Parallel Complexity of Iterated Morphisms and the Arithmetic of Small Numbers*

Carsten Damm† Markus Holzer† Klaus-Jörn Lange†

Abstract

We improve several upper bounds to the complexity of the membership problem for languages defined by iterated morphisms (D0L systems). The complexity bounds are expressed in terms of $\mathcal{DLOGTIME}$-uniform circuit families. We prove: 1) For polynomially growing D0L systems the membership problem is contained in $\mathcal{AC}^0$. 2) For arbitrary D0L systems the membership problem is contained in $\mathcal{NC}^1$. 3) The latter can be improved to $\mathcal{TC}^0$ if and only if upper bounds to a number of natural arithmetic problems can be improved to $\mathcal{TC}^0$. 4) The general D0L membership problem (the D0L system is part of the input) is contained in Cook's class $\mathcal{DET}$.

1 Introduction

We investigate languages defined by iterated homomorphisms or substitutions, i.e.: context-free Lindenmayer languages. They provide a whole framework of classes which have properties similar to the context-free (Chomsky) languages. The complexity of the membership problem for these languages is in most cases well determined, as it is for context-free languages. This can be seen by the following completeness-results[1], since the class of T0L languages is $\mathcal{NP}$-complete [15], the class of DT0L languages is $NSPACE(\log n)$-complete [12], and the class of 0L languages is $\mathcal{LOGCFL}$-complete [18].

For the case of D0L systems, i.e., of *one* iterated homomorphism, no such precise complexity bounds are known. Sudborough exhibited in [18] a logarithmically space bounded algorithm for the D0L membership problem. But at present there is no lower bound, i.e., no hardness result. This is caused by the fact, that LOG-reducibility used in [13, 18] is not adequate to treat classes below $DSPACE(\log n)$. In addition, D0L languages are sparse. *Sparseness* is a concept of growing interest in the field of complexity theory. A set $L \subseteq X^*$ is said to be sparse, if the cardinality of $L \cap X^n$ is bounded by some polynomial in n. Hardness results for sparse sets often would imply unlikely consequences. Thus, it is in general difficult to provide lower bounds for sparse sets (for an overview on results of this kind see [2, 10]). The best-known example for sparse sets are *Tally* languages, which are build over a one-letter alphabet.

D0L languages now represent one further natural example for sparseness. A slight extension of Sudborough's algorithm in [18] shows that D0L languages are not only sparse but indeed $DSPACE(\log n)$-printable, that is somehow "constructively sparse" (see [11]).

Thus, D0L languages are of a very low complexity. In order to characterize their membership problem, we have to look for complexity classes and reducibility notions

*Supported by DFG-La 618/1-1

†Institut für Informatik, Technische Universität München, Arcistr. 21, W-8000 München 2 (Germany)

[1]Usually, only *extended* systems are considered, which select only generated words over a *terminal* alphabet. Obviously, this is not important with respect to membership problems

below $DSPACE(\log n)$. This leads us in a natural way to parallel complexity theory and to classes and reductions defined by Boolean circuits, in particular to $\mathcal{AC}^0$, $\mathcal{TC}^0$ and $\mathcal{NC}^1$ (see [6]).

Because of the printability properties of D0L languages the construction of efficient $DSPACE(\log n)$-uniform circuits for the D0L membership problem is easy. But we achieve more: The upper bounds for the D0L membership problem heavily rely on upper bounds for certain arithmetic problems (multiplication, polynomial evaluation, etc.). The observation that only *small*, i.e., polynomially bounded numbers have to be dealt with, enables us to construct efficient *and* highly uniform, i.e., $\mathcal{DLOGTIME}$ -uniform circuits to solve the problem.

We further prove that D0L membership is as hard as small number arithmetic in some sense: w.r.t. $\mathcal{TC}^0$ -reductions recognition of D0L languages is equivalent to a number of arithmetic problems on small numbers. As a side result we obtain that recognizing tally D0L languages is as hard as recognizing arbitrary D0L languages.

Finally, we consider *general membership* problems, where the L system is part of the input. By [13] we know that GM_{T0L} as well as GM_{DT0L} are $\mathcal{PSPACE}$-complete and that GM_{0L} is $\mathcal{NP}$-complete. Again, the D0L case is open. By [13, 14] we know

$$NSPACE(\log n) \leq_{LOG} GM_{D0L} \in \mathcal{P} \cap DSPACE(\log^2 n).$$

We improve this in showing containment in the class $\mathcal{DET}$ defined by Cook in [6].

