# Space-efficient informational redundancy 

Christian Glaßer<br>Theoretische Informatik, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany

## A R T I C L E I N F O

## Article history:

Received 10 September 2009
Received in revised form 9 March 2010 Available online 16 March 2010

## Keywords:

Computational complexity
Log-space reductions
Autoreducibility
Mitoticity


#### Abstract

We study the relation of autoreducibility and mitoticity for polylog-space many-one reductions and log-space many-one reductions. For polylog-space these notions coincide, while proving the same for log-space is out of reach. More precisely, we show the following results with respect to nontrivial sets and many-one reductions: 1. polylog-space autoreducible $\Leftrightarrow$ polylog-space mitotic, 2. $\log$-space mitotic $\Rightarrow \log$-space autoreducible $\Rightarrow(\log n \cdot \log \log n)$-space mitotic, 3. relative to an oracle, log-space autoreducible $\nRightarrow$ log-space mitotic.

The oracle is an infinite family of graphs whose construction combines arguments from Ramsey theory and Kolmogorov complexity.


© 2010 Elsevier Inc. All rights reserved.

## 1. Introduction

Our investigations are motivated by the question of whether the ability to reformulate membership questions for a set implies that the set is redundant in the sense that it can be split into two equivalent parts. This question depends on the resources available for reformulation and for splitting. In this paper we focus on notions of redundancy that have low space complexity. Suppose we are given a set $A$ such that in log-space we can reformulate a given question $x \in A$ into an equivalent question $y \in A$. Is this enough to split $A$ in log-space into disjoint parts $A_{1}$ and $A_{2}$ such that $A, A_{1}$, and $A_{2}$ are log-space many-one equivalent? We study this question also with respect to polylog-space.

The reformulation of questions is made precise by the notion of autoreducibility. The idea of splitting a set into two parts that are equivalent to the original set is formalized by the notion of mitoticity. Hence our main questions can be formulated as follows:

Q1 Does log-space many-one autoreducibility imply log-space many-one mitoticity?
Q2 Does polylog-space many-one autoreducibility imply polylog-space many-one mitoticity?
Trakhtenbrot [14] defined a set $A$ to be autoreducible if there is an oracle Turing machine $M$ such that $A=L\left(M^{A}\right)$ and $M$ on input $x$ never queries $x$. Lachlan [7] introduced the notion of mitoticity which was studied comprehensively by Ladner [9,8]. A set $A$ is mitotic if there exists a recursive set $S$ such that $A, A \cap S$, and $A \cap \bar{S}$ are Turing equivalent. AmbosSpies [1] transferred autoreducibility and mitoticity to complexity theory and studied several versions of polynomial-time autoreducibility and polynomial-time mitoticity. A set $A$ is polynomial-time many-one autoreducible if $A$ is polynomial-time many-one reducible to $A$ via a function $f$ such that $f(x) \neq x$ for every $x$. Many-one autoreducible sets contain a local redundancy of information, since $x$ and $f(x)$ contain the same information with respect to membership in $A$. A set $A$

[^0]is polynomial-time many-one mitotic if there exists $S \in \mathrm{P}$ such that $A, A \cap S$, and $A \cap \bar{S}$ are polynomial-time many-one equivalent. Many-one mitoticity formalizes the aspect of informational redundancy in sets. Hence our questions Q1 and Q2 ask whether local (poly)log-space redundancy implies informational (poly)log-space redundancy. The converse implication holds in general, since mitoticity implies autoreducibility.

The question of whether local redundancy implies informational redundancy was first studied by Ladner [9,8] who showed that with respect to r.e. sets, autoreducibility and mitoticity coincide. Ambos-Spies [1] introduced the mentioned resource-bounded notions of redundancy and proved that polynomial-time Turing autoreducibility does not imply polynomial-time Turing mitoticity. Glaßer et al. [5,6] showed the same for all reducibility notions between polynomial-time 2-truth-table reducibility and polynomial-time Turing reducibility. In contrast, polynomial-time many-one autoreducibility and polynomial-time many-one mitoticity coincide [5]. The same holds for polynomial-time 1-truth-table reducibility.

In the present paper we shift the focus to logarithmic and polylogarithmic space, which brings us to the study of (poly)log-space autoreducibility and (poly)log-space mitoticity. We prove that polylog-space many-one autoreducibility and polylog-space many-one mitoticity coincide. This shows that even very restricted computational devices can exploit local redundancy and can transform it into informational redundancy. For log-space we show a similar, but weaker connection: Log-space many-one autoreducibility implies $(\log n \cdot \log \log n)$-space many-one mitoticity. The latter space bound can be even improved to $\left(\log n \cdot \log ^{(c)} n\right)$ for any fixed constant $c$, where $\log ^{(c)} n$ denotes the $c$-times composition of the log operation. On the technical side we obtain these results by developing a combination of the construction used for polynomial-time many-one reductions [5] and the repeated deterministic coin tossing by Cole and Vishkin [3].

So far we know that autoreducibility and mitoticity are equivalent with respect to unbounded Turing reductions [9,8], polynomial-time many-one reductions [5], and polylog-space many-one reductions (Corollary 3.9). Motivated by these equivalences, one could hope to turn the implications

## log-space many-one mitotic

$\Downarrow$

## log-space many-one autoreducible

$\Downarrow$
( $\log n \cdot \log \log n$ )-space many-one mitotic
into a full equivalence, by replacing $(\log n \cdot \log \log n)$-space with $\log$-space. We show that such an improvement is hard to obtain. In Section 4 we discuss in detail the reason for this and we make this precise with the construction of an oracle relative to which the equivalence does not hold. The oracle construction combines arguments from Ramsey theory and Kolmogorov complexity to make sure that log-space computable functions get lost in infinite graphs. Moreover, the constructed oracle separates log-space many-one autoreducibility and log-space many-one mitoticity with respect to all common models of log-space oracle machines. These include weak models like the ones by Ladner and Lynch [10] and Ruzzo, Simon, and Tompa [13], but also strong models like the model by Gottlob [4]. A discussion of all considered models is given in the preliminaries section.

Roughly speaking the oracle is a family of graphs whose existence follows from Ramsey theory. Each of these graphs is a cycle (i.e., a connected graph with indegree and outdegree 1) where the nodes are numbers whose lengths are polynomially bounded in the size of the graph. Our witness language $L$ is the set of all nodes that appear in some cycle of the graph family. The cycles are such that the successor of a given node can be computed in log-space which shows that $L$ is logspace many-one autoreducible. In contrast, for every unbounded function $t$, the $t(n)$-th successor cannot be computed in log-space (where $n$ is the size of the graph). So log-space functions can determine at most constantly many successors of a given node. Hence they see at most a constant-size part of the graph and act on the graph like a relation of constant arity. By the Ramsey theorem, we can choose our cycles such that log-space machines show the same acceptance behavior on several consecutive nodes $v_{1}, \ldots, v_{c}$. So a log-space separator $S$ puts all these nodes to the same side, either $S$ or $\bar{S}$. For a given log-space function $f$ we can ensure that $c$ is large enough such that $f$ on input $v_{1}$ can determine at most the nodes $v_{1}, \ldots, v_{c}$. The latter are all on the same side of $S$. So $f$ cannot be a reduction from $L \cap S$ to $L \cap \bar{S}$. In this way we diagonalize against all $f$ and obtain that $L$ is not log-space many-one mitotic.

## 2. Preliminaries

In the paper, all variables represent natural numbers, unless they are explicitly defined in a different way. We use the following abbreviations for intervals of natural numbers: $[n, m]=\{n, n+1, \ldots, m\},[n, m)=\{n, n+1, \ldots, m-1\},(n, m]=$ $\{n+1, n+2, \ldots, m\},(n, m)=\{n+1, n+2, \ldots, m-1\}$. For $n \in \mathbb{N}$ let $\log n$ be the logarithm of $n$ to base 2 . For $k \in \mathbb{Z}$ and $n \geqslant 1$ let $(k \bmod n)$ be the uniquely determined $m \in[0, n)$ such that $m \equiv k(\bmod n)$. Moreover, $\operatorname{sgn}(k)$ denotes the sign of $k$, abs $(k)$ denotes the absolute value of $k$, and $|k|$ denotes the length of the binary representation of $k$. For a function $f$, $f^{(i)}$ denotes the $i$-th superposition of $f$, i.e., $f^{(0)}(x)=x$ and $f^{(i+1)}(x)=f\left(f^{(i)}(x)\right)$. For a fixed function $f$, the sequence $f^{(0)}(x), f^{(1)}(x), \ldots$ is called the trajectory of $x$. The complement of a set $A$ is denoted by $\bar{A}$. A set $A$ is called nontrivial if $|A| \geqslant 2$ and $|\bar{A}| \geqslant 2$.

We distinguish between Turing machines and Turing transducers. Turing machines (TM for short) are used for accepting languages and so they output 0 or 1 . Turing transducers (TT for short) are machines that compute functions and hence can output arbitrary words. OTM (resp., OTT) is an abbreviation for oracle Turing machine (resp., oracle Turing transducer). If $M$ is a TM or TT, then $M(x)$ denotes the output of $M$ on input $x$. Similarly, if $M$ is an OTM or an OTT, then $M^{T}(x)$ denotes the output of $M$ on input $x$ where $T$ is used as oracle. If $M$ is a TM (resp., OTM), then $L(M)$ (resp., $L\left(M^{T}\right)$ ) denotes the set of words accepted by $M$.

### 2.1. Models of log-space oracle machines

There is a canonical way to define oracle access for time-bounded machines. However, for space-bounded machines there is no such distinguished way. We briefly discuss several models of log-space oracle machines; for a detailed comparison we refer to Buss [2]. The following properties are desirable for such models: The machine should not be able to use the query tape as additional storage, but it should be able to write long strings to the query tape (e.g., its own input). Moreover, we would like that log-space computable functions are closed under composition. Not all of the presented models satisfy these conditions.

LL-model by Ladner and Lynch [10]: The machine has one additional, one-way, write-only oracle tape which is not subject to the space bound and which is erased after asking a query.

RST-model by Ruzzo, Simon, and Tompa [13]: Like the model by Ladner and Lynch, but additionally it is required that the machine acts deterministically while anything is written on the oracle tape. So for deterministic machines, both models are equivalent.

L-model by Lynch [11]: The machine has an arbitrary, but fixed number of one-way, write-only oracle tapes. These tapes are not subject to the space bound and after asking a query, the corresponding tape is erased.

B-model by Buss [2]: The machine has many one-way, write-only query tapes and a single read-write index tape which is logarithmically bounded. If $i$ is written on the index tape, then all query operations (writing to a query tape, querying the oracle, erasing the query on the tape, and obtaining the answer) are with respect to tape $i$. If there are $k$ active queries of maximum length $m$, then this is considered as space $k \log m$ which must be of order $O(\log n)$.

W-model by Wilson [15]: The machine has a one-way, write-only oracle stack which we consider to write from left to right. The machine can write several (partial) queries to the stack such that neighboring queries are separated by \#. If the machine enters the query state, then the characters after the right-most \# are considered to be query. After querying, the \# and the query itself are erased so that the machine can continue to write the previous query at the stack. This nesting of queries may continue to any depth, but the stack contributes to the computation space as follows. If $q_{1} \# q_{2} \# \ldots \# q_{k}$ is the content of the stack, then this is considered as space $\sum_{i \in[1, k]} \max \left\{\log \left|q_{i}\right|, 1\right\}$ which must be of order $O(\log n)$.

G-model by Gottlob [4]: The machine has $O(\log n)$ one-way, write-only query tapes and a single read-write index tape which is logarithmically bounded. The query tapes are not subject to the space bound. If $i$ is written on the index tape, then all relativized operations are with respect to tape $i$.

For deterministic machines, the presented models compare as follows, where the strengths increase from bottom to top.


Throughout the paper (if not stated otherwise) we use the most powerful G-model for log-space OTMs and OTTs. Moreover, we assume the machines to have tape alphabet $\Sigma=\{0,1\}$ (which does not restrict the computational power). If we talk about the space used by such a machine, then this contains the space used by the working tape and the space used by the index tape. Recall that it does not contain the space used by query tapes. We may assume that a machine with space bound $d \log n$ has at most $d \log n$ query tapes. This latter assumption is motivated by the observation that each query tape can be used as a one-bit storage cell: For an oracle 0 , fix words $x_{0} \notin O$ and $x_{1} \in O$. A bit $b$ can be stored in the oracle tape by writing $x_{b}$ to the tape. We can read the bit $b$ by querying the tape and writing again $x_{b}$ to the tape (which was emptied by the query mechanism).

### 2.2. Complexity classes, reductions, autoreducibility, and mitoticity

For $s: \mathbb{N} \rightarrow \mathbb{N}$, let $\operatorname{FSPACE}(s)$ be the class of functions computable in deterministic space $O(s(n))$ and let $\operatorname{DSPACE}(s)$ be the class of languages that are decidable in deterministic space $O(s(n)$ ). Let FPLOG be the class of functions computable in deterministic polylog-space, i.e., FPLOG $=\bigcup_{k \geqslant 1}$ FSPACE $\left(\log ^{k} n\right)$. Let PLOG be the class of languages that are decidable in deterministic polylog-space, i.e., PLOG $=\bigcup_{k \geqslant 1}$ DSPACE $\left(\log ^{k} n\right)$. Moreover, let FL $=$ FSPACE $(\log n)$ and $L=\operatorname{DSPACE}(\log n)$. Observe that FPLOG and FL are closed under composition.

A set $A$ is polylog-space many-one reducible to a set $B$, in notation $A \leqslant_{\mathrm{m}}^{\mathrm{plog}} B$, if there exists a total $f \in$ FPLOG such that $x \in A \Leftrightarrow f(x) \in B$. Similarly, $A$ is log-space many-one reducible to $B$, in notation $A \leqslant_{\mathrm{m}}^{\log B \text {, if there exists a total }}$ $f \in$ FL such that $x \in A \Leftrightarrow f(x) \in B$. From FL's and FPLOG's closure under composition it follows that $\leqslant_{\mathrm{m}}^{\log }$ and $\leqslant_{\mathrm{m}}^{\text {plog }}$ are transitive. For integers $k \geqslant 1$ we write $A \leqslant_{\mathrm{m}}^{\log ^{k}} B$, if there exists a total $f \in \operatorname{FSPACE}\left(\log ^{k} n\right)$ such that $x \in A \Leftrightarrow f(x) \in B$. We
 the function classes $\operatorname{FSPACE}((\log n) \cdot \log \log n)$ and $\operatorname{FSPACE}\left(\log ^{k} n\right)$ for $k \geqslant 2$ are not closed under composition. Hence, the reductions $\leqslant_{\mathrm{m}}^{\log \cdot \log \log }$ and $\leqslant_{\mathrm{m}}^{\log ^{k}}$ for $k \geqslant 2$ are not transitive.

Let $\leqslant_{\mathrm{m}}^{\mathrm{r}}$ be a reduction from $\left\{\leqslant_{\mathrm{m}}^{\log }, \leqslant_{\mathrm{m}}^{\mathrm{plog}}, \leqslant_{\mathrm{m}}^{\log \cdot \log \log }, \leqslant_{\mathrm{m}}^{\log ^{\mathrm{k}}}\right\}$. We say $A$ and $B$ are $\leqslant_{\mathrm{m}}^{\mathrm{r}}$-equivalent, in notation $A \equiv_{\mathrm{m}}^{\mathrm{r}} B$, if $A \leqslant \leqslant_{\mathrm{m}}^{\mathrm{r}} B$ and $B \leqslant_{\mathrm{m}}^{\mathrm{r}} A . A$ is $\leqslant_{\mathrm{m}}^{\mathrm{r}}$-autoreducible, if $A \leqslant_{\mathrm{m}}^{\mathrm{r}} A$ via a reduction $f$ such that $f(x) \neq x$.
$A$ is $\leqslant_{\mathrm{m}}^{\log }$-mitotic (resp., $\leqslant_{\mathrm{m}}^{\mathrm{plog}}$-mitotic), if there exists a separator $S \in \mathrm{~L}$ (resp., $S \in \mathrm{PLOG}$ ) such that $A, A \cap S$, and $A \cap \bar{S}$ are pairwise $\leqslant_{\mathrm{m}}^{\log }$-equivalent (resp., $\leqslant_{\mathrm{m}}^{\mathrm{plog}}$-equivalent). ${ }^{1} A$ is $\leqslant_{\mathrm{m}}^{\log ^{\mathrm{k}}}$-mitotic (resp., $\leqslant_{\mathrm{m}}^{\log ^{\mathrm{k}} \cdot \log \log }$-mitotic), if there exists a separator $S \in \operatorname{DSPACE}\left(\log ^{k} n\right)$ (resp., $S \in \operatorname{DSPACE}\left(\left(\log ^{k} n\right) \cdot \log \log n\right)$ ) such that $A, A \cap S$, and $A \cap \bar{S}$ are pairwise $\leqslant_{\mathrm{m}}^{\log ^{k}}$-equivalent (resp., $\leqslant_{\mathrm{m}}^{\log ^{\mathrm{k}} \cdot \log \log }$-equivalent).

