Mitosis in Computational Complexity

Christian Glaßer[1], A. Pavan[2⋆], Alan L. Selman[3⋆⋆], and Liyu Zhang[4]

[1] Universität Würzburg, glasser@informatik.uni-wuerzburg.de
[2] Iowa State University, pavan@cs.iastate.edu
[3] University at Buffalo, selman@cse.buffalo.edu
[4] University at Buffalo, lzhang7@cse.buffalo.edu

Abstract. This expository paper describes some of the results of two recent research papers [GOP+05,GPSZ05]. The first of these papers proves that every NP-complete set is many-one autoreducible. The second paper proves that every many-one autoreducible set is many-one mitotic. It follows immediately that every NP-complete set is many-one mitotic. Hence, we have the compelling result that every NP-complete set A splits into two NP-complete sets A_1 and A_2.

1 Autoreducibility

We begin with the notion of autoreducibility. Trakhtenbrot [Tra70] defined a set A to be *autoreducible* if it can be reduced to itself by a Turing machine that does not ask its own input to the oracle. This means there is an oracle Turing machine M such that $A = L(M^A)$ and M on input x never queries x. Ladner [Lad73] showed that there exist Turing-complete recursively enumerable sets that are not autoreducible. We are interested in the polynomial-time variant of autoreducibility, introduced by Ambos-Spies [AS84], where we require the oracle Turing machine to run in polynomial time. Henceforth, by "autoreducible" we mean "polynomial-time autoreducible."

The question of whether complete sets for various complexity classes are autoreducible has been studied extensively [Yao90,BF92,BFvMT00]. Beigel and Feigenbaum [BF92] showed that Turing-complete sets for the classes that form the polynomial-time hierarchy are autoreducible. In particular, all Turing-complete sets for NP are autoreducible. Buhrman et al. [BFvMT00] showed that Turing-complete sets for EXP and Δ_i^{EXP} are autoreducible, whereas there exists a Turing-complete set for EESPACE that is not Turing autoreducible. They showed that answering questions about autoreducibility of intermediate classes results in interesting separation results.

Researchers have studied autoreducibility with various polynomial-time reducibilities. Regarding NP, Buhrman et al. [BFvMT00] showed that all truth-table-complete sets for NP are probabilistic truth-table autoreducible. Thus, all NP-complete sets are probabilistic truth-table autoreducible.

⋆ Research supported in part by NSF grants CCR-0344817 and CCF-0430807
⋆⋆ Research supported in part by NSF grant CCR-0307077

A set A is *polynomial-time m-autoreducible* (m-autoreducible, for short) if $A \leq_m^p A$ via a polynomial-time computable reduction function f such that for all x, $f(x) \neq x$. Note that m-autoreducibility is a strong form of autoreducibility. If a set is m-autoreducible, then it is T-autoreducible also. Buhrman and Torenvliet [BT94] asked whether all NP-complete sets are m-autoreducible and whether all PSPACE-complete sets are m-autoreducible. Glasser et al. [GOP$^+$05] resolve these questions positively. A set L is *nontrivial* if $\|L\| > 1$ and $\|\overline{L}\| > 1$. The proof that all nontrivial NP-complete sets are m-autoreducible is interesting, for it uses the left set technique of Ogihara and Watanabe [OW91].

Let L belong to the class NP and let M be a polynomial-time-bounded nondeterministic Turing machine that accepts L. For a suitable polynomial p, we can assume that all computation paths v on input x have length $p(|x|)$. Let

$$\begin{aligned}\mathrm{Left}(L) = \{\langle x, u\rangle \,\big|\, & |u| = p(|x|) \text{ and } \exists v, |v| = |u|, \text{such that} \\ & u \leq v \text{ and } M(x) \text{ accepts along path } v\}.\end{aligned}$$

Notice that Left(L) belongs to the class NP. So if L is NP-complete, then there is a polynomial-time-computable reduction f from Left(L) to L.

Theorem 1. *All nontrivial* NP*-complete sets are m-autoreducible.*

Proof. Let L be NP-complete. As we just described, let M be an NP-machine that accepts L, let p be a polynomial so that all computation paths of M on an input x have length $p(|x|)$, and let f be a polynomial-time-computable reduction from Left(L) to L. Since L is nontrivial, let $y_1, y_2 \in L$ and $\overline{y}_1, \overline{y}_2 \in \overline{L}$.

