The Power of Local Self-Reductions

Richard Beigel* Howard Straubing†
Yale University Boston College

Abstract

Identify a string x over $\{0,1\}$ with the positive integer whose binary representation is $1x$. We say that a self-reduction is k-local if on input x all queries belong to $\{x-1,\ldots,x-k\}$. We show that all k-locally self-reducible sets belong to PSPACE. However, the power of k-local self-reductions changes drastically between $k=2$ and $k=3$. Although all 2-locally self-reducible sets belong to $\mathrm{MOD}_6\mathrm{PH}$, some 3-locally self-reducible sets are PSPACE-complete. Furthermore, there exists a 6-locally self-reducible PSPACE-complete set whose self-reduction is an m-reduction (in fact, a permutation).

We prove all these results by showing that such languages are equivalent in complexity to the problem of multiplying an exponentially long sequence of uniformly generated elements in a finite monoid, and then exploiting the algebraic structure of the monoid.

1. Introduction

In this paper we identify a string x over $\{0,1\}$ with the positive integer whose binary representation is $1x$. Thus we do not distinguish between the sets $\{0,1\}^*$ and Z^+. Balcazar [1] introduced lexicographical self-reductions, also called wdq-self-reductions, which on input x query only strings that are less than x. Lexicographical self-reductions are an important tool in unifying certain connections between uniform and nonuniform complexity [1]. They are also important in the study of which complexity classes may have sparse complete sets [13].

Goldsmith, Joseph, and Young [9] (independent of Balcazar) introduced near testability, and subsequently Goldsmith, Joseph, Hemachandra, and Young [8] introduced near near-testability. Both notions are special cases of lexicographical self-reductions in which the queried string (if any) is always the immediate predecessor of the input string.

The complexity of lexicographically self-reducible sets is well understood: all of them belong to EXP and some of them are $\leq_m^p$-complete for EXP [1]. The complexity of near-testable sets is also well understood: all of them belong to PARITYP and some of them are $\leq_m^p$-complete for PARITYP [8]. So is the complexity of nearly near-testable sets: all of them belong to $\mathrm{PF}^{\mathrm{NP}} \circ$ PARITYP and some of them are $\leq_m^p$-complete for $\mathrm{PF}^{\mathrm{NP}} \circ$ PARITYP [10].

In order to better understand lexicographical self-reductions, we ask what happens when a self-reduction is allowed to look only at the k immediately preceding strings for some constant k.

Definition 1. $A \subseteq \mathsf{Z}^+$ is k-locally self-reducible if there if a polynomial time-bounded deterministic oracle Turing machine M such that

- M^A recognizes A, and
- On input x, M^A queries only elements of $\{x-1,\ldots,x-k\}$.

Remark: The 1-locally self-reducible sets are the same as the nearly near-testable sets.

We would like to define a many-one version of this notion of self-reducibility. Since no queries can be made on the empty string, we will allow our many-one reductions to be undefined on finitely many inputs.

Definition 2.

- A is *m-reducible* to B if there is a polynomial-time computable partial function $f : \mathsf{Z}^+ \to \mathsf{Z}^+$ such that for all but finitely many x we have $x \in A \iff f(x) \in B$.

*Yale University, Dept. of Computer Science, P.O. Box 208285, New Haven, CT 06520-8285, USA. Email: `beigel-richard@cs.yale.edu`. Research supported in part by the U.S.A. National Science Foundation under grant CCR-8958528 and by the Netherlands Organization for Scientific Research (NWO) under Visitors Grant B 62-403.

†Computer Science Dept., Boston College, Chestnut Hill, MA 02167, USA. Email: `straubin@bcuxs1.bc.edu`. Research supported in part by the U.S.A. National Science Foundation under grant CCR-9203208.

- If the partial function f above is 1–1 and onto, then A is *permutation-reducible* to B.

Definition 3. A is m-k-locally self-reducible if there is an m-reduction f from A to A such that $x-k \leq f(x) < x$ for all $x \in \mathsf{Z}^+$ for which $f(x)$ is defined.

Obviously, if A is m-k-locally self-reducible, then it is k-locally self-reducible.

Here are our main results:

- All k-locally self-reducible sets belong to PSPACE.
- All 2-locally self-reducible sets belong to MOD_6PH. (MOD_6PH is a generalization of the polynomial hierarchy where we allow a bounded number of MOD_6 quantifiers interspersed with the usual existential and universal quantifiers. It is an exponential analogue of the circuit complexity class ACC(6).) In fact, all 2-locally self-reducible sets belong to a fixed level of this hierarchy.
- There exists a 3-locally self-reducible PSPACE-complete set.
- There exists an m-6-locally self-reducible PSPACE-complete set whose self-reduction is in fact a permutation reduction. (The reader may be surprised that there is a self-reducible PSPACE-complete set whose self-reduction is even an m-reduction. The key to the reduction is determining which question to ask, rather than what to do with the answer.)