2 Some facts about D0L systems

Most of the following material is well known (see [16]).

Definition 1 A D0L system $G = (\Sigma, h, \alpha)$ consists of a finite alphabet Σ, a homomorphism $h : \Sigma^* \longrightarrow \Sigma^*$, and an "axiom" $\alpha \in \Sigma^*$. The language L_G generated by G is the set of words obtained by iteratively applying the homomorphism h to α: $L_G = h^*(\alpha) = \{\alpha, h(\alpha), h^2(\alpha), h^3(\alpha), \ldots\}$. The class of D0L languages is denoted by $D0L = \{L_G \mid G \text{ is a D0L system}\}$.

L_G is also called the membership problem of the D0L system G (G fixed). Most complexity questions dealt with in this paper refer to this situation.

Definition 2 The function $f_G(n) = \mid h^n(\alpha) \mid$ is called the growth function of G.

Lemma 3 For each D0L system G there is either a constant $c > 1$ such that for all n holds $f_G(n) \geq c^n$ or there is a polynomial $p(n)$ such that for all n holds $f_G(n) \leq p(n)$.

Let us call the D0L systems with polynomial upper bound on the growth function *polynomial* and the other ones *exponential* D0L systems. A D0L system $G = (\Sigma, h, \alpha)$ is called *conservative* if each letter a appears in every word in L_G. For special D0L systems Lemma 3 can be strengthened.

Lemma 4 ([7]) Let $G = (\Sigma, h, \alpha)$ be a polynomial conservative D0L system such that for all $a \in \Sigma$ holds $h(a) \in \Sigma^* a \Sigma^*$ or $h(a) = \lambda$. Then there is a polynomial $p(n)$ with rational coefficients and a number n_0 such that for each $n \geq n_0$ holds $f_G(n) = p(n)$.

By a technique called "slicing" or "tuning" each language generated by a polynomial D0L system can be decomposed into a disjoint union of languages, generated by D0L systems satisfying the assumptions of Lemma 4 (see [7]). We will assume therefore that all *polynomial* D0L systems under consideration fulfill the assumptions of Lemma 4: since we start from a fixed D0L system we can take the decomposition for free and need only to run the algorithms making the decision for the components of the decomposition.

3 Uniform circuits and small number arithmetic

The membership problem for D0L systems will be studied in terms of circuits. We assume familiarity with the definition of Boolean circuits as developed in [6].

A circuit C_n on n inputs in the usual way computes an n-variable function $f_{C_n} : \{0,1\}^n \longrightarrow \{0,1\}$, a circuit family $C = (C_n)$ accordingly computes a function $f_C : \{0,1\}^* \longrightarrow \{0,1\}$, where the nth circuit processes inputs of length n. The language or problem decided by C is the set of words in $\{0,1\}^*$ mapped to 1 under f_C.

This definition covers only two-letter languages, since Boolean circuits are based on two-valued logic. By binary encodings circuit families can also recognize more general languages. We will tacitly make use of this as a convention, without specifying the encoding.

We will also consider computation problems, e.g.: given two numbers, compute their product. In these cases the circuits to be considered have several output gates.

To define uniformity we make use of the terminology of [17]: Let $C = (C_n)$ be a circuit family. The *direct connection language* of C is the set $L_{DC}(C)$ of tuples $\langle 1^n, g_1, g_2, \tau \rangle$ where g_1 and g_2 are names of gates in C_n, g_2 is a child of g_1 and τ is the type of g_1 (for gates g_1 without predecessors g_2 is dummy). If $\mathcal{A}$ is an arbitrary class of languages we call a circuit family C to be $\mathcal{A}$-uniform if $L_{DC}(C) \in \mathcal{A}$. We will consider $\mathcal{DLOGTIME}$-uniform circuits. Here the direct connection language is recognized by deterministic Turing machines with logarithmic time bound (see [1]).