For an oracle $O$, let $L^{O}$ be the class of sets decidable by a log-space OTM that has access to oracle $O$. Similarly, $\mathrm{FL}^{O}$ is the class of functions computable by a log-space OTT that has access to oracle 0 . We also need the following relativized versions of log-space reducibilities, autoreducibility, and mitoticity.

Definition 2.1. Let $O$ be an oracle and let $A, B$ be sets of words.

$$
\begin{aligned}
& A \leqslant \leqslant_{\mathrm{m}}^{\log , O} B \quad \stackrel{d f}{\Longleftrightarrow} \text { there exists } f \in \mathrm{FL}^{O} \text { such that for all } x, \\
& (x \in A \Leftrightarrow f(x) \in B) \text {, } \\
& A \leqslant_{\mathrm{m}}^{\log -\operatorname{lin}, O} B \quad \stackrel{d f}{\Longleftrightarrow} \text { there exist } c>0 \text { and } f \in \mathrm{FL}^{O} \text { such that } \\
& \text { for all } x,|f(x)| \leqslant c|x| \text { and } \\
& (x \in A \Leftrightarrow f(x) \in B) \text {, } \\
& A \text { is } \leqslant_{\mathrm{m}}^{\log , O} \text {-autoreducible } \stackrel{d f}{\Longleftrightarrow} A \leqslant_{\mathrm{m}}^{\log , O} A \text { via a reduction } f \text { such that } \\
& f(x) \neq x \text {, } \\
& A \text { is } \leqslant_{\mathrm{m}}^{\log -l i n}, O \text {-autoreducible } \stackrel{d f}{\Longleftrightarrow} A \leqslant_{\mathrm{m}}^{\log -\operatorname{lin}, O} A \text { via a reduction } f \text { such that } \\
& f(x) \neq x, \\
& A \text { is } \leqslant \begin{array}{l}
\log , O \\
\mathrm{~m}
\end{array} \text {-mitotic } \stackrel{\text { df }}{\Longleftrightarrow} \text { there exists } S \in \mathrm{~L}^{O} \text { such that } \\
& A \equiv{ }_{\mathrm{m}}^{\log , O} A \cap S \equiv{ }_{\mathrm{m}}^{\log , O} A \cap \bar{S} .
\end{aligned}
$$

Proposition 2.2. If $L$ is $\leqslant_{\mathrm{m}}^{\log }$-mitotic such that $|\bar{L}| \geqslant 2$, then $L$ is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible.
Proof. If $L$ is $\leqslant_{\mathrm{m}}^{\log }$-mitotic, then there exist $S \in \mathrm{~L}$ and $f_{1}, f_{2} \in \mathrm{FL}$ such that $L \cap S \leqslant_{\mathrm{m}}^{\log } L \cap \bar{S}$ via $f_{1}$ and $L \cap \bar{S} \leqslant_{\mathrm{m}}^{\log } L \cap S$ via $f_{2}$. By assumption, there exist different words $v, w \in \bar{L}$. The following function is an $\leqslant_{\mathrm{m}}^{\log }$-autoreduction for $L$ (where min refers to quasi-lexicographic order).

$$
f^{\prime}(x) \stackrel{d f}{=} \begin{cases}f_{1}(x) & \text { if } x \in S \text { and } f_{1}(x) \notin S, \\ f_{2}(x) & \text { if } x \notin S \text { and } f_{2}(x) \in S, \\ \min (\{v, w\}-\{x\}) & \text { otherwise. }\end{cases}
$$

[^1]
## 3. The equivalence ( $\leqslant_{\mathrm{m}}^{\mathrm{plog}}$-autoreducible $\Leftrightarrow \leqslant_{\mathrm{m}}^{\text {plog }}$-mitotic)

This section establishes tight connections between autoreducibility and mitoticity in the (poly)log-space setting. With respect to nontrivial sets it holds that:

1. $\leqslant_{\mathrm{m}}^{\mathrm{plog}}$-autoreducible $\Leftrightarrow \leqslant_{\mathrm{m}}^{\mathrm{plog}}$-mitotic;
2. $\leqslant_{\mathrm{m}}^{\log }$-autoreducible $\Rightarrow \leqslant_{\mathrm{m}}^{\log \cdot \log \log }$-mitotic.

So for polylog-space many-one reductions we can prove an equivalence similar to the one that is known for polynomial-time many-one reductions [5]. However, for log-space many-one reductions we obtain mitoticity only if we grant the reduction a little more space than $O(\log n)$. Log-space many-one autoreducibility even implies $\left(\log n \cdot \log ^{(c)} n\right)$-space many-one mitoticity for every constant $c$. To obtain these results, we apply a combination of the construction used in the polynomial-time manyone setting [5] and the repeated deterministic coin tossing by Cole and Vishkin [3].

In the polynomial-time many-one setting, for a given set $L$ with polynomial-time many-one autoreduction $f$ (for simplicity we assume that $f$ is length-preserving) one has to show that $L$ is polynomial-time many-one mitotic. This is done by considering trajectories of the form $x, f(x), f(f(x)), \ldots$. Since $f$ is an autoreduction, the elements of a trajectory either all belong to $L$ or all belong to $\bar{L}$. One constructs a separator $S$ such that among the first $O(|x|)$ elements of an arbitrary trajectory $x, f(x), f(f(x)), \ldots$ there are elements from $S$ as well as elements from $\bar{S}$. Therefore, if one follows the trajectory of $x$ for $O(|x|)$ steps, one finds an element $y$ such that ( $x \in L \Leftrightarrow y \in L$ ) and ( $x \in S \Leftrightarrow y \notin S$ ). This is used to establish the polynomial-time many-one mitoticity of $L$.

For space-bounded many-one reductions a new difficulty comes up: In the log-space setting we can follow the trajectory only for constantly-many steps, but not for $O(|x|)$ steps. So if $f \in \mathrm{FL}$ and $c$ is constant, then we can compute $f^{(c)}$ in logspace, but we cannot compute $f^{(|x|)}(x)$ (since the intermediate results $f(x), f^{2}(x), \ldots$ cannot be stored in log-space). Here the repeated deterministic coin tossing [3] comes into play. With this technique it is possible to construct a well-balanced separator (which is used to establish mitoticity) such that instead of $|x|$ steps we only have to follow the trajectory for $\log \log |x|$ steps. This number can actually be dropped to any fixed number of repeated $\log$ operations, i.e., $\log ^{(c)}|x|$ where $c$ is constant.

We cannot prove that log-space many-one autoreducibility is equivalent to log-space many-one mitoticity. The lack of this equivalence is not due to our particular technique. In Section 4 we discuss in detail the deeper reason for the missing equivalence and we make this precise with the construction of an oracle relative to which the equivalence does not hold.

Theorem 3.1. Let $k \geqslant 1$ be an integer and let $L$ be $a \leqslant \leqslant_{\mathrm{m}}^{\log ^{\mathrm{k}}}$-autoreducible set such that $|\bar{L}| \geqslant 2$. Then there exist a total $g \in$ $\operatorname{FSPACE}\left(\left(\log ^{k^{7}} n\right) \cdot \log \log n\right)$ and a set $S \in \operatorname{DSPACE}\left(\log ^{k^{4}} n\right)$ such that for all $x$,

1. $x \in L \Leftrightarrow g(x) \in L$, and
2. $x \in S \Leftrightarrow g(x) \notin S$.

Proof. Let $f \in \operatorname{FSPACE}\left(\log ^{k} n\right)$ be $\mathrm{a} \leqslant \leqslant_{\mathrm{m}}^{\log ^{k}}$-autoreduction for $L$. Choose $c \geqslant 2$ such that $f$ can be computed in space $c \log ^{k} n$ and time $l(n) \stackrel{d f}{=} 2^{c \log ^{k} n}$. According to this time bound we now define a tower function.

$$
t(i) \stackrel{d f}{=} \begin{cases}2 & \text { for } i=0 \\ l(l(t(i-1))) & \text { for } i \geqslant 1\end{cases}
$$

Observe that the inverse tower function $t^{-1}(n) \stackrel{d f}{=} \min \{i \mid t(i) \geqslant n\}$ is computable in log-space in $n$ (i.e., linear space in the input length). Note that for all $n$,

$$
\begin{equation*}
t^{-1}(l(l(n)))=t^{-1}(n)+1 \tag{1}
\end{equation*}
$$

So from $f$ 's time bound we obtain for all $x$,

$$
\begin{equation*}
t^{-1}(|f(x)|) \leqslant t^{-1}(|x|)+1 \quad \text { and } \quad t^{-1}(|f(f(x))|) \leqslant t^{-1}(|x|)+1 \tag{2}
\end{equation*}
$$

We partition the set of all words as follows.

$$
\begin{aligned}
& S_{0} \stackrel{d f}{=}\left\{x \mid t^{-1}(|x|) \equiv 0(\bmod 2)\right\}, \\
& S_{1} \stackrel{d f}{=}\left\{x \mid t^{-1}(|x|) \equiv 1(\bmod 2)\right\} .
\end{aligned}
$$

Note that $S_{0}, S_{1} \in \mathrm{~L}$.

We use the following distance function for integers.

$$
d(x, y) \stackrel{d f}{=} \begin{cases}0 & \text { if } x=y \\ \operatorname{sgn}(y-x) \cdot\lfloor\log (2 \operatorname{abs}(y-x))\rfloor & \text { otherwise }\end{cases}
$$

This function is computable in log-space. Note that $d(x, y)=0$ if and only if $x=y$.
Claim 3.2. If $z_{1}, z_{2}$, and $z_{3}$ are integers such that $d\left(z_{1}, z_{2}\right)=d\left(z_{2}, z_{3}\right) \neq 0$, then there exist $i, j \in[1,3]$ such that for $r \stackrel{d f}{=} d\left(z_{1}, z_{2}\right)$,

$$
\left\lfloor z_{i} / 2^{\mathrm{abs}(r)}\right\rfloor \text { is even } \Leftrightarrow\left\lfloor z_{j} / 2^{\mathrm{abs}(r)}\right\rfloor \text { is odd. }
$$

Proof. Assume that the claim does not hold and let $z_{1}, z_{2}$, and $z_{3}$ be counter examples. Let $r \stackrel{d f}{=} d\left(z_{1}, z_{2}\right)=d\left(z_{2}, z_{3}\right), a_{1} \stackrel{d f}{=}$ $\left\lfloor z_{1} / 2^{\mathrm{abs}(r)}\right\rfloor, a_{2} \stackrel{d f}{=}\left\lfloor z_{2} / 2^{\mathrm{abs}(r)}\right\rfloor$, and $a_{3} \stackrel{d f}{=}\left\lfloor z_{3} / 2^{\mathrm{abs}(r)}\right\rfloor$. So by assumption, either $a_{1}, a_{2}$, and $a_{3}$ are all even or $a_{1}, a_{2}$, and $a_{3}$ are all odd. Without loss of generality let us assume that $a_{1}, a_{2}$, and $a_{3}$ are even (the other case is analogous).

Case 1: Assume $r>0$. So $z_{1}<z_{2}<z_{3}$ and hence $a_{1} \leqslant a_{2} \leqslant a_{3}$.
Assume $a_{1}=a_{3}$. From $d\left(z_{1}, z_{2}\right)=r$ it follows that $\log \left(2 \operatorname{abs}\left(z_{2}-z_{1}\right)\right) \geqslant r$ and hence $z_{2}-z_{1} \geqslant 2^{r-1}$. The same argument shows $z_{3}-z_{2} \geqslant 2^{r-1}$. So $z_{3} \geqslant z_{1}+2^{r}=z_{1}+2^{\text {abs }(r)}$ and hence $a_{3} \geqslant a_{1}+1$ which contradicts the assumption $a_{1}=a_{3}$.

So it holds that $a_{1}<a_{3}$. This implies $a_{3}-a_{1} \geqslant 2$, since both values are even. Since $a_{2}$ is even as well, we obtain $a_{2}-a_{1} \geqslant 2$ or $a_{3}-a_{2} \geqslant 2$. If $a_{2}-a_{1} \geqslant 2$, then $z_{2}-z_{1}>2^{r}$ and so $d\left(z_{1}, z_{2}\right) \geqslant r+1$. If $a_{3}-a_{2} \geqslant 2$, then $z_{3}-z_{2}>2^{r}$ and so $d\left(z_{2}, z_{3}\right) \geqslant r+1$. Both conclusions contradict the definition of $r$.

Case 2: Assume $r<0$. So $z_{1}>z_{2}>z_{3}$ and hence $a_{1} \geqslant a_{2} \geqslant a_{3}$.
Assume $a_{1}=a_{3}$. From $d\left(z_{1}, z_{2}\right)=r$ it follows that $\log \left(2 \operatorname{abs}\left(z_{2}-z_{1}\right)\right) \geqslant \operatorname{abs}(r)$ and hence $z_{1}-z_{2} \geqslant 2^{\mathrm{abs}(r)-1}$. The same argument shows $z_{2}-z_{3} \geqslant 2^{\text {abs }(r)-1}$. So $z_{1} \geqslant z_{3}+2^{\mathrm{abs}(r)}$ and hence $a_{1} \geqslant a_{3}+1$ which contradicts the assumption $a_{1}=a_{3}$.

So it holds that $a_{1}>a_{3}$. This implies $a_{1}-a_{3} \geqslant 2$, since both values are even. Since $a_{2}$ is even as well, we obtain $a_{1}-a_{2} \geqslant 2$ or $a_{2}-a_{3} \geqslant 2$. If $a_{1}-a_{2} \geqslant 2$, then $z_{1}-z_{2}>2^{\text {abs }(r)}$ and so $d\left(z_{1}, z_{2}\right) \leqslant-\operatorname{abs}(r)-1<r$. If $a_{2}-a_{3} \geqslant 2$, then $z_{2}-z_{3}>2^{\text {abs }(r)}$ and so $d\left(z_{2}, z_{3}\right) \leqslant-\operatorname{abs}(r)-1<r$. Both conclusions contradict the definition of $r$. This proves Claim 3.2.