We will also give a new proof of Hemachandra and Hoene's characterization of the nearly near-testable sets.

We prove these results by reducing questions about the Turing machine complexity of locally self-reducible sets to questions about the circuit complexity of multiplication in finite monoids, and then applying results of Barrington, Immerman, Straubing, and Thérien on circuits and monoids [3, 5, 4]. This algebraic approach to circuit complexity is proving to be a valuable tool in the study of Turing machine-based complexity classes; see, for example, [7, 11].

2. A Connection to Algebra

Let A be k-locally self-reducible, so there is a polynomial-time algorithm $\mathcal{A}$ that takes $x+1$ and $\chi_A(x-k+1), \ldots, \chi_A(x)$ as input and determines $\chi_A(x+1)$. This gives a polynomial-time algorithm $\mathcal{A}'$ that takes $x+1$ and $\chi_A(x-k+1), \ldots, \chi_A(x)$ as input and determines $\chi_A(x-k+2), \ldots, \chi_A(x+1)$.

Thus $x+1$ determines a mapping $\mathcal{A}^*(x+1)$ from $\{0,1\}^k$ into itself; the mapping is computable in polynomial time from the input $x+1$ by running the algorithm $\mathcal{A}'$ on all elements of $\{0,1\}^k$ in succession. So to determine $\chi_A(x+1)$ it suffices to compute

$$(\chi_A(1), \ldots, \chi_A(k))\mathcal{A}^*(k+1)\mathcal{A}^*(k+2)\cdots\mathcal{A}^*(x+1).$$

(Here we compose finite functions from left to right.) This expression can be evaluated left to right using a constant amount of space to store the k-tuple, linear space to store the input to $\mathcal{A}^*$, and polynomial space to compute each mapping $\mathcal{A}^*(i)$. We have thus proved:

Theorem 4. *If A is k-locally self-reducible then A is in* PSPACE.

We want to look at this proof in a somewhat different way, which will be more useful for the purposes of this paper. The set of maps from $\{0,1\}^k$ into itself forms a monoid (that is, a set with an associative product and an identity element) with composition as the product. It is easy to show that the problem of multiplying n elements of a fixed finite monoid is in NC^1 [3]. Thus the multiplication of exponentially many uniformly generated elements of a finite monoid can be carried out by uniform AND-OR circuits of linear depth. Our result then follows from the fact that uniform circuits of polynomial depth can be evaluated in polynomial space.

Let us be a bit more precise. If Q is a finite set, then the set of maps from Q into itself forms a finite monoid with composition as the operation. Actually there are two monoids we can define this way, one in which we compose maps from right to left, the other in which we compose maps from left to right. In what follows we will assume, as in the above discussion, that maps are composed from left to right, so that fg means "apply first f, then g." We thus also write the image of an element of Q under a map f as qf rather than $f(q)$. We use the term *transformation monoid* to mean any monoid of maps on a finite set, in which the identity element is the identity map.

Let $f : \{0,1\}^k \rightarrow \{0,1\}$. We define

$$F_f : \{0,1\}^k \rightarrow \{0,1\}^k$$

by

$$F_f(x_1, \ldots, x_k) = (x_2, \ldots, x_k, f(x_1, \ldots, x_k)),$$

where $f : \{0,1\}^k \rightarrow \{0,1\}$ is a map. Let us denote the monoid generated by the transformations F_f by M_k. That is, M_k consists of all possible products of finite sequences of the F_f, together with the identity transformation. We argued above that determining membership in a k-locally self-reducible set reduces to evaluation of products in M_k. In the next two sections we will analyze the structure of M_k for various values of k to obtain more detailed information about k-locally self-reducible sets.

3. PSPACE-Complete Locally Self-Reducible Sets

We will first obtain matching lower bounds to Theorem 4. Our proof depends on a result of Barrington, showing, in essence, that NC^1 is as easy as multiplication in finite monoids. Let S_5 denote the symmetric group of degree 5; that is, the group of all permutations of the set $\{1,2,3,4,5\}$. Let b, c be distinct elements of S_5, and let $L \in \mathrm{NC}^1$. Barrington [3] showed that for each $n > 0$ there is a function $f_n : \mathsf{Z}^+ \to \mathsf{Z}^+$, and for each $x \in \{0,1\}^*$ a sequence $\alpha(x)$ of elements of S_5, such that:

- The length of $\alpha(x)$ depends only on $|x|$, and is bounded by a polynomial in the size of the circuit for inputs of length $|x|$.
- The i^{th} element of $\alpha(x)$ depends only on the $f_{|x|}(i)^{th}$ bit of x.
- If $x \in L$, then $\prod \alpha(x)$, the product in S_5 of the elements of $\alpha(x)$, is equal to b. If $x \notin L$, then $\prod \alpha(x) = c$.

