Z(C(y_1,\ldots,y_k))$

we add a production

(2') $\quad A(x_1,\ldots,x_m) \to C(y_1,\ldots,y_k) \qquad 0 \le m \le k.$

Note that if $Z(x_1) \to x_1$ is the only production of the form 3.1(2.2) for Z, then we can replace (1) and (2) by (1') and (2') respectively, and remove Z from G.

In general we ought to iterate this construction in order to get rid of rules of the form $A(x_1) \to Z(Z'(x_1))$. Clearly, a finite number of iteration steps always suffices.

For the grammar G' obtained in this way it is straightforward to show that $L(G') = L(G_0)$.

In G' each production has one of the forms 3.1(2.1), 3.1(2.2), or

(*) $\quad A(x_1,\ldots,x_m) \to B(z_1,\ldots,z_\ell) \qquad 0 \le m \le \ell.$

Clearly, a rule of the form (*) is length-preserving if and only if $m = \ell$ or, equivalently, if either m = 0 or $(z_1,\ldots,z_\ell)$ is a permutation of $(x_1,\ldots,x_m)$.

Consider a derivation $S \overset{+}{\underset{IO}{\Rightarrow}} x$ according to G'. For type 3.1(2.2) productions we have either $m \le 1$ or $m \ge 2$. In case $m \le 1$ the rules are length-increasing, as G' is λ-free and length-preserving productions with m = 1 have been removed. It is now easy to see that during the derivation of x there are at most $|x|$ applications of type 3.1(2.2) rules, since these productions are either length-increasing, or length-preserving with $m \ge 2$. In the latter case the statement follows from the fact that G' is λ-free and argument-preserving.

In this derivation these $|x|$ applications remove $|x|$ occurrences of nonterminals in total. These $|x|$ occurrences can only be introduced by

$|x| - 1$ applications of type 3.1(2.1) rules, because productions of the form (*) preserve the number of nonterminals. So in $S \underset{IO}{\Rightarrow}^{+} x$ there are at most $2|x| - 1$ applications of productions of either type 3.1(2.1) or type 3.1(2.2). But in between there may occur any number of applications of type (*) rules.

The number of consecutively applied length-preserving productions of type (*) can be bounded by

$$M = \max\{(\#\Phi_m)^2.m! \mid 1 \leq m \leq p\}$$

where $\#\Phi_m$ is the cardinality of Φ_m and p is the largest natural number such that $\Phi_p \neq \emptyset$. This follows from the fact that in a subderivation purely obtained by length-preserving type (*) rules the same sentential form must occur twice within M + 1 steps.

Since G' is argument-preserving, we can use at most p length-increasing type (*) productions before applying again rules of the form 3.1(2.1) or 3.1(2.2).

Therefore, for each x in L(G'), G' allows a derivation of x of length at most $2p.M.|x|$. □

EXAMPLE 3.6. When we perform this construction to the grammar of Example 3.4, we replace the last two productions of that grammar by the single rule $A(x) \rightarrow B(x)$ while we remove F from the grammar. □

It is even possible to reduce the constant c in Theorem 3.5 to 2. But then we must introduce productions of a form different from 3.1(2.1), 3.1(2.2) and (*). Consequently, the algorithm to be presented in the next section becomes more complicated.

4. THE COMPLEXITY OF RECOGNIZING IO-MACRO LANGUAGES

The construction in the proof of Theorem 3.5 enables us to design a straightforward top-down recognition algorithm for IO-macro languages. This algorithm can easily be implemented on an auxiliary pushdown automaton. Finally, an analysis of the amount of time and space consumed by this auxiliary pushdown automaton together with Theorems 2.2(1) and 3.5(ii) yields the main result of this paper, viz.,

THEOREM 4.1. IO $\subseteq$ LOG(CF).

