Diagonalization, Uniformity, and Fixed-Point Theorems*

KENNETH W. REGAN

Department of Computer Science,
State University of New York at Buffalo

We derive new fixed-point theorems for subrecursive classes, together with a
theorem on the uniformity of certain reductions, from a general formulation of the
technique of delayed diagonalization. This formulation extends the main theorem of
U. Schöning (Theoret. Comput. Sci. 18 (1982), 95-103) to cases which involve
infinitely many diagonal classes $\%_k$, and which allow each $\%_k$ to contain uncount-
able many members. The main technical work ties the familiar concept of a witness
function directly to the often-studied Cantor-set topology on languages, and
provides a "delay construction" which refines those due to Schöning, S. Breidtbart,
and D. Schmidt. Our "a.e." fixed-point theorems do not require that the "programming system" for the subrecursive class in question be well-behaved; we compare
them to results which do. The other theorem is similar to the "uniform boundedness
theorem" of classical analysis, and extends work of J. Grollmann and A. Selman.

1. INTRODUCTION

This paper proves three closely related new theorems belonging to three
mathematical general which are ordinarily considered to be rather different.
Theorems whose gist is diagonalization are everywhere in computability and
complexity theory. Instances of uniformity are theorems of the form "If for
every $x$ there is a $y$ making $R(x, y)$ hold, then there is a single $y$ giving
$R(x, y)$ for every $x."$ Examples are the "uniform boundedness principle" in
classical real analysis, the quantifier-interchange lemmas used to collapse
some complexity-theoretic hierarchies to their second (i.e., $\forall \exists = \exists \forall$) level,
and most of interest in this paper, the theorem that a promise problem is
$NP$-hard iff it is uniformly $NP$-hard (Grollman and Selman, 1984, 1988).
Third, fixed-point theorems are also familiar in computing. The basic
example is the Recursion Theorem in its "Second" (Cutland, 1980) or
"Rogers" (Rogers, 1967; Machtey and Young, 1981) form: Let $[T_i]_{i=1}^{\infty}$ be
a standard recursive enumeration of Turing machines, and let $f: \mathbb{N}^+ \rightarrow \mathbb{N}^+$
be a total recursive function. Then there exists some $k \in \mathbb{N}^+$ such that $T_k$
and $T_{f(k)}$ compute the same partial function.

* Part of this research was conducted at Cornell University, supported by the U.S. Army
Research Office through the Cornell Mathematical Sciences Institute, Contract DAAG
29-84-C-0018.
The three main theorems, with references to stronger and more formal statements in the text, are as follows. Here "\( \mathcal{F}(Y, N) \)" denotes the class of languages which contain \( Y \) and are disjoint from \( N \), a class \( \mathcal{C} \) is cfv. if (like most complexity classes) it is closed under finite variation of its members, and for an enumeration \( \{A_k\}_{k=1}^\infty \) of languages, \( A_\omega \) denotes \( \{x \neq k \mid x \in A_k\} \). A programming system for a class \( \mathcal{C} \) is a recursive enumeration of Turing machines for all the languages or functions in the class (see Machtey and Young, 1978).

1. [Theorem 5.2]: Let \( \{A_k\}_{k=1}^\infty \) be a recursive presentation of cfv. classes. Let \( \{A_k\}_{k=1}^\infty \) be a recursive presentation of languages such that for each \( k \), \( A_k \notin \mathcal{C}_k \). Then there is a single language \( E \) such that \( E \notin \mathcal{C}_k \) for all \( k \), and yet \( E \) reduces "easily" to the language \( A_\omega \).

2. [Theorem 6.2]: Let \( \{P_k\}_{k=1}^\infty \) be a programming system for polynomial-time bounded Turing reductions. Given a language \( B \) put \( \mathcal{C}_k := \{A \mid L(P_k^A) = B\} \) for each \( k \), and let \( Y, N \) be disjoint languages. If for every language \( L \) in \( \mathcal{F}(Y, N) \) there is a class \( \mathcal{C}_k \) which contains \( L \), then some \( \mathcal{C}_k \) contains all of \( \mathcal{F}(Y, N) \).