We define the separator $S$ by the following algorithm which works on input $x$.

```
// Algorithm for the set S
y:= f(x), z:= fi(2)}(\textrm{x}),\quadu:=\mp@subsup{\textrm{f}}{}{(3)}(\textrm{x}),\quad\textrm{v}:=\mp@subsup{\textrm{f}}{}{(4)}(\textrm{x}
x}:=d(x,y),\quad\mp@subsup{y}{}{\prime}:=d(y,z),\quad\mp@subsup{z}{}{\prime}:=d(z,u),\quad\mp@subsup{u}{}{\prime}:=d(u,v
x
if }|x|<|y| and (x\inS S0\Leftrightarrowy\inS S ) then accep
if }|y|<|z| and (y\inS S \Leftrightarrow z G S S ) then rejec
// Phase 1
if d( }\mp@subsup{\textrm{x}}{}{\prime\prime},\mp@subsup{\textrm{y}}{}{\prime\prime})>d(\mp@subsup{y}{}{\prime\prime},\mp@subsup{z}{}{\prime\prime})\mathrm{ then accept
if }d(\mp@subsup{x}{}{\prime\prime},\mp@subsup{y}{}{\prime\prime})<d(\mp@subsup{y}{}{\prime\prime},\mp@subsup{z}{}{\prime\prime})\mathrm{ then reject
// Phase 2
if d( }\mp@subsup{\textrm{x}}{}{\prime\prime},\mp@subsup{\textrm{y}}{}{\prime\prime})\not=0\mathrm{ then
    r'\prime}:=d(\mp@subsup{x}{}{\prime\prime},\mp@subsup{y}{}{\prime\prime}
    accept iff \lfloor\mp@subsup{x}{}{\prime\prime}/\mp@subsup{2}{}{\textrm{abs}(\mp@subsup{r}{}{\prime\prime})}\rfloor}\mathrm{ is even
endif
// Phase 3
if }d(\mp@subsup{x}{}{\prime},\mp@subsup{y}{}{\prime})\not=0\mathrm{ then
    r':=d(\mp@subsup{x}{}{\prime},\mp@subsup{y}{}{\prime})
    accept iff \lfloor\mp@subsup{x}{}{\prime}/\mp@subsup{2}{}{abs(r')}\rfloor}\mathrm{ is even
endif
// Phase 4
r:=d(x,y)
accept iff \lfloorx/2abs(r)}\rfloor\mathrm{ is even
```

We argue that $S \in \operatorname{DSPACE}\left(\log ^{k^{4}} n\right)$. By assumption, $f \in \operatorname{FSPACE}\left(\log ^{k} n\right)$. It follows that $f^{(2)} \in \operatorname{FSPACE}\left(\log ^{k^{2}} n\right), f^{(3)} \in$ FSPACE $\left(\log ^{k^{3}} n\right)$, and $f^{(4)} \in \operatorname{FSPACE}\left(\log ^{k^{4}} n\right)$. So $x, y, z, u$, and $v$ can be computed in space $O\left(\log ^{k^{4}} n\right)$ (although these values are too large to be stored in space $O\left(\log ^{k^{4}} n\right)$ ). The distance function $d$ is computable in log-space. Therefore, $x^{\prime}, y^{\prime}, z^{\prime}$, and
$u^{\prime}$ can be computed in space $O\left(\log ^{k^{4}} n\right)$. Note that the lengths of these values are of order $O\left(\log ^{k^{4}} n\right)$. Hence $x^{\prime}, y^{\prime}, z^{\prime}$, and $u^{\prime}$ can be even stored in space $O\left(\log ^{k^{4}} n\right)$. So also $x^{\prime \prime}, y^{\prime \prime}, z^{\prime \prime}, d\left(x^{\prime \prime}, y^{\prime \prime}\right)$, and $d\left(y^{\prime \prime}, z^{\prime \prime}\right)$ can be computed and stored in space $O\left(\log ^{k^{4}} n\right)$. The tests for membership in $S_{0}$ and $S_{1}$ (lines 4 and 5) are possible in space $O\left(\log ^{k^{4}} n\right)$, since $S_{0}, S_{1} \in \mathrm{~L}$. All remaining steps can be carried out in space $O\left(\log ^{k^{4}} n\right)$, since they work on variables that are either stored on the working tape or (in case of $x$ ) are written on the input tape. This shows $S \in \operatorname{DSPACE}\left(\log ^{k^{4}} n\right)$.

Let $e \stackrel{d f}{=} 43 k^{2}\lceil\log c\rceil$. The function $g$ is defined by the following algorithm working on input $x$.

```
// Algorithm for function g
n}:=|x|,m:=e+6\mp@subsup{k}{}{3}\lceil\operatorname{log}\operatorname{log}n
for i:=1 to m
    // here }|\mp@subsup{f}{}{(i)}(x)|\leqslant\mp@subsup{I}{}{(3)}(n
    if }x\inS\Leftrightarrow\mp@subsup{f}{}{(i)}(\textrm{x})\not\inS\mathrm{ then return }\mp@subsup{\textrm{f}}{}{(i)}(\textrm{x}
next i
// this line is never reached
```

Claim 3.3. In the algorithm for $g$, the invariant in line 3 holds.

Proof. Assume there exists an $i \in[1, m]$ such that the algorithm reaches the $i$-th instance of the loop and there it holds that $\left|f^{(i)}(x)\right|>l^{(3)}(n)$. Choose the smallest such $i$ and let $s=t^{-1}(n)$. Note that $i \geqslant 4$, since $f$ 's computation time is bounded by $l(n)$. Observe that by $(1), t^{-1}\left(\left|f^{(i)}(x)\right|\right) \geqslant t^{-1}(l(l(n)))=s+1$. From (1) we also obtain that either

$$
\begin{equation*}
t^{-1}(l(n))=s \quad \text { and } \quad t^{-1}(l(l(n)))=s+1 \tag{3}
\end{equation*}
$$

or

$$
\begin{equation*}
t^{-1}(l(n))=t^{-1}(l(l(n)))=s+1 \quad \text { and } \quad t^{-1}\left(l^{(3)}(n)\right)=s+2 \tag{4}
\end{equation*}
$$

Recall that by (2), if $j_{0}<j_{2}$ such that $t^{-1}\left(\left|f^{\left(j_{0}\right)}(x)\right|\right)=s$ and $t^{-1}\left(\left|f^{\left(j_{2}\right)}(x)\right|\right)=s+2$, then there exists $j_{1} \in\left(j_{0}, j_{2}-2\right]$ such that $t^{-1}\left(\left|f^{\left(j_{1}\right)}(x)\right|\right)=s+1$ and $t^{-1}\left(\left|f^{\left(j_{1}+1\right)}(x)\right|\right)=s+1$. If (3) holds, then $t^{-1}(|f(x)|) \leqslant s$ and so there exists $j \in[2, i]$ such that for $u=f^{(j-2)}(x), v=f^{(j-1)}(x)$ and $w=f^{(j)}(x)$ it holds that $t^{-1}(|u|)=t^{-1}(|v|)=s$ and $t^{-1}(|w|)=s+1$. If (4) holds, then there exists $j \in[2, i]$ such that for $u=f^{(j-2)}(x), v=f^{(j-1)}(x)$ and $w=f^{(j)}(x)$ it holds that $t^{-1}(|u|)=t^{-1}(|v|)=s+1$ and $t^{-1}(|w|)=s+2$. In both cases we have $\left(u \in S_{0} \Leftrightarrow v \in S_{0}\right)$ and ( $v \in S_{0} \Leftrightarrow w \in S_{1}$ ). If we consider the algorithm for $S$, then we see that $u \notin S$ and $v \in S$. Therefore, in the algorithm for $g$, the condition in line 4 is either satisfied for $i=j-2$ or is satisfied for $i=j-1$. This contradicts our assumption that we reach the $i$-th instance of the loop. This proves Claim 3.3.

Claim 3.4. $g \in \operatorname{FSPACE}\left(\left(\log ^{k^{7}} n\right) \cdot \log \log n\right)$.
Proof. Let $n=|x|$ and $m=e+6 k^{3}\lceil\log \log |x|\rceil$. Choose $m^{\prime} \in[1, m]$ such that the computation $g(x)$ stops after the $m^{\prime}$-th instance of the loop. We describe how to compute the values $f^{(1)}(x), f^{(2)}(x), \ldots, f^{\left(m^{\prime}\right)}(x)$ in space $2 \cdot c \log ^{k}\left(l^{(3)}(n)\right) \cdot m$. For this, we use $m$ blocks of space $2 \cdot c \log ^{k}\left(l^{(3)}(n)\right)$. The right (resp., left) part of a block are the first (resp., $\left.\operatorname{last}\right) c \log ^{k}\left(l^{(3)}(n)\right)$ storage cells in the block. The $i$-th block is used to compute single bits of $f^{(i)}(x)$. More precisely, block $i$ interprets its left part as a number $j$ and it uses its right part to compute the $j$-th bit of $f^{(i)}(x)$. The latter computation uses the block $i-1$ to compute single bits of $f^{(i-1)}(x)$ (if $i=0$, then these bits can be directly read on the input tape). By Claim 3.3 , for $i \in\left[1, m^{\prime}\right]$ it holds that $\left|f^{(i)}(x)\right| \leqslant l^{(3)}(n)$. Hence, $f^{(i+1)}(x)$ is computable in space $c \log ^{k}\left(l^{(3)}(n)\right)$ if we have access to $f^{(i)}(x)$. So the space in the right part of the $i$-th block suffices to compute $f^{(i)}(x)$. This shows that we can compute $f^{(1)}(x), f^{(2)}(x), \ldots, f^{\left(m^{\prime}\right)}(x)$ in space $2 \cdot c \log ^{k}\left(l^{(3)}(n)\right) \cdot m$. Moreover, for the test $f^{(i)}(x) \notin S$ in line 4 we need space $O\left(\log ^{k^{4}}\left(l^{(3)}(n)\right)\right)$. Observe that

$$
\log \left(l^{(3)}(n)\right)=c^{k^{2}+k+1} \cdot \log ^{k^{3}} n
$$

So the space needed to compute $g(x)$ can be estimated by

$$
\begin{aligned}
2 \cdot c \log ^{k}\left(l^{(3)}(n)\right) \cdot m+O\left(\log ^{k^{4}}\left(l^{(3)}(n)\right)\right) & =O\left(m \cdot \log ^{k^{4}} n\right)+O\left(\log ^{k^{7}} n\right) \\
& \leqslant O\left(\left(\log ^{k^{7}} n\right) \cdot \log \log n\right)
\end{aligned}
$$

Note that the factor $\log \log n$ is needed only for $k=1$. This proves Claim 3.4.

Claim 3.5. The algorithm for $g$ never reaches line 6 .

Proof. Assume that for some input $x$, the algorithm reaches line 6. Let $n=|x|, m=e+6 k^{3}\lceil\log \log |x|\rceil$, and $x_{i}=f^{(i)}(x)$ for $i \geqslant 0$. Hence, for $i \in[1, m]$ it holds that $x \in S \Leftrightarrow x_{i} \in S$. Without loss of generality let us assume that $x_{i} \in S$ for all $i \in[0, m]$.

All remaining arguments refer to the algorithm for $S$. For $i \in[1, m]$ it holds that the algorithm on input $x_{i}$ does not stop in line 4 , since otherwise $x_{i-1}$ stops in line 5 which contradicts the assumption $x_{i-1} \in S$. (Here one has to note that if $x_{i}$ stops in line 4 , then by (2), $x_{i-1}$ cannot stop in line 4 as well.) So for all $i \in[1, m]$, the algorithm on input $x_{i}$ reaches phase 1 (line 6).

Phase 1: For $i \geqslant 0$ define $y_{i}$ to be the value of the program variable y when the algorithm for $S$ works on input $x_{i}$. In the same way we define $z_{i}, u_{i}, v_{i}, x_{i}^{\prime}, y_{i}^{\prime}, z_{i}^{\prime}, u_{i}^{\prime}, x_{i}^{\prime \prime}, y_{i}^{\prime \prime}$, and $z_{i}^{\prime \prime}$. Note that $y_{i}=x_{i+1}, z_{i}=y_{i+1}, u_{i}=z_{i+1}, v_{i}=u_{i+1}, y_{i}^{\prime}=x_{i+1}^{\prime}$, $z_{i}^{\prime}=y_{i+1}^{\prime}, u_{i}^{\prime}=z_{i+1}^{\prime}, y_{i}^{\prime \prime}=x_{i+1}^{\prime \prime}$, and $z_{i}^{\prime \prime}=y_{i+1}^{\prime \prime}$.

We show that there are not too many elements $i \in[1, m]$ such that the algorithm on input $x_{i}$ stops in line 7. By Claim 3.3, for $i \in[1, m],\left|x_{i}\right| \leqslant l^{(3)}(n)$. So the following holds for $i \in[1, m-4]$ : The lengths of $x_{i}, y_{i}, z_{i}, u_{i}$, and $v_{i}$ are bounded by $l^{(3)}(n)$. By the definition of the distance function $d$, the values $\operatorname{abs}\left(x_{i}^{\prime}\right)$, $\operatorname{abs}\left(y_{i}^{\prime}\right)$, abs $\left(z_{i}^{\prime}\right)$, and $\operatorname{abs}\left(u_{i}^{\prime}\right)$ are bounded by $1+\log \left(2^{l^{(3)}(n)}\right)=$ $1+l^{(3)}(n) \leqslant 2 \cdot l^{(3)}(n)$. So the values abs $\left(x_{i}^{\prime \prime}\right)$, abs $\left(y_{i}^{\prime \prime}\right)$, and $\operatorname{abs}\left(z_{i}^{\prime \prime}\right)$ are bounded by $3+\log l^{(3)}(n) \leqslant 2 \cdot \log l^{(3)}(n)$. Hence the values $\operatorname{abs}\left(d\left(x_{i}^{\prime \prime}, y_{i}^{\prime \prime}\right)\right)$ and $\operatorname{abs}\left(d\left(y_{i}^{\prime \prime}, z_{i}^{\prime \prime}\right)\right)$ are bounded as follows where $c^{\prime} \stackrel{d f}{=} 6 k^{2} \log c$ :

$$
\begin{aligned}
3+\log \log l^{(3)}(n) & =3+\log \log \left(2^{{c^{k^{2}+k+1} \cdot \log ^{k^{3}}} n}\right) \\
& =3+\log \left(c^{k^{2}+k+1} \cdot \log ^{k^{3}} n\right) \\
& =3+\left(k^{2}+k+1\right)(\log c)+k^{3} \log \log n \\
& \leqslant c^{\prime}+k^{3} \log \log n .
\end{aligned}
$$

We now consider the sequence of $d\left(x_{i}^{\prime \prime}, y_{i}^{\prime \prime}\right)$ for $i \in[1, m-4]$. This sequence is not decreasing, since otherwise we stop in line 8 which contradicts the assumption $x_{i} \in S$. We have seen that the values in this sequence are integers in $\left[-c^{\prime}-k^{3} \log \log n, c^{\prime}+k^{3} \log \log n\right]$. So the number of positions where the sequence increases is at most

$$
2\left(c^{\prime}+k^{3} \log \log n\right) \leqslant\left(12 k^{2} \log c\right)+2 k^{3} \log \log n \leqslant \frac{m-7}{3}
$$

This shows that the number of $i \in[1, m-4]$ such that the algorithm on input $x_{i}$ stops in line 7 is at most $(m-7) / 3$. By a pigeon hole argument, there exists a $j \in[1, m-4]$ such that the algorithm reaches phase 2 (line 9 ) for the inputs $x_{j}, x_{j+1}$, and $x_{j+2}$.

Phase 2: For $i \in[j, j+2]$ it holds that $d\left(x_{i}^{\prime \prime}, y_{i}^{\prime \prime}\right)=d\left(y_{i}^{\prime \prime}, z_{i}^{\prime \prime}\right)$. Let $r^{\prime \prime} \stackrel{d f}{=} d\left(x_{j}^{\prime \prime}, y_{j}^{\prime \prime}\right)$. It follows that

$$
r^{\prime \prime}=d\left(x_{j}^{\prime \prime}, x_{j+1}^{\prime \prime}\right)=d\left(x_{j+1}^{\prime \prime}, x_{j+2}^{\prime \prime}\right)=d\left(x_{j+2}^{\prime \prime}, x_{j+3}^{\prime \prime}\right)=d\left(x_{j+3}^{\prime \prime}, x_{j+4}^{\prime \prime}\right)
$$

Assume $r^{\prime \prime} \neq 0$. So the algorithm stops in line 12 , if the input is $x_{j}, x_{j+1}$, or $x_{j+2}$ (since $d\left(x_{j+1}^{\prime \prime}, y_{j+1}^{\prime \prime}\right)=d\left(y_{j+1}^{\prime \prime}, z_{j+1}^{\prime \prime}\right) \neq 0$ and $d\left(x_{j+2}^{\prime \prime}, y_{j+2}^{\prime \prime}\right)=d\left(y_{j+2}^{\prime \prime}, z_{j+2}^{\prime \prime}\right) \neq 0$. We apply Claim 3.2 to $x_{j}^{\prime \prime}, x_{j+1}^{\prime \prime}$, and $x_{j+2}^{\prime \prime}$. We obtain $i_{1}, i_{2} \in[j, j+2]$ such that

$$
\left\lfloor x_{i_{1}}^{\prime \prime} / 2^{\mathrm{abs}\left(r^{\prime \prime}\right)}\right\rfloor \text { is even } \Leftrightarrow\left\lfloor x_{i_{2}}^{\prime \prime} / 2^{\mathrm{abs}\left(r^{\prime \prime}\right)}\right\rfloor \text { is odd. }
$$

So on both inputs, $x_{i_{1}}$ and $x_{i_{2}}$, the algorithm stops in line 12, but one input is accepted and the other one is rejected. This contradicts our assumption $x_{i_{1}} \in S \Leftrightarrow x_{i_{2}} \in S$. So it must hold that $r^{\prime \prime}=0$. Hence the algorithm reaches phase 3 (line 14) for the inputs $x_{j}, x_{j+1}$, and $x_{j+2}$.

