Electronic Colloquium on Computational Complexity, Report No. 47 (2013)

Autoreducibility of Complete Sets for Log-Space and Polynomial-Time Reductions

Christian Glaßer* Dung T. Nguyen† Christian Reitwießner*‡ Alan L. Selman†
Maximilian Witek*

18th March 2013

Abstract

We investigate the autoreducibility and mitoticity of complete sets for several classes with respect to different polynomial-time and logarithmic-space reducibility notions.

Previous work in this area focused on polynomial-time reducibility notions. Here we obtain new mitoticity and autoreducibility results for the classes EXP and NEXP with respect to some restricted truth-table reductions (e.g., $\leq^{\mathrm{p}}_{2\text{-tt}}, \leq^{\mathrm{p}}_{\mathrm{ctt}}, \leq^{\mathrm{p}}_{\mathrm{dtt}}$).

Moreover, we start a systematic study of logarithmic-space autoreducibility and mitoticity which enables us to also consider P and smaller classes. Among others, we obtain the following results:

- Regarding $\leq^{\log}_{\mathrm{m}}$, $\leq^{\log}_{2\text{-tt}}$, $\leq^{\log}_{\mathrm{dtt}}$ and $\leq^{\log}_{\mathrm{ctt}}$, complete sets for PSPACE and EXP are mitotic, and complete sets for NEXP are autoreducible.
- All $\leq^{\log}_{1\text{-tt}}$-complete sets for NL and P are $\leq^{\log}_{2\text{-tt}}$-autoreducible, and all $\leq^{\log}_{\mathrm{btt}}$-complete sets for NL, P and Δ^{p}_k are $\leq^{\log}_{\text{log-T}}$-autoreducible.
- There is a $\leq^{\log}_{3\text{-tt}}$-complete set for PSPACE that is not even $\leq^{\log}_{\mathrm{btt}}$-autoreducible.

Using the last result, we conclude that some of our results are hard or even impossible to improve.

1 Introduction

A set C is called autoreducible if C can be reduced to itself by a reduction that does not query its own input. In this way, each reducibility induces a corresponding autoreducibility notion. The main question in connection with autoreducibility asks whether all complete sets of a certain complexity class are autoreducible. Interestingly, answering such questions often leads to new separations of complexity classes.

For example, consider the question of whether all polynomial-time truth-table-complete sets for EXP are polynomial-time truth-table-autoreducible. Buhrman et al. [5] show that a positive answer results in NL $\neq$ NP while a negative answer implies PH $\neq$ EXP. So the study of the autoreducibility of complete sets is a fascinating and important topic.

Mitoticity is another structural property of complete sets that could lead to separations of complexity classes. A set C is mitotic if it can be partioned into sets C_1 and C_2 such that C, C_1, and C_2 are equivalent. Here again each reducibility induces a corresponding notion of mitoticity.

*Julius-Maximilians-Universität Würzburg, {glasser,reitwiessner,witek}@informatik.uni-wuerzburg.de
†University at Buffalo, The State University of New York, {dtn3,selman}@buffalo.edu
‡CCTVAL, Universidad Técnica Federico Santa María, Valparaíso. Supported by Basal Project PB-821 CCTVal – Centro Científico Tecnológico de Valparaíso

ISSN 1433-8092

Over the past 20 years, researchers were able to solve autoreducibility and mitoticity questions for several complexity classes and with respect to several polynomial-time reducibility notions. With our paper we further develop this knowledge in two ways: First, we extend techniques by Buhrman et al. [5] to show new mitoticity results for EXP and NEXP. Second, we start a systematic investigation of autoreducibility and mitoticity for logarithmic-space reductions. Since the previous research was concerned with polynomial-time reductions, it did not produce conclusions about P or smaller classes.

With respect to polynomial-time 2-truth-table reducibility ($\leq^{\mathrm{p}}_{\text{2-tt}}$) we show that all EXP-complete sets are mitotic and all NEXP-complete sets are autoreducible. With respect to logspace reducibilities (e.g., $\leq^{\log}_{\mathrm{m}}$, $\leq^{\log}_{\text{2-tt}}$, $\leq^{\log}_{\text{btt}}$, $\leq^{\log}_{\mathrm{T}}$) we obtain several autoreducibility and mitoticity results for complete sets of the classes NL, P, PSPACE, EXP, and NEXP. Table 1 summarizes previously known results and their references together with results newly obtained in this paper. For example, we show:

(i) All $\leq^{\log}_{\text{2-tt}}$-complete sets for PSPACE are $\leq^{\log}_{\text{2-tt}}$-mitotic.
(ii) All $\leq^{\log}_{\text{2-tt}}$-complete sets for EXP are $\leq^{\log}_{\text{2-tt}}$-mitotic.

Note that in both cases, mitoticity implies autoreducibility.

The restriction of the reduction allows us to show stronger negative results. We prove:

(iii) There exists a $\leq^{\log}_{\text{3-tt}}$-complete set for PSPACE that is not $\leq^{\log}_{\text{btt}}$-autoreducible.

This result is particularly interesting, since it shows that statement (i) does not hold for $\leq^{\log}_{\text{3-tt}}$ and that (ii) cannot be improved to $\leq^{\log}_{\text{3-tt}}$, unless one separates EXP from PSPACE. Furthermore, for logspace bounded-truth-table reducibility we obtain that resolving the autoreducibility or mitoticity of complete sets for classes between L and PSPACE in one or the other way implies new separations of complexity classes:

(iv) For every $\mathcal{C} \in \{\mathrm{P}, \mathrm{NP}, \Delta^{\mathrm{p}}_k, \mathrm{PP}\}$ it holds that
  - if all $\leq^{\log}_{\text{btt}}$-complete sets for $\mathcal{C}$ are $\leq^{\log}_{\text{btt}}$-autoreducible, then $\mathcal{C} \neq \mathrm{PSPACE}$
  - otherwise, $\mathrm{L} \neq \mathcal{C}$.

The paper is organized as follows. Section 2 contains the preliminary definitions and some basic propositions about autoreducibility and mitoticity. In section 3 we use search techniques in computation trees to establish autoreducibility of complete sets for NL and P. In section 4 we use local checkability to obtain further autoreducibility results for NL, P, and Δ^{p}_k. Moreover, we argue that some of those results are difficult to improve, as such an improvement would separate P or Δ^{p}_k from PSPACE. In section 5 we consider higher complexity classes such as $\mathrm{PSPACE}, \mathrm{EXP}, \mathrm{NEXP}$ and use diagonalization to obtain mitoticity and autoreducibility results, some of which again are hard or even impossible to improve.

2 Preliminaries

Let $\log 0 = 0$ and $\log n = \lceil \log_2 n \rceil$ for $n \geq 1$. A set is called *trivial* if it is finite or cofinite; otherwise the set is called *nontrivial*. The characteristic function of a set A is denoted by c_A or simply A. If M is a machine, then $M(x)$ denotes the computation of M on input x and $L(M)$ denotes the language accepted by M. Let $\langle \cdots \rangle$ be a standard pairing function computable in logarithmic space.

The operators $\wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow$ denote the usual 2-ary Boolean functions and $\neg\wedge, \neg\vee, \not\rightarrow, \not\leftarrow, \oplus$ denote the negations of these functions.

reduction	NL	P	Δ_k^{p}	PSPACE	EXP	NEXP	references
$\leq_{\mathrm{m}}^{\log}$	$\mathrm{A}_{\text{1-tt}}^{\log}$, $\mathrm{A}_{\text{2-dtt}}^{\log}$, $\mathrm{A}_{\text{2-ctt}}^{\log}$	$\mathrm{A}_{\text{1-tt}}^{\log}$		$\mathrm{M}_{\mathrm{m}}^{\log}$	$\mathrm{M}_{\mathrm{m}}^{\log}$	$\mathrm{A}_{\mathrm{m}}^{\log}$	2.7, 3.5, 5.3, 5.6
$\leq_{\text{1-tt}}^{\log}$	$\mathrm{A}_{\text{2-tt}}^{\log}$	$\mathrm{A}_{\text{2-tt}}^{\log}$		$\mathrm{M}_{\mathrm{m}}^{\log}$	$\mathrm{M}_{\mathrm{m}}^{\log}$	$\mathrm{A}_{\mathrm{m}}^{\log}$	3.5, 5.15
$\leq_{\alpha\text{tt}}^{\log}$	$\mathrm{A}_{\text{btt}}^{\log}$	$\mathrm{A}_{\text{btt}}^{\log}$					4.11
$\leq_{\text{2-tt}}^{\log}$				$\mathrm{M}_{\text{2-tt}}^{\log}$	$\mathrm{M}_{\text{2-tt}}^{\log}$	$\mathrm{A}_{\text{2-tt}}^{\log}$	5.7, 5.11
$\leq_{k\text{-ctt}}^{\log}$	$\mathrm{A}_{k\text{-tt}}^{\log}$, $\mathrm{A}_{2k\text{-ctt}}^{\log}$	$\mathrm{A}_{k\text{-tt}}^{\log}$		$\mathrm{M}_{k\text{-ctt}}^{\log}$	$\mathrm{M}_{k\text{-ctt}}^{\log}$	$\mathrm{A}_{k\text{-ctt}}^{\log}$	2.7, 3.5, 5.11
$\leq_{k\text{-dtt}}^{\log}$	$\mathrm{A}_{k\text{-tt}}^{\log}$, $\mathrm{A}_{2k\text{-dtt}}^{\log}$	$\mathrm{A}_{k\text{-tt}}^{\log}$		$\mathrm{M}_{k\text{-dtt}}^{\log}$	$\mathrm{M}_{k\text{-dtt}}^{\log}$	$\mathrm{A}_{k\text{-dtt}}^{\log}$	2.7, 3.5, 5.11
$\leq_{\text{btt}}^{\log}$	$\mathrm{A}_{\text{log-T}}^{\log}$	$\mathrm{A}_{\text{log-T}}^{\log}$	$\mathrm{A}_{\text{log-T}}^{\log}$	X_1	X_2		4.1, 4.2, 4.8, [5]
$\leq_{\text{ctt}}^{\log}$	$\mathrm{A}_{\text{ctt}}^{\log}$			$\mathrm{M}_{\text{ctt}}^{\log}$	$\mathrm{M}_{\text{ctt}}^{\log}$	$\mathrm{A}_{\text{ctt}}^{\log}$	3.5, 5.6, 5.11
$\leq_{\text{dtt}}^{\log}$	$\mathrm{A}_{\text{dtt}}^{\log}$			$\mathrm{M}_{\text{dtt}}^{\log}$	$\mathrm{M}_{\text{dtt}}^{\log}$	$\mathrm{A}_{\text{dtt}}^{\log}$	3.5, 5.6, 5.11
$\leq_{\text{tt}}^{\log}$	$\mathrm{A}_{\text{tt}}^{\log}$	$\mathrm{A}_{\text{tt}}^{\log}$	$\mathrm{A}_{\text{tt}}^{\log}$				3.5, 4.8
$\leq_{\mathrm{T}}^{\log}$	$\mathrm{A}_{\text{tt}}^{\log}$	$\mathrm{A}_{\text{tt}}^{\log}$	$\mathrm{A}_{\text{tt}}^{\log}$				3.5, 4.8

reduction	NP	Δ_k^{p}	PSPACE	EXP	NEXP	references
$\leq_{\mathrm{m}}^{\mathrm{p}}$	$\mathrm{M}_{\mathrm{m}}^{\mathrm{p}}$	$\mathrm{M}_{\mathrm{m}}^{\mathrm{p}}$	$\mathrm{M}_{\mathrm{m}}^{\mathrm{p}}$	$\mathrm{M}_{\mathrm{m}}^{\mathrm{p}}$	$\mathrm{M}_{\mathrm{m}}^{\mathrm{p}}$	[3, 6, 7, 8]
$\leq_{\text{1-tt}}^{\mathrm{p}}$	$\mathrm{M}_{\text{1-tt}}^{\mathrm{p}}$		$\mathrm{M}_{\text{1-tt}}^{\mathrm{p}}$	$\mathrm{M}_{\mathrm{m}}^{\mathrm{p}}$	$\mathrm{M}_{\mathrm{m}}^{\mathrm{p}}$	[4, 7, 8, 9], 5.15
$\leq_{\text{2-tt}}^{\mathrm{p}}$				$\mathrm{M}_{\text{2-tt}}^{\mathrm{p}}$	$\mathrm{A}_{\text{2-tt}}^{\mathrm{p}}$	[5], 5.7, 5.11
$\leq_{k\text{-ctt}}^{\mathrm{p}}$				$\mathrm{M}_{k\text{-ctt}}^{\mathrm{p}}$	$\mathrm{A}_{k\text{-ctt}}^{\mathrm{p}}$	5.6, 5.11
$\leq_{k\text{-dtt}}^{\mathrm{p}}$	$\mathrm{A}_{k\text{-dtt}}^{\mathrm{p}}$		$\mathrm{A}_{k\text{-dtt}}^{\mathrm{p}}$	$\mathrm{M}_{k\text{-dtt}}^{\mathrm{p}}$	$\mathrm{A}_{k\text{-dtt}}^{\mathrm{p}}$	[7], 5.6, 5.11
$\leq_{\text{btt}}^{\mathrm{p}}$				X_3		[5]
$\leq_{\text{ctt}}^{\mathrm{p}}$				$\mathrm{M}_{\text{ctt}}^{\mathrm{p}}$	$\mathrm{A}_{\text{ctt}}^{\mathrm{p}}$	5.6, 5.11
$\leq_{\text{dtt}}^{\mathrm{p}}$	$\mathrm{A}_{\text{dtt}}^{\mathrm{p}}$		$\mathrm{A}_{\text{dtt}}^{\mathrm{p}}$	$\mathrm{M}_{\text{dtt}}^{\mathrm{p}}$	$\mathrm{A}_{\text{dtt}}^{\mathrm{p}}$	[7], 5.6, 5.11
$\leq_{\text{tt}}^{\mathrm{p}}$	$\mathrm{A}_{\text{tt}}^{\mathrm{BPP}}$	$\mathrm{A}_{\text{tt}}^{\mathrm{p}}$		$\mathrm{A}_{\text{tt}}^{\mathrm{BPP}}$		[2, 5]
$\leq_{\mathrm{T}}^{\mathrm{p}}$	$\mathrm{A}_{\mathrm{T}}^{\mathrm{p}}$		$\mathrm{A}_{\mathrm{T}}^{\mathrm{p}}$	$\mathrm{A}_{\mathrm{T}}^{\mathrm{p}}$		[2, 5]

Table 1: Are all complete sets for certain classes autoreducible or even mitotitc? For each reduction $\leq$ in the first column and each class $\mathcal{C}$ in the first row, the corresponding cell shows if all $\leq$-complete sets for $\mathcal{C}$ are autoreducible or mitotic, where M_y^x means $\leq_y^x$-mitotic, and A_y^x means $\leq_y^x$-autoreducible. From the first cell in the upper table we know for instance that all $\leq_{\mathrm{m}}^{\log}$-complete sets for NL are $\leq_{\text{1-tt}}^{\log}$-autoreducible. $k \geq 2$ is a fixed integer, α is an arbitrary binary boolean function. For the cells marked with X_1, X_2, and X_3, negative results are known: There is a $\leq_{\text{btt}}^{\log}$-complete set for PSPACE that is not $\leq_{\text{btt}}^{\log}$-autoreducible (Theorem 4.2) and a $\leq_{\text{btt}}^{\log}$-complete set for EXP that is not $\leq_{\text{btt}}^{\mathrm{p}}$-autoreducible [5]. Results implied by universal relations between reductions are omitted. For the definitions of the reductions and the classes, see section 2.