Barrington in his original paper, and later Barrington, Immerman and Straubing [5] studied the complexity of this circuit simulation for uniform NC^1. For our purposes, we only need the following facts, easily derivable from Barrington's original construction: Suppose we have a fan-in 2 circuit whose underlying graph is the full binary tree of depth d. Thus there are $2^d - 1$ leaves, each interior node is either an AND gate or an OR gate, and each leaf node is labeled by an input bit or its negation. (Typically the same input bit appears as the label of many different leaves.) We can determine the j^{th} element of the string $\alpha(x)$ in $d^{O(1)}$ time steps, assuming that in a single step we can do each of the following:

- Determine from the $d+1$-bit encoding of a node whether it is an AND gate, an OR gate or a leaf.
- Determine from i, j whether the i^{th} leaf is labeled by the j^{th} input bit, or whether it is labeled by the negation of the j^{th} input bit.

We will also need the following fact about the connection between space used by a Turing machine and circuit depth: Every language in PSPACE is recognized by a uniform polynomial-depth family of circuits [6, 2]. In this context, "uniform" means that we can determine in polynomial time from the encoding of a gate whether it is an AND gate, an OR gate, or a leaf, and, if it is a leaf, which bit of the input it queries.

Lemma 5. *Let K be a set that generates S_5 and includes both an odd and an even permutation. Let T be a nonempty proper subset of $\{1,2,3,4,5\}$. There is a polynomial-time computable function $\phi : \mathsf{Z}^+ \to K$ such that the set*

$$S_\phi = \{x : 1\phi(1)\cdots\phi(x) \in T\}$$

is PSPACE-complete.

Proof: Let L be any language in PSPACE. Let $k \in T$, $k' \notin T$, and let b be a permutation that maps 1 to k, and c a permutation that maps 1 to k'. By the preceding remarks, for each $x \in \{0,1\}^*$, there is a sequence $\alpha(x)$ of elements of S_5 such that the length of $\alpha(x)$ depends only on $|x|$ and is exponential in $|x|$, $\prod \alpha(x)$ is b if $x \in L$ and c otherwise, and the map that sends the pair (i, x) to the i^{th} element of $\alpha(x)$ is polynomial-time computable.

We can strengthen this slightly and suppose that the elements of the sequence $\alpha(x)$ all belong to the set K. To show this, we must argue that every element of S_5 can be written as a product of a fixed number $r > 0$ of elements of K. Let H be the set of elements of S_5 that can be written as a product of a sequence of generators of length divisible by $120 = |S_5|$. For any sequences β and γ of generators, H contains the product of the sequence $\beta\gamma\beta^{119}\gamma^{119}$. Thus H contains all the commutators in S_5, and since H is obviously closed under multiplication, H contains the commutator subgroup A_5, which consists of all the even permutations. Since H contains at least one odd permutation as well, $H = S_5$. We now have each element of S_5 expressed as a product of a sequence of length divisible by 120. Let $g \in K$; we can pad each of the sequences by a sufficient number of copies of g^{120} so that they all have the same length.

Now consider the infinite sequence

$$\alpha(1)^{120}\alpha(2)^{120}\cdots$$

of elements of K. (Note that in writing the argument of α as an integer we are using the identification between $\{0,1\}^*$ and Z^+.) Let $\phi(i)$ denote the i^{th} element of this sequence. Observe that ϕ is polynomial-time computable.

We have

$$\prod \alpha(x) = \prod [\alpha(1)^{120}\cdots\alpha(x-1)^{120}\alpha(x)] = \prod_{i=1}^{z(x)} \phi(i).$$

Here $z(x)$ is the length of the concatenated sequence

$$\alpha(1)^{120}\cdots\alpha(x-1)^{120}\alpha(x);$$

$z(x)$ is polynomial-time computable. It follows that $x \in L$ if and only if $z(x) \in S_\phi$, so that S_ϕ is PSPACE-hard. Since ϕ is computable in polynomial time, our remarks in the preceding section show that S_ϕ is in PSPACE, so S_ϕ is PSPACE-complete. ▌

Theorem 6. *There exists an m-6-locally self-reducible* PSPACE-*complete language whose self-reduction is a permutation reduction.*

Proof: We define a self-reducible set S recursively. Let K consist of the transpositions (1 2), (2 3), (3 4), and (4 5), together with the identity of S_5. K generates S_5 and contains odd permutations (the transpositions) and an even permutation (the identity). Thus Lemma 5 can be applied; let $\phi : \mathsf{Z}^+ \to K$ be as in the lemma. Let

- $1 \in S$
- $\{2,3,4,5\} \subseteq \overline{S}$
- if $q \geq 1$ and $1 \leq r \leq 5$ then, $5q + r \in S$ iff $5(q-1) + r\phi(q) \in S$