Phase 3: From $r^{\prime \prime}=0$ it follows that $x_{j}^{\prime \prime}=x_{j+1}^{\prime \prime}=x_{j+2}^{\prime \prime}$. Let $r^{\prime} \stackrel{d f}{=} x_{j}^{\prime \prime}$. It follows that

$$
r^{\prime}=d\left(x_{j}^{\prime}, x_{j+1}^{\prime}\right)=d\left(x_{j+1}^{\prime}, x_{j+2}^{\prime}\right)=d\left(x_{j+2}^{\prime}, x_{j+3}^{\prime}\right)
$$

Assume $r^{\prime} \neq 0$. So the algorithm stops in line 17, if the input is $x_{j}, x_{j+1}$, or $x_{j+2}$. We apply Claim 3.2 to $x_{j}^{\prime}, x_{j+1}^{\prime}$, and $x_{j+2}^{\prime}$. We obtain $i_{1}, i_{2} \in[j, j+2]$ such that

$$
\left\lfloor x_{i_{1}}^{\prime} / 2^{\mathrm{abs}\left(r^{\prime}\right)}\right\rfloor \text { is even } \Leftrightarrow\left\lfloor x_{i_{2}}^{\prime} / 2^{\mathrm{abs}\left(r^{\prime}\right)}\right\rfloor \text { is odd. }
$$

So on both inputs, $x_{i_{1}}$ and $x_{i_{2}}$, the algorithm stops in line 17, but one input is accepted and the other one is rejected. This contradicts our assumption $x_{i_{1}} \in S \Leftrightarrow x_{i_{2}} \in S$. So it must hold that $r^{\prime}=0$. Hence the algorithm reaches phase 4 (line 19) for the inputs $x_{j}, x_{j+1}$, and $x_{j+2}$.

Phase 4: From $r^{\prime}=0$ it follows that $x_{j}^{\prime}=x_{j+1}^{\prime}=x_{j+2}^{\prime}$. Let $r \stackrel{d f}{=} x_{j}^{\prime}$. It follows that

$$
r=d\left(x_{j}, x_{j+1}\right)=d\left(x_{j+1}, x_{j+2}\right)=d\left(x_{j+2}, x_{j+3}\right)
$$

If $r=0$, then $x_{j}=x_{j+1}=f\left(x_{j}\right)$ which contradicts the assumption $f(x) \neq x$. So $r \neq 0$ and we can apply Claim 3.2 to $x_{j}, x_{j+1}$, and $x_{j+2}$. We obtain $i_{1}, i_{2} \in[j, j+2]$ such that

$$
\left\lfloor x_{i_{1}} / 2^{\mathrm{abs}(r)}\right\rfloor \text { is even } \Leftrightarrow\left\lfloor x_{i_{2}} / 2^{\mathrm{abs}(r)}\right\rfloor \text { is odd. }
$$

So on both inputs, $x_{i_{1}}$ and $x_{i_{2}}$, the algorithm stops in line 21, but one input is accepted and the other one is rejected. This contradicts our assumption $x_{i_{1}} \in S \Leftrightarrow x_{i_{2}} \in S$ and finishes the proof of Claim 3.5.

With Claim 3.5 at hand we can easily finish the proof of the theorem. Let $x$ be the input to the algorithm for $g$. Since we do not reach line 6, we must stop in line 4 and therefore $(x \in S \Leftrightarrow g(x) \notin S)$. Also, it holds that $g(x)=f^{(i)}(x)$ for some $i \geqslant 1$. So ( $x \in L \Leftrightarrow g(x) \in L$ ), since $f$ is an autoreduction for $L$. This proves Theorem 3.1.

Corollary 3.6. Let $k \geqslant 1$ be an integer and let $L$ be $a \leqslant_{\mathrm{m}}^{\log ^{k}}$-autoreducible set such that $|\bar{L}| \geqslant 1$. Then $L$ is $\leqslant_{\mathrm{m}}^{\log ^{k^{7}}} \cdot \log \log$-mitotic.

Proof. Note that if $L$ is $\leqslant \leqslant_{\mathrm{m}} \log ^{k}$-autoreducible and $|\bar{L}| \geqslant 1$, then $|\bar{L}| \geqslant 2$. From Theorem 3.1 we obtain $g \in \operatorname{FSPACE}\left(\left(\log ^{k^{7}} n\right)\right.$. $\log \log n$ ) and $S \in \operatorname{DSPACE}\left(\log ^{k^{4}} n\right)$ such that $(x \in L \Leftrightarrow g(x) \in L)$ and $(x \in S \Leftrightarrow g(x) \notin S)$. Thus $L \cap S \leqslant_{\mathrm{m}}^{\log ^{k^{7}} \cdot \log \log } L \cap \bar{S}$ and $L \cap \bar{S} \leqslant_{\mathrm{m}}^{\log ^{k^{7}}} \cdot \log \log L \cap S$, both via $g$. This shows $L \cap S \equiv \equiv_{\mathrm{m}}^{\log ^{k^{7}}} \cdot \log \log L \cap \bar{S}$.

The following function $g^{\prime}$ witnesses $L \leqslant_{\mathrm{m}}^{\log ^{7^{7}}} \cdot \log \log L \cap S$ : If $x \in S$, then $g^{\prime}(x)=x$ else $g^{\prime}(x)=g(x)$. Moreover, the following function $g^{\prime \prime}$ witnesses $L \cap S \leqslant \leqslant_{\mathrm{m}}^{\log ^{\mathrm{k}^{7}} \cdot \log \log } L$ : If $x \in S$, then $g^{\prime \prime}(x)=x$ else $g^{\prime \prime}(x)=w_{1}$, where $w_{1}$ is a fixed word in $\bar{L}$. This shows $L \equiv \equiv_{\mathrm{m}}^{\log ^{\mathrm{k}^{7}}} \cdot \log \log L \cap S$ and analogously we obtain $L \equiv \equiv_{\mathrm{m}}^{\log ^{k^{7}}} \cdot \log \log L \cap \bar{S}$.

Corollary 3.7. Let $L$ be any set such that $|\bar{L}| \geqslant 1$. If $L$ is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible, then $L$ is $\leqslant \frac{\log \cdot \log \log \text {-mitotic. }}{}$
Remark 3.8. Corollary 3.7 can be improved in the sense that for every $k \geqslant 1$, if $L$ is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible and $|\bar{L}| \geqslant 1$, then $L$ is mitotic with respect to many-one reductions that belong to

$$
\operatorname{FSPACE}((\log n) \cdot \underbrace{\log \log \cdots \log }_{k \text { times }} n) .
$$

This is achieved by the following modification of the algorithm for $S$ (Theorem 3.1). Compute values $x^{\prime \prime \prime}, x^{\prime \prime \prime \prime}, \ldots$ and $y^{\prime \prime \prime}, y^{\prime \prime \prime \prime}, \ldots$ and $z^{\prime \prime \prime}, z^{\prime \prime \prime \prime}, \ldots$ until one arrives at variables with $k$ primes. Change phase 1 to consider these latter variables instead of $x^{\prime \prime}, y^{\prime \prime}$, and $z^{\prime \prime}$. Proceed with phases that consider variables with $k-1$ primes, then with $k-2$ primes, and so on. So the notions $\leqslant_{\mathrm{m}}^{\log }$-autoreducibility and $\leqslant_{\mathrm{m}}^{\log }$-mitoticity are even closer than stated in Corollary 3.7.

Corollary 3.9. Let $L$ be any set such that $|\bar{L}| \geqslant 2 . L$ is $\leqslant_{\mathrm{m}}^{\mathrm{plog}}$-autoreducible if and only if $L$ is $\leqslant_{\mathrm{m}}^{\mathrm{plog}}$-mitotic.
Proof. If $L$ is $\leqslant_{\mathrm{m}}^{\text {plog }}$-mitotic, then there exist $S \in$ PLOG and $f_{1}, f_{2} \in$ FPLOG such that $L \cap S \leqslant \leqslant_{\mathrm{m}}^{\text {plog }} L \cap \bar{S}$ via $f_{1}$ and $L \cap \bar{S} \leqslant \leqslant_{\mathrm{m}}^{\text {plog }}$ $L \cap S$ via $f_{2}$. By assumption, there exist different words $v, w \in \bar{L}$. The following function $f^{\prime}$ is a $\leqslant_{\mathrm{m}}^{\text {plog }}$-autoreduction for $L$ : If $x \in S$ and $f_{1}(x) \notin S$, then $f^{\prime}(x)=f_{1}(x)$. If $x \notin S$ and $f_{2}(x) \in S$, then $f^{\prime}(x)=f_{2}(x)$. Otherwise, $f^{\prime}(x)=\min (\{v, w\}-\{x\})$.

The other direction follows from Corollary 3.6.

## 4. The difficulty of $\left(\leqslant_{\mathrm{m}}^{\log }\right.$-autoreducible $\Rightarrow \leqslant_{\mathrm{m}}^{\log }$-mitotic)

We know that the notions of $\leqslant_{\mathrm{m}}^{\mathrm{p}}$-autoreducibility and $\leqslant_{\mathrm{m}}^{\mathrm{p}}$-mitoticity are equivalent [5]. In the preceding section we showed that with respect to log-space many-one reductions, these notions are almost equivalent:

$$
\begin{aligned}
\leqslant_{\mathrm{m}}^{\log } \text {-mitotic } & \Rightarrow \leqslant_{\mathrm{m}}^{\log } \text {-autoreducible } ; \\
\leqslant_{\mathrm{m}}^{\log \text {-autoreducible }} & \Rightarrow \leqslant_{\mathrm{m}}^{\log \cdot \log \log } \text {-mitotic. }
\end{aligned}
$$

In this section we explain in detail the reason why it is difficult to establish the full equivalence. This is done in three steps. First, in Section 4.1 we describe this difficulty on an intuitive level. Then, in Section 4.2 we sketch the construction of a relativized world where this difficulty becomes provable. Finally, in Section 4.3 we give the detailed oracle construction. Relative to our oracle, $\leqslant_{\mathrm{m}}^{\log }$-autoreducibility and $\leqslant_{\mathrm{m}}^{\log }$-mitoticity are not equivalent. This result holds with respect to all models of log-space oracle machines that were discussed in the preliminaries section.

It is difficult to show unconditionally that $\leqslant_{\mathrm{m}}^{\log }$-autoreducibility does not imply $\leqslant_{\mathrm{m}}^{\log }$-mitoticity, since such a proof separates $L$ from $P$.

$$
\mathrm{L}=\mathrm{P} \Rightarrow \leqslant_{\mathrm{m}}^{\log } \text {-autoreducibility and } \leqslant_{\mathrm{m}}^{\log } \text {-mitoticity are equivalent. }
$$

This is seen as follows: $\mathrm{L}=\mathrm{P}$ implies $\mathrm{FL}=\mathrm{FP}$. If $A$ is $\leqslant_{\mathrm{m}}^{\mathrm{log}}$-autoreducible, then it is $\leqslant_{\mathrm{m}}^{\mathrm{p}}$-autoreducible and hence $\leqslant_{\mathrm{m}}^{\mathrm{p}}$ mitotic [5]. So there exists a separator $S \in \mathrm{P}=\mathrm{L}$ such that $A \equiv_{\mathrm{m}}^{\mathrm{p}} A \cap S \equiv_{\mathrm{m}}^{\mathrm{p}} A \cap \bar{S}$. From FL $=\mathrm{FP}$ it follows that $A \equiv{ }_{\mathrm{m}}^{\log }$ $A \cap S \equiv \equiv_{\mathrm{m}}^{\log } A \cap \bar{S}$ and hence $A$ is $\leqslant_{\mathrm{m}}^{\log }$-mitotic.

So in the case of log-space reductions we observe a behavior that differs from the experience we had with polynomialtime reductions. For the latter, either autoreducibility and mitoticity coincide (i.e., for $\leqslant_{\mathrm{m}}^{\mathrm{p}}$ ) or it is possible to separate the notions unconditionally (i.e., for all reductions between $\leqslant_{2-\mathrm{tt}}^{\mathrm{p}}$ and $\leqslant_{\mathrm{T}}^{\mathrm{p}}$ ). In contrast, with respect to log-space manyone reductions, it appears as a plausible possibility that autoreducibility and mitoticity are different, but we cannot prove
 inequivalent, but very similar notions.

### 4.1. Explanation on an intuitive level

We give an intuitive explanation of the difficulty of transforming $\leqslant_{m}^{\log _{m}}$-autoreducibility into $\leqslant_{m}^{\log _{m}}$-mitoticity. It is in the nature of such explanations that our arguments will be simplified and informal. For the exact and detailed construction we refer to Section 4.3.

We say that a function $f \in$ FL has difficult, detached cycles if for every $g \in$ FL there exists a constant $c>0$ such that for infinitely many $x$ :

1. $T_{x} \stackrel{d f}{=}\left\{f^{(0)}(x), f^{(1)}(x), \ldots, f^{(|x|-1)}(x)\right\}$ has cardinality $|x|$ and it holds that $f^{(|x|)}(x)=x$;
2. $f^{-1}\left(T_{x}\right) \subseteq T_{x}$;
3. $\forall y \in T_{x},\left[g(y) \in T_{x} \Rightarrow g(y) \in\left\{f^{(0)}(y), f^{(1)}(y), \ldots, f^{(c)}(y)\right\}\right]$.

Item 1 states that the trajectory of $x$ is a cycle of length $|x|$. Item 2 says that no other arguments are mapped to $T_{X}$ and therefore, the trajectory of $x$ is not connected to other trajectories. Item 3 describes a certain hardness of $f$ : For a given element in the trajectory, a log-space machine can only compute constantly-many successors. This is consistent with the fact that $f^{(c)}(x) \in$ FL for all $f \in$ FL and all constants $c$, and it is also consistent with our impression that $f^{(t(x))}(x)$ is not necessarily in FL if $t$ is not constant.

At first glance, the property of having difficult, detached cycles might appear artificial and very strong. However, there is no reason to exclude the existence of functions $f \in$ FL that have this property and that satisfy $f(x) \neq x$. For example, with our construction below we demonstrate a relativized world in which such functions exist.