PROOF. Consider an IO-macro language generated by a λ-free IO-macro grammar $G' = (\Phi', \Sigma, X, P', S)$ obtained in the way described in proving Theorem 3.5. Thus each production in P' has one of the forms 3.1(2.1), 3.1(2.2) or (*). We present a nondeterministic algorithm which decides whether a given string $x \in \Sigma^+$ is in L(G'). This algorithm uses an array of length p, denoted by $x_1,\ldots,x_p$ (p is the largest rank of any symbol in Φ'.), a pushdown store PDS, and two variables nt and π of type "nonterminal" and "production" respectively. The structure of the algorithm looks as follows.

```
nt: = S;
for i:=1 step 1 to p do x_i := λ od;
while |x_1| ≤ |x| do
   guess π ∈ P';
   case .
        .
        .
        .
   end case;
od;
reject
```

For each π in P' there is a "case" in the case-statement. We give examples of typical entries in this case-statement according to the form of π. First, if π is of the form 3.1(2.1), i.e., $A(x_1,\ldots,x_m) \to B(C(y_1,\ldots,y_k), z_2,\ldots,z_\ell)$ with $\ell \geq 1$, the corresponding "case" reads

```
if nt = A then
   for i:= ℓ step - 1 to 2 do PUSH(z_i) od;
   PUSH(B);
   <x_1,...,x_k>:= <y_1,...,y_k>; nt:=C;
fi;
```

(We use <...> to denote parallel assignment).

Second, when π is a type 3.1(2.2) production, viz. $A(x_1,\ldots,x_m) \to w$ with $m \geq 0$, then its "case" looks like

```
if nt = A then
  x_1 := w;
  if PDS = empty and x_1 = x
     then accept
     else nt:= POP;
          for i:=2 step 1 to ρ(nt) do x_i := POP od;
  fi;
fi;
```

Here $\rho(nt)$ denotes the rank of the nonterminal in nt.

Finally, for a rule π of the form (*), e.g. $A(x_1,\ldots,x_m) \to B(z_1,\ldots,z_\ell)$ with $0 \leq m \leq \ell$, the corresponding entry in the case-statement is

```
if nt = A then
   <x_1,...,x_ℓ>:= <z_1,...,z_ℓ>;
   nt:=B;
fi;
```

We now show how this algorithm can be implemented on a nondeterministic auxiliary pushdown automaton. Firstly, the values of nt and π do not depend on $|x|$ but on G' only, and therefore they can be stored in the finite control. In a derivation of G' leading to x, the value of each argument is a substring of x, and it is determined completely by its beginning and its ending position in x, i.e., by two natural numbers in between 1 and $|x| = n$. So we can store each variable x_i using $O(\log n)$ bits only (cf. [18,11,14,21,3]). The pushdown alphabet equals $\Phi \cup \{0,1,\$\}$ where Φ is now considered to be a non-ranked alphabet, and \$ is a marker to separate the bit strings.

Starting from the algorithm, the construction of a nondeterministic log-space bounded auxiliary pushdown automaton which accepts L(G') is now straightforward except for the following point. The assignment x_1:=w (cf. type 3.1(2.2) rules) may result in a value of x_1 which is not a substring of x. Therefore we guess a substring of x and put it in x_1. Then we test whether this guess is unequal to the value of the expression w. In case it is, we reject x; otherwise we continue.

By Theorem 2.2(1) it now suffices to show that this log-space bounded

auxiliary pushdown automaton recognizes L(G') in polynomial time. From the linear bound on the derivation length (Theorem 3.5) it follows that the while-loop is executed $O(n)$ times. Thus the total time consumed in recognizing L(G') nondeterministically is $O(n^2)$, since the test and each "case" in the while-loop requires at most $O(n)$ time. □

From Theorems 2.2(2) and 4.1 we obtain the following consequences, the former of which was already established in [17] using a different argument.

COROLLARY 4.2. (1) IO $\subseteq$ P,
(2) IO $\subseteq$ DSPACE($(\log n)^2$),
(3) IO $\subseteq$ NTIME(n^2). □

It is an interesting open question whether the n^2 bound in 4.2(3) can be improved to n log n or even to n (Remember that the context-free languages are in NTIME(n)).

Obviously, Theorem 4.1 and Corollary 4.2 also hold for subclasses of the family IO. In particular we mention the family of basic languages [12,13] and the several families generated by bounded nested IO-macro grammars [9]. With respect to the linear basic [12,13] or, equivalently, the EDTOL languages [7] it was shown in [18] that their membership problem can be solved in NSPACE(log n); cf. [11,14,21,3].

ACKNOWLEDGMENT.

The author is indebted to Joost Engelfriet and Jan van Leeuwen for their comments on a preliminary version of this paper.

REFERENCES

[1] AHO, A.V., "Indexed grammars - an extension of context-free grammars", *J. Assoc. Comp. Mach.* 15 (1968) 647-671.