Definition 5 $\mathcal{AC}^0$ is the set of problems solvable by $\mathcal{DLOGTIME}$-uniform unbounded fan-in polynomial size circuits of constant depth. If the uniformity-condition is removed, this class is denoted *nonuniform-*$\mathcal{AC}^0$. $\mathcal{NC}^1$ consists of all problems solvable by $\mathcal{DLOGTIME}$-uniform bounded fan-in polynomial size circuits of logarithmic depth.

We will show that certain arithmetic problems can be solved in $\mathcal{AC}^0$ or $\mathcal{NC}^1$ in case the numerical inputs are small compared to the input length n. "Small" means here to be polynomial bounded in n. This can formally be met by supplying the numbers in unary notation: input number $m \leq n$ is given in the form $1^m 0^{n-m}$. Further one considers 1^n to be part of the input.

We consider the following problems (all numbers are nonnegative integers):

1. UNARY-TO-BINARY (BINARY-TO-UNARY): given m, compute $bin(m)$ (the opposite to the former)

2. MULTIPLY-SMALL-NUMBERS: given (l, m), compute $bin(lm)$ if $lm \leq n$, return a fault signal otherwise

3. POLYNOMIAL-EVALUATION-ON-SMALL-NUMBERS: given m, compute $bin(p(m))$ for a fixed polynomial $p(x)$ with integer coefficients

4. DIVISION-OF-SMALL-NUMBERS: (l, m), compute $(bin(d), bin(r))$, with $l = dm + r, 0 \leq r \leq m - 1$

5. GCD-OF-SMALL-NUMBERS: given (l, m), $0 < l, m$, compute $bin(GCD(l, m))$.

6. POWER-TO-SMALL-NUMBERS: given m, for a fixed integer $a > 0$ compute $bin(a^m)$ if $a^m \leq n$, otherwise return a fault signal

7. MATRIX POWER-TO-SMALL-NUMBERS: given m, for a fixed $s \times s$ matrix A and a fixed number $a > 1$ compute a binary representation of A^m if $a^m \leq n$ and all entries of A^m do not exceed n, otherwise return a fault signal

Please note that 3, 6, and 7 are *classes* of problems rather than single problems.

Proposition 6 The above mentioned problems are computable in $\mathcal{NC}^1$. Problems 1. – 5. are even computable in $\mathcal{AC}^0$

Sketch of proof: We give only the proof for Problem 2. For Problem 1. the reader is referred to [3]. For the other problems one makes use of the following observation: once circuits have been proved to be $\mathcal{DLOGTIME}$-uniform they can be used as parts (modules) of new circuits, which turn out to be $\mathcal{DLOGTIME}$-uniform if the local interconnection structure between the modules can be recognized in $\mathcal{DLOGTIME}$. So starting with a "small-numbers-multiplier" one can construct highly uniform circuits for more complex problems. The multiplier circuit works as follows:

W.l.o.g. we assume $lm \leq n$. The problem can easily be reduced to computing in parallel all products $l_1 \cdot m$ (where l_1 ranges from 0 to n) by fixed factor multipliers $M_n^{l_1}$ and simultaneously testing whether. The proper result will be passed through. We are left with constructing the fixed-factor multipliers $M_n^{l_1}$. M_n^0 is easily constructed. Suppose $l_1 > 0$. We consider copy gates $C_n^{i,j}$ for $1 \leq i \leq l_1, 1 \leq j \leq n$. For each i gate $C_n^{i,j}$ simply reproduces the jth bit (from left) of $1^m 0^{n-m}$. Obviously for each j all gates in block $(C_n^{1,j}, C_n^{2,j}, \ldots, C_n^{l_1,j})$ output 1 if $j \leq m$ and 0 otherwise. Therefore the outcome of the gates

$$(C_n^{1,1}, C_n^{2,1}, \ldots, C_n^{l_1,1}), (C_n^{1,2}, C_n^{2,2}, \ldots, C_n^{l_1,2}), \ldots, (C_n^{l_1,j}, C_n^{l_1,j}, \ldots, C_n^{l_1,j})$$

(in this order) is $1^{l_1 m} 0^{l_1(n-m)}$. A final UNARY-TO-BINARY yields the desired output. □

Corollary 7 Given D0L system G, the problem to decide upon input 1^n, whether there is some k such that $n = f_G(k)$ and, if so, to compute the least such k, is computable in $\mathcal{NC}^1$. If G is polynomial, the problem is even computable in $\mathcal{AC}^0$, depth 11.