Suppose $f$ has difficult, detached cycles. We use $f$ for the construction of a set $L$ that is $\leqslant_{\mathrm{m}}^{\log _{\text {a }}}$-autoreducible via $f$, but not $\leqslant_{\mathrm{m}}^{\log }$-mitotic. By item 3 (the hardness condition), every log-space machine can only compute a constant-size preview of $f$ 's trajectory. Therefore, every log-space computable separator $S$ that claims to establish the $\leqslant_{\mathrm{m}}^{\log }$-mitoticity of $L$ can only compute such a constant-size preview of $f$ 's trajectory. This implies that with respect to the trajectory of $f$, every separator $S$ of $L$ acts like a relation of constant arity, since it depends only on constantly-many successors of the input $x$. From Ramsey theory (more precisely, the existence of the generalized Ramsey numbers) it follows that for every $c \geqslant 0$ there exists an $x$ such that $x, f(x), \ldots, f^{(c)}(x)$ have the same membership with respect to $S$ (i.e., either all belong to $S$ or all belong to $\bar{S}$ ). Again by item 3, every log-space computable function $g$ that claims to establish $L \cap S \leqslant{ }_{\mathrm{m}}^{\log } L \cap \bar{S}$ (which is needed for $\leqslant_{\mathrm{m}}^{\log }$-mitoticity) can only compute a constant-size preview of $f$ 's trajectory. So by choosing $c$ large enough we can enforce that either $g(x) \in\left\{x, f(x), \ldots, f^{(c)}(x)\right\}$ and hence

$$
x \in S \quad \Leftrightarrow \quad f(x) \in S
$$

or $g(x)$ does not belong to $x$ 's trajectory with respect to $f$ in which case we can (by diagonalization) construct $L$ such that

$$
x \in L \quad \Leftrightarrow \quad g(x) \notin L .
$$

So $g$ is not a $\leqslant_{\mathrm{m}}^{\log }$-reduction from $L \cap S$ to $L \cap \bar{S}$. In this way we diagonalize against all pairs $(S, g)$ and obtain a set $L$ that is not $\leqslant_{\mathrm{m}}^{\log }$-mitotic. More precisely, our diagonalization is in such a way that we put whole trajectories inside or outside $L$. This implies that $L$ is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible via $f$.

In Section 4.3 below we make the described scenario precise and construct an oracle relative to which there exists a set $L$ that is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible, but not $\leqslant_{\mathrm{m}}^{\log }$-mitotic.

### 4.2. Road map for the oracle construction

While neglecting technical details we sketch the main arguments of the construction. In the first part, with the stagewise construction of an oracle $O$ we create a suitable relativized environment. Then, in the second part, we use this environment and construct a language $L$ that is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible, but not $\leqslant_{\mathrm{m}}^{\log }$-mitotic.

We start with the description of stage $s$ of the construction of $O$. There we diagonalize against two log-space machines $M_{1}$ (a possible log-space separator) and $M_{2}$ (a possible log-space reduction function). At the beginning we choose $n$ large enough such that changing the oracle with respect to words of length $\geqslant n^{2}$ does not affect separations made in earlier stages. Then we choose a set $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$ and $S$ has maximal Kolmogorov complexity. In particular, all subsets of $S$ have high Kolmogorov complexity.

Now let us observe that each $T \subseteq S$ of cardinality $\geqslant 2$ induces a particular set $\langle T\rangle \subseteq \Sigma^{n^{2}}$ which can be used to define the oracle with respect to words of length $n^{2}$. This set $\langle T\rangle$ is defined as follows: If $w_{0}, \ldots, w_{k-1}$ are the words in $T$ in ascending order, then the characteristic sequence of $\langle T\rangle$ (considered as a subset of $\Sigma^{n^{2}}$ ) is

$$
0 \cdots 01 w_{1} 0 \cdots 01 w_{2} 0 \cdots 01 w_{k-1} 0 \cdots 01 w_{0} 0 \cdots 0
$$

where the factor $1 w_{i+1}$ starts after the $w_{i}$-th letter and the factor $1 w_{0}$ starts after the $w_{k-1}$-th letter. Thus a word $w$ of length $n^{2}$ belongs to $\langle T\rangle$ if and only if in the sequence above there is a 1 at position $w$.

This encoding of $T$ has the advantage that for a given $w_{i}$, the successor $w_{i+1}$ can be computed by a log-space machine that has access to the oracle $\langle T\rangle$. For this, the machine just has to query the words $w_{i}+1, w_{i}+2, \ldots, w_{i}+n^{2}$ and has to interpret the vector of answers as the word $w_{i+1}$. This property results in the $\leqslant_{\mathrm{m}}^{\log }$-autoreducibility of $T$ and finally this will translate into the $\leqslant_{\mathrm{m}}^{\log _{\mathrm{m}}}$-autoreducibility of $L$.

Since log-space computable functions are closed under composition, for every constant $c>1$, there exists a log-space machine with oracle $\langle T\rangle$ that on input $w_{i}$ computes the $c$-th next word $w_{i+c}$. We show that for log-space machines this $c$-times composition of the successor function is expensive: No log-space machine can compute successors that are farther away than a constant. Hence, there exists a constant $c$ such that on input $w_{i}$ and with access to the oracle $\langle T\rangle$ the machines $M_{1}$ and $M_{2}$ can only gain knowledge about the words $w_{i}, w_{i+1}, \ldots, w_{i+c}$, but not about the words $w_{0}, w_{1}, \ldots, w_{i-1}$ and $w_{i+c+1}, w_{i+c+2}, \ldots, w_{k-1} .^{2}$ In particular, the machines will not notice if we change the oracle with respect to the latter words. Therefore, if we consider $M_{1}$ on input $w_{i}$ and with oracle $\langle T\rangle$, then this computation will not change if we replace the oracle by $\left\{w_{i}, w_{i+1}, \ldots, w_{i+c}\right\}$. In this sense, $M_{1}^{\langle T\rangle}\left(w_{i}\right)$ computes exactly the $(c+1)$-ary relation

$$
R\left(w_{i}, w_{i+1}, \ldots, w_{i+c}\right) \stackrel{d f}{=} M_{1}^{\left\langle\left\{w_{i}, w_{i+1}, \ldots, w_{i+c}\right\}\right\rangle}\left(w_{i}\right)
$$

We now apply Ramsey theory and obtain a set $T_{S}=\left\{w_{0}, \ldots, w_{3 c-1}\right\}$ such that $T_{S} \subseteq S$ and all words $w_{0}, \ldots, w_{2 c-1}$ are equivalent with respect to the relation, i.e., are all inside or all outside the relation. Note that this property of $T_{S}$ is very strong. It means that either

$$
w_{0}, \ldots, w_{2 c-1} \in L\left(M_{1}^{\left\langle T_{S}\right\rangle}\right)
$$

or

$$
w_{0}, \ldots, w_{2 c-1} \in \overline{L\left(M_{1}^{\left\langle T_{s}\right\rangle}\right)}
$$

We let $O_{s} \stackrel{d f}{=}\left\langle T_{s}\right\rangle$ which defines our oracle with respect to words of length $n^{2}$. This finishes stage $s$ of our construction. The final oracle $O$ is the union of all $O_{s}$.

We enter the second part, i.e., the construction of $L$. All remaining arguments are now relative to the oracle $O$. We have to construct $L$ such that it is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible, but not $\leqslant_{\mathrm{m}}^{\log }$-mitotic. On the one hand, $L$ will be the union of sets $T_{s}$ for certain $s$ which immediately results in $L$ ' $s \leqslant_{m}^{\log }$-autoreducibility. On the other hand, we diagonalize against all possible log-space-computable separators $S=L\left(M_{1}\right)$ and all log-space-computable functions $f(x)=M_{2}(x)$ that claim to reduce $L \cap S$ to $L \cap \bar{S}$. The latter destroys $\leqslant_{\mathrm{m}}^{\log }$-mitoticity.

We sketch the diagonalization argument. Let $s$ be the stage of 0 's construction in which we diagonalized against $M_{1}$ and $M_{2}$. In this stage we constructed the set $T_{s}=\left\{w_{0}, \ldots, w_{3 c-1}\right\}$. We already observed that $M_{2}\left(w_{0}\right)$ cannot gain knowledge about the words $w_{c+1}, w_{c+2}, \ldots, w_{3 c-1}$. If $M_{2}\left(w_{0}\right) \notin T_{s}$, then by putting $T_{s}$ inside or outside $L$, we can enforce that $f$ does not reduce $L$ to $L$. Otherwise, $M_{2}\left(w_{0}\right) \in T_{s}$ and hence $M_{2}\left(w_{0}\right) \in\left\{w_{0}, \ldots, w_{c}\right\}$, since it cannot gain knowledge about the other words. However, as seen above, either $w_{0}, \ldots, w_{2 c-1} \in L\left(M_{1}\right)$ or $w_{0}, \ldots, w_{2 c-1} \in \overline{L\left(M_{1}\right)}$. So in this case, $f$ does not reduce $S$ to $\bar{S}$. Therefore, in any case, $f$ does not reduce $L \cap S$ to $L \cap \bar{S}$. This shows that $L$ is not $\leqslant_{\mathrm{m}}^{\log }$-mitotic.

[^2]
### 4.3. The detailed construction

This section contains the detailed construction of an oracle relative to which $\leqslant_{\mathrm{m}}^{\log }$-autoreducibility and $\leqslant_{\mathrm{m}}^{\log }$-mitoticity are not equivalent.

Fix a universal Turing transducer $U$. For a finite set $S=\left\{w_{1}, \ldots, w_{m}\right\} \subseteq \Sigma^{*}$ where $w_{1}<\cdots<w_{m}$, let code( $S$ ) be the quasi-lexicographically smallest word $w \in \Sigma^{*}$ such that $U(w)$ outputs the string $w_{1} \# w_{2} \# \cdots \# w_{m}$ and stops. So $|\operatorname{code}(S)|$ is the Kolmogorov complexity of the set $S$.

Suppose we have to encode two nonempty words $w_{1}$ and $w_{2}$ into one word. For this it is not enough to just concatenate both words, since then it is not clear where $w_{1}$ ends. An easy way to mark the border between both words is to use the repetition code for $w_{1}$. More precisely, all bits of $w_{1}$ (except the last one) are stored twice such that 0 becomes 00 and 1 becomes 11. If the last bit is 0 , then this is encoded by 01 , otherwise this is encoded by 10 . In this way, the concatenation of the repetition code of $w_{1}$ and the normal code of $w_{2}$ completely describes both strings.

For our construction we need to consider sets $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$. We start with a proposition that gives an upper bound for the Kolmogorov complexity of such sets. Moreover, it guarantees the existence of sets having a high Kolmogorov complexity.

## Proposition 4.1.

1. There exists $c \geqslant 0$ such that for all $n \geqslant 1$ and all $S \subseteq \Sigma^{n^{2}}$,

$$
|\operatorname{code}(S)| \leqslant n^{2}|S|+2 \log n+c .
$$

2. There exists $c \geqslant 0$ such that for all $n \geqslant 1$ and all $S \subseteq \Sigma^{n^{2}}$ where $|S| \leqslant n$,

$$
|\operatorname{code}(S)| \leqslant n\left(n^{2}-\frac{\log n}{2}\right)+c
$$

3. For all $n \geqslant 4$ there exists $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$ and

$$
|\operatorname{code}(S)| \geqslant n\left(n^{2}-\log n\right)
$$

4. There exists $c \geqslant 0$ such that for all $n \geqslant 4$ there exists $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$ and

$$
n\left(n^{2}-\log n\right) \leqslant|\operatorname{code}(S)| \leqslant n\left(n^{2}-\frac{\log n}{2}\right)+c .
$$

Proof. For statement 1 , use $c$ bits for encoding a constant size decoding program (via repetition code), use the $2 \log n$ bits for encoding $n$ (via repetition code), and use $n^{2}|S|$ bits for the concatenation of all words in $S$.

For statement 2, first observe that for $n \geqslant 10$,

$$
n!\geqslant 4^{n / 2}\left(\frac{n}{2}\right)^{n / 2}=(2 n)^{n / 2}
$$

For $n \geqslant 10$, we estimate an upper bound for the number of sets $S \subseteq \Sigma^{n^{2}}$ such that $|S| \leqslant n$.

$$
\begin{aligned}
\sum_{i=0}^{n}\binom{2^{n^{2}}}{i} & \leqslant n\binom{2^{n^{2}}}{n}=n \cdot \frac{2^{n^{2}}\left(2^{n^{2}}-1\right) \cdots\left(2^{n^{2}}-n+1\right)}{n!} \\
& \leqslant \frac{n \cdot 2^{n^{3}}}{(2 n)^{n / 2}} \leqslant \frac{2^{n^{3}}}{n^{n / 2}}=2^{n\left(n^{2}-\frac{\log n}{2}\right)}
\end{aligned}
$$

So for $n \geqslant 10$, each $S \subseteq \Sigma^{n^{2}}$ where $|S| \leqslant n$ can be encoded by a constant size decoding program (via repetition code) followed by $n\left(n^{2}-\frac{\log n}{2}\right)$ bits. This shows statement 2 , since the inequality also holds for $n<10$, if we choose $c$ large enough.

For the third statement, we start with the following estimation for $n \geqslant 4$.

$$
\begin{align*}
& 2^{n^{2}} \geqslant 4 n \geqslant \frac{n}{1-2^{-1 / 2}} \\
& 2^{n^{2}}\left(1-2^{-1 / 2}\right) \geqslant n \\
& 2^{n^{2}}-n \geqslant 2^{n^{2}-(1 / 2)} \tag{5}
\end{align*}
$$

With help of (5), we can show the following lower bound for the number of $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$ where $n \geqslant 4$.

$$
\begin{aligned}
\binom{2^{n^{2}}}{n}=\frac{2^{n^{2}}\left(2^{n^{2}}-1\right) \cdots\left(2^{n^{2}}-n+1\right)}{1 \cdot 2 \cdots \cdot n} & \geqslant \frac{\left(2^{n^{2}}-n\right)^{n}}{\frac{1}{2} \cdot\left(\frac{n}{2}\right)^{n / 2} \cdot n^{n / 2}} \\
& \geqslant \frac{2^{n^{3}-(n / 2)}}{\frac{1}{2} \cdot n^{n} \cdot 2^{-n / 2}} \\
& =\frac{2^{n^{3}}}{2^{(n \log n)-1}}=2^{n\left(n^{2}-\log n\right)+1} .
\end{aligned}
$$

However, there exist less than $2^{n\left(n^{2}-\log n\right)+1}$ words over $\{0,1\}$ whose length is $\leqslant n\left(n^{2}-\log n\right)$. Therefore, for at least one $S$, $|\operatorname{code}(S)| \geqslant n\left(n^{2}-\log n\right)$. This shows the third statement.

Statement 4 is an immediate consequence of the statements 2 and 3.
We now show that if $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$ and $S$ has a high Kolmogorov complexity, then all subsets of $S$ have a high Kolmogorov complexity.

Proposition 4.2. The following holds for almost all $n$. Let $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n, S=\left\{w_{1}, \ldots, w_{n}\right\}$ where $w_{1}<\cdots<w_{n}$, and $|\operatorname{code}(S)| \geqslant n\left(n^{2}-\log n\right)$.

1. $\forall T \subseteq S,|\operatorname{code}(T)| \geqslant n^{2}\left(|T|-\frac{1}{2}\right)$.
2. $\forall i \in[2, n], w_{i}-w_{i-1}>3 n^{2}$.

Proof. For the first statement, assume there exist an $S$ as in the proposition and a $T \subseteq S$ such that $|\operatorname{code}(T)|<n^{2}\left(|T|-\frac{1}{2}\right)$. The set $S$ is completely described by the pair $(\operatorname{code}(T), \operatorname{code}(S-T))$. Hence

$$
|\operatorname{code}(S)| \leqslant O(\log |\operatorname{code}(T)|)+|\operatorname{code}(T)|+|\operatorname{code}(S-T)| .
$$

To see this, use the $O(\log |\operatorname{code}(T)|)$ bits to encode the constant size decoding program and the length of code( $(T)$ (both via repetition code). The latter allows us to separate the code $(T)$-part from the code $(S-T)$-part. By assumption, $\log |\operatorname{code}(T)|<$ $2 \log n+\log |T| \leqslant 3 \log n$. Moreover, by Proposition 4.1.1, $|\operatorname{code}(S-T)| \leqslant O(\log n)+n^{2}|S-T|$. So we obtain

$$
|\operatorname{code}(S)| \leqslant O(\log n)+n^{2}\left(|T|-\frac{1}{2}\right)+n^{2}|S-T|=n^{3}-\frac{n^{2}}{2}+O(\log n) .
$$

Hence, for sufficiently large $n,|\operatorname{code}(S)|<n\left(n^{2}-\log n\right)$ which contradicts our assumption. So the first statement holds for almost all $n$.