Then $x \in S_\phi$ iff $5x + 1 \in S$, so by Lemma 5, S is PSPACE-complete. Clearly, S has a 6-local self-reduction that is 1–1 and onto from $\mathsf{Z}^+ - \{1,2,3,4,5\}$ to Z^+. ▌

Theorem 7. *There exists a 3-locally self-reducible language that is PSPACE-complete.*

Proof: We first take a closer look at the structure of the monoid M_3. We define functions $f, g : \{0,1\}^3 \to \{0,1\}$ by

$$f(0,0,0) = f(0,0,1) = f(0,1,0) = f(1,1,1) = 1.$$

$$f(0,1,1) = f(1,0,0) = f(1,0,1) = f(1,1,0) = 0.$$

$$g(0,0,1) = g(0,1,0) = g(1,0,0) = g(1,1,1) = 1.$$

$$g(0,0,0) = g(0,1,1) = g(1,0,1) = g(1,1,0) = 0.$$

Let us identify the triples $(0,0,0)$, $(0,0,1)$, $(0,1,0)$, $(0,1,1)$, $(1,0,0)$, $(1,0,1)$, $(1,1,0)$, $(1,1,1)$ with $2,1,6,3,4,7,5,8$, respectively. With these identifications, F_f is the permutation (2 1 3 5 4)(6 7) and F_g is the permutation (1 3 5 4)(6 7). Note that F_f and F_g both map $\{1,2,3,4,5\}$ into itself. In fact, the permutations (2 1 3 5 4) and (1 3 5 4) generate S_5, because

$$(2\ 1\ 3\ 5\ 4)^2(1\ 3\ 5\ 4)(2\ 1\ 3\ 5\ 4)^2 = (3\ 5),$$

and S_5 is generated by any pair consisting of a 5-cycle and a transposition.

Since we have both an odd and an even permutation, we can apply Lemma 5, with $T = \{1,3\}$, and obtain both a polynomial-time computable map ϕ and a PSPACE-complete set S_ϕ. We define a set S recursively by $1 \notin S$, $2 \notin S$, $3 \in S$, and, for $x > 3$, $x \in S$ if and only if $(\chi_S(x-3), \chi_S(x-2), \chi_S(x-1))\phi(x-3) = (\chi_S(x-2), \chi_S(x-1), 1)$.

Since ϕ is polynomial-time computable, S is 3-locally self-reducible. For $x > 3$, we have $x \in S$ if and only if the third component of $(0,0,1)\phi(1)\cdots\phi(x-3)$ is 1. Under the identification of triples and integers, this happens if and only if $1\phi(1)\cdots\phi(x-3) \in T$, since this product of permutations always maps 1 into $\{1,2,3,4,5\}$. Thus for any positive integer y, $y \in S_\phi$ if and only if $y + 3 \in S$. It follows from Lemma 5 that S is PSPACE-complete. ▌

4. An Upper Bound for 2-Locally Self-Reducible Sets

In this section we will give an upper bound on the complexity of 2-locally self-reducible sets. This bound suggests that such sets are not PSPACE-complete, unless certain hierarchies contained within PSPACE collapse to a fixed level.

Let us review a few complexity-theoretic notions: If $\mathcal{C}$ is a class of languages and then we denote by $\mathrm{MOD}_m \cdot \mathcal{C}$ the class of languages L for which there is a polynomial p and a language $L' \in \mathcal{C}$ such that for each x,

$$|\{y : y \in \{0,1\}^{p(|x|)}, xy \in L'\}| \equiv 0 \pmod{m}.$$

We denote by $\forall \cdot \mathcal{C}$ (respectively, $\exists \cdot \mathcal{C}$) the class of languages L for which there is a polynomial p and a language $L' \in \mathcal{C}$ such that $x \in L$ if and only if for every (respectively, for some) $y \in \{0,1\}^{p(|x|)}$, $xy \in L'$. We can define hierarchies of complexity classes by beginning with the class P and successively applying the operators $\mathrm{MOD}_m\cdot$, $\exists\cdot$, and $\forall\cdot$. If we do not use the modular operators, we get the usual polynomial time hierarchy. If we allow the modular operators for a fixed modulus m, we obtain a hierarchy of complexity classes that we denote by MOD_mPH. That is, MOD_mPH is the union of all the classes

$$Q_1 \cdot Q_2 \cdots Q_t \cdot \mathrm{P},$$

where each Q_i is one of the operators $\forall$, $\exists$, or MOD_m.

The class MOD_2P is commonly denoted $\oplus$P and PARITYP in the literature.