4 The Membership Problem for D0L systems is in $\mathcal{NC}^1$

In this section, we first give the idea to the proof that the membership problem for polynomial D0L systems is contained in $\mathcal{AC}^0$. After this we proof that the membership problem for arbitrary, but fixed D0L systems is solvable in $\mathcal{NC}^1$, by showing that every language generated by an exponential D0L system is contained in $\mathcal{NC}^1$.

Let G be a conservative D0L system $G = (\Sigma, h, \alpha)$ as described in Lemma 4. First we introduce some notations. Let P be the predicate

$$P(k, i, a) \Longleftrightarrow \text{the } i\text{th letter of } h^k(\alpha) \text{ is } a.$$

Observe that

$$w = b_1 b_2 \dots b_n \in L_G \Longleftrightarrow \exists\, k \forall_{1 \leq i \leq n} P(k, i, b_i), \text{where } b_j \in \Sigma \text{ with } 1 \leq j \leq n.$$

The idea of the proof will be to determine the value of the predicate $P(k, i, b_i)$ very efficiently. In the polynomial case we make use of a special structure of the words generated by G (see [7]).

To determine the value of the predicate P, we make use of the structure of the words generating process. One can show by induction, that every position i in a word w generated by a polynomial D0L system G of rank[2] ρ can be described by a sequence $(g_1, p_1) \dots (g_s, p_s)$ with $s \leq \rho + 1$, where g_j, for $1 \leq j \leq s$, only depends on the D0L system G, and p_j, for $1 \leq j \leq s$, on the derivation length of w.

Now the crucial observation, which is stated without proof, is the following.

Fact 8 Let $G = (\Sigma, h, \alpha)$ be a D0L system of rank ρ. Then there exists a constant number of polynomials $\{q_1, \dots, q_t\}$ such that for every valid[3] sequence $(g_1, p_1) \dots (g_s, p_s)$ with $s \leq \rho + 1$ there is a polynomial q_j with $1 \leq j \leq t$ such that $q_j((g_1, p_1) \dots (g_s, p_s)) = i$, and a letter $a \in \Sigma$ such that the ith letter of $h^n(\alpha)$ is the letter a.

Knowing this, it is easy to design $\mathcal{AC}^0$ circuits for the predicate P, since the parameters of the polynomials involved are position numbers in a string of length at most n (hence they are small numbers) and all the computations can be performed in $\mathcal{AC}^0$ by Proposition 6. Note that the constructed circuit is of fixed depth. This lead us to:

Proposition 9 The membership problem for polynomial D0L systems is solvable in $\mathcal{AC}^0$ with depth 14. $\square$

Now let G be a fixed exponential D0L system $G = (\Sigma, h, \alpha)$. By the exponential growth the derivation length k is bounded by $c \cdot \log(f_G)$, where c is a constant only depending on the G. Therefore $w = b_1 b_2 \dots b_n \in L_G$ is equivalent to: There exists a $k \leq c \cdot \log n$ such that for every $1 \leq i \leq n$ holds $P(k, i, b_i)$.

We follow here the ideas of Sudborough [18]. The basic idea of his algorithm is, that if b_i is the appropriate letter on position i, then there exists a sequence of letters $a_0, a_1, \dots, a_k \in \Sigma$ such that a_0 is the i_0th letter in the initial string α, every a_j, with $1 \leq j \leq k$, is the i_jth letter in $h(a_{k-1})$, and a_k is equal to b_i.

Note that every i_j with $0 \leq j \leq k$ is bounded by the constant $m = \max_{a \in \Sigma}\{\mid h(a) \mid\}$. So the first and second conditions can be tested very easily once the a_j's and i_j's are chosen. Therefore we concentrate on the third condition — the test whether these sequences lead to the ith letter in w.

It is easy to see, that the global position number i is uniquely determined by the sequences $a_0, a_1, \dots, a_k$ and $i_0, i_1, \dots i_k$. This dependence can be described in a very

[2]The rank ρ corresponds to the degree of the growth function (see [7]).