The following algorithm defines a function g to be an m-autoreduction for L: Let x be an input, and define $n = |x|$ and $m = p(|x|)$.

```
1   if f(⟨x, 0^m⟩) ≠ x then output f(⟨x, 0^m⟩)
2   if f(⟨x, 1^m⟩) = x then
3      if M(x) accepts along 1^m then
4         output a string from {y_1, y_2} − {x}
5      else
6         output a string from {ȳ_1, ȳ_2} − {x}
7      endif
8   endif
9   // here f(⟨x, 0^m⟩) = x ≠ f(⟨x, 1^m⟩)
10  determine z of length m such that f(⟨x, z⟩) = x ≠ f(⟨x, z + 1⟩)
11  if M(x) accepts along z then output a string from {y_1, y_2} − {x}
12  else output f(⟨x, z + 1⟩)
```

Step 10 is accomplished by an easy binary search algorithm: Start with $z_1 := 0^m$ and $z_2 := 1^m$. Let z' be the middle element between z_1 and z_2. If $f(z') = x$ then $z_1 := z'$ else $z_2 := z'$. Again, choose the middle element between z_1 and z_2, and so on. This shows that g is computable in polynomial time. Clearly, $g(x) \neq x$, so it remains to show that $L \leq_m^p L$ via g.

If the algorithm stops in step 1, then

$$x \in L \iff \langle x, 0^m \rangle \in \text{Left}(L) \Leftrightarrow g(x) = f(\langle x, 0^m \rangle) \in L.$$

If the algorithm stops in step 4 or step 6, then $f(\langle x, 0^m \rangle) = f(\langle x, 1^m \rangle)$. Hence

$$x \in L \iff \langle x, 1^m \rangle \in \text{Left}(L) \iff M(x) \text{ accepts along } 1^m \iff g(x) \in L.$$

Assuming we reach step 9, it holds that $f(\langle x, 0^m \rangle) = x \neq f(\langle x, 1^m \rangle)$. If the algorithm stops in step 11, then $x \in L$ and $g(x) \in L$. Now consider the possibility that the algorithm stops in step 12. We treat two cases:

Assume $x \notin L$. So there is no accepting computation. So $\langle x, z+1 \rangle \notin \text{Left}(L)$. So $g(x) = f(\langle x, z+1 \rangle) \notin L$ also.

Assume $x \in L$. Then $f(\langle x, 0^m \rangle) = f(\langle x, z \rangle) = x \in L$. The rightmost accepting computation of M on x is to the right of z. So $f(\langle x, z+1 \rangle) \in L$, because either $z+1$ is accepting or something to its right is accepting. □

The paper of Glasser et al. extends the basic technique to show that the nontrivial complete sets of several additional complexity classes are m-autoreducible: $\oplus$P, the levels of the polynomial-time hierarchy, the Boolean hierarchy over NP, and MODPH.

2 Mitotic Sets

A recursively enumerable set is *mitotic* if it can be divided into two disjoint r.e. sets A_0 and A_1 such that A, A_0, and A_1 are all Turing equivalent. The set A consists of two parts that *each contain the same amount of information* as the original set. Ladner [Lad73] showed that autoreducibility and mitoticity coincide for the r.e. sets.

Ambos-Spies [AS84] defined two notions of mitoticity in the polynomial-time setting:

Definition 1 (Ambos-Spies). *A decidable set A is* polynomial-time m(T)-mitotic *(m(T)-mitotic, for short) if there exists a set $B \in$ P such that*

$$A \equiv^p_{m(T)} A \cap B \equiv^p_{m(T)} A \cap \overline{B}.$$

A decidable set A is polynomial-time weakly m(T)-mitotic *(weakly m(T)-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$, and*

$$A \equiv^p_{m(T)} A_0 \equiv^p_{m(T)} A_1.$$

Ambos-Spies proved that m-mitotic implies m-autoreducible and asked whether the converse holds. Also, he proved that T-autoreducible does not imply T-mitotic. Buhrman, Hoene, and Torenvliet [BHT98] proved that all m-complete sets for EXP are weakly m-mitotic, while Glasser et al. [GOP$^+$05] strengthened this result to show that all m-complete sets for EXP are m-mitotic.