There is a simple connection to circuit complexity: Suppose we have a language L that is polynomial-time reducible to evaluating a uniform, constant-depth, exponential-size family of circuits with unbounded fan-in AND, OR and MOD_m gates. Each gate and input node can be encoded by a bit string whose length is polynomial in the input size. "Uniform" in this context means that we can determine in polynomial time from the encoding of a gate what type of gate it is, from x and the encoding of an input node the bit at that input node, and from the encodings of two gates g_1 and g_2 whether g_1 is a child of g_2. Then L is in $\mathrm{MOD}_m\mathrm{PH}$. At the cost of an increase in the depth by a constant factor, we may assume that all the gates on a given level are of the same type; in this case, we can write down explicitly a level in $\mathrm{MOD}_m\mathrm{PH}$ that L belongs to. For example, if L is reducible in this way to a depth-3 family of circuits with a MOD_3 gate at the output, AND gates at the next level, and OR gates at the input level, then $L \in MOD_3 \cdot \forall \cdot \exists \cdot \mathrm{P}$.

A monoid typically contains many subsets that are closed under multiplication in the monoid, and are thus *subsemigroups* of the monoid. When the subsemigroup is a group, we call it a *group in the monoid.* (We avoid the term "subgroup" in this context, since this is often taken to mean that the identity of the group coincides with that of the monoid.)

Lemma 8 (Folklore). *Let M be a transformation monoid on a finite set Q. Every group in M is isomorphic to a permutation group on $|Q|$ elements.*

Proof: Let G be a group in M. Let e be the identity of G. If $g \in G$ then $Qg = (Qg)e$ and thus the image of g is contained in the image of e. Conversely, $Qe = (Qg^{-1})g$, and thus the image of e is contained in that of g. So all elements of G have the same image I. For any $g \in G$, $Ig = (Qe)g = Qg = I$, so every element of G permutes I. If $g, h \in G$ induce the same permutation on I, then for all $q \in Q$, $qg = (qe)g = (qe)h = qh$ and thus $g = h$. Thus G is isomorphic to a group of permutations of $I \subseteq Q$, which can be embedded in the group of permutations on $|Q|$ elements. ∎

We say that a finite monoid M is *solvable* if every group in M is solvable. There is more to this terminology than meets the eye: Every monoid M admits a certain kind of decomposition (the Krohn–Rhodes decomposition) in which the factors are simple composition factors of the groups contained in M. When M is a solvable monoid, all of these composition factors are cyclic groups of prime order. Thus solvable monoids are in a precise sense a semigroup-theoretic analogue of solvable groups. Barrington and Thérien [4] show that in this case we can compute the product of a sequence of elements of M in $\mathrm{ACC}(r)$—that is, with a constant-depth, polynomial-size (in the length of the sequence) uniform family of unbounded fan-in circuits with AND, OR and MOD_r gates—where r is the product of the distinct prime divisors of the cardinalities of the groups in M. In fact it is possible, although somewhat cumbersome, to write down the depth of the circuit family and the gates that occur on each level directly from a Krohn-Rhodes decomposition of M.

Proposition 9. *M_2 is solvable, and the prime divisors of the orders of the groups in M_2 are 2 and 3.*

Proof: By Lemma 8, every group in M_2 is isomorphic to a subgroup of S_4, the symmetric group on four letters. Thus M_2 is solvable, and the only primes that divide the order of a group in M_2 are 2 and 3. We need only show that M_2 contains both a group of order 2 and a group of order 3.

Let $f(0,1) = 0$, $f(1,0) = 1$, $f(0,0) = 0$, and $f(1,1) = 1$. Then F_f transposes $(1,0)$ and $(0,1)$ and leaves $(0,0)$ and $(1,1)$ fixed,thus generating a group of order 2 in M_2.

Let $g(0,0) = 1$, $g(0,1) = 0$, $g(1,0) = 0$, and $g(1,1) = 1$. Then F_g cycles $(0,0)$, $(0,1)$, and $(1,0)$, and fixes $(1,1)$. Thus F_g generates a group of order 3 in M_2. ∎

As an easy consequence we obtain:

Theorem 10. *Every 2-locally self-reducible set belongs to a fixed level of the* $\mathrm{MOD}_6\mathrm{PH}$ *hierarchy.*

Proof: Let A be 2-locally self-reducible. As remarked in Section 2, determining if $x \in A$ reduces to evaluating a uniformly-generated exponentially long product in M_2. By the preceding proposition and the result of Barrington and Thérien cited above, this can be carried out by a particular uniform exponential-size ACC(6) circuit family. Thus A belongs to a particular level of the $\mathrm{MOD}_6\mathrm{PH}$ hierarchy. ∎

We will not try to explicitly determine the lowest level of $\mathrm{MOD}_6\mathrm{PH}$ that contains all 2-locally self-reducible sets. However, we can fairly easily obtain an explicit upper bound by noting that the monoid of all transformations on a 4-element set has a Krohn-Rhodes decomposition as a wreath product

$$U \circ C_2 \circ U \circ C_3 \circ C_2 \circ U \circ C_2 \circ C_2 \circ C_3 \circ C_2,$$

where C_m denotes the cyclic group of order m, and U denotes a group-free finite monoid. This leads to a circuit with seven levels of MOD_2 and MOD_3 gates, and a similar number of levels of AND and OR gates to simulate the monoids U and to glue everything together. We could get a lower level of the hierarchy by using a decomposition of the smaller monoid M_2, although we have no reason to believe that this upper bound is the best possible.