3. [Theorem 7.2]: Let \( \{Q_k\}_{k=1}^\infty \) be a programming system for a class \( \mathcal{C} \) which meets some minimal closure conditions (e.g., \( \mathcal{C} := \mathcal{P} \) or \( \mathcal{P} \mathcal{F} \mathcal{P} \mathcal{A} \mathcal{C} \)). Let \( \{A_k\}_{k=1}^\infty \) be any enumeration of languages such that \( A_\omega \) belongs to \( \mathcal{C} \). Then there are infinitely many \( k \) such that \( L(Q_k) \) is a finite variation of \( A_k \).

We give applications for each theorem in its respective section. We also present the theorems in both "effective" and "noneffective" forms. For example, the version of (2) originally given in (Grollman and Selman, 1984) states that if for each \( k \) there is a recursive language \( A_k \in \mathcal{F}(Y, N) \) such that \( A_k \notin \mathcal{C}_k \), then there is a recursive language \( E \in \mathcal{F}(Y, N) \) which is not in \( \mathcal{C}_k \) for any \( k \). (It also takes \( B := \text{SAT} \) in referring to \( \mathcal{N} \mathcal{P} \)-hardness.) The version given above, which was discovered independently in (Regan, 1984, 1985), is noneffective, having a weaker hypothesis and a weaker conclusion. Despite the differences, we show that both forms of the result arise from essentially the same technique.

More than this, we show that many familiar diagonalization results hold in what one may call a uniformly relativized form. For example, the familiar simple statement of "Ladner's theorem" (Ladner, 1975) is that if \( \mathcal{N} \mathcal{P} \neq \mathcal{P} \), then there are languages \( E \) in \( \mathcal{N} \mathcal{P} \) which are neither \( \mathcal{N} \mathcal{P} \)-complete nor in \( \mathcal{P} \). The stronger form tells one how to construct a single total OTM \( M_e \) such that if \( X \) is any oracle making \( \mathcal{N} \mathcal{P}^X \neq \mathcal{P}^X \), then \( L(M_e^X) \) is in \( \mathcal{N} \mathcal{P}^X \), but is neither \( \mathcal{N} \mathcal{P}^X \)-complete nor in \( \mathcal{P}^X \). We suspect that this and similar particular facts may be known, but this does not diminish our main point, which is that we tie them to a general phenomenon. An important goal in
the field is to determine which diagonalization results "relativize" to all oracle sets, and which do not. We offer our work as a preliminary step in classifying them.

The sense of "uniform" in the terms "uniform, effective" or "uniformly relativized" is not the same as the idea of "uniformity" discussed above, and we prefer to emphasize that many of our results are "constructive," insofar as they come from (primitive) recursive operations on the codes of oracle Turing machines. We believe that attention to the effectiveness of such constructions will pay dividends when enough research accumulates to allow an in-depth analysis of complexity theory from the point of view of constructive formal systems. For the moment, we note at the end of the paper that our techniques fail to yield a constructive form of the fixed-point theorem. Hence we ask, with reference to (3) above, whether one can compute \( k \) giving \( L(Q_k) \equiv \langle A \rangle \) as a function of the codes \( a, q \) of total machines such that \( M_a \) accepts \( A_\omega \) and \( M_q \) generates the programming system. This stands in contrast to the classical proof of the Recursion Theorem, which shows one how to compute a fixed point. We suspect that the answer to our open problem is "no."