The main result of the second paper by Glasser et al. [GPSZ05] settles the open question of Ambos-Spies with the following result:

Theorem 2 (GPSZ, 2005). *Let L be any set such that $\|\overline{L}\| \geq 2$. L is m-autoreducible if and only if L is m-mitotic.*

The following corollaries follow immediately:

Corollary 1. *Every nontrivial set that is m-complete for one of the following complexity classes is m-mitotic.*

- NP, $\oplus$P, PSPACE, EXP, NEXP
- *any level of the polynomial-time hierarchy,* MODPH*, or the Boolean hierarchy over* NP

The corollary settles several long-standing open questions raised by Ambos-Spies [AS84], Buhrman, Hoene, and Torenvliet [BHT98], and Buhrman and Torenvliet [BT94]. With specific regard to the class NP, we have the following:

Corollary 2. *For every nontrivial* NP*-complete set L, there is a set $S \in$ P such that $L \cap S$ and $L \cap \overline{S}$ are* NP*-complete.*

Corollary 2 holds for all *known* natural NP-complete sets.[5] Our contribution is that it holds unconditionally for all NP-complete sets.

Corollary 3. *A nontrivial set L is* NP*-complete if and only if L is the union of two disjoint* P*-separable* NP*-complete sets.*

The proof of Theorem 2 in one direction is straightforward. However the proof that m-autoreducible implies m-mitotic is too complex to present here. Nevertheless, we can illustrate some of the issues that arise in the proof and suggest how these issues are addressed.

Assume that L is m-autoreducible via reduction function f. Given x, the repeated application of f yields a sequence of words $x, f(x), f(f(x)), \ldots$, which we call the trajectory of x. These trajectories either are infinite or end in a cycle of length at least 2. Note that as f is an autoreduction, $x \neq f(x)$.

At first glance it seems that m-mitoticity can be easily established by the following idea: In every trajectory, label the words at even positions with $+$ and all other words with $-$. Define S to be the set of strings whose label is $+$. With this 'definition' of S it seems that f reduces $L \cap S$ to $L \cap \overline{S}$ and $L \cap \overline{S}$ to $L \cap S$.

However, this labeling strategy has at least two problems. First, it is not clear that $S \in$ P; because given a string y, we have to compute the parity of the position of y in a trajectory. As trajectories can be of exponential length, this might take exponential time. The second and more fundamental problem is the following: The labeling generated above is *inconsistent* and not well defined. For example, let $f(x) = y$. To label y which trajectory should we use? The trajectory of x or the trajectory of y? If we use trajectory of x, y gets a label of $+$, whereas

[5] All known natural NP-complete sets are p-isomorphic to SAT. It is easy to see that the corollary holds for SAT, from which it follows that it holds for all sets that are p-isomorphic to SAT.

if we use the trajectory of y, then it gets a label of $-$. Thus S is not well defined and so this idea does not work. It fails because the labeling strategy is a global strategy. To label a string we have to consider all the trajectories in which x occurs. Every single x gives rise to a labeling of possibly infinitely many words, and these labelings may overlap in an inconsistent way.

We resolve this by using a *local labeling strategy*. More precisely, we compute a label for a given x just by looking at the neighboring values x, $f(x)$, and $f(f(x))$. It is immediately clear that such a strategy is well-defined and therefore defines a consistent labeling. We also should guarantee that this local strategy strictly alternates labels, i.e., x gets $+$ if and only if $f(x)$ gets $-$. Such an alternation of labels would help us to establish the m-mitoticity of L.

Thus our goal will be to find a local labeling strategy that has a nice alternation behavior. However, we settle for something less. Instead of requiring that the labels strictly alternate, we only require that given x, at least one of $f(x), f(f(x)), \cdots, f^m(x)$ gets a label that is different from the label of x, where m is polynomially bounded in the length of x. This suffices to show m-mitoticity.

The most difficult part in our proof is to show that there exists a local labeling strategy that has this weaker alternation property.

We now formulate the core underlying problem. To keep this proof sketch simpler, we make several assumptions and ignore several technical but important details. If we assume (for simplicity) that on strings $x \notin 1^*$ the autoreduction is length preserving such that $f(x) > x$, then we arrive at the following graph labeling problem.

Core Problem: Let G_n be a directed graph with 2^n vertices such that every string of length n is a vertex of G_n. Assume that 1^n is a sink, that nodes $u \neq 1^n$ have outdegree 1, and that $u < v$ for edges (u, v). For $u \neq 1^n$ let $s(u)$ denote u's unique successor, i.e., $s(u) = v$ if (u, v) is an edge. Find a strategy that labels each node with either $+$ or $-$ such that:

(i) Given a node u, its label can be computed in polynomial time in n.
(ii) There exists a polynomial p such that for every node u, at least one of the nodes $s(u), s(s(u)), \ldots, s^{p(n)}(u)$ gets a label that is different from the label of u.