[2] ASVELD, P.R.J., "*Iterated Context-Independent Rewriting - An Algebraic Approach to Families of Languages*" (Doctoral dissertation), Twente University of Technology, Enschede, The Netherlands (1978).

[3] ASVELD, P.R.J., "Space-bounded complexity classes and iterated deterministic substitution", *Inform. Contr.* 44 (1980) 282-299.

[4] ASVELD, P.R.J., "Generalizations of context-free grammars and the non-self-embedding property", Report (1980), Mathematical Centre, Amsterdam, The Netherlands.

[5] ASVELD, P.R.J. and J. ENGELFRIET, "Extended linear macro grammars, iteration grammars, and register programs", *Acta Informatica* 11 (1979) 259-285.

[6] COOK, S.A., "Characterizations of pushdown machines in terms of time-bounded computers", *J. Assoc. Comp. Mach.* 18 (1971) 4-18.

[7] DOWNEY, P.J., "*Formal Languages and Recursion Schemes*", (Doctoral dissertation) Harvard University, Cambridge, Mass. (1974). See also: Proceedings IEEE Conf. on Biologically Motivated Automata Theory (1974) 54-58.

[8] ENGELFRIET, J. and E. MEINECHE SCHMIDT, "IO and OI", Part I *J. Comput. System Sci.* 15 (1977) 328-353, Part II *J. Comput. System Sci.* 16 (1978) 67-99.

[9] ENGELFRIET, J. and G. SLUTZKI, "Bounded nesting in macro grammars", *Inform. Contr.* 42 (1979) 157-193.

[10] ERNI, W.J., "Some further languages log-tape reducible to context-free languages", Report 45 (1976), Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, FGR.

[11] ERNI, W.J., "Auxiliary pushdown acceptors and regulated rewriting systems", Report 59 (1977), Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, FGR.

[12] FISCHER, M.J., "*Grammars with Macro-like Productions*", (Doctoral dissertation), Harvard University, Cambridge, Mass. Reprinted in: "Mathematical Linguistics and Automatic Translation", Harvard University Computation Laboratory Report NSF-22 (1968).

[13] FISCHER, M.J., "Grammars with macro-like productions", Proceedings 9th Ann. IEEE Symp. on Switching and Automata Theory (1968) 131-142.

[14] HARJU, T., "A polynomial recognition algorithm for the EDTOL languages", *Elektron. Informationsverarbeit. Kybernetik* 13 (1977) 169-177.

[15] HARRISON, M.A., "*Introduction to Formal Language Theory*", Addison-Wesley, Reading, Mass. (1978).

[16] HOPCROFT, J.E. and J.D. ULLMAN, "*Introduction to Automata Theory, Languages and Computation*", Addison-Wesley, Reading, Mass. (1979).

[17] HUNT III, H.B., "On the complexity of finite, pushdown, and stack automata", *Math. Systems Theory* 10 (1976) 33-52.

[18] JONES, N.D. and S. SKYUM, "Recognition of deterministic ETOL languages in logarithmic space", *Inform. Contr.* 35 (1977) 177-181.

[19] MASLOV, A.N., "The hierarchy of indexed languages of an arbitrary level", *Soviet Math.* 15 (1974) 1170-1174.

[20] ROUNDS, W.C., "Complexity of recognition in intermediate-level languages", Proceedings 14th Ann. IEEE Symp. on Switching and Automata Theory (1973) 145-158.

[21] SUDBOROUGH, I.H., "The time and tape complexity of developmental languages", in A. Salomaa and M. Steinby (Eds.), "Automata, Languages and Programming, 4th colloquium", Lecture Notes in Computer Science 52 (1977) 509-523, Springer-Verlag, Berlin/ Heidelberg/New York.

[22] SUDBOROUGH, I.H., "The complexity of the membership problem for some extensions of context-free languages", *Internat. J. Computer Math.* 6 (1977) 191-215.

[23] SUDBOROUGH, I.H., "On the tape complexity of deterministic context-free languages", *J. Assoc. Comp. Mach.* 25 (1978) 405-414.

[24] WAND, M., "*Mathematical Foundations of Formal Language Theory*", (Doctoral dissertation), Massachusetts Institute of Technology, Project MAC, TR-108, Cambridge, Mass. (1973).