Our result shows that we cannot reduce the 3-locally self-reducible sets to the 2-locally self-reducible sets, unless PSPACE collapses to this fixed level of $\mathrm{MOD}_6\mathrm{PH}$. This would happen if there were a uniform way to compute products in S_5 using ACC(6) circuits, but this is considered unlikely. A theorem of Smolensky [14] shows that for polynomial-size circuit families, ACC(6) is different from both ACC(3) and ACC(2). The analogous separation for uniform exponential-size families would show that the 2-locally self-reducible sets are not in either $\mathrm{MOD}_2\mathrm{PH}$ or $\mathrm{MOD}_3\mathrm{PH}$.

5. 1-Locally Self-Reducible Sets

Hemachandra and Hoene [10] isolated the complexity of the 1-locally self-reducible sets by showing that they belong to $\oplus$OptP and that some of them are $\oplus$OptP-hard. This result (and the related result of [8]) falls out of our algebraic framework in a nearly mechanical way.

We first recall a few definitions. OptP denotes the class of functions f from $\{0,1\}^*$ to itself for which there exist a polynomial p and a language $L \in \mathrm{P}$ such that for all $x \in \{0,1\}^*$,

$$f(x) = \max\{y \in \{0,1\}^{p(|x|)} : xy \in L\},$$

if such a y exists, and $f(x) = 0^{p(|x|)}$ otherwise. $\oplus$OptP is the class of all languages L for which there exist a polynomial p, a language $L' \in \mathrm{P}$, and a function $f \in$ OptP such that for all x, $x \in L$ if and only if

$$|\{y \in \{0,1\}^{p(|x|)} : xf(x)y \in L'\}| \equiv 0 \pmod 2).$$

Hemachandra and Hoene show that $\oplus$OptP is identical to $\mathrm{PF}^{\mathrm{NP}} \circ \mathrm{PARITYP}$.

Theorem 11 (Hemachandra–Hoene). *Every 1-locally self-reducible language is in* $\oplus$OptP, *and every language in* $\oplus$OptP *is* $\leq_m^P$*-reducible to a 1-locally self-reducible language.*

Proof: Both parts of the proof are based on the structure of the monoid M_1, which is just the monoid of all transformations on the set $\{0,1\}$. Observe that M_1 has four elements: the identity, the transposition, the constant map to 0, and the constant map to 1. We denote these elements by $1, \tau, c_0$, and c_1, respectively.

For the first part of the proof, let A be 1-locally self-reducible. Since $\oplus$OptP is closed under finite variation, we may assume $1 \in A$. There is then a polynomial-time computable map α from the positive integers to M_1 such that $x \in A$ if and only if the product $\alpha(2)\alpha(3)\cdots\alpha(x)$ fixes 1; that is, if and only if this product is equal to 1 or c_1. We define a map ζ from strings of odd length to M_1 as follows: If $|x|+1 = |y|$ then $\zeta(xy) = 1$ if $y = 0^{|x|+1}$, $y = 0^{|x|}1$ or $1x < y$ in the lexicographic order. (Remember that we view x alternatively as a string, or as the integer whose binary representation is $1x$.) In all other cases $\zeta(xy) = \alpha(u)$, where u is the integer whose binary representation is y. ζ is clearly polynomial time computable, and $x \in A$ if and only if

$$\zeta(x0^{|x|+1}) \cdot \zeta(x0^{|x|}1) \cdots \zeta(x1^{|x|+1}) \in \{1, c_1\}.$$

Let $f(x)$ be the largest y in the lexicographic order, with $|y| = |x|+1$, such that $\zeta(xy)$ is a constant map, if such a y exists; $f(x) = 0^{|x|+1}$ otherwise. Clearly $f \in$ OptP. Observe that if there is no y with $|y| = |x|+1$ such that $\zeta(xy)$ is a constant map, then $f(x) = 0^{|x|+1}$, and $x \in A$ if and only if an even number of elements of the sequence

$$\zeta(x0^{|x|+1}), \ldots, \zeta(x1^{|x|+1})$$

are equal to τ. On the other hand, if $\zeta(xf(x)) = c_1$ (respectively, c_0), then $x \in A$ if and only if an even (respectively, odd) number of elements $\zeta(xy)$, with $|y| = |x|+1$ and $y > f(x)$ are equal to τ.