Our last motivation is to help bind together the work of many papers in the literature devoted to delayed diagonalization, witness functions, the Cantor-set topology on languages, oracle constructions, and various structural properties of complexity classes. Theorem 5.2 extends the "uniform diagonalization" technique of U. Schöning (1981, 1982) so that one may diagonalize out of infinitely many rather than finitely many classes. It also sharpens the "easy" reducibility involved in its statement, perhaps as far as possible. Our development synthesizes methods of S. Breidtbart (1978), R. Landweber, R. Lipton, and E. Robertson, (1981), P. Chew and M. Machtey (1981), and D. Schmidt (1985) through the concept of "out-running" witness functions. A technical innovation, namely "running a clock backwards," enables us to avoid the explicit uses of recursion in (Breidbtart, 1978; Schmidt, 1985) (the former directly appeals to the classical Recursion Theorem), and transmutes the time-constructibility requirement of (Schöning, 1981, 1982) into a space-constructibility requirement. We also extend the technique to diagonalize over certain classes having \( 2^\infty \)-many members, by drawing the connection between witness functions and nowhere-dense classes, which is inherent in (Mehlhorn, 1973; Kozen and Machtey, 1980; Dowd, 1982; Blum and Impagliazzo, 1987) and in a subrecursive setting in (Lutz, 1987).

Section 2 provides useful background information, notably on the Cantor-set topology \( \mathcal{Z} \). Section 3 defines the various witness functions we are concerned with, and outlines the diagonalization technique. Section 4 presents the technical work that goes into our refined and general "delay construction," and can be skipped by those interested only in its conse-
quences. Sections 5, 6, and 7 present the theorems making up the respective pieces of the title, together with some applications. Section 8 concludes by summarizing the similarities among these theorems, and by relating them to open problems in the field.

2. Preliminaries

We refer to a finite basic alphabet $\Sigma$, and to the augmented alphabet $\Gamma := \Sigma \cup \{\#\}$. Here "#" is called the separator symbol. We use the operation $(x, y) \mapsto x \# y$ to encode tuples, even though it is not 1–1 when $x$, $y$ themselves contain "#" signs. We use it mostly when $y \in \Sigma^*$, and then it is 1–1. Any subset $A$ of $\Sigma^*$ of $\Gamma^*$ is called a language, and any collection $\mathcal{C}$ of languages, a class. We write either $\sim A$ or $\bar{A}$ to denote the complement of $A$ when the set being complemented over is clear; similarly $\sim \mathcal{C}$ denotes $\mathcal{P}(\Gamma^*) \setminus \mathcal{C}$ or $\mathcal{P}(\Sigma^*) \setminus \mathcal{C}$ according to context.

Our typography follows some general conventions: $x$, $y$, $z$, $w$, ... denote strings; $m$, $n$, $q$, $r$, ..., numbers; $a$, $b$, $c$, $d$, ..., either numbers or strings; $i$, $j$, $k$, $l$, ..., numbers used as indices; $f$, $g$, $h$, $p$, ..., functions; $\alpha$, $\beta$, $\gamma$, $\delta$, ..., 0–1 characteristic vectors; $A$, $B$, $C$, $D$, ..., languages used in examples; $I$, $J$, $K$, ..., sets of indices; $X$, $Y$, $Z$, $W$, ..., language variables; $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, ..., classes of languages; and $\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}$, ..., classes of functions. A boldface $\emptyset$ distinguishes the empty class from the empty language $\emptyset$. When a class of languages has an analogous class of functions, we add an "F" somewhere in its name; thus $\mathcal{F} \mathcal{R} \mathcal{C} \mathcal{E}$ denotes the class of total recursive functions, $\mathcal{F} \mathcal{P}$ the polynomial-time computable functions, and $\text{DTIME}_t^F[t(n)]$ the class of functions computable in time $t(n)$ with oracle set $A$.

Taking some total ordering of $\Sigma$, and putting "#" last in $\Gamma$, gives us the canonical orderings of $\Sigma^*$ and $\Gamma^*$, which are defined by letting shorter strings precede longer ones, and ordering strings of the same length lexicographically. We write num$_\Sigma$ and num$_\Gamma$ for the resulting bijections from $\Sigma^*$ to $\mathbb{N}^+$ and $\Gamma^*$ to $\mathbb{N}^+$, and str$_\Sigma$, str$_\Gamma$ for their respective inverses. The empty string $\lambda$ corresponds to 1 in either case. We drop the subscripts "\Sigma" and "\Gamma" when the context is clear, and more worthy of note, sometimes drop "num" or "str" altogether in using numbers and strings interchangeably. It is understood that a natural number $k$ stands for str$_\Sigma(k)$, not str$_\Gamma(k)$. Typical uses of this convention are "$x \neq k$" in place of "$x \neq \text{str}_\Sigma(k)$", and "$k, l \leq x$" to abbreviate "$k, l \leq \text{num}_\Gamma(x)$."