[3]A sequence is called valid, if it corresponds to a position number i.

concise manner. To do this we introduce the following notation. Let $pref_i(v)$ be the prefix of length i of the word v. Using this, we get for the position i of a_k in the word $h^k(\alpha)$ the following formula[4]:

$$(\pi(pref_{i_0-1}(\alpha))(A_G)^k + \pi(pref_{i_1-1}(h(a_0)))(A_G)^{k-1} + \dots$$
$$\dots + \pi(pref_{i_{k-1}-1}(h(a_{k-2})))(A_G) + \pi(pref_{i_k-1}(h(a_{k-1}))))\eta^T + 1.$$

The first three conditions above are easy to test with the $\mathcal{DLOGTIME}$ circuit constructor. Since all parameters of the formula without k are constants, and k is bounded by $O(n)$ and therefore k is a small number, this computation can be performed in $\mathcal{NC}^1$ by Proposition 6.

We describe the circuit for the predicate P. First model with OR-gates the existential branch for the sequences $a_0, a_1, \dots, a_k$ and $i_0, i_1, \dots, i_k$ and then test with an AND-gate whether i is equal to the value of the formula above by using the above $\mathcal{NC}^1$ circuit and test that the ith letter is a_k. Now it is easy to design $\mathcal{NC}^1$ circuits for the membership problem for G.

Proposition 10 The membership problem for exponential D0L systems is in $\mathcal{NC}^1$. □

Theorem 11 The membership problem for D0L systems is solvable in $\mathcal{NC}^1$. □

5 Possible improvements

The higher the growth rate of a D0L system, the more sparse is its language. Thus exponentially growing D0L languages are more sparse than polynomially growing ones. BUT the upper bounds proved so far in this paper seem to contradict the feeling that the more sparse a language is the more easy is it to recognize it. Indeed our upper bound for polynomial D0L systems is $\mathcal{AC}^0$, fixed depth and for the exponential case, where the languages under consideration contain at most $O(\log n)$ words of length n we could not do better than $\mathcal{NC}^1$.

On the other hand there is no language generated by a D0L system that is complete for $\mathcal{NC}^1$ under $\mathcal{DLOGTIME}$ -reductions[5].Both observations indicate that $D0L$ might already be contained in some subclass of $\mathcal{NC}^1$. A candidate is $\mathcal{TC}^0$, the class of problems solvable by constant depth, polynomial size $\mathcal{DLOGTIME}$ -uniform[6] circuit families containing NOT- and MAJORITY-gates (see [9]). A MAJORITY-gate outputs 1 if more than halve of its inputs are 1.

The key point for any improvement seems to be to avoid matrix powering in the algorithm for the exponential case. But it turns out that with respect to $\mathcal{TC}^0$ -reductions (defined below) recognizing D0L languages is as hard as MATRIXPOWER-TO-SMALL-NUMBERS. So the answer to the above question is NO, unless this problem can be placed into $\mathcal{TC}^0$.

We use a reducibility notion similar to the $\mathcal{AC}^0$ -reductions of Chandra et al. [4].

[4]The growth function $f_G(n)$ can be expressed by $\pi(\alpha)(A_G)^n\eta^T$, where $\pi(\alpha)$ is the Parikh-vector of the initial word α, A_G is a matrix depending on the D0L system, and η is the all-one vector (see [16]).

[5]This because $D0L \subseteq$ *nonuniform*-$\mathcal{AC}^0$, but $PARITY \in \mathcal{NC}^1 \setminus$ *nonuniform*-$\mathcal{AC}^0$ ([8]).

[6]The direct connection language has to be defined appropriately – also multiple wires is have to be dealt with.

Definition 12 Problem L is said to be $\mathcal{TC}^0$-reducible to problems $L_1, L_2, \ldots, L_m$ if there is a $\mathcal{TC}^0$ circuit family with additional L_1-, L_2-, …, and L_m-oracle gates which solves L. A set $\mathcal{C}$ of problems is $\mathcal{TC}^0$-reducible to a set $\mathcal{C}'$ of problems if for any problem L in $\mathcal{C}$ there are problems $L_1, L_2, \ldots, L_m$ in $\mathcal{C}'$, such that L is $\mathcal{TC}^0$-reducible to $L_1, L_2, \ldots, L_m$. Two problems or sets of problems are said to be equivalent with respect to $\mathcal{TC}^0$-reductions if they are $\mathcal{TC}^0$-reducible to each other.