We define for $|z| = |y| = |x|+1$,

$$\theta(xzy) = \begin{cases} 1 & \text{if } y < z \\ 1 & \text{if } y = z \text{ and } \zeta(xz) \in \{1, c_1\} \\ \tau & \text{if } y = z \text{ and } \zeta(xz) \in \{\tau, c_0\} \\ \zeta(xy) & \text{if } y > z \end{cases}$$

It follows now that $x \in A$ if and only if there is an even number of y with $y = |x|+1$ such that $\theta(xf(x)y) = \tau$. Let us define A' by $xzy \in A'$ if and only if $|z| = |y| = |x|+1$ and $\theta(xzy) = \tau$. Since $A' \in \mathrm{P}$, $A \in \oplus$OptP, as claimed.

For the converse, suppose $A \in \oplus$OptP. Then there exist $A' \in \mathrm{P}$, $f \in$ OptP, and a polynomial p such that $x \in A$ if and only if $xf(x)y \in A'$ for an even number of $y \in \{0,1\}^{p(|x|)}$. Since $f \in$ OptP there exist a polynomial q and a set $T \in \mathrm{P}$ such that $f(x)$ is the lexicographically largest y of length $q(|x|)$ such that $xy \in T$, if such a y exists, and $f(x) = 0^{q(|x|)}$ otherwise. We may assume that p and q are strictly increasing functions.

We now define a function from strings whose length is an integer of the form $n + p(n) + q(n)$, where $n > 0$, to elements of M_1 : Let x, y, z be strings with $|y| = p(|x|)$, $|z| = q(|x|)$. Let z' denote the successor of z in the lexicographic order on strings of length $q(|x|)$; z' is not defined if $z = 1^{q(|x|)}$. If $xz' \in T$ then we define $\theta(xzy) = c_1$. Otherwise we set $\theta(xzy) = 1$ if $xzy \notin A'$ and either $xz \in T$ or $z = 0^{q(|x|)}$, $\theta(xzy) = \tau$ if $xzy \in A'$ and either $xz \in T$ or $z = 0^{q(|x|)}$, and $\theta(xzy) = 1$ in all other cases. Consider the sequence

$$\theta(x0^{p(|x|)+q(|x|)}), \ldots, \theta(x1^{p(|x|)+q(|x|)}).$$

It follows from the definitions made above that the product of this sequence in M_1 is equal to 1 or c_1 if and only if $x \in A$. Let us denote this sequence by α_x. Now consider the infinite sequence formed by concatenating all the α_x with copies of c_1 in between:

$$\alpha_1 c_1 \alpha_2 c_1 \cdots$$

Let $\psi(i)$ denote the i^{th} element of this sequence. Observe that ψ is computable in polynomial time.

We define a set S of positive integers recursively as follows: $1 \in S$, and for $j > 1$, $j \in S$ if and only if $\chi_S(j-1)\psi(j-1) = 1$. Thus $j \in S$ if and only if the product $\psi(1)\cdots\psi(j-1)$ is 1 or c_1. Obviously S is 1-locally self-reducible. We now have $x \in A$ if and only if

$$1 = 1 \cdot \alpha_x = 1 \cdot \alpha_1 c_1 \cdots \alpha_x = \psi(1)\cdots\psi(y),$$

where y is computable from x in polynomial time. Thus $x \in A$ if and only if $y + 1 \in S$, giving us the desired reduction to a 1-locally self-reducible set. ∎

An analogous (although much simpler) argument, using the permutation group $S_2 = \{1, \tau\}$, will prove that the near-testable sets are all in $\oplus$P, and that every language in $\oplus$P is $\leq_m^P$-reducible to a near-testable set. This result is originally due to Goldsmith, Hemachandra, Joseph, and Young [8].

6. Conclusions and Related Research

We have shown that the class of k-locally self-reducible sets coincides with PSPACE for all $k \geq 3$. We have also found an upper bound on the complexity of 2-locally reducible sets and given a new proof of an exact characterization of the 1-locally self-reducible sets.

In all cases, our results were obtained by studying the complexity of computing the product of an exponentially long, uniformly generated sequence of elements in the finite monoid M_k, for $k = 1, 2, 3$. In essence, we showed that every k-locally self-reducible set is polynomial-time, many-one reducible to the computation of such a product, and conversely, every such product computation is reducible to testing membership in a k-locally self-reducible set.

As we remarked in the introduction, several other papers in the literature have employed this algebraic approach to Turing machine-based complexity classes. After we completed a preliminary version of this paper, we received a technical report by M. Ogihara [12], which is best understood in terms of this approach. Ogihara studies "bottleneck Turing machines". On inputs of length n, a width-k bottleneck Turing machine runs for $2^{n^{O(1)}}$ phases, where each phase takes $n^{O(1)}$ steps. At the end of each phase, the machine is reset. The only information passed from one phase to the next is the value in a 'safe storage', capable of holding one of k different values. Our locally self-reducible sets are a special case, since every k-locally self-reducible set is recognized by a width-2^k bottleneck Turing machine.