The notions of polynomial-time (resp., logspace) oracle Turing machine and polynomial-time-computable (resp., logspace-computable) function are defined according to Ladner, Lynch, and Selman [10, 11].

For sets A and B we say that A is polynomial-time Turing reducible to B ($A \leq^{\mathrm{p}}_{\mathrm{T}} B$), if there exists a polynomial-time oracle Turing machine that accepts A with B as its oracle. If M on input x asks at most $O(\log |x|)$ queries, then A is polynomial-time log-Turing reducible to B ($A \leq^{\mathrm{p}}_{\text{log-T}} B$). If M's queries are nonadaptive (i.e., independent of the oracle), then A is polynomial-time truth-table reducible to B ($A \leq^{\mathrm{p}}_{\mathrm{tt}} B$). If M asks at most k nonadaptive queries, then A is polynomial-time k-truth-table reducible to B ($A \leq^{\mathrm{p}}_{k\text{-tt}} B$). A is polynomial-time bounded-truth-table reducible to B ($A \leq^{\mathrm{p}}_{\mathrm{btt}} B$), if $A \leq^{\mathrm{p}}_{k\text{-tt}} B$ for some k. A is polynomial-time disjunctive-truth-table reducible to B ($A \leq^{\mathrm{p}}_{\mathrm{dtt}} B$), if there exists a polynomial-time-computable function f such that for all x, $f(x) = (q_1, \ldots, q_n)$ for some $n \geq 1$ and ($x \in A \iff c_B(q_1) \vee \cdots c_B(q_n)$). If n is bounded by some constant k, then A is polynomial-time k-disjunctive-truth-table reducible to B ($A \leq^{\mathrm{p}}_{k\text{-dtt}} B$). The polynomial-time conjunctive-truth-table reducibilities $\leq^{\mathrm{p}}_{\mathrm{ctt}}$ and $\leq^{\mathrm{p}}_{k\text{-ctt}}$ are defined analogously. A is polynomial-time many-one reducible to B ($A \leq^{\mathrm{p}}_{\mathrm{m}} B$), if there exists a polynomial-time-computable function f such that ($x \in A \iff f(x) \in B$). For a k-ary Boolean function α, A is polynomial-time α-truth-table reducible to a set B ($A \leq^{\mathrm{p}}_{\alpha\mathrm{tt}} B$), if there exists a polynomial-time-computable function f such that $f(x) = (q_1, \ldots, q_k)$ and ($x \in A \iff \alpha(c_B(q_1), \ldots, c_B(q_k))$). We also use the following logspace reducibilities which are defined analogously in terms of logspace oracle Turing machines and logspace-computable functions: $\leq^{\log}_{\mathrm{T}}$, $\leq^{\log}_{\text{log-T}}$, $\leq^{\log}_{\mathrm{tt}}$, $\leq^{\log}_{k\text{-tt}}$, $\leq^{\log}_{\mathrm{btt}}$, $\leq^{\log}_{\mathrm{dtt}}$, $\leq^{\log}_{k\text{-dtt}}$, $\leq^{\log}_{\mathrm{ctt}}$, $\leq^{\log}_{k\text{-ctt}}$, $\leq^{\log}_{\mathrm{m}}$, $\leq^{\log}_{\alpha\mathrm{tt}}$.

Consider a logspace oracle Turing machine M on input x. If we run through all configurations and compute the next string that is queried, we obtain a list L of all strings that are possibly queried during the computation of M on x. Now we can simulate the computation of M on x such that each time a string q is queried, we query *all* strings from L and use the answer to q in the further computation. This shows that we may assume that logspace oracle Turing machines query nonadaptively.

Proposition 2.1 ([10]) *$A \leq^{\log}_{\mathrm{tt}} B$ if and only if $A \leq^{\log}_{\mathrm{T}} B$.*

Definition 2.2 (autoreducibility) *For $\leq \in \{\leq^{\mathrm{p}}_{\mathrm{T}}, \leq^{\mathrm{p}}_{\text{log-T}}, \leq^{\mathrm{p}}_{\mathrm{tt}}, \leq^{\mathrm{p}}_{k\text{-tt}}, \leq^{\mathrm{p}}_{\mathrm{btt}}, \leq^{\log}_{\mathrm{T}}, \leq^{\log}_{\text{log-T}}, \leq^{\log}_{\mathrm{tt}}, \leq^{\log}_{k\text{-tt}}, \leq^{\log}_{\mathrm{btt}}\}$, a set A is $\leq$-autoreducible, if $A \leq A$ via an oracle Turing machine that on input x does not query x.*

For $\leq \in \{\leq^{\mathrm{p}}_{\mathrm{dtt}}, \leq^{\mathrm{p}}_{k\text{-dtt}}, \leq^{\mathrm{p}}_{\mathrm{ctt}}, \leq^{\mathrm{p}}_{k\text{-ctt}}, \leq^{\mathrm{p}}_{\mathrm{m}}, \leq^{\mathrm{p}}_{\alpha\mathrm{tt}}, \leq^{\log}_{\mathrm{dtt}}, \leq^{\log}_{k\text{-dtt}}, \leq^{\log}_{\mathrm{ctt}}, \leq^{\log}_{k\text{-ctt}}, \leq^{\log}_{\mathrm{m}}, \leq^{\log}_{\alpha\mathrm{tt}}\}$, a set A is $\leq$-autoreducible, if $A \leq A$ via a function f such that if $f(x) = (q_1, \ldots, q_n)$, then $x \notin \{q_1, \ldots, q_n\}$.

Definition 2.3 (mitoticity) *For any polynomial-time reducibility $\leq^{\mathrm{p}}$, a set A is $\leq^{\mathrm{p}}$-mitotic, if there exists a separator $S \in \mathrm{P}$ such that $A \equiv^{\mathrm{p}} A \cap S \equiv^{\mathrm{p}} A \cap \overline{S}$. Analogously we define mitoticity for logspace reducibilities at which the separator is chosen from* L.

Proposition 2.4 *For every reduction $\leq$ and every non-trivial set A, if A is $\leq$-mitotic, then A is $\leq$-autoreducible.*

Proposition 2.5 *Let $\mathcal{C}$ be a complexity class and $k \geq 1$.*

1. *For all reducibility notions $\leq \in \{\leq^{\log}_{\mathrm{m}}, \leq^{\log}_{k\text{-tt}}, \leq^{\log}_{k\text{-T}}, \leq^{\log}_{\mathrm{btt}}, \leq^{\log}_{\mathrm{tt}}, \leq^{\log}_{\text{log-T}}, \leq^{\log}_{\mathrm{T}}\}$, if A is $\leq$-complete for $\mathcal{C}$, then $\overline{A}$ is $\leq$-complete for* co$\mathcal{C}$.
2. *If A is $\leq^{\log}_{k\text{-ctt}}$-complete for $\mathcal{C}$, then $\overline{A}$ is $\leq^{\log}_{k\text{-dtt}}$-complete for* co$\mathcal{C}$.
3. *If A is $\leq^{\log}_{\mathrm{ctt}}$-complete for $\mathcal{C}$, then $\overline{A}$ is $\leq^{\log}_{\mathrm{dtt}}$-complete for* co$\mathcal{C}$.

4. *If A is $\leq^{\log}_{k\text{-dtt}}$-complete for $\mathcal{C}$, then $\overline{A}$ is $\leq^{\log}_{k\text{-ctt}}$-complete for* co$\mathcal{C}$.
5. *If A is $\leq^{\log}_{\text{dtt}}$-complete for $\mathcal{C}$, then $\overline{A}$ is $\leq^{\log}_{\text{ctt}}$-complete for* co$\mathcal{C}$.

Proposition 2.6 *The following holds for $k \geq 1$.*

1. *For all reducibility notions $\leq \in \{\leq^{\log}_{\text{m}}, \leq^{\log}_{k\text{-tt}}, \leq^{\log}_{k\text{-T}}, \leq^{\log}_{\text{btt}}, \leq^{\log}_{\text{tt}}, \leq^{\log}_{\log\text{-T}}, \leq^{\log}_{\text{T}}\}$, if A is $\leq$-autoreducible, then $\overline{A}$ is $\leq$-autoreducible.*
2. *A is $\leq^{\log}_{k\text{-ctt}}$-autoreducible $\Rightarrow$ $\overline{A}$ is $\leq^{\log}_{k\text{-dtt}}$-autoreducible.*
3. *A is $\leq^{\log}_{\text{ctt}}$-autoreducible $\Rightarrow$ $\overline{A}$ is $\leq^{\log}_{\text{dtt}}$-autoreducible.*
4. *A is $\leq^{\log}_{k\text{-dtt}}$-autoreducible $\Rightarrow$ $\overline{A}$ is $\leq^{\log}_{k\text{-ctt}}$-autoreducible.*
5. *A is $\leq^{\log}_{\text{dtt}}$-autoreducible $\Rightarrow$ $\overline{A}$ is $\leq^{\log}_{\text{ctt}}$-autoreducible.*

Proposition 2.7 *Let $k \geq 1$ and let $\mathcal{C}$ be a complexity class closed under complementation.*

1. *All $\leq^{\log}_{\text{m}}$-complete sets for $\mathcal{C}$ are $\leq^{\log}_{1\text{-tt}}$-autoreducible.*
2. *All $\leq^{\log}_{k\text{-dtt}}$-complete sets for $\mathcal{C}$ are $\leq^{\log}_{k\text{-tt}}$-autoreducible.*
3. *All $\leq^{\log}_{k\text{-ctt}}$-complete sets for $\mathcal{C}$ are $\leq^{\log}_{k\text{-tt}}$-autoreducible.*

Proof We first show item 2, so let A be $\leq^{\log}_{k\text{-dtt}}$-complete for $\mathcal{C}$ and some $k \geq 1$. $\mathcal{C}$ is closed under complementation, hence $\overline{A} \in \mathcal{C}$ and thus $\overline{A} \leq^{\log}_{k\text{-dtt}} A$ via some f, i.e., $x \notin A \iff \bigvee_{i=1}^{k} y_i \in A$, where $f(x) = (y_1, \ldots, y_k)$. If $x \notin \{y_1, \ldots, y_k\}$, this yields a $\leq^{\log}_{k\text{-tt}}$-autoreduction for A. If $x \in f(x)$, then $x \notin A$, since otherwise the reduction shows a contradiction.

To show item 3, let A be $\leq^{\log}_{k\text{-ctt}}$-complete for $\mathcal{C}$. By Proposition 2.5, $\overline{A}$ is $\leq^{\log}_{k\text{-dtt}}$-complete for co$\mathcal{C} = \mathcal{C}$. So $\overline{A}$ is $\leq^{\log}_{k\text{-tt}}$-autoreducible by the argument above. From Proposition 2.6 we obtain that A is $\leq^{\log}_{k\text{-tt}}$-autoreducible.

Finally, item 1 follows from item 2 and item 3 for $k = 1$. □

Proposition 2.8 *Let $\mathcal{C}$ be a complexity class that is closed under complementation. For every Boolean function α, if A is $\leq^{\log}_{\alpha\text{tt}}$-complete for $\mathcal{C}$, then A is $\leq^{\log}_{(\neg\alpha)\text{tt}}$-complete for $\mathcal{C}$.*

Proof Let A be $\leq^{\log}_{\alpha\text{tt}}$-complete for $\mathcal{C}$ and $B \in \mathcal{C}$, where α is a k-ary Boolean function. We have to show that $B \leq^{\log}_{(\neg\alpha)\text{tt}} A$. Since $\overline{B} \in \mathcal{C}$, there is a function f that shows $\overline{B} \leq^{\log}_{\alpha\text{tt}} A$. On input x, consider $\langle y_1, y_2, \ldots, y_k\rangle = f(x)$. We now have $x \in \overline{B} \iff \alpha(c_A(y_1), c_A(y_2), \ldots, c_A(y_k))$. This means that $x \in B \iff (\neg\alpha)(c_A(y_1), c_A(y_2), \ldots, c_A(y_k))$ and thus f also shows $B \leq^{\log}_{(\neg\alpha)\text{tt}} A$. □

3 Autoreducibility by Self-Reducibility

We use the notion of self-reducibility to show the autoreducibility of complete sets. Observe that for NL and P there exist self-reducible, $\leq^{\log}_{\text{m}}$-complete sets, which follows from the characterizations in terms of nondeterministic and alternating logspace machines. In this section we argue that this implies that all complete sets for NL and P are autoreducible (not only for $\leq^{\log}_{\text{m}}$, but also several other logspace reducibility notions). For example, we obtain that all $\leq^{\log}_{\text{tt}}$-complete sets for NL and P are $\leq^{\log}_{\text{tt}}$-autoreducible.

The following notion of $\leq^{\log}_{\text{T}}$-self-reducibility is a restriction of $\leq^{\log}_{\text{T}}$-autoreducibility which demands that oracle queries have a certain structure.

Definition 3.1 ([1]) *A is $\leq^{\log}_{\mathrm{T}}$-self-reducible if there is a logspace oracle Turing machine M that accepts A with oracle A such that on input x, the queries asked by M are of the same length as x, lexicographically smaller than x, and differ from x at most in the last $\log|x|$ symbols.*

The notions of $\leq^{\log}_{\mathrm{tt}}$-self-reducibility and $\leq^{\log}_{k\text{-tt}}$-self-reducibility are defined analogously. By Proposition 2.1, a set is $\leq^{\log}_{\mathrm{T}}$-self-reducible if and only if it is $\leq^{\log}_{\mathrm{tt}}$-self-reducible.

There is a technical difficulty in defining self-reducibility for disjunctive and conjunctive truth-table reducibilities. In these cases, the reduction cannot simply accept or reject, but must generate queries that represent the answer. However, a self-reduction on input $x = y0^{|y|}$ is not allowed to make any query, since the last $\log|x|$ symbols of x already reached the minimal possible value. Therefore, in the definition below the self-reduction may accept or reject without asking any queries.