For the second statement, assume $w_{i}-w_{i-1} \leqslant 3 n^{2}$. So $T \stackrel{d f}{=}\left\{w_{i}, w_{i-1}\right\}$ is a subset of $S$ such that

$$
|\operatorname{code}(T)| \leqslant n^{2}+O(\log n),
$$

since we only have to encode a constant size decoding program (via repetition code), the number $n$ (via repetition code), the word $w_{i}$, and the number $w_{i}-w_{i-1}$. So for sufficiently large $n,|\operatorname{code}(T)|<n^{2}+\frac{n^{2}}{2}$. By the first statement, this is only possible for finitely many $n$.

We now encode a sequence of words of length $n^{2}$ into an oracle. The encoding will be such that for a given word, one can compute the next word in the sequence in log-space (where the computation has access to the oracle). This will result in the autoreducibility of a particular set. However, we will see that a log-space OTT can only make a constant number of such moves to the right. We will exploit this to show that the mentioned set is not mitotic.

We start with a definition that describes how to encode a sequence of words into an oracle.
Definition 4.3. Let $n, k \geqslant 1$ and $T \subseteq \Sigma^{n^{2}}$ such that $T=\left\{w_{0}, \ldots, w_{k-1}\right\}, w_{0}<\cdots<w_{k-1}$, and $w_{i}=a_{i, 1} \cdots a_{i, n^{2}}$ for $i \in[0, k)$ and $a_{i, j} \in \Sigma$. Moreover, for all $i \in[1, k), w_{i}-w_{i-1} \geqslant 3 n^{2}$.

$$
\langle T\rangle \stackrel{d f}{=} T \cup\left\{w_{i}+j \in \Sigma^{n} \mid i \in[0, k), j \in\left[1, n^{2}\right], a_{r, j}=1 \text { where } r=(i+1 \bmod k)\right\} .
$$

This definition can be visualized as follows. For $S \subseteq \Sigma^{n^{2}}$, let $c(S)$ denote the characteristic sequence that corresponds to the membership in $S$ for all words of length $n^{2}$, i.e.,

$$
c(S)=\chi_{S}(0 \cdots 00) \chi_{S}(0 \cdots 01) \chi_{S}(0 \cdots 10) \cdots \chi_{S}(1 \cdots 10) \chi_{S}(1 \cdots 11) .
$$

So $c(S) \in \Sigma^{2^{n^{2}}}$. Let $T=\left\{w_{0}, \ldots, w_{k-1}\right\}$ be as in Definition 4.3 and let $S=\langle T\rangle$. Observe that

$$
c(S) \in 0^{*} 1 w_{1} 0^{*} 1 w_{2} 0^{*} \cdots 0^{*} 1 w_{k-1} 0^{*} 1 w_{0} 0^{*}
$$

such that for $i \in[0, k)$, the factor $1 w_{i+1} \bmod k$ starts after the $w_{i}$-th letter of $c(S)$, i.e.,

$$
1 w_{i+1 \bmod k}=\chi_{S}\left(w_{i}\right) \chi_{S}\left(w_{i}+1\right) \cdots \chi_{S}\left(w_{i}+n^{2}\right)
$$

For fixed $n, k \geqslant 1$ and $T \subseteq \Sigma^{n^{2}}$ such that $|T| \leqslant n$ and $T=\left\{w_{0}, \ldots, w_{k-1}\right\}$ where $w_{0}<\cdots<w_{k-1}$ we use the abbreviation

$$
Q_{j} \stackrel{d f}{=}\left\{w_{j}, w_{j}+1, \ldots, w_{j}+n^{2}\right\}
$$

for $j \in[0, k)$. So querying the oracle for all words in $Q_{j}$ reveals the right neighbor of $w_{j}$, i.e., the word $w_{r}$ where $r=$ $(j+1 \bmod k)$.

Recall that we encode a sequence of words of length $n^{2}$ into an oracle such that, given a word from the sequence, one can determine the next word in the sequence in log-space. Note that log-space computable functions are closed under composition. So for every constant $k>1$, in log-space we can also compute the $k$-th next word. The following lemma shows that this $k$-times composition of the successor function is expensive if our sequence of words has high Kolmogorov complexity. Roughly speaking, if on input $w_{1}$ a log-space OTT queries for $w_{5}$, then, in before, it must have queried $n$ times for the predecessor $w_{4}$. By repeating this argument we obtain that the machine must have queried $n^{2}$ times for $w_{3}, n^{3}$ times for $w_{2}$, and $n^{4}$ times for $w_{1}$. This argument shows that for every log-space OTT $M$ there exists a $k$ (namely the constant $8 d$ in Lemma 4.5 ) such that $M$ cannot compute the $k$-th next word in the sequence. So $M$ only acts locally which in turn shows that $M$ cannot be used to establish mitoticity of a particular set.

Lemma 4.4. For every $\log$-space OTM $M$ with space bound $d \log n$ there exists $n_{0}$ such that the following holds for all $n \geqslant n_{0}$, all $S \subseteq \Sigma^{n^{2}}$, and all $T \subseteq S$ such that $|S|=n$, $\operatorname{code}(S) \geqslant n\left(n^{2}-\log n\right)$, and $T=\left\{w_{0}, \ldots, w_{k-1}\right\}$ where $k \geqslant 2$ and $w_{0}<\cdots<w_{k-1}$. Let $i, j \in[0, k)$ such that $j^{\prime} \stackrel{\text { df }}{=}(j+1 \bmod k) \neq i$ and consider $M^{\langle T\rangle}\left(w_{i}\right)$ between the steps $t_{1}$ and $t_{2}$ where $t_{1}<t_{2}$.

1. If a query tape $\tau$ is empty after step $t_{1}$ and if in step $t_{2}$, the machine uses $\tau$ to query a word $q \in Q_{j^{\prime}}$, then at least $n$ words in $Q_{j}$ are queried between the steps $t_{1}$ and $t_{2}$.
2. If more than $d \log n$ words $q \in Q_{j^{\prime}}$ are queried between the steps $t_{1}$ and $t_{2}$, then at least $n$ words in $Q_{j}$ are queried between these steps.

Proof. If statement 1 does not hold, then for infinitely many $n$, there exist $S, T, i, j, t_{1}, t_{2}$, and $\tau$ such that (i) $\tau$ is empty after step $t_{1}$, (ii) in step $t_{2}$, the machine uses $\tau$ to query a word $q \in Q_{j^{\prime}}$, and (iii) less than $n$ words in $Q_{j}$ are queried between the steps $t_{1}$ and $t_{2}$. If for fixed $n, S, T, i$, and $j$ there is more than one possibility for $t_{1}, t_{2}, \tau$, and $q$ to satisfy (i)-(iii), then we choose the one where $t_{2}$ is minimal. We show that our assumption implies that $\operatorname{code}(T)$ is too short which is a contradiction.

We encode the configuration of $M^{\langle T\rangle}\left(w_{i}\right)$ after step $t_{1}$ by the following string: $O(1)$ bits for the machine state (repetition code), $O(\log n)$ bits for the head position on the input tape (repetition code), $O(\log n)$ bits for the working tape and the index tape including the head positions (repetition code), and 1 bit for each query tape (repetition code). This latter bit describes the answer that is given by the oracle when the machine queries the corresponding tape next time. Hence the configuration of $M^{\langle T\rangle}\left(w_{i}\right)$ after step $t_{1}$ is described by a string $z$ of length $O(\log n)$. By assumption, the machine queries at most $n$ words $q \in Q_{j}$. So the corresponding answers can be described by a string $y \in \Sigma^{n}$. Let $x_{1}$ be the binary representation of $t_{2}-t_{1}$ (in repetition code) and let $x_{2}$ be the binary representation of $q-w_{j^{\prime}}$ (in repetition code). Note that $\left|x_{1}\right|=$ $O(d \log n)$, since otherwise the computation would have run into a loop. Also, $x_{2} \leqslant n^{2}$ and hence $\left|x_{2}\right|=O(\log n)$. Finally, let $v$ be a string of length $O(\log n)$ that consists of the constant-size listing of the decoding algorithm $\mathcal{A}$ described below followed by the binary representations of $n, i$, and $j$ (all in repetition code). Note that $i, j<n$. We claim that for sufficiently large $n$, the string

$$
u \stackrel{d f}{=} v w_{0} \cdots w_{j} w_{j+2} \cdots w_{k-1} x_{1} x_{2} y z
$$

is a code for $T$, i.e., $U(u)$ outputs the string $w_{0} \# w_{1} \# \cdots \# w_{k-1}$ and stops.
To see this, the decoding algorithm $\mathcal{A}$ uses $v$ to obtain $j$ and it uses $u$ to construct the set $T^{\prime}=T-\left\{w_{j^{\prime}}\right\}$. Then $\mathcal{A}$ uses $v$ to obtain $i$ and hence $w_{i}$ (note that $w_{i} \in T^{\prime}$, since $i \neq j^{\prime}$ ). With help of $z$, the algorithm reconstructs the configuration of $M^{\langle T\rangle}\left(w_{i}\right)$ after step $t_{1}$. The query tapes cannot be reconstructed, but thanks to $z$, for each tape we know the answer of the next query made by this tape. Now $\mathcal{A}$ starts with the reconstructed configuration and simulates $M$ 's computation for $x_{1}$ steps. During this simulation, the first query made by a tape is answered according to $z$, oracle queries in $Q_{j}$ are answered according to $y$, and all remaining oracle queries are answered according to $\left\langle T^{\prime}\right\rangle$. Observe that the sets $\langle T\rangle$ and $\left\langle T^{\prime}\right\rangle$ differ only with respect to words in $Q_{j} \cup Q_{j^{\prime}}$. The following factorization visualizes the situation if $j+1<k$, i.e., $j^{\prime}=j+1$.

$$
\begin{aligned}
& c(\langle T\rangle)=0 \cdots 01 w_{1} 0 \cdots 01 w_{j} 0 \cdots 01 w_{j+1} 0 \cdots 01 w_{j+2} 0 \cdots 01 w_{j+3} 0 \cdots 0 w_{k-1} 0 \cdots 0 w_{0} 0 \cdots 0 \\
& c\left(\left\langle T^{\prime}\right\rangle\right)=0 \cdots 01 w_{1} 0 \cdots 01 w_{j} 0 \cdots 01 w_{j+2} 0 \cdots 00 \cdots 00 \cdots 01 w_{j+3} 0 \cdots 0 w_{k-1} 0 \cdots 0 w_{0} 0 \cdots 0 \\
& \uparrow \quad \uparrow \quad \uparrow \\
& \text { position: } w_{j} \quad w_{j+1} \quad w_{j+2}
\end{aligned}
$$

$\mathcal{A}$ makes sure that queries in $Q_{j}$ are answered correctly. So it remains to argue for queries $q^{\prime} \in Q_{j^{\prime}}$. Let $\tau^{\prime}$ be the query tape that is used for querying $q^{\prime}$, and let $t^{\prime} \in\left[t_{1}, t_{2}\right)$ be the step in which $\tau^{\prime}$ queries $q^{\prime}$. If $\tau^{\prime}$ queries the first time in our simulation, then $\mathcal{A}$ gives the right answer, since it uses $z$ to answer the first query made by a tape. Otherwise, $\tau^{\prime}$ was already used for querying earlier in our simulation. So after the first query, $\tau^{\prime}$ became empty and after the $t^{\prime}$-th step, the machine queries $q^{\prime}$ which is written on $\tau^{\prime}$. This contradicts our choice of $t_{1}, t_{2}, \tau$, and $q$ such that $t_{2}$ is minimal. Therefore, in our simulation, queries $q^{\prime} \in Q_{j^{\prime}}$ only appear as a first query made by a query tape and therefore, these queries are answered correctly. It follows that $\mathcal{A}$ correctly simulates the work of $M^{\langle T\rangle}\left(w_{i}\right)$ even though it does not know the word $w_{j+1}$.

After the simulation, $\mathcal{A}$ finds the word $q \in Q_{j^{\prime}}$ written on the tape $\tau$ which is the tape that was queried in the last simulation step. So $\mathcal{A}$ can reconstruct the missing word $w_{j^{\prime}}=q-x_{2}$. Now $\mathcal{A}$ has complete knowledge about $T$ and can output $w_{0} \# w_{1} \# \cdots \# w_{k-1}$. This shows that $u$ is a code for $T$, since the listing of $\mathcal{A}$ is encoded in a prefix of $u$, and since $U$ is a universal Turing transducer.

Therefore, for sufficiently large $n$,

$$
|\operatorname{code}(T)| \leqslant|u|=O(\log n)+n+n^{2}(k-1)<n^{2}\left(k-\frac{1}{2}\right)=n^{2}\left(|T|-\frac{1}{2}\right)
$$

From Proposition 4.2.1 it follows that this is only possible if $n$ is bounded by a constant. This contradicts our assumption that we can choose $n$ arbitrarily large. This shows statement 1 of the lemma.

Statement 2 follows from statement 1: If more than $d \log n$ words $q \in Q_{j^{\prime}}$ are queried between the steps $t_{1}$ and $t_{2}$, then the number of queried words in $Q_{j^{\prime}}$ is greater than the number of query tapes and hence, one tape $\tau$ is used at least twice to query a word from $Q_{j^{\prime}}$. In particular, there must exist $t_{1}^{\prime}, t_{2}^{\prime} \in\left[t_{1}, t_{2}\right]$ where $t_{1}^{\prime}<t_{2}^{\prime}$ such that after step $t_{1}^{\prime}$ the tape $\tau$ is empty and in step $t_{2}^{\prime}$, the machine uses $\tau$ to query a word $q \in Q_{j^{\prime}}$. This proves statement 2 and finishes the proof of Lemma 4.4.

We use the observation made in Lemma 4.4 and show that if we use a sequence of words with high Kolmogorov complexity as an oracle, then a log-space OTM can only access a very small part of the sequence. So the computation will not notice changes that are made outside this part.

Lemma 4.5. For every log-space OTM $M$ with space bound $d \cdot \log n$ where $d>1$ there exists an $n_{0}$ such that the following holds for all $n \geqslant n_{0}$. If $T \subseteq S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$, $\operatorname{code}(S) \geqslant n\left(n^{2}-\log n\right)$, and $T=\left\{w_{0}, \ldots, w_{k+8 d-1}\right\}$ where $k \geqslant 1$ and $w_{0}<\cdots<$ $w_{k+8 d-1}$, then

$$
\forall i \in[0, k), \quad M^{\langle T\rangle}\left(w_{i}\right)=M^{\left\langle\left\{w_{i}, w_{i+1}, \ldots, w_{i+8 d}\right\}\right\rangle}\left(w_{i}\right) .
$$

Proof. Choose $n_{0}$ as the maximum of $2^{d}$ and the constant assured by Lemma 4.4. Let $n \geqslant n_{0}$ and $T \subseteq S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$, $\operatorname{code}(S) \geqslant n\left(n^{2}-\log n\right)$, and $T=\left\{w_{0}, \ldots, w_{k+8 d-1}\right\}$ where $k \geqslant 1$ and $w_{0}<\cdots<w_{k+8 d-1}$. Moreover, fix some $i \in[0, k)$. We have to prove

$$
\begin{equation*}
M^{\langle T\rangle}\left(w_{i}\right)=M^{\left\langle\left\{w_{i}, w_{i+1}, \ldots, w_{i+8 d}\right\}\right\rangle}\left(w_{i}\right) . \tag{6}
\end{equation*}
$$

Claim 4.6. Let $l \geqslant 2\lceil 1+d \log n\rceil, s \in[0, k+8 d)$, and $r \stackrel{d f}{=}(s-i \bmod k+8 d)$. If $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least $l$ words from $Q_{s}$, then it queries at least $\ln ^{r / 2}$ words from $Q_{i}$.

Proof. The proof is by induction on $r=0, \ldots, k+8 d-1$. If $r=0$, then $i=s$. So if $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least $l$ words from $Q_{s}$, then these are at least $l^{0}$ words from $Q_{i}$ which shows the induction base.