Now every language recognized by a width-k bottleneck Turing machine is in PSPACE, since we need only polynomially many bits to keep track of the phase, polynomial space to perform the computation in each phase, and only a constant amount of additional space for the safe storage. It is also easy to see that testing membership in such a language is equivalent to evaluating a uniformly generated, exponentially long product in the monoid T_k of all maps from $\{1, \ldots, k\}$ into itself, since in each phase we are, in effect, computing such a map. (Machines for which this map is always a permutation are called permutation bottleneck Turing machines.) Cai and Furst [7] used Barrington's completeness result for NC^1 to show that every language in PSPACE is recognized by a width-5 bottleneck Turing machine. Ogihara studies the structure of the classes of languages width-k bottleneck Turing machines for $k < 5$. What is interesting here is that many of his results can be obtained by translating fundamental facts about the structure of the monoids T_k and the permutation groups S_k into Turing machine language.

Let us give a few examples of this: The class of languages recognized by width-k bottleneck Turing machines is denoted SF_k by Ogihara, and the class of languages recognized by width-k permutation bottleneck Turing machines is denoted *perm*-SF_k. Ogihara shows that *perm*-SF_3 coincides with the class $\mathrm{MOD}_3 \cdot \mathrm{MOD}_2 \cdot P$. This translates the fact that the permutation group S_3 is an extension of the cyclic group of order 3 by the cyclic group of order 2. He further shows first, that,

$$\mathrm{MOD}_2 \cdot \mathrm{MOD}_3 \cdot \mathrm{MOD}_2 \cdot \mathrm{P} \subseteq SF_3,$$

which translates the fact that the semigroup of non-permutations in T_3 contains copies of the cyclic group of order 2, and then that

$$(\mathrm{MOD}_2 \cdot \mathrm{MOD}_3 \cdot \mathrm{MOD}_2)\mathrm{OptP},$$

(which is defined analogously to $\oplus\mathrm{OptP} = \mathrm{MOD}_2\mathrm{OptP}$ in the preceding section) is contained in SF_3. This is because, as we saw in the last section, passage from P to OptP in these complexity classes corresponds to the adjunction of constant maps to a transformation monoid, and T_3 contains all the constant maps. We can go further and use a wreath product decomposition of T_3 to show

$$SF_3 \subseteq (\Sigma_2^P \cup \Pi_2^P) \cdot \mathrm{MOD}_2 \cdot (\Sigma_2^P \cup \Pi_2^P) \cdot \mathrm{MOD}_3 \cdot \mathrm{MOD}_2.$$

There is, in all this, the beginnings of a detailed theory, in which structural properties of finite monoids are translated into complexity-theoretic properties of the classes of languages defined by multiplication of exponentially long, uniformly generated sequences in these monoids.

Acknowledgments. We are grateful to Bill Gasarch for proofreading the manuscript.

References

[1] J. L. Balcázar. Self-reducibility. *JCSS*, 41, 1990.

[2] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, New York, 1988.

[3] D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *JCSS*, 38(1):150–164, Feb. 1989.

[4] D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, Oct. 1988.

[5] D. Barrington, N. Immerman, and H. Straubing. On Uniformity in NC^1. *JCSS*, 41:274–306, 1990.

[6] A. Borodin. On relating time and space to size and depth, *SICOMP*, 6(4):733–743, 1977.

[7] J. Cai and M. Furst. PSPACE survives three-bit bottlenecks. *International Journal of Foundations of Computer Science*, 1990.

[8] J. Goldsmith, L. Hemachandra, D. Joseph, and P. Young. Near-testable sets. *SICOMP*, 20(3):506–523, June 1991.

[9] J. Goldsmith, D. Joseph, and P. Young. Self-reducible, p-selective, near-testable, and p-cheatable sets: The effect of internal structure on the complexity of a set. TR 87-06-02, Dept. of Computer Science, University of Washington, Seattle, June 1987. An extended abstract appeared in *Proc. of the 2nd Annual Conference on Structure in Complexity Theory*, IEEE Computer Society Press, June 1987, pp. 50–59.

[10] L. A. Hemachandra and A. Hoene. On sets with efficient implicit membership tests. *SICOMP*, 20(6):1148–1156, Dec. 1991.

[11] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. Wagner. On the power of polynomial time bit-reductions. In *Proc. 8th Structures*, pp. 200–207, 1993.

[12] M. Ogihara. On serializable languages University of Rochester Technical Report, June, 1994.

[13] M. Ogiwara and A. Lozano. On one query self-reducible sets. In *Proc. 6th Structures*, pp. 139–151, 1991.

[14] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19^{th} ACM STOC*, pp. 77–82, 1987.