Definition 3.2 *A set A is $\leq^{\log}_{\mathrm{dtt}}$-self-reducible if there is a logspace-computable function f whose values can be 0, 1, or a list of words $(y_1, \ldots y_n)$ where $n \geq 1$ such that the following holds: If $f(x) \in \{0,1\}$, then $c_A(x) = f(x)$. Otherwise, it holds that $f(x) = (y_1, \ldots y_n)$ such that the y_i are of the same length as x, are lexicographically smaller than x, differ from x at most in the last $\log|x|$ symbols, and $x \in A \Leftrightarrow (c_A(y_1) \vee \cdots \vee c_A(y_n))$. If n is bounded by some constant k, then A is $\leq^{\log}_{k\text{-dtt}}$-self-reducible. The notions of $\leq^{\log}_{\mathrm{ctt}}$-self-reducibility and $\leq^{\log}_{k\text{-ctt}}$-self-reducibility are defined analogously. A is $\leq^{\log}_{\mathrm{m}}$-self-reducible if it is $\leq^{\log}_{1\text{-dtt}}$-self-reducibile.*

Each nontrivial self-reducible set B is autoreducible. The lemma below says that if a set A is in some sense equivalent to a self-reducible set B, then also A is autoreducible. The proof for the easiest case of $\leq^{\log}_{\mathrm{dtt}}$ works by first executing the reduction $A \leq^{\log}_{\mathrm{m}} B$ and then it iteratively follows exactly one of the self-reducibility-queries of B. For each of these queries, the $\leq^{\log}_{\mathrm{dtt}}$-reduction to A is computed. If x does not occur among the queries of this reduction, we can complete the reduction. Otherwise, we continue the self-reduction on this path, as it positively depends on whether $x \in A$.

Lemma 3.3 *Let $l \geq 1$ and A, B be sets.*

1. *$A \leq^{\log}_{\mathrm{m}} B \leq^{\log}_{\mathrm{tt}} A$ and B is $\leq^{\log}_{\mathrm{tt}}$-self-reducible $\implies$ A is $\leq^{\log}_{\mathrm{tt}}$-autoreducible.*
2. *$A \leq^{\log}_{\mathrm{m}} B \leq^{\log}_{1\text{-tt}} A$ and B is $\leq^{\log}_{2\text{-tt}}$-self-reducible $\implies$ A is $\leq^{\log}_{2\text{-tt}}$-autoreducible.*
3. *$A \leq^{\log}_{\mathrm{m}} B \leq^{\log}_{\mathrm{dtt}} A$ and B is $\leq^{\log}_{\mathrm{dtt}}$-self-reducible $\implies$ A is $\leq^{\log}_{\mathrm{dtt}}$-autoreducible.*
4. *$A \leq^{\log}_{\mathrm{m}} B \leq^{\log}_{l\text{-dtt}} A$, B is $\leq^{\log}_{2\text{-dtt}}$-self-reducible $\implies$ A is $\leq^{\log}_{2l\text{-dtt}}$-autoreducible.*

Proof 1. Let $A \leq^{\log}_{\mathrm{m}} B$ via f. Let h^q_x be the unary Boolean function that results from the reduction $B \leq^{\log}_{\mathrm{tt}} A$ on input q, when all queries $r \neq x$ are substituted by their answers $c_A(r)$. Hence h^q_x is the unary Boolean function with the property $c_B(q) = h^q_x(c_A(x))$. Moreover, let M be the oracle Turing machine performing the $\leq^{\log}_{\mathrm{tt}}$-self-reduction of B. The following oracle machine computes a $\leq^{\log}_{\mathrm{T}}$-autoreduction for A on input x:

1. $s := f(x)$, $\beta := \mathrm{id}$
2. let $q_1, \ldots, q_k$ be the words queried by M on input s
3. if $h^{q_1}_x, \ldots, h^{q_k}_x$ are all constants then
4. return the result of M on s, where queries are answered according to $h^{q_1}_x, \ldots, h^{q_k}_x$
5. choose i such that $h^{q_i}_x$ is not constant
6. let $s := q_i$ and $\beta := h^{q_i}_x$
7. goto 2

Observe that this computation can be executed in logarithmic space without querying the input x. The machine only needs logarithmic space, since the queries q_i differ from $f(x)$ only in the last $\log|x|$ symbols. Note that the computation eventually stops in line 4, since s is always replaced by a lexicographically smaller string.

For the correctness note that in line 2 we always have $c_A(x) = \beta(c_B(s))$: It is true at the beginning of the computation and in line 5 it holds that $c_B(q_i) = h_x^{q_i}(c_A(x))$ and since $h_x^{q_i}$ is either non or id, we obtain $c_A(x) = h_x^{q_i}(c_B(q_i))$. This shows that A is $\leq_{\mathrm{T}}^{\log}$-autoreducible and hence $\leq_{\mathrm{tt}}^{\log}$-autoreducible by Proposition 2.1.

2. We argue analogously. Note that in this case $k \leq 2$ and we can check whether the functions $h_x^{q_i}$ are constants without asking any queries (since we consider $\leq_{1\text{-tt}}^{\log}$-reductions).

3. Let $h(q)$ be the set of words queried by the reduction $B \leq_{\mathrm{dtt}}^{\log} A$ on input q. In line 3 of the above algorithm, it suffices to check whether x belongs to at least one of the sets $h(q_1), \ldots, h(q_k)$. If so, say $x \in h(q_i)$, then we continue with $s := q_i$. Otherwise, we return $h(q_1) \cup \cdots \cup h(q_k)$.

For the correctness of the modified algorithm note that in line 2 we always have $c_A(x) = c_B(s)$: This is true at the beginning of the computation and in line 5 it holds that $c_A(x) = c_B(s) = c_B(q_1) \vee \cdots \vee c_B(q_k) \geq c_B(q_i) \geq c_A(x)$ and thus $c_A(x) = c_B(q_i)$.

4. We argue similar to 3., while observing that in this case $k \leq 2$ and $|h(q_1) \cup h(q_2)| \leq 2l$. □

Proposition 3.4 *Let $k \geq 1$.*

1. *There is a $\leq_{\mathrm{m}}^{\log}$-complete set for* NL *that is $\leq_{2\text{-dtt}}^{\log}$-self-reducible.*
2. *There is a $\leq_{\mathrm{m}}^{\log}$-complete set for* NL *that is $\leq_{2\text{-ctt}}^{\log}$-self-reducible.*
3. *There is a $\leq_{\mathrm{m}}^{\log}$-complete set for* P *that is $\leq_{2\text{-tt}}^{\log}$-self-reducible.*
4. *There is a $\leq_{\mathrm{m}}^{\log}$-complete set for* AC^k *(resp.,* SAC^k*,* NC^k*) that is $\leq_{\mathrm{tt}}^{\log}$-self-reducible.*

Proof The evaluation of Boolean circuits over binary OR (resp., AND) is $\leq_{\mathrm{m}}^{\log}$-complete for NL and $\leq_{2\text{-dtt}}^{\log}$-self-reducible (resp., $\leq_{2\text{-ctt}}^{\log}$-self-reducible). The evaluation of Boolean circuits over binary OR and AND is $\leq_{\mathrm{m}}^{\log}$-complete for P and is $\leq_{2\text{-tt}}^{\log}$-self-reducible.

For the classes AC^k, SAC^k, and NC^k, the following problem has the desired properties: For a given circuit C and a given number i determine whether the i-th gate of C has value 1. □

With Lemma 3.3 and Proposition 3.4 we show the autoreducibility of logspace complete sets.

Theorem 3.5 *Let $k \geq 1$.*

1. *All $\leq_{\mathrm{tt}}^{\log}$-complete sets for* NL*,* P*,* AC^k*,* SAC^k*, or* NC^k *are $\leq_{\mathrm{tt}}^{\log}$-autoreducible.*
2. *All $\leq_{1\text{-tt}}^{\log}$-complete sets for* NL *or* P *are $\leq_{2\text{-tt}}^{\log}$-autoreducible.*
3. *All $\leq_{\mathrm{dtt}}^{\log}$-complete sets for* NL *are $\leq_{\mathrm{dtt}}^{\log}$-autoreducible.*
4. *All $\leq_{\mathrm{ctt}}^{\log}$-complete sets for* NL *are $\leq_{\mathrm{ctt}}^{\log}$-autoreducible.*
5. *All $\leq_{k\text{-dtt}}^{\log}$-complete sets for* NL *are $\leq_{2k\text{-dtt}}^{\log}$-autoreducible.*
6. *All $\leq_{k\text{-ctt}}^{\log}$-complete sets for* NL *are $\leq_{2k\text{-ctt}}^{\log}$-autoreducible.*
7. *All $\leq_{\mathrm{m}}^{\log}$-complete sets for* NL *are $\leq_{2\text{-dtt}}^{\log}$- and $\leq_{2\text{-ctt}}^{\log}$-autoreducible.*

Proof Let A be $\leq_{\mathrm{tt}}^{\log}$-complete for NL (resp., P). By Proposition 3.4, there exists a $\leq_{2\text{-tt}}^{\log}$-self-reducible set B that is $\leq_{\mathrm{m}}^{\log}$-complete for NL (resp., P). By Lemma 3.3, A is $\leq_{\mathrm{tt}}^{\log}$-autoreducible.

The remaining statements are shown analogously. The statements about conjunctive autoreducibility are obtained by the Propositions 2.5 and 2.6 and the fact that NL is closed under complement. □

We now use self-reducibility to show that for restricted $\leq_{2\text{-tt}}^{\log}$ reducibility it holds that complete sets for NL and P are autoreducible.

Theorem 3.6 *All sets that are* $\leq^{\log}_{(\to)\mathrm{tt}}$*-complete for* NL *(resp.,* P*) are* $\leq^{\log}_{\mathrm{btt}}$*-autoreducible.*

Proof We argue for P (the case of NL is shown analogously). Let $\alpha = (\to)$, let A be $\leq^{\log}_{\alpha\mathrm{tt}}$-complete for P, and let M be some alternating machine that accepts A. Consider the set

$$R = \{\langle x, C\rangle \mid \text{configuration } C \text{ is the root of an accepting subtree in } M(x)\}$$

and observe that $R, \overline{R} \in \mathrm{P}$. Hence there are logspace-computable functions f, g that show $R, \overline{R} \leq^{\log}_{\alpha\mathrm{tt}} A$. Let C_0 denote the start configuration of M on input x. We may assume:

- $f(\langle x, C_0\rangle)$ queries $(a \to x)$ for some fixed $a \in A$
- for every stop configuration C_s, neither $f(\langle x, C_s\rangle)$ nor $g(\langle x, C_s\rangle)$ queries x

The following algorithm traverses $M(x)$ and keeps the invariant $x \in A \iff \langle x, C\rangle \in R$.

1. $C := C_0$
2. $(C_1, C_2) :=$ successor configurations of C in $M(x)$
3. $\beta :=$ type of node C in $M(x)$
4. if $\beta = \vee$ then:
5. let $f(\langle x, C_1\rangle) = (y_1 \to y_2)$ and $f(\langle x, C_2\rangle) = (y_3 \to y_4)$
6. if $x \notin \{y_1, y_2, y_3, y_4\}$ then return $((y_1 \to y_2) \vee (y_3 \to y_4))$
7. else if $x \in \{y_1, y_3\}$ then return some fixed value $a \in A$
8. else set $C := C_i$, where $y_{2i} = x$, and continue with step 2
9. else $(\beta = \wedge)$:
10. let $g(\langle x, C_1\rangle) = (y_1 \to y_2)$ and $g(\langle x, C_2\rangle) = (y_3 \to y_4)$
11. if $x \notin \{y_1, y_2, y_3, y_4\}$ then return $(\neg(y_1 \to y_2) \wedge \neg(y_3 \to y_4))$
12. else if $x \in \{y_2, y_4\}$ then return some fixed value $b \notin A$
13. else set $C := C_i$, where $y_{2i-1} = x$, and continue with step 2

After line 1, the invariant clearly holds. Assume that we reach line 8, so $\beta = \vee$ and $f(\langle x, C_i\rangle) = (y_{2i-1} \to x)$. If $x \notin A$, then $\langle x, C\rangle \notin R$ and hence $\langle x, C_i\rangle \notin R$. If $x \in A$, then $c_A(y_{2i-1}) \to c_A(x)$ is true and hence $\langle x, C_i\rangle \in R$. So after setting $C := C_i$, the invariant still holds. Assume now that we reach line 13. so $\beta = \wedge$ and $g(\langle x, C_i\rangle) = (x \to y_{2i})$. If $x \in A$, then $\langle x, C\rangle \in R$ and hence $\langle x, C_i\rangle \in R$. If $x \notin A$, then $c_A(x) \to c_A(y_{2i})$ is true and hence $\langle x, C_i\rangle \in \overline{R}$. So after setting $C := C_i$, the invariant still holds.

Since the invariant always holds, the return statements in line 6 and line 11 are obviously correct. So suppose we stop in line 7 and it holds that $x \in \{y_1, y_3\}$, hence $f(\langle x, C_i\rangle) = (x \to y_{2i})$. Then $x \notin A$ leads to the following contradiction: if $x \notin A$, then $(x \to y_{2i})$ is true, hence $\langle x, C_i\rangle \in R$, which implies $\langle x, C\rangle \in R$, which implies $x \in A$. Suppose now that we stop in line 12 and it holds that $x \in \{y_2, y_4\}$, hence $g(\langle x, C_i\rangle) = (y_{2i-1} \to x)$. Then $x \in A$ leads to the following contradiction: if $x \in A$, then $(y_{2i-1} \to x)$ is true, hence $\langle x, C_i\rangle \in \overline{R}$, which implies $\langle x, C\rangle \notin R$, which implies $x \notin A$.

Observe that the algorithm will eventually return, because we assumed that stop configurations do not query x. □

4 Autoreducibility by Local Checkability of Computations

If we represent computations of NL, P, and Δ^{p}_k machines in tableaus or configuration graphs, we can locally check the consistency of these computations. This technique allows us to show

(i) the $\leq^{\log}_{\log\text{-}\mathrm{T}}$-autoreducibility of all $\leq^{\log}_{\mathrm{btt}}$-complete sets for NL, P, and Δ^{p}_k, and

(ii) the $\leq^{\log}_{\text{btt}}$-autoreducibility of all $\leq^{\log}_{\alpha\text{tt}}$-complete sets for NL and P, for all 2-ary Boolean α.

Using techniques by Buhrman et al. [5] we show that not all $\leq^{\log}_{\text{btt}}$-complete sets for PSPACE are $\leq^{\log}_{\text{btt}}$-autoreducible. Hence certain improvements of (i) and (ii) are difficult to obtain: Improving (i) to $\leq^{\log}_{\text{btt}}$-autoreducibility for P (resp., Δ^{p}_k) implies P $\neq$ PSPACE (resp., $\Delta^{\text{p}}_k \neq$ PSPACE), and improving (ii) to 3-ary Boolean α for P implies P $\neq$ PSPACE.

Moreover, we obtain that resolving the $\leq^{\log}_{\text{btt}}$-autoreducibility of $\leq^{\log}_{\text{btt}}$-complete sets for P, NP, PP, Δ^{p}_k, Σ^{p}_k, or Π^{p}_k leads to unknown separations of complexity classes.