Now let $r \geqslant 1$ and hence $i \neq s$. Assume that $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least $l$ words from $Q_{s}$. If we define $j=(s-1 \bmod k+8 d)$ and $j^{\prime}=s$, then our assumption says that $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least $l$ words from $Q_{j^{\prime}}$. So the computation can be partitioned into $l^{\prime}=\lfloor l /\lceil 1+d \log n\rceil\rfloor \geqslant 2$ intervals such that each interval queries for more than $d \log n$ words in $Q_{j^{\prime}}$. We can apply Lemma 4.4.2, since $i \neq j^{\prime}$. Therefore, each interval queries at least $n$ words in $Q_{j}$. So the whole computation $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least

$$
l^{\prime} n \geqslant\left(\frac{l}{\lceil 1+d \log n\rceil}-1\right) n \geqslant \frac{l n}{2\lceil 1+d \log n\rceil}>l \sqrt{n}
$$

words in $Q_{j}$ (the last estimation holds, since $d \leqslant \log n$ ). Note that $(j-i \bmod k+8 d)=r-1$. So from the induction hypothesis it follows that $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least

$$
(l \sqrt{n}) n^{(r-1) / 2}=\ln ^{r / 2}
$$

words from $Q_{i}$. This finishes the induction step and proves Claim 4.6.
Claim 4.7. For $s \in[0, i) \cup[i+8 d, k+8 d), M^{\langle T\rangle}\left(w_{i}\right)$ does not query for words in $Q_{s}$.
Proof. Assume there exists $s \in[0, i) \cup[i+8 d, k+8 d)$ such that $M^{\langle T\rangle}\left(w_{i}\right)$ queries for some $q \in Q_{s}$. If we define $j=$ $(s-1 \bmod k+8 d)$ and $j^{\prime}=s$, then our assumption says that $M^{\langle T\rangle}\left(w_{i}\right)$ queries for some $q \in Q_{j^{\prime}}$. We can apply Lemma 4.4.1, since $i \neq j^{\prime}$. So $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least $n$ words in $Q_{j}$. Let $r \stackrel{d f}{=}(j-i \bmod k+8 d)$ and observe that $r \geqslant 8 d-1$ (since $j \in[0, i-1) \cup[i+8 d-1, k+8 d))$. From Claim 4.6 it follows that $M^{\langle T\rangle}\left(w_{i}\right)$ queries at least $n \cdot n^{r / 2}>n^{4 d}$ words from $Q_{i}$. However, the latter is impossible, since $\left|w_{i}\right|=n^{2}$ and so the computation $M^{\langle T\rangle}\left(w_{i}\right)$ uses only $\log n^{2 d}$ bits of working space which implies a time bound of $n^{2} \cdot n^{2 d}<n^{4 d}$. This proves Claim 4.7.

Observe (as in the proof of Lemma 4.4) that $\langle T\rangle$ and $\left\langle\left\{w_{i}, w_{i+1}, \ldots, w_{i+8 d}\right\}\right\rangle$ differ only with respect to words in

$$
Q=Q_{0} \cup Q_{1} \cup \cdots \cup Q_{i-1} \cup Q_{i+8 d} \cup Q_{i+8 d+1} \cup \cdots \cup Q_{k+8 d-1}
$$

By Claim 4.7, $M^{\langle T\rangle}\left(w_{i}\right)$ does not query for words in $Q$. This shows Eq. (6) and finishes the proof of Lemma 4.5.

We now transfer Lemma 4.5 from OTMs to OTTs: If we use a sequence of words with high Kolmogorov complexity as an oracle, then a log-space OTT can only compute a very small part of this sequence.

Corollary 4.8. For every log-space OTT $M$ with space bound $d \cdot \log n$ for $d>1$ there exists an $n_{0}$ such that the following holds for all $n \geqslant n_{0}$. If $T \subseteq S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$, $\operatorname{code}(S) \geqslant n\left(n^{2}-\log n\right)$, and $T=\left\{w_{0}, \ldots, w_{k+7+8 d}\right\}$ where $k \geqslant 1$ and $w_{0}<\cdots<$ $w_{k+7+8 d}$, then

$$
\forall i \in[0, k), \quad M^{\langle T\rangle}\left(w_{i}\right) \notin\left\{w_{0}, \ldots, w_{i-1}, w_{i+8 d+8}, \ldots, w_{k+7+8 d}\right\} .
$$

Proof. Let $M$ and $d$ be as above. Let $M^{\prime}$ be the modification of $M$ that on inputs of length $n$ uses $(d \log n)+1$ query tapes such that the output is written to the last query tape and at the end of the computation, the machine queries the word on the last tape. Note that $M^{\prime}$ is a $\log$-space OTM with space bound $(d+1) \log n$. Now consider the proof of Lemma 4.5 applied to $M^{\prime}$. First, in that proof, we define a certain constant $n_{0}$. Then we choose arbitrary $n \geqslant n_{0}$ and $T \subseteq S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$, $\operatorname{code}(S) \geqslant n\left(n^{2}-\log n\right)$, and $T=\left\{w_{0}, \ldots, w_{k+8(d+1)-1}\right\}$ where $k \geqslant 1$ and $w_{0}<\cdots<w_{k+8(d+1)-1}$. Finally, in Claim 4.7 we show that for all $i \in[0, k)$ and all $s \in[0, i) \cup[i+8(d+1), k+8(d+1))$ it holds that $M^{\prime\langle T\rangle}\left(w_{i}\right)$ does not query for words in $Q_{s}$. In particular, for these $i$ and $s, M^{\prime\langle T\rangle}\left(w_{i}\right)$ does not query for $w_{s}$ and therefore, $M^{\langle T\rangle}\left(w_{i}\right) \neq w_{s}$. We obtain

$$
\forall i \in[0, k), \quad M^{\langle T\rangle}\left(w_{i}\right) \notin\left\{w_{0}, \ldots, w_{i-1}, w_{i+8(d+1)}, \ldots, w_{k+8(d+1)-1}\right\} .
$$

This proves the corollary.

So far we learned that sequences of words of high Kolmogorov complexity allow us to set up relativized worlds in such a way that log-space machines cannot learn more than a constant-size part of the original sequence. Hence, the behavior of a log-space OTM $M$ on input of some word $w_{i}$ on the list depends only on a constant number of successors $w_{i+1}, \ldots, w_{i+c}$ on the list. In this sense, $M$ on $w_{i}$ computes nothing more than a $(c+1)$-ary relation on words, i.e., $M\left(w_{i}\right)=R\left(w_{i}, w_{i+1}, \ldots, w_{i+c}\right)$. By the Ramsey theorem, if we fix an arity $d$ for relations and if we choose a sequence of words that is large enough, then for every $d$-ary relation on words we will find a subsequence all of its words are equivalent with respect to the relation (i.e., are all inside or all outside the relation). In our proof we will use the following formulation of Ramsey's theorem.

Theorem 4.9. (See [12].) For all d, $a_{0}, a_{1} \geqslant 1$ there exists a minimal natural number $R^{(d)}\left(a_{0}, a_{1}\right)$ (the generalized Ramsey number for two colors) such that no matter how each d-element subset of an $R^{(d)}\left(a_{0}, a_{1}\right)$-element set $S$ is colored with 0 and 1 , there exists $b \in\{0,1\}$ and $T \subseteq S$ such that $|T| \geqslant a_{b}$ and all d-element subsets of $T$ have color $b$.

We have argued that log-space machines must act very locally on sequences of words of high Kolmogorov complexity. As a consequence of this and the Ramsey theorem we can now show that one can always find a sequence that (if it is used as oracle) simultaneously achieves two things:

1. Unbalance: A given log-space OTM $M_{1}$ (intuitively, a possible separator) cannot separate the sequence in a balanced way. There will always be large parts in the sequence where $M_{1}$ either always accepts or always rejects.
2. Locality: A given log-space OTT $M_{2}$ (intuitively, a possible reduction establishing mitoticity) cannot map to words that are in the sequence, but far from the input word.

These statements say that we can successfully diagonalize against every pair ( $M_{1}, M_{2}$ ) where $M_{1}$ is a separator and $M_{2}$ is a reduction that possibly establish mitoticity of our witness language.

Lemma 4.10. Let $M_{1}$ be a log-space OTM and let $M_{2}$ be a log-space OTT such that both machines have space bound $d \log n$ for $d>1$. For $e \geqslant 8(d+1)$ and for all sufficiently large $n$ there exists $T=\left\{w_{0}, \ldots, w_{3 e-1}\right\} \subseteq \Sigma^{n^{2}}$ such that the following holds:

1. $w_{0}<\cdots<w_{3 e-1}$ and $\forall i \in[1,3 e), w_{i}-w_{i-1} \geqslant 3 n^{2}$;
2. either $\left\{w_{0}, \ldots, w_{2 e-1}\right\} \subseteq L\left(M_{1}^{\langle T\rangle}\right)$ or $\left\{w_{0}, \ldots, w_{2 e-1}\right\} \subseteq \overline{L\left(M_{1}^{\langle T\rangle}\right)}$;
3. $\forall i \in[0, e], M_{2}^{\langle T\rangle}\left(w_{i}\right) \notin\left\{w_{2 e}, \ldots, w_{3 e-1}\right\}$.

Proof. Let $n_{1,0}$ be the constant $n_{0}$ that arises if we apply Lemma 4.5 to $M_{1}$ and let $n_{2,0}$ be the constant $n_{0}$ that arises if we apply Corollary 4.8 to $M_{2}$. Moreover, let $n$ be large enough such that $n \geqslant \max \left\{3 e, n_{1,0}, n_{2,0}, R^{(8 d+1)}(3 e, 3 e)\right\}$. By Proposition 4.1.3, there exists $S \subseteq \Sigma^{n^{2}}$ such that $|S|=n$ and $\operatorname{code}(S) \geqslant n\left(n^{2}-\log n\right)$. From Lemma 4.5 it follows that if $Q=\left\{u_{0}, \ldots, u_{3 e-1}\right\} \subseteq S$ where $u_{0}<\cdots<u_{3 e-1}$, then

$$
\begin{equation*}
\forall i \in[0,2 e), \quad M_{1}^{\langle Q\rangle}\left(u_{i}\right)=M_{1}^{\left\langle\left\{u_{i}, u_{i+1}, \ldots, u_{i+8 d}\right\}\right\rangle}\left(u_{i}\right) \tag{7}
\end{equation*}
$$

$M_{1}$ induces the following relation on $(8 d+1)$-element subsets of $S$.

$$
R \stackrel{d f}{=}\left\{V \mid V=\left\{v_{0}, \ldots, v_{8 d}\right\} \subseteq S, v_{0}<\cdots<v_{8 d}, \text { and } M_{1}^{\langle V\rangle}\left(v_{0}\right) \text { accepts }\right\}
$$

Note that $\chi_{R}$ (i.e., the characteristic function of $R$ ) induces a $0-1$-coloring of all ( $8 d+1$ )-element subsets of $S$. Also note that $|S| \geqslant R^{(8 d+1)}(3 e, 3 e)$. By Theorem 4.9, there exists $T \subseteq S$ of cardinality $3 e$ and $b \in\{0,1\}$ such that

$$
\begin{equation*}
\forall V \subseteq T \quad\left(|V|=8 d+1 \Rightarrow \chi_{R}(V)=b\right) \tag{8}
\end{equation*}
$$

Choose words $w_{0}<\cdots<w_{3 e-1}$ such that $T=\left\{w_{0}, \ldots, w_{3 e-1}\right\}$. So from (7) we obtain

$$
\begin{equation*}
\forall i \in[0,2 e), \quad M_{1}^{\langle T\rangle}\left(w_{i}\right)=M_{1}^{\left\langle\left\langle w_{i}, w_{i+1}, \ldots, w_{i+8 d}\right\}\right\rangle}\left(w_{i}\right) . \tag{9}
\end{equation*}
$$

Statement 1 of the lemma is an immediate consequence of Proposition 4.2. Moreover, (8) and (9) imply

$$
\begin{equation*}
\forall i \in[0,2 e), \quad M_{1}^{\langle T\rangle}\left(w_{i}\right)=M_{1}^{\left\langle\left\{w_{i}, w_{i+1}, \ldots, w_{i+8 d}\right\}\right\rangle}\left(w_{i}\right)=b . \tag{10}
\end{equation*}
$$

This shows statement 2 of the lemma.
We turn to statement 3 . If $k \stackrel{d f}{=} 3 e-8 d-8$, then $T=\left\{w_{0}, \ldots, w_{k+7+8 d}\right\}$. From Corollary 4.8 we obtain

$$
\forall i \in[0, k), \quad M_{2}^{\langle T\rangle}\left(w_{i}\right) \notin\left\{w_{0}, \ldots, w_{i-1}, w_{i+8 d+8}, \ldots, w_{k+7+8 d}\right\}
$$

In particular,

$$
\forall i \in[0, e], \quad M_{2}^{\langle T\rangle}\left(w_{i}\right) \notin\left\{w_{2 e}, \ldots, w_{3 e-1}\right\}
$$

This proves the statement 3.

We now apply the argument given in Lemma 4.10 in an oracle construction. More precisely, we construct the oracle such that no log-space machine (considered as a separator) can act in a balanced way and no log-space transducer (considered as a reduction) can act in a non-local way. Relative to this oracle, we then construct a language $L$ by diagonalizing against all possible separators and all possible reduction functions that might witness the $\leqslant_{\mathrm{m}}^{\log }$-mitoticity of $L$. In addition, the construction of $L$ is such that $L$ is $\leqslant_{\mathrm{m}}^{\log }$-autoreducible.

Theorem 4.11. There exists an oracle 0 relative to which $\leqslant_{\mathrm{m}}^{\log }$-autoreducibility does not imply $\leqslant_{\mathrm{m}}^{\log }$-mitoticity (i.e., there is an $L$ that is $\leqslant_{\mathrm{m}}^{\log , O}$-autoreducible, but not $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic).

Proof. We use a stagewise construction such that at stage $s$, the oracle is constructed up to words of length $n_{s}^{2}$ where $n_{0}<n_{1}<\cdots$ (the numbers $n_{s}$ will be chosen in the construction). At stage $s$ we choose a $T_{s} \subseteq \Sigma^{n_{s}^{2}}$ such that $\left|T_{s}\right| \in\left[2, n_{s}\right]$. Then we let $O_{s}=\left\langle T_{s}\right\rangle$ and finally we define $0=\bigcup_{s \in \mathbb{N}} O_{s}$.

After the construction of $O$, we will choose a suitable $I \subseteq \mathbb{N}$ and we will show that

$$
L \stackrel{d f}{=} \bigcup_{s \in I} T_{S}
$$

is $\leqslant_{\mathrm{m}}^{\log , O}$-autoreducible, but not $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic. The detailed description of the construction follows.
Let $p_{0}, p_{1}, \ldots$ be a list of all pairs $\left(N_{1}, N_{2}\right)$ such that $N_{1}$ is a log-space OTM and $N_{2}$ is a log-space OTT. At stage $s$ we choose $T_{s}$ and $O_{s}=\left\langle T_{s}\right\rangle$ in such a way that relative to the completed oracle $O$ we can successfully diagonalize against the pair $p_{s}=\left(N_{1}, N_{2}\right)$ in the following sense. We interpret $N_{1}$ as a machine for a separator $S$ of $L$, and we interpret $N_{2}$ as a log-space many-one reduction $f$ that reduces $L \cap S$ to $L \cap \bar{S}$. A successful diagonalization means that by putting $s$ inside or outside $I$ we can enforce that $L \cap S$ does not $\leqslant_{\mathrm{m}}^{\log , O}$-reduce to $L \cap \bar{S}$ via reduction function $f$.