We exhibit a labeling strategy with these properties. To define this labeling, we use the following *distance function*: $d(x, y) \stackrel{df}{=} \lfloor \log |y - x| \rfloor$. The core problem is solved by the following local strategy.

```
0   // Strategy for labeling node x
1   let y = s(x) and z = s(y).
2   if d(x,y) > d(y,z) then output −
3   if d(x,y) < d(y,z) then output +
4   r := d(x,y)
5   output + iff ⌊x/2^(r+1)⌋ is even
```

Clearly, this labeling strategy satisfies condition (i). We give a sketch of the proof that it also satisfies condition (ii). Define $m = 5n$ and let $u_1, u_2, \ldots, u_m$

be a path in the graph. It suffices to show that not all the nodes $u_1, u_2, \ldots, u_m$ obtain the same label. Assume that this does not hold, say all these nodes get label +. So no output is made in line 2 and therefore, the distances $d(u_i, u_{i+1})$ do not decrease. Note that the distance function maps to natural numbers. If we have more than n increases, then the distance between u_{m-1} and u_m is bigger than n. Therefore, $u_m - u_{m-1} > 2^{n+1}$, which is impossible for words of length n. So along the path $u_1, u_2, \ldots, u_m$ there exist at least $m - n = 4n$ positions where the distance stays the same. By a pigeon hole argument there exist four consecutive such positions, i.e., nodes $v = u_i$, $w = u_{i+1}$, $x = u_{i+2}$, $y = u_{i+3}$, $z = u_{i+4}$ such that $d(v, w) = d(w, x) = d(x, y) = d(y, z)$. So for the inputs v, w, and x, we reach line 4 where the algorithm will assign $r = d(v, w)$. Observe that for all words w_1 and w_2, the value $d(w_1, w_2)$ allows an approximation of $w_2 - w_1$ up to a factor of 2. More precisely, $w - v$, $x - w$, and $y - x$ belong to the interval $[2^r, 2^{r+1})$. It is an easy observation that this implies that not all of the following values can have the same parity: $\lfloor v/2^{r+1} \rfloor$, $\lfloor w/2^{r+1} \rfloor$, and $\lfloor x/2^{r+1} \rfloor$. According to line 5, not all words v, w, and x obtain the same label. This is a contradiction that shows that not all the nodes $u_1, u_2, \ldots, u_m$ obtain the same label. This proves (ii) and solves the core labeling problem.

We mention again that Turing autoreducible does not imply Turing mitotic [AS84]. Glasser et al. [GPSZ05] proved that autoreducibility does not imply mitoticity for all polynomial-time-bounded reducibilities beween 3-tt-reducibility and Turing-reducibility. (There exists L in EXP such that L is 3-tt-autoreducible, but L is not weakly T-mitotic.) It is not known what happens when we consider 2-tt-reductions. Is every 2-tt-autoreducible set 2-tt-mitotic? This is an open question. Much is not known about truth-table reductions and autoreducibility. For example, it is not known whether all $\leq^p_{tt}$-complete sets for NP are $\leq^p_{tt}$-autoreducible. It is not known whether all $\leq^p_{btt}$-complete sets for NP are $\leq^p_{btt}$-autoreducible.

References

[AS84] K. Ambos-Spies. P-mitotic sets. In E. Börger, G. Hasenjäger, and D. Roding, editors, *Logic and Machines, Lecture Notes in Computer Science 177*, pages 1–23. Springer-Verlag, 1984.

[BF92] R. Beigel and J. Feigenbaum. On being incoherent without being very hard. *Computational Complexity*, 2:1–17, 1992.

[BFvMT00] H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet. Using autoreducibility to separate complexity classes. *SIAM Journal on Computing*, 29(5):1497–1520, 2000.

[BHT98] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness, and structure of complete sets. *SIAM Journal on Computing*, 27:637–653, 1998.

[BT94] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Proceedings 9th Structure in Complexity Theory*, pages 118–133, 1994.

[GOP+05] C. Glasser, M. Ogihara, A. Pavan, A. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.

[GPSZ05] C. Glasser, A. Pavan, A. Selman, and L. Zhang. Redundancy in complete sets. Technical Report 05-068, Electronic Colloquium on Computational Complexity, 2005.

[Lad73] R. Ladner. Mitotic recursively enumerable sets. *Journal of Symbolic Logic*, 38(2):199–211, 1973.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal of Computing*, 20(3):471–483, 1991.

[Tra70] B. Trakhtenbrot. On autoreducibility. *Dokl. Akad. Nauk SSSR*, 192, 1970. Translation in Soviet Math. Dokl. 11: 814– 817, 1970.

[Yao90] A. Yao. Coherent functions and program checkers. In *Proceedings of the 22nd Annual Symposium on Theory of Computing*, pages 89–94, 1990.