Theorem 4.1 *1. All $\leq^{\log}_{\text{btt}}$-complete sets for* NL *are $\leq^{\log}_{\text{log-T}}$-autoreducible.*
2. All $\leq^{\log}_{\text{btt}}$-complete sets for P *are $\leq^{\log}_{\text{log-T}}$-autoreducible.*

Proof We show item 2 (the proof of 1 is analoguous), so let A be $\leq^{\log}_{\text{btt}}$-complete for P. Recall that P = AL and let M be an alternating logspace TM with $L(M) = A$. Consider the set

$$R = \{\langle x, C\rangle \mid \text{configuration } C \text{ is the root of an accepting subtree in the configuration graph of } M \text{ on input } x\}.$$

Note that the configuration graph of M on input x also contains those configurations of M on input x of length $O(\log |x|)$ that are not reachable from the start configuration of M on input x. Observe that $R \in$ AL, hence $R \leq^{\log}_{\text{btt}} A$ via some logspace oracle Turing machine. Let h^C_x be the unary Boolean function that results from the reduction $R \leq^{\log}_{\text{btt}} A$ on input $\langle x, C\rangle$ when all queries $q \neq x$ are substituted by their answers $c_A(q)$. Hence $c_R(\langle x, C\rangle) = h^C_x(c_A(x))$. We may assume that for every input x:

- $h^{C_0}_x = \text{id}$, where C_0 is the start configuration of M on input x
- $h^{C_s}_x$ is constant for every stop configuration C_s of M on input x

So for every x, there exists a configuration C in the configuration graph of M on input x with successors C_1 and C_2 such that h^C_x is not constant, while $h^{C_1}_x$ and $h^{C_2}_x$ are constant. Let $C(x)$ be the smallest such configuration and let $B = \{\langle x, i\rangle \mid \text{the } i\text{-th bit of } C(x) \text{ is one}\}$. Observe that $B \in$ AL and hence $B \leq^{\log}_{\text{btt}} A$. Thus the function $g(x) \stackrel{df}{=} C(x)$ can be computed by a logspace oracle Turing machine with oracle A such that on input x, the machine queries $O(\log |x|)$ words.

We describe the $\leq^{\log}_{\text{log-T}}$-autoreduction on input x:

1. compute $C(x)$ using g's machine with oracle $A \cup \{x\}$
2. verify that C is a configuration with successors C_1, C_2 of M on input x such that
   - h^C_x is not constant and
   - $h^{C_1}_x$ and $h^{C_2}_x$ are constant

   otherwise return 0
3. if C is an existential configuration, then return $h^C_x(h^{C_1}_x \vee h^{C_2}_x)$
   if C is a universal configuration, then return $h^C_x(h^{C_1}_x \wedge h^{C_2}_x)$

Note that in the case $x \notin A$, it could be that in step 1, the algorithm does not correctly compute the configuration $C(x)$. However, the algorithm never rejects erroneously in step 2, since the verification in step 2 always passes for the correct configuration $C(x)$.

If the algorithm reaches step 3, then h^C_x is not constant while $h^{C_1}_x$ and $h^{C_2}_x$ are constant. Hence for an existential (resp., universal) C it holds that $h^{C_1}_x \vee h^{C_2}_x = c_R(\langle x, C\rangle) = h^C_x(c_A(x))$ (resp., $h^{C_1}_x \wedge h^{C_2}_x = c_R(\langle x, C\rangle) = h^C_x(c_A(x))$). So the algorithm accepts if and only if $x \in A$ (which follows from the fact that either $h^C_x = \text{id}$ or $h^C_x = \text{non}$).

The algorithm works in logspace and queries $O(\log |x|)$ words. Hence A is $\leq^{\log}_{\text{log-T}}$-autoreducible. □

Theorem 4.1 raises the question of whether one can improve this result to obtain a *non-adaptive* autoreduction. The next theorem and its corollary show that at least in the case of P (Theorem 4.1.2) such an improvement is difficult to obtain as it separates P from PSPACE. The proof is based on an idea by Buhrman et al. [5] who show that not all $\leq^{\text{p}}_{\text{btt}}$-complete sets for EXP are $\leq^{\text{p}}_{\text{btt}}$-autoreducible.

Theorem 4.2 *For every k there is a $\leq^{\log}_{\text{2-T}}$-complete set for* PSPACE *that is not $\leq^{\log}_{n^k\text{-tt}}$-autoreducible. In particular, not all $\leq^{\log}_{\text{btt}}$-complete sets for* PSPACE *are $\leq^{\log}_{\text{btt}}$-autoreducible.*

Proof We construct a set $A \in$ PSPACE that contains two tracks, one track encodes a $\leq^{\log}_{\text{m}}$-complete set K and the other one diagonalizes against all $\leq^{\log}_{n^k\text{-tt}}$-reductions. Because these reductions are able to ask polynomially large queries, a straightforward encoding of K into A fails, since the diagonalization would have to decide K in the simulation of the reduction. The trick by Buhrman et al. is to include switch points that swap the tracks and always send the reduction onto the track that contains barriers.

Let $\Sigma = \{0,1\}$ and let $\{M_i\}_i$ be an enumeration of all $\leq^{\log}_{n^k\text{-tt}}$-autoreductions such that the computation of M_i on x can be simulated in space $i \log |x|$ and asks queries of length at most $|x|^i$. Furthermore, we may assume that M_i on inputs of length n queries exactly n^k distinct words. The diagonalization against M_i will be done on input $0^{t(i)}$ for $t(i) = 2^{2^{2^i}}$. This ensures that the length of the largest query on this input is $(t(i))^i < t(i+1)$.

Let $K \subseteq \{0,1\}^*$ be $\leq^{\log}_{\text{m}}$-complete for PSPACE such that $K \cap \Sigma^{t(i)-1} = \emptyset$ for all $i \in \mathbb{N}$. Furthermore, we assume that $K \in$ DSPACE(n).

We define $A_{-1} = \emptyset$ and iteratively construct the stages $A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots$ such that $A = \bigcup_{i\in\mathbb{N}} A_i$.

Stage i: Let $n = t(i)$ and $Q = \{q_1, \ldots, q_{n^k}\}$ be the words queried by $M_i(0^n)$. Let $L = Q \cap 1\Sigma^{\geq n}$ and $R = Q \cap 0\Sigma^{\geq n}$ be the "left" and "right" queries and subdivide them further into blocks according to their length as follows: $L_j = L \cap \{x \mid n^{2^{j-1}} < |x| \leq n^{2^j}\}$, $R_j = R \cap \{x \mid n^{2^{j-1}} < |x| \leq n^{2^j}\}$ for $1 \leq j \leq \lceil \log i \rceil$. We claim that at least one of the following holds:

$X)$ $\forall l_1 \subseteq L_1 \exists r_1 \subseteq R_1 \forall l_2 \subseteq L_2 \exists r_2 \subseteq R_2 \ldots \forall l_{\lceil \log i \rceil} \subseteq L_{\lceil \log i \rceil} \exists r_{\lceil \log i \rceil} \subseteq R_{\lceil \log i \rceil} :$
$M_i^{A_{i-1} \cup l_1 \cup \cdots \cup l_{\lceil \log i \rceil} \cup r_1 \cup \cdots \cup r_{\lceil \log i \rceil}}(0^n)$ rejects

$Y)$ $\forall r_1 \subseteq R_1 \exists l_1 \subseteq L_1 \forall r_2 \subseteq R_2 \exists l_2 \subseteq L_2 \ldots \forall r_{\lceil \log i \rceil} \subseteq R_{\lceil \log i \rceil} \exists l_{\lceil \log i \rceil} \subseteq L_{\lceil \log i \rceil} :$
$M_i^{A_{i-1} \cup l_1 \cup \cdots \cup l_{\lceil \log i \rceil} \cup r_1 \cup \cdots \cup r_{\lceil \log i \rceil}}(0^n)$ accepts

This follows by basic logic: $\neg X$ has the structure $\exists l_1 \forall r_1 \cdots : M_i(0^n)$ accepts. From this we obtain Y, since $\exists l_i \forall r_i H$ implies $\forall r_i \exists l_i H$ for all H.

If X holds, then we define $A_i = \{0^n\} \cup \{1x \mid x \in K \wedge t(i) \leq |x| < t(i+1) - 1\} \cup r_1 \cup \cdots \cup r_{\lceil \log i \rceil}$, where the l_j result from K and the r_j are chosen in the order $r_1, \ldots, r_{\lceil \log i \rceil}$ as the lexicographically minimal $r_j \subseteq R_j$ such that

$$\forall l_{j+1} \subseteq L_{j+1} \exists r_{j+1} \subseteq R_{j+1} \forall l_{j+2} \subseteq L_{j+2} \cdots : M_i^{A_{i-1} \cup l_1 \cup \cdots \cup l_{\lceil \log i \rceil} \cup r_1 \cup \cdots \cup r_{\lceil \log i \rceil}}(0^n) \text{ rejects.}$$

If X does not hold, then Y holds and we analogously define $A_i = \{0x \mid x \in K \wedge t(i) \leq |x| < t(i+1) - 1\} \cup l_1 \cup \cdots \cup l_{\lceil \log i \rceil}$. We show following:

1. $K \leq^{\log}_{\text{2-T}} A$
2. $A \in$ PSPACE

3. A is not $\leq^{\log}_{n^k\text{-tt}}$-autoreducible

1.: Let x be some input and $i \in \mathbb{N}$ such that $t(i) \leq |x| < t(i+1) - 1$ (all other inputs are trivial). First we query $0^{t(i)} \in A$. If the answer is "yes", then we return the answer to the query $1x \in A$. Otherwise, we return the answer to the query $0x \in A$.

2.: By induction on i we show that $A \in \mathrm{DSPACE}(n^{k+1})$. The induction base is trivial, since it is a finite case. For the induction step, we first show that we can decide inputs of the form $0^{t(i)}$ in $\mathrm{DSPACE}(n^{k+1})$ and then we show it for inputs of length between $t(i)$ and $t(i+1)$.

For inputs of the form 0^n where $n = t(i)$ we only have to decide whether situation X holds or not. Note that we cannot write down all queries of $M_i(0^n)$, as some may be too large. We can, though, record the first symbol and the length of each query and thus determine the set R_j or L_j it lies in or whether it is shorter than n. Queries that are shorter than n can even be answered by the induction hypothesis. Furthermore, queries can be identified by their sequence number, since all queries are distinct and the reduction is non-adaptive. This means that we can find out whether situation X holds or not by a depth-first-search of the expression tree: Each node in the tree can be encoded as a string of answers to the at most n^k queries that are larger than n and thus takes space n^k. The value of a leaf can be determined by simulating $M_i(0^n)$ on this set of answers, which uses only $i \log n$ space. In total, we can decide 0^n at least in space n^{k+1}.

Now let x be an input such that $t(i) < |x| < t(i+1)$. Without loss of generality, assume that X holds (we can again determine that). If $x \in 1\Sigma^*$ just simulate the algorithm that decides K in linear space. If x starts with 0, but is not a query of $M_i(0^{t(i)})$, we can reject. Otherwise, there is some j such that $x \in R_j$. We now recursively compute all sets $r_1, \ldots, r_{j-1}$ and $l_1, \ldots, l_j$, which is possible in $\mathrm{DSPACE}(n^{k+1})$, since $n > t(i)^{2^{j-1}}$ is now large enough. We again search the expression tree for X in depth-first-manner, but now only for the given values of $r_1, \ldots, r_{j-1}$ and $l_1, \ldots, l_j$ and we search for the lexicographically smallest value of r_j for which X holds. We can find out the sequence number of the query x, since our input x is large enough and accept if and only if $x \in r_j$.

3.: For every i we ensured that $M_i^A(0^{t(i)})$ accepts if and only if $0^{t(i)} \notin A$, and thus A is not $\leq^{\log}_{n^k\text{-tt}}$-autoreducible. □

It follows that improving Theorem 4.1.2 to $\leq^{\log}_{\log\text{-tt}}$-autoreducibility or $\leq^{\log}_{n^c\text{-tt}}$-autoreducibility separates P from PSPACE. At this point we also observe two similar statements (4.3.2 and 4.3.3) which will explain the difficulty of improving the Theorems 4.8 and 4.11 below.

Corollary 4.3 *Let $c \geq 1$ and $k \geq 2$.*

1. *If all $\leq^{\log}_{\mathrm{btt}}$-complete sets for P are $\leq^{\log}_{n^c\text{-tt}}$-autoreducible, then* $\mathrm{P} \neq \mathrm{PSPACE}$.
2. *If all $\leq^{\log}_{3\text{-tt}}$-complete sets for P are $\leq^{\log}_{\mathrm{btt}}$-autoreducible, then* $\mathrm{P} \neq \mathrm{PSPACE}$.
3. *If all $\leq^{\log}_{\mathrm{btt}}$-complete sets for Δ_k^{p} are $\leq^{\log}_{n^c\text{-tt}}$-autoreducible, then $\Delta_k^{\mathrm{p}} \neq \mathrm{PSPACE}$.*

Trivially, all $\leq^{\log}_{\mathrm{btt}}$-complete sets for L are $\leq^{\log}_{\mathrm{btt}}$-autoreducible and $\leq^{\log}_{\mathrm{btt}}$-mitotic. This shows that proving or refuting Theorem 4.2 for class like P, NP, or Δ_k^{p} leads to unknown separations of complexity classes.

Corollary 4.4 *Let $k \geq 1$ and $\mathcal{C} \in \{\mathrm{P}, \mathrm{NP}, \mathrm{PP}, \Delta_k^{\mathrm{p}}, \Sigma_k^{\mathrm{p}}, \Pi_k^{\mathrm{p}}\}$.*

1. *If all $\leq^{\log}_{\mathrm{btt}}$-complete sets for $\mathcal{C}$ are $\leq^{\log}_{\mathrm{btt}}$-autoreducible, then $\mathcal{C} \neq \mathrm{PSPACE}$, otherwise $\mathcal{C} \neq \mathrm{L}$.*
2. *If all $\leq^{\log}_{\mathrm{btt}}$-complete sets for $\mathcal{C}$ are $\leq^{\log}_{\mathrm{btt}}$-mitotic, then $\mathcal{C} \neq \mathrm{PSPACE}$, otherwise $\mathcal{C} \neq \mathrm{L}$.*

Using another technique by Buhrman at al. [5] we generalize the proof of Theorem 4.1 and obtain similar results for the classes Δ_k^{p}. By Corollary 4.3.3, improving the theorem to $\leq^{\log}_{n^c\text{-tt}}$-autoreducibility or even $\leq^{\log}_{\mathrm{btt}}$-autoreducibility separates Δ_k^{p} from PSPACE.

For $z = z_1 \cdots z_m \in \{0,1\}^m$ and an $(m+n)$-ary Boolean formula φ with variables $y_1, \ldots, y_{m+n}$, $\varphi(z)$ denotes the n-ary Boolean formula obtained by substituting y_i for z_i where $i \in \{1, \ldots, m\}$.