Stage $s$ : Assume $p_{s}=\left(N_{1}, N_{2}\right)$. Let $M_{1}$ (resp., $M_{2}$ ) be the modification of $N_{1}$ (resp., $N_{2}$ ) that does not query for words of length $\leqslant n_{s-1}^{2}$, but answers such queries according to the hardwired set $O_{0} \cup \cdots \cup Q_{s-1}$. Choose $d>0$ such that $M_{1}$ and $M_{2}$ have the space bound $d \log n$. Let $t$ be the maximum running time of all computations that have been considered in the construction so far. Choose $n_{s}$ large enough such that $n_{s}>\max \left(n_{s-1}, t\right)$ and that Lemma 4.10 can be applied to $M_{1}$ and $M_{2}$ for $e \stackrel{d f}{=} 8(d+1)$ and $n \stackrel{d f}{=} n_{s}$. From Lemma 4.10 we obtain $T_{s}=\left\{w_{0}, \ldots, w_{3 e-1}\right\} \subseteq \Sigma^{n_{s}^{2}}$ such that:

- $w_{0}<\cdots<w_{3 e-1}$ and $\forall i \in[1,3 e), w_{i}-w_{i-1} \geqslant 3 n_{s}^{2}$;
- either $\left\{w_{0}, \ldots, w_{2 e-1}\right\} \subseteq L\left(M_{1}^{\left\langle T_{s}\right\rangle}\right)$ or $\left\{w_{0}, \ldots, w_{2 e-1}\right\} \subseteq \overline{L\left(M_{1}^{\left\langle T_{s}\right\rangle}\right)}$;
- $\forall i \in[0, e], M_{2}^{\left\langle T_{s}\right\rangle}\left(w_{i}\right) \notin\left\{w_{2 e}, \ldots, w_{3 e-1}\right\}$.

Let $O_{s}=\left\langle T_{s}\right\rangle$. This finishes the stage $s$.

$$
O \stackrel{d f}{=} \bigcup_{s \in \mathbb{N}} O_{s}
$$

Let $s \geqslant 1, p_{s}=\left(N_{1}, N_{2}\right)$, and $T_{s}=\left\{w_{0}, \ldots, w_{3 e-1}\right\}$ be as above. In the oracle construction, $n_{s}$ was chosen such that $n_{s}^{2}>n_{s-1}^{2}$. Therefore, $M_{1}^{\left\langle T_{s}\right\rangle}\left(w_{i}\right)$ accepts if and only if $N_{1}^{O_{0} \cup \ldots \cup O_{s}}\left(w_{i}\right)$ accepts. Similarly, $M_{2}^{\left\langle T_{s}\right\rangle}\left(w_{i}\right)=N_{2}^{O_{0} \cup \ldots \cup O_{s}}\left(w_{i}\right)$. Moreover, $n_{s+1}$ is chosen large enough such that changing the oracle with respect to words of lengths $\geqslant n_{s+1}$ will not affect the computations $N_{1}^{O}\left(w_{i}\right)$ and $N_{2}^{O}\left(w_{i}\right)$. Hence $M_{1}^{\left\langle T_{s}\right\rangle}\left(w_{i}\right)$ accepts if and only if $N_{1}^{O}\left(w_{i}\right)$ accepts. Also, $M_{2}^{\left\langle T_{s}\right\rangle}\left(w_{i}\right)=N_{2}^{O}\left(w_{i}\right)$. Together with (12) and (13) we obtain

$$
\begin{equation*}
\text { if } N_{2}^{O}\left(w_{0}\right) \in T_{s}, \quad \text { then }\left(w_{0} \in L\left(N_{1}^{O}\right) \Leftrightarrow N_{2}^{O}\left(w_{0}\right) \in L\left(N_{1}^{O}\right)\right) \tag{14}
\end{equation*}
$$

We now describe the choice of a suitable $I \subseteq \mathbb{N}$ such that the set

$$
L_{I} \stackrel{d f}{=} \bigcup_{s \in I} T_{S}
$$

is $\leqslant_{\mathrm{m}}^{\log , O}$-autoreducible, but not $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic. The index set $I$ is constructed in stages such that at stage $s$, we determine whether or not $s$ belongs to $I$. Assume we are at stage $s$ such that $p_{s}=\left(N_{1}, N_{2}\right)$ and $T_{s}=\left\{w_{0}, \ldots, w_{3 e-1}\right\}$ as above. Let $v_{s} \stackrel{d f}{=} w_{0}$ be the quasi-lexicographically minimal word in $T_{s}$.

Assume $N_{2}^{O}\left(w_{0}\right) \notin T_{s}$. In this case we put $s$ to $I$ if and only if for all $s^{\prime}<s$ where $s^{\prime} \in I$ it holds that $N_{2}^{O}\left(w_{0}\right) \notin T_{s}$. This makes sure that $w_{0} \in L_{I} \Leftrightarrow N_{2}^{O}\left(w_{0}\right) \notin L_{I}$. (Note that if $N_{2}^{O}\left(w_{0}\right) \in L_{I}$, then $N_{2}^{O}\left(w_{0}\right) \in T_{0} \cup \cdots \cup T_{s}$, since by the construction, $n_{s+1}$ is large enough such that $N_{2}^{O}\left(w_{0}\right)$ cannot belong to $T_{s+1}$.)

Now let us assume $N_{2}^{O}\left(w_{0}\right) \in T_{s}$. In this case we put $s$ to $I$. From (14) we obtain $w_{0} \in L\left(N_{1}^{O}\right) \Leftrightarrow N_{2}^{O}\left(w_{0}\right) \in L\left(N_{1}^{O}\right)$.
Our construction ensures the following, no matter whether or not $N_{2}^{O}\left(v_{s}\right)$ belongs to $T_{s}$.

$$
\begin{equation*}
\left(v_{s} \in L_{I} \Leftrightarrow N_{2}^{O}\left(v_{s}\right) \notin L_{I}\right) \vee\left(v_{s} \in L\left(N_{1}^{O}\right) \Leftrightarrow N_{2}^{O}\left(v_{s}\right) \in L\left(N_{1}^{O}\right)\right) . \tag{15}
\end{equation*}
$$

From (15) we obtain the non-mitoticity of $L \stackrel{d f}{=} L_{I}$ as follows: If $L$ is $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic, then there exists a separator $S \in \mathrm{~L}^{0}$ and a reduction $f \in \mathrm{FL}^{O}$ such that $L \cap S \leqslant \leqslant_{\mathrm{m}}^{\log , O} L \cap \bar{S}$ via $f$. Let $N_{1}$ be a log-space OTM and let $N_{2}$ be a log-space OTT such that $S=L\left(N_{1}^{O}\right)$ and $f(x)=N_{2}^{O}(x)$. Choose $s$ such that $p_{s}=\left(N_{1}, N_{2}\right)$. By (15), $\left(v_{s} \in L \Leftrightarrow f\left(v_{s}\right) \notin L\right)$ or $\left(v_{s} \in S \Leftrightarrow f\left(v_{s}\right) \in S\right)$. This contradicts the assumption that $L \cap S \leqslant_{\mathrm{m}}^{\log , O} L \cap \bar{S}$ via $f$. Therefore, $L$ is not $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic.

It remains to show that $L$ is $\leqslant_{\mathrm{m}}^{\log , O}$-autoreducible which actually holds for all $L_{J}$ where $J \subseteq \mathbb{N}$.

Claim 4.12. $L^{\prime} \stackrel{d f}{=} \bigcup_{s \in \mathbb{N}} T_{s}$ belongs to $\mathrm{L}^{O}$. More precisely, $L^{\prime}$ can be decided by a log-space OTM with only one query tape.

Proof. By the choice of the $T_{s}$, all neighboring words in $T_{s}$ have a distance $w_{i}-w_{i-1} \geqslant 3 n_{s}^{2}$. After Definition 4.3 we already observed that if $T_{s}=\left\{w_{0}, \ldots, w_{k-1}\right\} \subseteq \Sigma^{n_{s}^{2}}$, then in the characteristic sequence of $\left\langle T_{s}\right\rangle$, the word $1 w_{i+1} \bmod k$ starts at position $w_{i}$. So the distance between the start of $1 w_{i}$ and the start of $1 w_{i+1}$ is at least $3 n_{s}^{2}$. From $\left|1 w_{i}\right|=n_{s}^{2}+1$ it follows that the distance between the end of $1 w_{i}$ and the start of $1 w_{i+1}$ is at least $2 n_{s}^{2}-1$. Therefore, in the characteristic sequence of $\left\langle T_{s}\right\rangle$, there are at least $2 n_{s}^{2}-2$ letters 0 between the words $1 w_{i}$ and $1 w_{i+1}$. So a word $w$ belongs to $T_{s}$ if and only if $w \in\left\langle T_{s}\right\rangle$ and for all $i \in\left[1,2 n_{s}^{2}-2\right]$ it holds that $w-i \notin\left\langle T_{s}\right\rangle$. It follows that a word $w$ belongs to $L^{\prime}$ if and only if $w \in O$ and for all $i \in[1,2|w|-2]$ it holds that $w-i \notin O$. This shows $L^{\prime} \in \mathrm{L}^{0}$. Note that the latter condition can be tested in log-space with one query tape.

Claim 4.13. $L$ is $\leqslant_{\mathrm{m}}^{\log , O}$-autoreducible. More precisely, $L$ is $\leqslant_{\mathrm{m}}^{\log -\operatorname{lin}, O}$-autoreducible by a reduction that can be computed by a logspace OTT with only one query tape.

Proof. We describe $\mathrm{a} \leqslant_{\mathrm{m}}^{\log , O}$-autoreduction $f$ for $L=L_{I}$. Let $w$ be the input. If $w \notin L^{\prime}$, then $f$ outputs a fixed word that is not in $L^{\prime}$ and that is different from $w$. Otherwise, $w \in L^{\prime}$ and hence $|w|=n_{s}^{2}$ and $w \in T_{s}$ for some $s$. Note that the characteristic sequence of $\left\langle T_{s}\right\rangle$ is such that at position $w_{i}$ we find the next word $w_{i+1}$. The same holds for $O$, since by our construction, a word of length $n_{s}^{2}$ belongs to $O$ if and only if it belongs to $\left\langle T_{s}\right\rangle$. Therefore, by querying the oracle for the words $w+1, w+2, \ldots, w+|w|$ and by interpreting the answers as bits, we obtain a word $w^{\prime}$ which is the successor of $w$ in $T_{s}$. The function $f$ outputs $w^{\prime}$.

Observe that $f \in \mathrm{FL}^{O}$ and that one query tape suffices to compute $f$. Moreover, $f$ is not length-increasing.
We argue that $f$ is a $\leqslant_{\mathrm{m}}^{\log , O}$-autoreduction for $L$. Clearly, by the definition of $f$ it holds that $f(w) \neq w$. If $w \notin L^{\prime}$, then $w \notin L$ and $f(w) \notin L$. So assume $w \in L^{\prime}$. In this case, $w \in T_{s}$ and $f$ outputs $w$ 's successor in $T_{s}$. By our construction, either $T_{s} \subseteq L$ or $T \cap L_{I}=\emptyset$. So $w \in L \Leftrightarrow f(w) \in L$.

This finishes the proof of Theorem 4.11.
Corollary 4.14. There exists an oracle $O$ and a language $L$ such that all of the following holds.

- L is $\leqslant_{\mathrm{m}}^{\log -l i n, O}$-autoreducible by a reduction function that is computable by a log-space OTT with only one query tape.
- L is not $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic.

Corollary 4.15. There exists an oracle $O$ such that relative to $O$ and with respect to every machine model $\mu \in\{L L, R S T, L, B, W, G\}$ it holds that $\leqslant_{\mathrm{m}}^{\log }$-autoreducibility does not imply $\leqslant_{\mathrm{m}}^{\log }$-mitoticity.

Proof. Let $O$ and $L$ be as in Corollary 4.14 and choose a machine model $\mu$. From the first item of Corollary 4.14 it follows that $L$ is $\leqslant_{\mathrm{m}}^{\log , O}$-autoreducible with respect to $\mu$. By the second item of Corollary 4.14, $L$ is not $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic with respect to the G-model (which is the default machine model in this paper). Among all considered machine models, the G-model is the most powerful one. Hence, $L$ is not $\leqslant_{\mathrm{m}}^{\log , O}$-mitotic with respect to $\mu$.

## 5. Conclusions

We know that autoreducibility and mitoticity are equivalent with respect to polynomial-time many-one reductions [5]. The present paper proves the same for polylog-space many-one reductions. Moreover, with respect to log-space manyone reductions, the notions are almost equivalent, but it is difficult to prove or refute the equivalence (proving requires nonrelativizable methods and refuting is as hard as separating L from P). However, we do not know the relationship of autoreducibility and mitoticity with respect to (poly)log-space truth-table reductions and (poly)log-space Turing reductions. The polynomial-time setting allows separations in these cases [5,6]. It remains an open question if similar separations can be proved in the (poly)log-space setting.

## References

[1] K. Ambos-Spies, P-mitotic sets, in: E. Börger, G. Hasenjäger, D. Roding (Eds.), Logic and Machines, in: Lecture Notes in Comput. Sci., vol. 171, SpringerVerlag, 1984, pp. 1-23.
[2] J.F. Buss, Relativized alternation and space-bounded computation, J. Comput. System Sci. 36 (3) (1988) 351-378.
[3] R. Cole, U. Vishkin, Deterministic coin tossing with applications to optimal parallel list ranking, Inform. Control 70 (1) (1986) 32-53.
[4] G. Gottlob, Collapsing oracle-tape hierarchies, in: Proceedings 11th Conference on Computational Complexity, IEEE Computer Society Press, 1996, pp. 33-42.
[5] C. Glaßer, A. Pavan, A.L. Selman, L. Zhang, Redundancy in complete sets, in: Proceedings 23rd Symposium on Theoretical Aspects of Computer Science, in: Lecture Notes in Comput. Sci., vol. 3884, Springer-Verlag, 2006, pp. 444-454.
[6] C. Glaßer, A.L. Selman, S. Travers, L. Zhang, Non-mitotic sets, in: Proceedings 27th Conference on Foundations of Software Technology and Theoretical Computer Science, in: Lecture Notes in Comput. Sci., vol. 4855, Springer-Verlag, 2007, pp. 146-157.
[7] A.H. Lachlan, The priority method I, Z. Math. Logik Grundlagen Math. 13 (1967) 1-10.
[8] R.E. Ladner, A completely mitotic nonrecursive r.e. degree, Trans. Amer. Math. Soc. 184 (1973) 479-507.
[9] R.E. Ladner, Mitotic recursively enumerable sets, J. Symbolic Logic 38 (2) (1973) 199-211.
[10] R.E. Ladner, N.A. Lynch, Relativization of questions about log space computability, Math. Systems Theory 10 (1976) 19-32.
[11] N.A. Lynch, Log space machines with multiple oracle tapes, Theoret. Comput. Sci. 6 (1978) 25-39.
[12] F.P. Ramsey, On a problem of formal logic, Proc. London Math. Soc. 30 (1930) 264-286.
[13] W.L. Ruzzo, J. Simon, M. Tompa, Space-bounded hierarchies and probabilistic computations, J. Comput. System Sci. 28 (2) (1984) $216-230$.
[14] B. Trakhtenbrot, On autoreducibility, Dokl. Akad. Nauk SSSR 192 (6) (1970) 1224-1227; translation in: Soviet Math. Dokl. 11 (3) (1970) 814-817.
[15] C.B. Wilson, A measure of relativized space which is faithful with respect to depth, J. Comput. System Sci. 36 (3) (1988) $303-312$.


[^0]:    E-mail address: glasser@informatik.uni-wuerzburg.de.
    0022-0000/\$ - see front matter © 2010 Elsevier Inc. All rights reserved.

[^1]:    ${ }^{1}$ This pairwise equivalence can be written as $A \equiv \equiv_{\mathrm{m}}^{\log } A \cap S \equiv_{\mathrm{m}}^{\log } A \cap \bar{S}$ (resp., $A \equiv_{\mathrm{m}}^{\mathrm{plog}} A \cap S \equiv_{\mathrm{m}}^{\mathrm{plog}} A \cap \bar{S}$ ), since $\leqslant_{\mathrm{m}}^{\log }$ (resp., $\leqslant{ }_{\mathrm{m}}^{\text {plog }}$ ) is transitive. This is not possible in the definitions of $\leqslant_{\mathrm{m}}^{\log ^{\mathrm{k}}}$-mitoticity and $\leqslant_{\mathrm{m}}^{\log \cdot \log \log }$-mitoticity.

[^2]:    2 This is the point where we need $S$ and hence $T$ to have high Kolmogorov complexity, since otherwise some information about the latter words could be contained in $w_{i}, w_{i+1}, \ldots, w_{i+c}$ which would allow $M_{1}$ and $M_{2}$ to obtain this information.