Theorem 13 The following classes of problems are equivalent with respect to $\mathcal{TC}^0$-reductions: $D0L$, POWER-TO-SMALL-NUMBERS, MATRIXPOWER-TO-SMALL-NUMBERS.

Hence $D0L \subseteq \mathcal{TC}^0$ iff POWER-TO-SMALL-NUMBERS $\subseteq \mathcal{TC}^0$ iff MATRIXPOWER-TO-SMALL-NUMBERS $\subseteq \mathcal{TC}^0$.

Sketch of Proof: Computing the sum of n numbers, polynomially bounded in n can be computed in $\mathcal{TC}^0$: compute the unary representation of the summands, concatenate these, and use the reduction of UNARY-COUNT to MAJORITY in [4] to compute the sum.

By this observation and the proof of Proposition 10 $D0L$ is $\mathcal{TC}^0$-reducible to MATRIXPOWER-TO-SMALL-NUMBERS. The latter in turn is $\mathcal{TC}^0$-reducible to POWER-TO-SMALL-NUMBERS (expand the entries of the mth power of a fixed matrix A into a sum of products of A's entries and rearrange these products into products of powers.)

For the reduction of POWER-TO-SMALL-NUMBERS to $D0L$ let a be an arbitrary positive integer and let $G = (\{1\}, \{1 \mapsto 1^a\}, 1)$. Let $\{1^l \in L_G \mid l \leq n\} = \{1, 1^a, 1^{a^2}, \ldots, 1^{a^r}\}$. From the following two facts one can easily construct $\mathcal{TC}^0$ circuits with L_G-gates that compute the mth power of a, when 1^m0^{n-m} is given as input :

- $a^m \leq n$ iff $m \leq r$ iff $m + 1 \leq \#\{1^l \in L_G \mid l \leq n\}$. This can be easily tested by $\mathcal{TC}^0$-circuits with L_G-gates.

- If $a^m \leq n$, then a^m equals the length of the $(m+1)$st string in the sequence $1, 1^a, 1^{a^2}, \ldots, 1^{a^r}$ that belongs to L_G.

Remembering that polynomial D0L systems are in $\mathcal{AC}^0$ which is contained in $\mathcal{TC}^0$ completes the proof. □

Corollary 14 $D0L$ is $\mathcal{TC}^0$-equivalent to $\mathcal{TALLY}$-$D0L = \{L \in D0L \text{ and } L \subseteq \{1\}^*\}$. □

6 The General Membership Problem for D0L systems is in $\mathcal{DET}$

In the previous sections we constructed algorithms which solved the membership problem for an arbitrary, but fixed D0L system. If we regard G as a part of the input we come to a *general* membership problem (see [13]). In the following we will use an appropriate coding $\langle G \rangle \in \Sigma^\star$ of a system G for some fixed alphabet Σ without giving any details.

Definition 15 The *general membership problem* for D0L systems is the set

$$GM_{D0L} := \{\langle G, w \rangle \mid G = (\Sigma, h, \alpha) \text{ is a D0L system, } w \in \Sigma^\star \text{ and } w \in L_G\}.$$

Jones and Skyum provided upper and lower bounds for this problem.

Theorem 16 ([13, 14]) $GM_{D0L} \in \mathcal{P} \cap DSPACE(log^2 n)$ and it is $NSPACE(\log n)$-hard.

Theorem 16 says that there exists both a polynomial time-bounded algorithm and one which uses $O(\log^2 n)$ space. We should not look for an algorithm which obeys both resource bounds simultaneously, i.e. an algorithm showing membership in $\mathcal{SC}^2$, since the second part would then give the unexpected inclusion $NSPACE(\log n) \subseteq \mathcal{SC}^2$ (see [6]).

We improve the upper bound of Theorem 16 by showing that GM_{D0L} is contained in the subclass $\mathcal{DET}$ of $\mathcal{P} \cap DSPACE(\log^2 n)$. $\mathcal{DET}$ was originally defined by Cook in [5] — it is the class of all problems $\mathcal{NC}^1$ -reducible to the computation of integer determinants.

Theorem 17 $GM_{D0L} \in \mathcal{DET}$.