Definition 4.5 *Let $k \geq 0$. If k is even, let $Q := \forall, R := \exists$, and $Q := \exists, R := \forall$ otherwise.*

1. Σ_k-3SAT $:= \{\varphi \mid \varphi$ *is a Boolean formula in 3-DNF if k is even and in 3-CNF otherwise, has km variables, and* $\exists z_k \in \{0,1\}^m \forall z_{k-1} \in \{0,1\}^m \cdots Q z_1 \in \{0,1\}^m\ \varphi(z_k, \ldots, z_1) = 1\}$
2. Π_k-3SAT $:= \{\varphi \mid \varphi$ *is a Boolean formula in 3-CNF if k is even and in 3-DNF otherwise, has km variables, and* $\forall z_k \in \{0,1\}^m \exists z_{k-1} \in \{0,1\}^m \cdots R z_1 \in \{0,1\}^m\ \varphi(z_k, \ldots, z_1) = 1\}$
3. Σ_k-EMW $:= \{\varphi \in \Sigma_k$-3SAT $\mid$ *if $k \geq 1$ and φ has km variables, then the minimal $z_k \in \{0,1\}^m$ such that $\varphi(z_k) \in \Pi_{k-1}$-3SAT is even*$\}$

Note that Σ_0-EMW $= \Sigma_0$-3SAT $= \{\varphi \mid \varphi$ is a true Boolean sentence in 3-DNF$\}$.

Theorem 4.6 ([13]) *For $k \geq 1$ and $L \in \Sigma_k^{\mathrm{p}}$ there exists an $f \in$ FL such that for all n, $f(2^n) = \varphi_n$ is an $(n+km)$-ary Boolean formula such that for all $x \in \{0,1\}^n$, $x \in L \Leftrightarrow \varphi_n(x) \in \Sigma_k$-3SAT.*

Theorem 4.7 *For $k \geq 1$, Σ_k-EMW is $\leq^{\log}_{\mathrm{m}}$-complete for $\Delta_{k+1}^{\mathrm{p}}$.*

Proof Wagner [12] shows that the problem of whether the minimal satisfying assignment of a given Boolean formula is even is $\leq^{\mathrm{p}}_{\mathrm{m}}$-complete for Δ_2^{p}. The same technique combined with Theorem 4.6 shows that Σ_k-EMW is $\leq^{\log}_{\mathrm{m}}$-complete for $\Delta_{k+1}^{\mathrm{p}}$. □

Theorem 4.8 *For $k \geq 1$, all sets A that are $\leq^{\log}_{\mathrm{btt}}$-complete for $\Delta_{k+1}^{\mathrm{p}}$ are $\leq^{\log}_{\text{log-T}}$-autoreducible.*

Proof For every $h \in \mathrm{F}\Delta_{k+1}^{\mathrm{p}}$ we define a $\leq^{\log}_{\mathrm{btt}}$-reduction R_h and functions $h^+, h^- \in \mathrm{F}\Delta_{k+1}^{\mathrm{p}}$ as follows: Choose the smallest $c \in \mathbb{N}$ such that $|h(x)| \leq |x|^c + c$. Let B_h be the set of pairs (x,i) such that bit i in $h(x)$'s binary representation is 1. $B_h \in \Delta_{k+1}^{\mathrm{p}}$ and hence $\leq^{\log}_{\mathrm{btt}}$-reduces to A via a machine R_h, where R_h is the lexicographically first such machine. So the values $R_h^A(x,i)$ for $i < |x|^c + c$ tell us the binary representation of $h(x)$. If the query x is not allowed, we can only compute the following candidates for $h(x)$: $h^+(x) \stackrel{df}{=} \sum_{i<|x|^c+c} 2^i \cdot R_h^{A\cup\{x\}}(x,i)$ and $h^-(x) \stackrel{df}{=} \sum_{i<|x|^c+c} 2^i \cdot R_h^{A-\{x\}}(x,i)$. If $x \in A$, then $h^+(x) = h(x)$. If $x \notin A$, then $h^-(x) = h(x)$.

By Theorem 4.7, there is an $f \in$ FL and a polynomial r such that $|f(x)| < r(|x|)$ and $x \in A \iff f(x) \in \Sigma_k$-EMW for all x. Let $\varphi_x := f(x)$. We may assume that φ_x has the right format (3-DNF if k is even, 3-CNF otherwise), has the m-bit variables $y_k, \ldots, y_1$, and for all i and all $z_k, \ldots, z_1 < 2^m$, the value $\varphi(z_k, \ldots, z_1)$ is independent of z_i's highest bit. For every i, let $\varphi_{x,i} := \varphi_x$ if $(k-i)$ is even, and $\varphi_{x,i} := \neg\varphi_x$ otherwise. For $i = k, \ldots, 1$, we define:

$$
\begin{aligned}
z_i(x) &:= \min(\{z < 2^m \mid \varphi_{x,i}(\overline{z}_k, \ldots, \overline{z}_{i+1}, z) \in \Pi_{i-1}\text{-3SAT}\} \cup \{2^{m-1}\}) \\
s_i(x) &:= \max(\{j \leq m \mid z_i^+(x) \text{ and } z_i^-(x) \text{ differ at the } j\text{-th bit from right}\} \cup \{0\}) \\
\overline{z}_i(x) &:= \min(z_i^+(x), z_i^-(x))
\end{aligned}
$$

For fixed i, $\varphi_{x,i}$ can be computed in space $O(\log(|x|))$ and $z_i, s_i, \overline{z}_i \in \mathrm{F}\Delta_{k+1}^{\mathrm{p}}$.

$$
\begin{aligned}
F_i &:= \{x \mid \varphi_{x,i}(\overline{z}_k(x), \ldots, \overline{z}_{i+1}(x)) \in \Sigma_i\text{-3SAT}\} \\
E_i &:= \{x \mid \varphi_{x,i}(\overline{z}_k(x), \ldots, \overline{z}_{i+1}(x)) \in \Sigma_i\text{-EMW}\}
\end{aligned}
$$

Observe that $F_i, E_i \in \Delta^{\mathrm{p}}_{k+1}$ and $x \in A \iff \varphi_x \in \Sigma_k\text{-EMW} \iff x \in E_k$. So the theorem is implied by the following statement, which we show by induction.

$$F_i, E_i \leq^{\log}_{\log\text{-T}} A \text{ for } i = 0, \ldots, k, \text{ where on input } x \text{ the reduction does not query } x.$$

First, let $i = 0$. Since $E_0 = F_0$, it suffices to argue for F_0. If k is even, then $\varphi_{x,0} = \varphi_x$, and φ_x is in 3-DNF. If k is odd, then $\varphi_{x,0} = \neg\varphi_x$, and φ_x is in 3-CNF. Hence in both cases, after moving the negation to the literals, $\varphi_{x,0}$ is in 3-DNF. Define

$$s(x) := \min(\{j < r(|x|) \mid \text{conjunction } j \text{ in } \varphi_{x,0}(\overline{z}_k, \ldots, \overline{z}_1) \text{ is satisfied}\} \cup \{r(|x|)\})$$

and note that $s \in \mathrm{F}\Delta^{\mathrm{p}}_{k+1}$, hence $s^+(x)$ and $s^-(x)$ are computable in logarithmic space with $O(\log|x|)$ queries to $A - \{x\}$. So $x \in F_0 \iff \varphi_{x,0}(\overline{z}_k, \ldots, \overline{z}_1) = 1 \iff s^+(x)$ or $s^-(x)$ point to a conjunction in $\varphi_{x,0}(\overline{z}_k, \ldots, \overline{z}_1)$ that is satisfied. We argue that the right-hand side of this equivalence can be tested with $O(\log|x|)$ queries to $A - \{x\}$: Both conjunctions consist of 3 literals. The value of each such literal is determined by one bit of some $\overline{z}_j(x)$. The index j and the position of these bits can be determined in logarithmic space (without oracle queries), since φ_x is computable in logarithmic space. Using $s^+_j(x)$, $s^-_j(x)$, and the reduction R_{s_j} we can determine the value of each bit in $\overline{z}_j(x)$ with $O(\log|x|)$ queries to $A - \{x\}$.

For the induction step, suppose the claim holds for some $i < k$. We show the claim for $i+1$. On input x, we determine $s^+_{i+1}(x)$ and $s^-_{i+1}(x)$ with $O(\log|x|)$ queries to $A - \{x\}$.

Case 1: $s^+_{i+1}(x) \neq s^-_{i+1}(x)$.
Without loss of generality we assume $s^+_{i+1}(x) > s^-_{i+1}(x)$. Using $R_{z_{i+1}}$ we can test with $O(\log|x|)$ queries to $A \cup \{x\}$ whether $z^+_{i+1}(x)$ and $z^-_{i+1}(x)$ differ at the $s^+_{i+1}(x)$-th bit from right. If this holds, then $s_{i+1}(x) \neq s^-_{i+1}(x)$ and $x \in A$. Otherwise, $s_{i+1}(x) \neq s^+_{i+1}(x)$ and hence $x \notin A$. Since $F_{i+1}, E_{i+1} \in \Delta^{\mathrm{p}}_{k+1}$ and we know $c_A(x)$, we can determine with $O(1)$ queries to $A - \{x\}$ whether $x \in F_{i+1}$ and whether $x \in E_{i+1}$.

Case 2: $s^+_{i+1}(x) = s^-_{i+1}(x) = s_{i+1}(x) = 0$.
In this case, $z_{i+1}(x) = z^+_{i+1}(x) = z^-_{i+1}(x)$, hence

$$x \in F_{i+1} \iff \varphi_{x,i+1}(\overline{z}_k(x), \ldots, \overline{z}_{i+2}(x)) \in \Sigma_{i+1}\text{-3SAT} \iff z^-_{i+1}(x) < 2^{m-1}, \text{ and}$$
$$x \in E_{i+1} \iff \varphi_{x,i+1}(\overline{z}_k(x), \ldots, \overline{z}_{i+2}(x)) \in \Sigma_{i+1}\text{-EMW} \iff z^-_{i+1}(x) \text{ is even and } < 2^{m-1}.$$

The right-hand sides correspond to bit $m-1$ and bit 0 of $z^-_{i+1}(x)$, which can be determined via $R_{z_{i+1}}$ with a constant number of queries to $A - \{x\}$.

Case 3: $s^+_{i+1}(x) = s^-_{i+1}(x) = s_{i+1}(x) > 0$.
With $R_{z_{i+1}}$ we test with $O(\log|x|)$ queries to $A \cup \{x\}$ whether $z^+_{i+1}(x), z^-_{i+1}(x) \geq 2^{m-1}$. If so, then $z_{i+1}(x) = 2^{m-1}$, $\varphi_{x,i+1}(\overline{z}_k, \ldots, \overline{z}_{i+2}) \notin \Sigma_{i+1}$-3SAT, and $\varphi_{x,i+1}(\overline{z}_k, \ldots, \overline{z}_{i+2}) \notin \Sigma_{i+1}$-EMW, which means $x \notin F_{i+1}$ and $x \notin E_{i+1}$. Otherwise, $z^+_{i+1}(x) < 2^{m-1}$ or $z^-_{i+1}(x) < 2^{m-1}$.

$$x \in A \wedge z^+_{i+1}(x) < z^-_{i+1}(x) \implies \overline{z}_{i+1}(x) = z^+_{i+1}(x) = z_{i+1}(x) < 2^{m-1},\ \varphi_{x,i+1}(\overline{z}_k, \ldots, \overline{z}_{i+1}) \in \Pi_i\text{-3SAT}$$
$$x \in A \wedge z^+_{i+1}(x) > z^-_{i+1}(x) \implies z_{i+1}(x) = z^+_{i+1}(x),\ \overline{z}_{i+1}(x) = z^-_{i+1}(x),\ \varphi_{x,i+1}(\overline{z}_k, \ldots, \overline{z}_{i+1}) \notin \Pi_i\text{-3SAT}$$
$$x \notin A \wedge z^+_{i+1}(x) < z^-_{i+1}(x) \implies z_{i+1}(x) = z^-_{i+1}(x),\ \overline{z}_{i+1}(x) = z^+_{i+1}(x),\ \varphi_{x,i+1}(\overline{z}_k, \ldots, \overline{z}_{i+1}) \notin \Pi_i\text{-3SAT}$$
$$x \notin A \wedge z^+_{i+1}(x) > z^-_{i+1}(x) \implies \overline{z}_{i+1}(x) = z^-_{i+1}(x) = z_{i+1}(x) < 2^{m-1},\ \varphi_{x,i+1}(\overline{z}_k, \ldots, \overline{z}_{i+1}) \in \Pi_i\text{-3SAT}$$

If $z^+_{i+1}(x) < z^-_{i+1}(x)$ then let $v(x) := 1$ else let $v(x) := 0$. We obtain

$$\begin{aligned} x \in A &\iff \neg v(x) \oplus \varphi_{x,i+1}(\overline{z}_k, \ldots, \overline{z}_{i+1}) \in \Pi_i\text{-3SAT} \\ &\iff v(x) \oplus \varphi_{x,i}(\overline{z}_k, \ldots, \overline{z}_{i+1}) \in \Sigma_i\text{-3SAT} \iff v(x) \oplus x \in F_i. \end{aligned}$$

Note that $v(x) = R_{z_{i+1}}^{A-\{x\}}(x, s_{i+1}(x))$. Together with the induction hypothesis, we can test the right-hand side of the equivalence with $O(\log |x|)$ queries to $A - \{x\}$. So we obtain $c_A(x)$ which allows us to determine with $O(1)$ queries to $A-\{x\}$ whether $x \in F_{i+1}$ and whether $x \in E_{i+1}$. □

Corollary 4.9 *All $\leq_{\mathrm{tt}}^{\log}$-complete sets for $\Delta_{k+1}^{\mathrm{p}}$ are $\leq_{\mathrm{tt}}^{\log}$-autoreducible.*

Proof The proof of the $\leq_{\mathrm{btt}}^{\log}$-case also works for the $\leq_{\mathrm{tt}}^{\log}$-case, as the bounded number of queries is only used when we count the number of queries in the autoreduction. □

In the proof of the next lemma we locally check configuration graphs of nondeterministic (resp., alternating) logspace machines to obtain the $\leq_{\mathrm{btt}}^{\log}$-autoreducibility of $\leq_{(\leftrightarrow)\mathrm{tt}}^{\log}$-complete sets for NL and P. Together with the results from previous sections this shows Theorem 4.11, which states that for every fixed 2-ary Boolean function α, all $\leq_{\alpha\mathrm{tt}}^{\log}$-complete sets for NL and P are $\leq_{\mathrm{btt}}^{\log}$-autoreducible. By Theorem 4.2, improving the theorem to 3-ary Boolean functions separates P from PSPACE.

Lemma 4.10 *All sets $\leq_{(\leftrightarrow)\mathrm{tt}}^{\log}$-complete for* NL *or $\leq_{(\leftrightarrow)\mathrm{tt}}^{\log}$-complete for* P *are $\leq_{\mathrm{btt}}^{\log}$-autoreducible.*

Proof We will argue for P (the case of NL is shown analogously). Let A be $\leq_{(\leftrightarrow)\mathrm{tt}}^{\log}$-complete for P and let M be some alternating machine that shows $A \in \mathrm{P}$. Consider the set $R = \{\langle x, C\rangle \mid C \text{ is the root of an accepting subtree in } M(x)\}$ and observe that $R \in \mathrm{P}$, hence $R \leq_{(\leftrightarrow)\mathrm{tt}}^{\log} A$ via some logspace-computable function f. We assume:

- for every configuration C of $M(x)$, $f(\langle x, C\rangle)$ queries two distinct words
- for the start configuration C_0 of $M(x)$, $f(\langle x, C_0\rangle)$ queries x
- for every stop configuration C_s of $M(x)$, $f(\langle x, C_s\rangle)$ does not query x

Hence there must be a configuration C in the configuration graph of M on input x with successors C_1, C_2 such that $f(\langle x, C\rangle)$ queries x, and both $f(\langle x, C_1\rangle)$ and $f(\langle x, C_2\rangle)$ do not query x. We loop over all configurations until we find such a configuration C. Now we can determine $x \in A$ by at most five queries different from x. □

Theorem 4.11 *For every $\alpha\colon \{0,1\}^2 \to \{0,1\}$, all $\leq_{\alpha\mathrm{tt}}^{\log}$-complete sets for* NL *and all $\leq_{\alpha\mathrm{tt}}^{\log}$-complete sets for* P *are $\leq_{\mathrm{btt}}^{\log}$-autoreducible.*

Proof Let $\alpha\colon \{0,1\}^2 \to \{0,1\}$, $\mathcal{C} \in \{\mathrm{NL}, \mathrm{P}\}$, and let A be $\leq_{\alpha\mathrm{tt}}^{\log}$-complete for $\mathcal{C}$. We have the following cases:

- α is constant: there are no $\leq_{\alpha\mathrm{tt}}^{\log}$-complete sets for $\mathcal{C}$.
- $\alpha(x,y) = x$, $\alpha(x,y) = y$, $\alpha(x,y) = \neg x$, or $\alpha(x,y) = \neg y$: Theorem 3.5.2 shows that A is $\leq_{2\text{-tt}}^{\log}$-autoreducible.
- $\alpha \in \{\wedge, \vee, (\neg\wedge), (\neg\vee)\}$: Proposition 2.7 shows that $\leq_{\alpha\mathrm{tt}}^{\log}$-complete sets for $\mathcal{C}$ are $\leq_{2\text{-tt}}^{\log}$-autoreducible for $\alpha \in \{\wedge, \vee\}$. Since $\mathcal{C}$ is closed under complementation, the remaining functions follow by Proposition 2.8.
- $\alpha \in \{\leftrightarrow, \oplus\}$: Lemma 4.10 shows that for $\alpha = \leftrightarrow$, $\leq_{\alpha\mathrm{tt}}^{\log}$-complete sets for $\mathcal{C}$ are $\leq_{\mathrm{btt}}^{\log}$-autoreducible. Since $\mathcal{C}$ is closed under complementation, the case of $\alpha = \oplus$ follows again by Proposition 2.8.
- $\alpha \in \{\rightarrow, \leftarrow, (\neg\rightarrow), (\neg\leftarrow)\}$: Theorem 3.6 shows that for $\alpha = \rightarrow$, $\leq_{\alpha\mathrm{tt}}^{\log}$-complete sets for $\mathcal{C}$ are $\leq_{\mathrm{btt}}^{\log}$-autoreducible. By symmetry, this also holds for $\alpha = \leftarrow$. Since $\mathcal{C}$ is closed under complementation, the case of the remaining functions follows again by Proposition 2.8.

□

5 Mitoticity and Autoreducibility by Diagonalization

We use diagonalization to obtain logspace mitoticity and autoreducibility results. Generally speaking, the diagonalization prevents difficult cases that could not be handled. Buhrman et al. [5] use this technique to show for EXP and other classes that all $\leq^{\mathrm{p}}_{\text{2-tt}}$-complete sets are $\leq^{\mathrm{p}}_{\text{2-tt}}$-autoreducible. We extend this technique and show the statement for NEXP.

Moreover, we consider several logspace reducibilities and obtain results for PSPACE, EXP, and NEXP, as those classes are powerful enough to diagonalize against logspace reductions. Since PSPACE and EXP are closed under complementation, we obtain that their many-one complete sets are mitotic. For NEXP we can only show that many-one complete sets are autoreducible.

5.1 Complete Sets for Classes closed under Complementation

We will first show mitoticity and autoreducibility results for $\leq^{\log}_{\mathrm{m}}$-complete sets and for $\leq^{\log}_{\text{2-tt}}$-complete sets, that generally apply to classes that have certain closure properties. For this purpose, we first define length-restricted reductions computable in space $\log^2$.

Definition 5.1 *Let A and B be sets.*

1. *We define $A \leq^{\log^2\text{-lin}}_{\mathrm{m}} B$ similarly to $A \leq^{\log}_{\mathrm{m}} B$, except that, on input x, the computation of $f(x)$ is allowed to use $(\log|x|)^2$ space, but it must hold that $|f(x)| \leq c \cdot |x|$, where $c > 0$ is some constant.*
2. *We define $A \leq^{\log^2\text{-lin}}_{\text{1-tt}} B$ similarly to $A \leq^{\log}_{\text{1-tt}} B$, except that, on input x, the oracle machine is allowed to use $(\log|x|)^2$ space, but may only ask one oracle question of length at most $c \cdot |x|$, where $c > 0$ is some constant.*

We obtain the following results.

Theorem 5.2 *If a class $\mathcal{C}$ is closed under union, complement, and $\leq^{\log^2\text{-lin}}_{\mathrm{m}}$-reducibility, then all $\leq^{\log}_{\mathrm{m}}$-complete sets in $\mathcal{C}$ are $\leq^{\log}_{\mathrm{m}}$-mitotic.*

Proof Let A be $\leq^{\log}_{\mathrm{m}}$-complete, we show that A is $\leq^{\log}_{\mathrm{m}}$-mitotic. Let $f_1, f_2, \ldots$ be an enumeration of logarithmic-space-bounded Turing transducers such that the computation of f_i on x can be simulated in space $\log i \cdot (1 + \log|x|)$ using a binary alphabet.

$$\begin{aligned} B_1 &\stackrel{df}{=} \{y \mid y = \langle 0^i, x, 0^{|x|^2}\rangle, |f_i(y)| \leq |y|, \text{ and } f_i(y) \notin A\} \\ B_2 &\stackrel{df}{=} \{y \mid y = \langle 0^i, x, 0^{|x|^2}\rangle, |f_i(y)| > |y|, \text{ and } x \in A\} \\ B &\stackrel{df}{=} B_1 \cup B_2 \end{aligned}$$

$B_1 \in \mathcal{C}$, since $B_1 \leq^{\log^2\text{-lin}}_{\mathrm{m}} \overline{A} \in \mathcal{C}$. $B_2 \in \mathcal{C}$, since $B_2 \leq^{\log^2\text{-lin}}_{\mathrm{m}} A \in \mathcal{C}$. Hence $B \in \mathcal{C}$ and there exists some j such that $B \leq^{\log}_{\mathrm{m}} A$ via f_j. If there exists a $y = \langle 0^i, x, 0^{|x|^2}\rangle$ such that $|f_j(y)| \leq |y|$, then $y \in B \Leftrightarrow f_j(y) \notin A \Leftrightarrow y \notin B$, which is a contradiction. Therefore, $|f_j(y)| > |y|$ for all y. This shows that $A \leq^{\log}_{\mathrm{m}} B$ via $g(x) \stackrel{df}{=} \langle 0^j, x, 0^{|x|^2}\rangle$ and $A \leq^{\log}_{\mathrm{m}} A$ via $h(x) = f_j(g(x))$. Note that $|h(x)| \geq |x|^2$ for all x. Since $h \in \mathrm{FL}$, there exists a $c \geq 3$ such that for all $x \in \Sigma^{\geq 2}$,

$$|x|^2 \leq |h(x)| < |x|^c. \tag{1}$$

Let

$$S \stackrel{df}{=} \{x \mid \min\{i \in \mathbb{N} \mid |x| \leq 2^{c^i}\} \text{ is even}\}$$

and note that $S \in \mathrm{L}$. From (1) it follows that for every $x \in \Sigma^{\geq 2}$ there exists an $i \in \{1, \ldots, \lfloor \log c \rfloor\}$ such that $x \in S \Leftrightarrow h^i(x) \notin S$ (where h^i is the i-th superposition of h). Let

$$r(x) \stackrel{df}{=} \begin{cases} h^i(x), & \text{if } |x| \geq 2 \\ a, & \text{if } |x| < 2 \text{ and } x \in A \\ a', & \text{if } |x| < 2 \text{ and } x \notin A \end{cases}$$

where $i = \min\{1, \ldots, \lfloor \log c \rfloor \mid x \in S \Leftrightarrow h^i(x) \notin S\}$, a is a fixed element in A, and a' is a fixed element in $\overline{A}$. Note that $r \in \mathrm{FL}$, since c is constant. Moreover, for all x it holds that $(x \in S \Leftrightarrow r(x) \notin S)$ and $(x \in A \Leftrightarrow r(x) \in A)$. Therefore, $A \cap S \leq_{\mathrm{m}}^{\log} A \cap \overline{S}$ and $A \cap \overline{S} \leq_{\mathrm{m}}^{\log} A \cap S$ both via r. Moreover $A \equiv_{\mathrm{m}}^{\log} A \cap S$, since $S \in \mathrm{L}$. This shows that A is $\leq_{\mathrm{m}}^{\log}$-mitotic. □

Corollary 5.3 *All $\leq_{\mathrm{m}}^{\log}$-complete sets for the following classes are $\leq_{\mathrm{m}}^{\log}$-mitotic:* QP = DTIME($2^{\mathrm{polylog}(n)}$), PSPACE, EXP, REC, DSPACE(s) *and* NSPACE(s) *for all space-constructible $s \geq \log^2$.*

Proof The classes satisfy the requirements in Theorem 5.2. □

We adapt a technique by Buhrman et al. [5] to the logspace setting and obtain the following result.

Theorem 5.4 *If a class $\mathcal{C}$ is closed under $\leq_{\text{1-tt}}^{\log^2\text{-lin}}$-reducibility, then all $\leq_{\text{2-tt}}^{\log}$-complete sets for $\mathcal{C}$ are $\leq_{\text{2-tt}}^{\log}$-autoreducible.*

Proof Let $\{M_i\}_i$ be an enumeration of all $\leq_{\text{2-tt}}^{\log}$-reductions such that the truth-table used by M_i on input x and the number of queries different from x can be obtained in space $\log i \cdot (1 + \log |x|)$ using a binary alphabet. Without loss of generality, we assume that all reductions always query exactly two distinct values.
Let A be $\leq_{\text{2-tt}}^{\log}$-complete for $\mathcal{C}$ and let D be the set of words accepted by the following algorithm.

1. If the input is not of the form $(0^i, x)$, then reject.
2. Determine $v := A(x)$ with one query to the oracle A.
3. Simulate $M_i(0^i, x)$ and let t be the truth-table that is obtained if we replace the query x with the value v (if applicable).
4. If t is constant:
   accept if and only if $t \equiv$ false.
5. If t is of the form $y \notin A$ (for $y \neq x$):
   accept if and only if $v \equiv$ false.
6. If t is of the form $y \in A$ (for $y \neq x$):
   accept if and only if $v \equiv$ true.
7. If the value of t depends on two queries different from x:
   accept if and only if $v \equiv$ true.

The algorithm describes a $\leq_{\text{1-tt}}^{\log^2\text{-lin}}$-reduction to A and so $D \in \mathcal{C}$. Hence, $D \leq_{\text{2-tt}}^{\log} A$ via some machine M_j. We describe an autoreduction for A on input x: If $M_j(0^j, x)$ does not query x, then act just as M_j. Otherwise, accept if and only if the other query belongs to A.

Note that the reduction can be computed in logarithmic space and does not query its own input. If M_j does not query x, the autoreduction is obviously correct. If M_j has a truth-table t that is constant once the value of $x \in A$ is substituted, then by line 4, $(0^j, x) \in D \iff t \equiv$ false which contradicts the fact that $D \leq_{\text{2-tt}}^{\log} A$ via M_j. Therefore, M_j accepts if and only if the query different from x lies in A and thus the autoreduction is correct. □

Corollary 5.5 *All $\leq^{\log}_{2\text{-tt}}$-complete sets for the following classes are $\leq^{\log}_{2\text{-tt}}$-autoreducible:* QP = DTIME($2^{\text{polylog}(n)}$), PSPACE, EXP, REC, DSPACE(s) *and* NSPACE(s) *for all space-constructible $s \geq \log^2$.*

Proof Follows from Theorem 5.4. □

5.2 Complete Sets for NEXP

It is unknown whether the above results apply to NEXP, so here we cannot conclude mitoticity. We can, however, show that sets complete for NEXP are at least autoreducible.

Theorem 5.6 *1. For every $k \geq 1$ and $\leq \in \{\leq^{\text{p}}_{k\text{-dtt}}, \leq^{\text{p}}_{k\text{-ctt}}, \leq^{\text{p}}_{\text{dtt}}, \leq^{\text{p}}_{\text{ctt}}\}$, every $\leq$-complete set for* NEXP *is $\leq$-autoreducible.*

2. For every $k \geq 1$ and $\leq \in \{\leq^{\log}_{k\text{-dtt}}, \leq^{\log}_{k\text{-ctt}}, \leq^{\log}_{\text{dtt}}, \leq^{\log}_{\text{ctt}}\}$, every $\leq$-complete set for NEXP *is $\leq$-autoreducible.*

Proof We first show the first statement of the theorem. Here, we argue for $\leq^{\text{p}}_{\text{dtt}}$, as the other cases are shown analogously. Let A be a $\leq^{\text{p}}_{\text{dtt}}$-complete set for NEXP. Recall that $A \leq^{\text{p}}_{\text{dtt}} B \iff$ there exists a polynomial-time computable function f such that for all x, $f(x) = \langle q_1, \ldots, q_n\rangle$ and $(x \in A \iff B(q_1) \vee \cdots B(q_n))$. Let $\{f_i\}_{i\geq 1}$ be an enumeration of all polynomial-time Turing transducers such that the computation of f_i on x can be simulated in time $|x|^i + i$. Let B be the set of inputs $\langle 0^i, x\rangle$ accepted by the following nondeterministic algorithm in exponential time.

- $Q :=$ set of all queries of f_i on input $\langle 0^i, x\rangle$.
- If $x \notin Q$, then accept $\langle 0^i, x\rangle \iff x \in A$.
- Otherwise, reject $\langle 0^i, x\rangle$.

Obviously $B \in$ NEXP. So $B \leq^{\text{p}}_{\text{dtt}} A$ via some disjunctive truth-table reduction f_j.