Idea of proof : The gist of the construction is to use the algorithm of Jones and Skyum in [14]. They distinguish the two cases whether the given D0L system is infinite or finite[7]. It is not to hard to show, that their procedure to treat the infinite case, is computable by a $\mathcal{DET}$ function. A careful analysis shows, that the constructions given in [14, 19] to settle the finite case, can be replaced by $\mathcal{DET}$ computable functions. The details cannot be given here for lack of space. □

Very essential for the $NSPACE(\log n)$-hardness in [14] was the use of erasing productions in the constructing D0L system. In fact, we were not able to show this for D0L systems without erasing, i.e. PD0L systems[8]. Nevertheless, we get $DSPACE(\log n)$-hardness, which we state without proof.

Theorem 18 GM_{PD0L} is $DSPACE(\log n)$-hard with respect to $\mathcal{AC}^0$ -reductions[9].

This leaves open the possibility that GM_{PD0L} might be contained in $\mathcal{SC}^2$. Theorems 16 and 18 express that general membership problems are provable more difficult than fixed ones for (P)D0L systems, since the latter are contained in $\mathcal{ALOGTIME}$*-uniform-*$\mathcal{AC}^0$ which is a proper subset of $DSPACE(\log n)$ (see [8]).

Acknowledgments We would like to thank the following people for many hints and helpful discussions: Birgit Jenner, Bernd Kreußler, Peter Rossmanith, Wojciech Rytter, and Thomas Zeugmann.

References

[1] D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within $\mathcal{NC}^1$. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[2] R. V. Book. Sparse sets, tally sets, and polynomial reducibilities. In *Proc.of 13th MFCS*, volume 324 of *LNCS*, pages 1–13. Springer, 1988.

[7]The infinity problem is well known to be $\mathcal{NP}$-complete – but with respect to ED0L languages. For D0L systems this question can easily be seen to be $NSPACE(\log n)$-complete. Hence the question can be decided by a $\mathcal{DET}$ procedure.

[8]The letter P stands for *propagating*; see e.g. [16]

[9]In fact, it is possible to use many-one $\mathcal{DLOGTIME}$ -reducibilites !

[3] S. R. Buss. The boolean formula value problem is in ALOGTIME. In *Proc. 19th Ann. ACM Symp. on Theory of Computing*, pages 123–131, 1987.

[4] A. K. Chandra, L. Stockmeyer, and U. Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, May 1984.

[5] S. A. Cook. Towards a complexity theory of synchronous parallel computation. *L'Enseignement Mathématique*, XXVII(1–2):75–100, 1981.

[6] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1–3):2–22, 1985.

[7] A. Ehrenfeucht and G. Rozenberg. D0L system with rank. In *L Systems*, volume 15 of *LNCS*, pages 136–141. Springer, 1974.

[8] M. Furst, J. B. Saxe, and M. Sipser. Parity circuits and the polynomial-time hierarchy. In *Proc. 22th IEEE Symp. FOCS*, pages 260–270, 1981.

[9] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turan. Threshold circuits of bounded detph. In *Proc. 28th IEEE Symp. FOCS*, pages 99–110, 1987.

[10] J. Hartmanis. Some observations about $\mathcal{NP}$ complete sets. In *Proc. of 6th FCT*, volume 278 of *LNCS*, pages 185–196. Springer, 1987.

[11] B. Jenner and B. Kirsig. *Alternierung und Logarithmischer Platz.* Dissertation (in German), Universität Hamburg, 1989.

[12] N. D. Jones and S. Skyum. Recognition of deterministic ET0L languages in logarithmic space. *Information and Computation*, 35:177–181, 1977.

[13] N. D. Jones and S. Skyum. Complexity of some problems concerning L systems. *Math. Systems Theory*, 13:29–43, 1979.

[14] N. D. Jones and S. Skyum. A note on the complexity of general D0L membership. *SIAM Journal on Computing*, 10(1):114–117, February 1981.

[15] Jan Van Leeuwen. The membership problem for ET0L-languages is polynomially complete. *Information Processing Letters*, 3(5):138–143, May 1975.

[16] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*, volume 90. Academic Press, 1980.

[17] W. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–338, 1981.

[18] I. Sudborough. The time and tape complexity of developmental languages. In *Proc. of 4th ICALP*, volume 52 of *LNCS*, pages 509–521. Springer, 1977.

[19] P. M. B. Vintányi. *Lindenmayer Systems: Structure, Languages, and Growth Functions.* PhD thesis, University of Amsterdam, 1978.