For every x, if x is one of the queries of $f_j(\langle 0^j, x\rangle)$, then, by the above algorithm, $\langle 0^j, x\rangle \notin B$. Hence for each query q of $f_j(\langle 0^j, x\rangle)$ we have $q \notin A$. In particular $x \notin A$. On the other hand, if $f_j(\langle 0^j, x\rangle) = \langle q_1, \ldots, q_m\rangle$ and $x \neq q_i$ for all i, then $x \in A \Leftrightarrow \langle 0^j, x\rangle \in B \Leftrightarrow c_A(q_1) \vee \cdots \vee c_A(q_m)$. Based on this observation, we obtain the following autoreduction for A, where x is the input.

- $Q :=$ set of all queries of f_j on input $\langle 0^j, x\rangle$.
- If $x \notin Q$, then return $f_j(\langle 0^j, x\rangle)$.
- Otherwise, return some fixed value $y \in \overline{A} - \{x\}$.

Hence A is $\leq^{\text{p}}_{\text{dtt}}$-autoreducible.

In order to show the second statement for $\leq^{\log}_{\text{dtt}}$, suppose that A is $\leq^{\log}_{\text{dtt}}$-complete for NEXP. Observe that the above enumeration $\{f_i\}_{i\geq 1}$ includes all $\leq^{\log}_{\text{dtt}}$-reductions. If $B \leq^{\log}_{\text{dtt}} A$ via the $\leq^{\log}_{\text{dtt}}$ reduction f_j, then f_j is logspace computable, and hence the described autoreduction of A on input x can be computed in logspace. The other cases for logspace are shown analogously. □

Note that every non-trivial $\leq^{\log}_{1\text{-dtt}}$-autoreducible set is also $\leq^{\log}_{\text{m}}$-autoreducible, and similarly every non-trivial $\leq^{\text{p}}_{1\text{-dtt}}$-autoreducible set is also $\leq^{\text{p}}_{\text{m}}$-autoreducible, hence Theorem 5.6 covers $\leq^{\log}_{\text{m}}$ and $\leq^{\text{p}}_{\text{m}}$ as a special case.

Theorem 5.7 *1. Every $\leq^{\text{p}}_{2\text{-tt}}$-complete set for* NEXP *is $\leq^{\text{p}}_{2\text{-tt}}$-autoreducible.*

2. *Every* $\leq^{\log}_{2\text{-tt}}$*-complete set for* NEXP *is* $\leq^{\log}_{2\text{-tt}}$*-autoreducible.*

Proof We first show that $\leq^{\mathrm{p}}_{2\text{-tt}}$-complete sets for NEXP are $\leq^{\mathrm{p}}_{2\text{-tt}}$-autoreducible. Let A be a $\leq^{\mathrm{p}}_{2\text{-tt}}$-complete set for NEXP. Let $\{M_i\}_{i\geq 1}$ be an enumeration of all $\leq^{\mathrm{p}}_{2\text{-tt}}$ reductions such that the computation of M_i on x can be simulated in time $|x|^i+i$. Observe that $L_1 \leq^{\mathrm{p}}_{2\text{-tt}} L_2$ if and only if there exist polynomial-time computable functions $f\colon \Sigma^* \to (\Sigma^*)^2$ and $g\colon \Sigma^*\times\{0,1\}^2 \to \{0,1\}$ such that for all x, $f(x) = \langle q_1, q_2\rangle$ and $x \in L_1 \Leftrightarrow g(x, L_2(q_1), L_2(q_2)) = 1$. In this sense, let f_i and g_i be the functions that correspond to the reduction M_i. Without loss of generality we assume that if $f_i(x) = \langle q_1, q_2\rangle$, then $q_1 \neq q_2$. Let B be the set of inputs $\langle 0^i, x\rangle$ accepted by the following nondeterministic algorithm N in exponential time.

- Compute $f_i(\langle 0^i, x\rangle) = \langle q_1, q_2\rangle$ and let $Q = \{q_1, q_2\}$.
- If $x \notin Q$ then: accept $\langle 0^i, x\rangle \iff x \in A$.
- Otherwise $x \in Q$ and without loss of generality we assume $x = q_1$. Let g^x_i be the 2-ary Boolean function defined by $g^x_i(\alpha,\beta) \stackrel{df}{=} g_i(x,\alpha,\beta)$ and consider all possible cases for g^x_i:
  1. g^x_i is constant 0 or constant 1: accept $\langle 0^i, x\rangle \iff g^x_i = 0$
  2. $g^x_i(\alpha,\beta) = \beta$ or $g^x_i(\alpha,\beta) = \neg\beta$: accept $\langle 0^i, x\rangle \iff x \in A$
  3. $g^x_i(\alpha,\beta) = \alpha$ or $g^x_i(\alpha,\beta) = \neg\alpha$: reject $\langle 0^i, x\rangle$
  4. $g^x_i \in \{\wedge, \not\to, \not\leftarrow, \neg\vee, \leftrightarrow\}$: accept $\langle 0^i, x\rangle$
  5. $g^x_i \in \{\neg\wedge, \to, \leftarrow, \vee, \oplus\}$: reject $\langle 0^i, x\rangle$

Observe that $B \in$ NEXP. So $B \leq^{\mathrm{p}}_{2\text{-tt}} A$ via some $\leq^{\mathrm{p}}_{2\text{-tt}}$ reduction M_j. Compute $f_j(\langle 0^j, x\rangle) = \langle q_1, q_2\rangle$ and let $Q = \{q_1, q_2\}$. Before we describe a $\leq^{\mathrm{p}}_{2\text{-tt}}$ autoreduction for A, we have to observe some facts. Suppose $q_1 = x$ and consider the following cases:

1. g^x_j is constant 0 or constant 1: $\langle 0^j, x\rangle \in B \iff g^x_j = 0$, contradicting $B \leq^{\mathrm{p}}_{2\text{-tt}} A$ via M_j.
2. $g^x_j(\alpha,\beta) = \beta$ or $g^x_j(\alpha,\beta) = \neg\beta$: If $g^x_j(\alpha,\beta) = \beta$, we obtain $x \in A \iff \langle 0^j, x\rangle \in B \iff g^x_j(A(x), A(q_2)) = 1 \iff A(q_2) = 1 \iff q_2 \in A$. Similarly, if $g^x_j(\alpha,\beta) = \neg\beta$, we obtain $x \in A \iff \langle 0^j, x\rangle \in B \iff g^x_j(A(x), A(q_2)) = 1 \iff \neg A(q_2) = 1 \iff q_2 \notin A$.
3. $g^x_j(\alpha,\beta) = \alpha$ or $g^x_j(\alpha,\beta) = \neg\alpha$: In both cases $\langle 0^j, x\rangle \notin B$, hence $g^x_j(A(x), A(q_2)) = 0$. Thus if $g^x_j(\alpha,\beta) = \alpha$, then $x \notin A$, and if $g^x_j(\alpha,\beta) = \neg\alpha$, then $x \in A$.
4. $g^x_j \in \{\wedge, \not\to, \not\leftarrow, \neg\vee, \leftrightarrow\}$: Here $\langle 0^j, x\rangle \in B$, hence $g^x_j(A(x), A(q_2)) = 1$. If $g^x_j \in \{\wedge, \not\to\}$, then $x \in A$. If $g^x_j \in \{\not\leftarrow, \neg\vee\}$, then $x \notin A$. If $g^x_j = \leftrightarrow$, then $x \in A \iff q_2 \in A$.
5. $g^x_j \in \{\neg\wedge, \to, \leftarrow, \vee, \oplus\}$: Here $\langle 0^j, x\rangle \notin B$, hence $g^x_j(A(x), A(q_2)) = 0$. If $g^x_j \in \{\neg\wedge, \to\}$, then $x \in A$. If $g^x_j \in \{\leftarrow, \vee\}$, then $x \notin A$. If $g^x_j = \oplus$, then $x \in A \iff q_2 \in A$.

We describe a $\leq^{\mathrm{p}}_{2\text{-tt}}$ autoreduction of A on input x:

- Compute $f_j(\langle 0^j, x\rangle) = \langle q_1, q_2\rangle$ and let $Q = \{q_1, q_2\}$.
- If $x \notin Q$: accept iff M^A_j accepts $\langle 0^j, x\rangle$.
- Otherwise, suppose $x = q_1$ (the case $x = q_2$ is analogous). By the above arguments, g^x_j cannot be constant and we have the following cases:
  - $g^x_j(\alpha,\beta) = \beta$: accept $\iff q_2 \in A$
  - $g^x_j(\alpha,\beta) = \neg\beta$: accept $\iff q_2 \notin A$
  - $g^x_j(\alpha,\beta) = \alpha$: reject
  - $g^x_j(\alpha,\beta) = \neg\alpha$: accept
  - $g^x_j \in \{\wedge, \neg\wedge, \to, \not\to\}$: accept
  - $g^x_j \in \{\vee, \neg\vee, \leftarrow, \not\leftarrow\}$: reject
  - $g^x_j \in \{\leftrightarrow, \oplus\}$: accept $\iff q_2 \in A$

Hence A is $\leq^{\mathrm{p}}_{\text{2-tt}}$-autoreducible.

In order to show the second statement, suppose that A is $\leq^{\log}_{\text{2-tt}}$-complete for NEXP. Observe that the above enumeration $\{M_i\}_{i\geq 1}$ includes all $\leq^{\log}_{\text{2-tt}}$-reductions. If $B \leq^{\log}_{\text{2-tt}} A$ via the $\leq^{\log}_{\text{2-tt}}$ reduction M_j, then f_j and g_j are logspace computable, and hence the described autoreduction of A on input x can be computed in logspace. □

Note that by Theorem 4.2 (resp., [5]), there exist $\leq^{\log}_{\text{3-tt}}$-complete sets for PSPACE (resp., $\leq^{\mathrm{p}}_{\text{3-tt}}$-complete sets for EXP) that are not $\leq^{\log}_{\text{btt}}$-autoreducible (resp., $\leq^{\mathrm{p}}_{\text{btt}}$-autoreducible), hence improving Theorem 5.7.1 to $\leq^{\mathrm{p}}_{\text{3-tt}}$ separates NEXP from EXP, and improving Theorem 5.7.2 to $\leq^{\log}_{\text{3-tt}}$ separates NEXP from PSPACE.

5.3 Complete Sets for PSPACE and EXP

We show that for some restricted polynomial-time truth-table reductions, complete sets for EXP are complete under length-increasing reductions, and the same holds considering logspace reductions for PSPACE and EXP. By carefully repeating the length-increasing reductions in such a way that we switch between stages defined by a separator set we obtain mitoticity for PSPACE and EXP.

Definition 5.8 *Given two sets A and B, we define $A \leq^{\mathrm{p}}_{\text{T-li}} B$ if there is a Turing machine M such that $A = L(M^B)$ and all queries made by $M^B(x)$ are of length strictly greater than $|x|$. The notions $\leq^{\mathrm{p}}_{\text{2-tt-li}}$, $\leq^{\mathrm{p}}_{k\text{-ctt-li}}$, $\leq^{\mathrm{p}}_{k\text{-dtt-li}}$, $\leq^{\mathrm{p}}_{\text{dtt-li}}$, $\leq^{\mathrm{p}}_{\text{ctt-li}}$, $\leq^{\mathrm{p}}_{\text{m-li}}$ and $\leq^{\log}_{\text{T-li}}$, $\leq^{\log}_{\text{2-tt-li}}$, $\leq^{\log}_{k\text{-ctt-li}}$, $\leq^{\log}_{k\text{-dtt-li}}$, $\leq^{\log}_{\text{dtt-li}}$, $\leq^{\log}_{\text{ctt-li}}$, $\leq^{\log}_{\text{m-li}}$ are defined similarly.*

Berman [3] and Ganesan and Homer [6] show that all many-one complete sets for EXP are many-one length-increasing equivalent. In the following lemma, we generalize to show that it also holds for some certain polynomial-time reductions and logspace reductions under EXP and PSPACE.

Lemma 5.9
1. *For every $k \geq 2$ and $\leq \in \{\leq^{\mathrm{p}}_{\text{2-tt}}, \leq^{\mathrm{p}}_{k\text{-ctt}}, \leq^{\mathrm{p}}_{k\text{-dtt}}, \leq^{\mathrm{p}}_{\text{ctt}}, \leq^{\mathrm{p}}_{\text{dtt}}\}$, all $\leq$-complete sets for* EXP *are $\leq$-li equivalent.*
2. *For every $k \geq 2$ and $\leq \in \{\leq^{\log}_{\text{2-tt}}, \leq^{\log}_{\text{k-ctt}}, \leq^{\log}_{\text{k-dtt}}, \leq^{\log}_{\text{ctt}}, \leq^{\log}_{\text{dtt}}\}$, all $\leq$-complete sets for* EXP *(resp.,* PSPACE*) are $\leq$-li equivalent.*

Proof We show the lemma for $\leq^{\mathrm{p}}_{\text{2-tt}}$, the other cases are shown analogously. Let A be any $\leq^{\mathrm{p}}_{\text{2-tt}}$-complete set for EXP and K be the canonical $\leq^{\mathrm{p}}_{\text{m}}$-complete set for EXP. Observe that $A \leq^{\mathrm{p}}_{\text{m-li}} K$, so it suffices to show $K \leq^{\mathrm{p}}_{\text{2-tt-li}} A$.

Let $\{M_i\}_{i\geq 1}$ be an enumeration of all $\leq^{\mathrm{p}}_{\text{2-tt}}$ reductions such that the computation of M_i on x can be simulated in time $|x|^i + i$. Note that $L_1 \leq^{\mathrm{p}}_{\text{2-tt}} L_2$ if and only if there exist polynomial-time computable functions $f\colon \Sigma^* \to (\Sigma^*)^2$ and $g\colon \Sigma^* \times \{0,1\}^2 \to \{0,1\}$ such that for all x, $f(x) = \langle q_1, q_2\rangle$ and $x \in L_1 \Leftrightarrow g(x, L_2(q_1), L_2(q_2)) = 1$. Let f_i and g_i be the functions that correspond to the reduction M_i. Without loss of generality we assume that if $f_i(x) = \langle q_1, q_2\rangle$, then $|q_1| \leq |q_2|$. This is not a restriction, because we can always switch q_1 with q_2 and modify g_i on x accordingly. Let B be the set of inputs $\langle 0^i, x\rangle$ accepted by the following nondeterministic algorithm N in exponential time.

1. compute $f_i(\langle 0^i, x\rangle) = \langle q_1, q_2\rangle$
2. if $|\langle 0^i, x\rangle| < |q_1| \leq |q_2|$, then accept $\iff$ $x \in K$
3. if $|q_1| \leq |q_2| \leq |\langle 0^i, x\rangle|$, then accept $\iff$ $g_i(x, A(q_1), A(q_2)) = 0$
4. if $|q_1| \leq |\langle 0^i, x\rangle| < |q_2|$, then consider the following cases:

(a) $g_i(x, A(q_1), 0) = g_i(x, A(q_1), 1)$: accept $\iff$ $g_i(x, A(q_1), 0) = 0$.
(b) $g_i(x, A(q_1), \beta) = \beta$ for all β: accept $\iff$ $x \in K$.
(c) $g_i(x, A(q_1), \beta) = \neg\beta$ for all β: accept $\iff$ $x \notin K$.

Claim 5.10 $B \in \mathrm{EXP}$.

Proof We analyze the running time of the above algorithm on input $\langle 0^i, x\rangle$ where $n = |\langle 0^i, x\rangle|$.

- M_i on input x can be simulated in time $|x|^i + i$, hence computing $f_i(\langle 0^i, x\rangle)$ in step 1 and $g_i(x, \alpha, \beta)$ in step 3 and step 4 is possible in exponential time in n.
- $K \in \mathrm{EXP}$, so deciding $x \in K$ in step 2 and step 4 takes exponential time in n.
- $A \in \mathrm{EXP}$ and in step 3 we have $|q_1| \leq |q_2| \leq n$. Hence computing $A(q_1)$ and $A(q_2)$ in step 3 takes time exponential in n.
- Analogously, computing $A(q_1)$ in step 3 takes time exponential in n.

□

Because $B \in \mathrm{EXP}$ and A is $\leq^{\mathrm{p}}_{2\text{-tt}}$-complete for EXP, we have $B \leq^{\mathrm{p}}_{2\text{-tt}} A$ by some reduction M_j. Let $f_j(\langle 0^j, x\rangle) = \langle q_1, q_2\rangle$. Observe the following for B's algorithm on input $\langle 0^j, x\rangle$:

1. If $|\langle 0^j, x\rangle| < |q_1| \leq |q_2|$, then $x \in K \iff \langle 0^j, x\rangle \in B \iff g_j(x, A(q_1), A(q_2)) = 1$. Notice that in this case, both q_1 and q_2 are longer than x.
2. If $|q_1| \leq |q_2| \leq |\langle 0^i, x\rangle|$, we have $\langle 0^j, x\rangle \in B \iff g_j(x, A(q_1), A(q_2)) = 0$, which contradicts the fact that M_j reduces B to A. Hence this case cannot occur.
3. If $|q_1| \leq |\langle 0^i, x\rangle| < |q_2|$, consider each case in the algorithm:
   (a) If $g_j(x, A(q_1), 0) = g_j(x, A(q_1), 1)$, we have $\langle 0^j, x\rangle \in B \iff g_j(x, A(q_1), A(q_2)) = 0$, which contradicts the fact that M_j reduces B to A. Hence this case cannot occur.
   (b) If $g_j(x, A(q_1), \beta) = \beta$ for all β, we obtain $x \in K \iff \langle 0^j, x\rangle \in B \iff g_j(A(q_1), A(q_2)) = A(q_2) = 1 \iff q_2 \in A$.
   (c) If $g_j(x, A(q_1), \beta) = \neg\beta$ for all β, we obtain $x \in K \iff \langle 0^j, x\rangle \notin B \iff g_j(A(q_1), A(q_2)) = 1 - A(q_2) = 0 \iff q_2 \in A$.

   So, in this case we have $x \in K \iff q_2 \in A$ with $|x| < |\langle 0^j, x\rangle| < |q_2|$.

We describe a $\leq^{\mathrm{p}}_{2\text{-tt-li}}$ reduction from K to A on input x:

1. compute $f_j(\langle 0^j, x\rangle) = \langle q_1, q_2\rangle$
2. if $|\langle 0^j, x\rangle| < |q_1| \leq |q_2|$, then accept $\iff g_j(x, A(q_1), A(q_2)) = 1$
3. if $|q_1| \leq |\langle 0^j, x\rangle| < |q_2|$, then accept $\iff A(q_2) = 1$.

Hence $K \leq^{\mathrm{p}}_{2\text{-tt-li}} A$. □

Theorem 5.11 *1. For every $k \geq 2$ and $\leq \in \{\leq^{\mathrm{p}}_{2\text{-tt}}, \leq^{\mathrm{p}}_{k\text{-ctt}}, \leq^{\mathrm{p}}_{k\text{-dtt}}, \leq^{\mathrm{p}}_{\mathrm{ctt}}, \leq^{\mathrm{p}}_{\mathrm{dtt}}\}$, every $\leq$-complete set for* EXP *is $\leq$-mitotic.*
2. For every $k \geq 2$ and $\leq \in \{\leq^{\log}_{2\text{-tt}}, \leq^{\log}_{k\text{-ctt}}, \leq^{\log}_{k\text{-dtt}}, \leq^{\log}_{\mathrm{ctt}}, \leq^{\log}_{\mathrm{dtt}}\}$, every $\leq$-complete set for EXP *(resp.,* PSPACE*) is $\leq$-mitotic.*

Proof We prove the theorem for $\leq^{\mathrm{p}}_{2\text{-ctt}}$, other cases will follow similarly. Let A be $\leq^{\mathrm{p}}_{2\text{-ctt}}$-complete for EXP. By Lemma 5.9, $A \times \Sigma^* \leq^{\mathrm{p}}_{2\text{-ctt-li}} A$. Hence there exist polynomial-time computable functions f and g such that $x \in A \times \Sigma^* \iff (f(x) \in A \wedge g(x) \in A)$. Since f, g are polynomial-time computable, there is an $l > 0$ such that $|x|^{1/l} < |f(x)|, |g(x)| < |x|^l$ for all x.

Define $t_1 = 2$ and $t_{i+1} = (t_i)^{2l^2}$, and let $S = \{x |$ for some odd $i, t_i \leq |x| < t_{i+1}\}$. Note that $S \in \mathrm{P}$. We show that A, $A \cap S$, and $A \cap \overline{S}$ are $\leq^{\mathrm{p}}_{2\text{-ctt}}$-equivalent.

Claim 5.12 *$A \equiv^{\mathrm{p}}_{2\text{-ctt}} A \cap S$ and $A \equiv^{\mathrm{p}}_{2\text{-ctt}} A \cap \overline{S}$.*

Proof Since $S \in \mathrm{P}$, $A \cap S \leq^{\mathrm{p}}_{2\text{-ctt}} A$ and $A \cap \overline{S} \leq^{\mathrm{p}}_{2\text{-ctt}} A$. We describe the reduction $A \leq^{\mathrm{p}}_{2\text{-ctt}} A \cap S$, the reduction $A \leq^{\mathrm{p}}_{2\text{-ctt}} A \cap \overline{S}$ is similar.

On input x with $|x| \geq 2$, compute i such that $t_i \leq |x| < t_{i+1}$. Consider the following cases:

Case 1: i is odd, i.e., $x \in S$. Choose y sufficiently large such that $(t_{i+2})^l < |\langle x, y\rangle| < (t_{i+2})^{2l}$. This is possible in polynomial time, since $(t_{i+2})^l = (((t_i)^{2l^2})^{2l^2})^l = (t_i)^{4l^5} \leq |x|^{4l^5}$.

Let $F(x) = f(\langle x, y\rangle)$ and $G(x) = g(\langle x, y\rangle)$. So $|F(x)| = |f(\langle x, y\rangle)| > |\langle x, y\rangle|^{1/l} > t_{i+2}$ and $|F(x)| = |f(\langle x, y\rangle)| < |\langle x, y\rangle|^l < (t_{i+2})^{2l^2} = t_{i+3}$. Hence $t_{i+2} < |F(x)| < t_{i+3}$, i.e., $F(x) \in S$. Similarly we have $G(x) \in S$. Therefore, $x \in A \iff \langle x, y\rangle \in A \times \Sigma^* \iff F(x) = f(\langle x, y\rangle) \in A \wedge G(x) = g(\langle x, y\rangle) \in A \iff F(x) \in A \cap S \wedge G(x) \in A \cap S$.

Case 2: i is even, i.e., $x \notin S$. Similar to case 1, we choose y such that $(t_{i+1})^l < |\langle x, y\rangle| < (t_{i+1})^{2l}$.

Let $F(x) = f(\langle x, y\rangle)$ and $G(x) = g(\langle x, y\rangle)$. So $t_{i+1} < |F(x)|, |G(x)| < t_{i+2}$ and $F(x), G(x) \in S$. Therefore, $x \in A \iff \langle x, y\rangle \in A \times \Sigma^* \iff F(x) = f(\langle x, y\rangle) \in A \wedge G(x) = g(\langle x, y\rangle) \in A \iff F(x) \in A \cap S \wedge G(x) \in A \cap S$.

Combining the two cases, we obtain that $A \leq^{\mathrm{p}}_{2\text{-ctt}} A \cap S$. □

Claim 5.13 *$A \cap S \leq^{\mathrm{p}}_{2\text{-ctt}} A \cap \overline{S}$ and $A \cap \overline{S} \leq^{\mathrm{p}}_{2\text{-ctt}} A \cap S$.*

Proof We show $A \cap S \leq^{\mathrm{p}}_{2\text{-ctt}} A \cap \overline{S}$, the other reduction is similar. Let a_1, a_2 be fixed elements from $\overline{A}$. On input x, compute i such that $t_i \leq |x| < t_{i+1}$ and consider the following cases:

Case 1: i is odd, i.e., $x \in S$. Similar to Claim 5.12, we choose y such that $(t_{i+1})^l < |\langle x, y\rangle| < (t_{i+1})^{2l}$.

Let $F(x) = f(\langle x, y\rangle)$ and $G(x) = g(\langle x, y\rangle)$. By similar reasoning, $t_{i+1} < |F(x)|, |G(x)| < t_{i+2}$. Since i is odd, $F(x) \in \overline{S}$ and $G(x) \in \overline{S}$. So $x \in A \cap S \iff x \in A \iff \langle x, y\rangle \in A \times \Sigma^* \iff F(x) = f(\langle x, y\rangle) \in A \wedge G(x) = g(\langle x, y\rangle) \in A \iff F(x) \in A \cap \overline{S} \wedge G(x) \in A \cap \overline{S}$.

Case 2: i is even, i.e., $x \notin S$. Hence $x \notin A \cap S$. Let $F(x) = a_1$ and $G(x) = a_2$. It holds that $x \in A \cap S \iff F(x) \in A \cap \overline{S} \wedge G(x) \in A \cap \overline{S}$.

This shows $A \cap S \leq^{\mathrm{p}}_{2\text{-ctt}} A \cap \overline{S}$ by the functions F and G. □

We conclude that A is $\leq^{\mathrm{p}}_{2\text{-ctt}}$-mitotic. □

Note that by Theorem 4.2 (resp., [5]), there exist $\leq^{\log}_{3\text{-tt}}$-complete sets for PSPACE (resp., $\leq^{\mathrm{p}}_{3\text{-tt}}$-complete sets for EXP) that are not $\leq^{\log}_{\mathrm{btt}}$-autoreducible (resp., $\leq^{\mathrm{p}}_{\mathrm{btt}}$-autoreducible), hence Theorem 5.11 cannot be improved to $\leq^{\log}_{3\text{-tt}}$ or $\leq^{\mathrm{p}}_{3\text{-tt}}$.

5.4 Complete Sets for 1-Truth-Table Reductions

Homer, Kurtz, and Royer [9] and Buhrman [4] showed that for EXP and NEXP, every $\leq^{\mathrm{p}}_{1\text{-tt}}$-complete set is also $\leq^{\mathrm{p}}_{\mathrm{m}}$-complete. Their approach also applies to $\leq^{\log}_{1\text{-tt}}$-complete sets for PSPACE, EXP and NEXP, so we have the following theorem.

Theorem 5.14 ([4, 9]) *1. All $\leq^{\log}_{1\text{-tt}}$-complete sets for* PSPACE *(resp.,* EXP, NEXP*) are $\leq^{\log}_{\mathrm{m}}$-complete for* PSPACE *(resp.,* EXP*,* NEXP*).*

2. All $\leq^{\mathrm{p}}_{1\text{-tt}}$-complete sets for EXP *(resp.,* NEXP*) are $\leq^{\mathrm{p}}_{\mathrm{m}}$-complete for* EXP *(resp.,* NEXP*).*

Proof Similar to [4, Theorem 3.2 and Theorem 3.4]. □

This means that most of the obtained results for many-one complete sets also hold for 1-truth-table complete sets. We obtain the following corollary.

Corollary 5.15 *1. All* $\leq^{\log}_{\text{1-tt}}$*-complete sets for* PSPACE, EXP *are* $\leq^{\log}_{\text{m}}$*-mitotic.*
2. All $\leq^{\log}_{\text{1-tt}}$*-complete sets for* NEXP *are* $\leq^{\log}_{\text{m}}$*-autoreducible.*
3. All $\leq^{\text{p}}_{\text{1-tt}}$*-complete sets for* EXP, NEXP *are* $\leq^{\text{p}}_{\text{m}}$*-mitotic.*

Proof Recall that:

- every $\leq^{\log}_{\text{m}}$-complete set for PSPACE, EXP is $\leq^{\log}_{\text{m}}$-mitotic (see Corollary 5.3)
- every $\leq^{\log}_{\text{m}}$-complete set for NEXP is $\leq^{\log}_{\text{m}}$-autoreducible (see Theorem 5.6)
- every $\leq^{\text{p}}_{\text{m}}$-complete set for EXP, NEXP is $\leq^{\text{p}}_{\text{m}}$-mitotic [7, 8]

Applying Theorem 5.14 we obtain:

- every $\leq^{\log}_{\text{1-tt}}$-complete set for PSPACE, EXP is $\leq^{\log}_{\text{m}}$-mitotic
- every $\leq^{\log}_{\text{1-tt}}$-complete set for NEXP is $\leq^{\log}_{\text{m}}$-autoreducible
- every $\leq^{\text{p}}_{\text{1-tt}}$-complete set for EXP, NEXP is $\leq^{\text{p}}_{\text{m}}$-mitotic

□

References

[1] J. L. Balcázar. Self-reducibility. *Journal of Computer and System Sciences*, 41(3):367–388, 1990.

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

[3] L. Berman. *Polynomial Reducibilities and Complete Sets.* PhD thesis, Cornell University, Ithaca, NY, 1977.

[4] H. Buhrman. *Resource Bounded Reductions.* PhD thesis, University of Amsterdam, 1993.

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

[6] K. Ganesan and S. Homer. Complete problems and strong polynomial reducibilities. *SIAM Journal on Computing*, 21:733–742, 1992.

[7] C. Glaßer, M. Ogihara, A. Pavan, A. L. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. *Journal of Computer and System Sciences*, 73(5):735–754, 2007.

[8] C. Glaßer, A. Pavan, A. L. Selman, and L. Zhang. Splitting NP-complete sets. *SIAM Journal on Computing*, 2008.

[9] S. Homer, S. A. Kurtz, and J. S. Royer. On 1-truth-table-hard languages. *Theoretical Computer Science*, 115(2):383–389, 1993.

[10] R. E. Ladner and N. A. Lynch. Relativization of questions about log space computability. *Mathematical Systems Theory*, 10:19–32, 1976.

[11] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[12] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51:53–80, 1987.

[13] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de