Our Turing machines have single input, output, and oracle tapes with alphabet $\Gamma$, and any finite number of worktapes having alphabet $\Upsilon \supseteq \Gamma \cup \{\lambda\}$, where $\lambda$ denotes the blank. All tapes are semi-infinite extending to the right. The input tape is read-only, the output tape write-only, and both takes are on-line, meaning that their heads cannot
move leftward. We do not allow multiple inputs separated by blanks; an input \( x \in \Gamma^* \) may be interpreted as a tuple of strings in \( \Sigma^* \) separated by "#" signs. We suppose the input is initially left-justified on the input tape, with the head scanning the first cell, and say the first blank encountered "marks the end of the input." We adopt the convention of (Hopcroft and Ullman, 1979) that every TM reads all of its input.

We consider every TM \( T \) be an oracle Turing machine (OTM), always supposing that the oracle tape is present and that \( T \)'s finite control contains the query states \( q_? \), \( q_y \), and \( q_n \). Whenever \( T \) with oracle \( A \subseteq \Gamma^* \) enters \( q_? \), \( T \) next enters \( q_y \) or \( q_n \) according to whether \( z \in A \), where \( z \) is identifiable as the string over \( \Gamma \) extending from the left end of the oracle tape to the first blank on that tape. We identify non-oracle TM's with OTM's having \( \emptyset \) as oracle set. An OTM \( T \) is \( X \)-total if \( T \) halts for all inputs with oracle set \( X \), and simply total (as an OTM) if \( T \) is \( X \)-total for all oracles \( X \).

We distinguish between transducers, which compute partial functions from \( \Gamma^* \) to \( \Gamma^* \), and acceptors of languages. We sometimes identify acceptors with transducers whose range is binary (typically \{"yes," "no"\}, \{0, 1\}, or \{1, 2\}), and most often denote them by "M" rather than "T." As usual, \( L(M^A) \) denotes the language accepted by \( M \) with oracle set \( A \). All TM's used in this paper are deterministic; any references to nondeterminism are for exposition only. Our machine model follows the standard multitape TM model of (Hopcroft and Ullman, 1979) in respects other than those above.

Any "reasonable" means of coding up tape labels, worktape alphabets, and finite controls yields a recursive enumeration \( [T_j]_{j=1}^{\infty} \) of transducers which is acceptable in the sense of (Rogers, 1967). We similarly represent the acceptors by \( [M_i]_{i=1}^{\infty} \), which one can regard as a recursive sublist of \( [T_j]_{j=1}^{\infty} \). The partial time and space functions, relativized to an oracle set \( X \), are denoted by Time\(^X\)(\( j, x \)) and Space\(^X\)(\( j, x \)), and respectively denote the number of steps and the number of distinct worktape cells used in the computation of \( T_j \) with oracle \( X \) on input \( x \). Both are undefined if this computation fails to halt. The more familiar forms of these functions are denoted by \( t^X(j, n) := \max\{\text{Time}(j, x) \mid |x| \leq n\} \), and \( s^X(j, n) := \max\{\text{Space}(j, x) \mid |x| \leq n\} \). (Some technical notes: Writing "...|x| = n..." in these definitions would serve equally well. Our results also hold under the more stringent condition that oracle-tape cells are charged against the space bound, though we have preferred to adopt the loosest of the conventions about oracle log-space discussed in (Ladner and Lynch, 1976; Ruzzo, Simon, and Tompa, 1984).)

### 2.1. Minimum Time/Space Complexity Bounds

By the input convention of (Hopcroft and Ullman, 1979) adopted above, \( t^A(j, n) \geq n + 1 \) for all \( n \in \mathbb{N}, j \in \mathbb{N}^+, \) and \( A \subseteq \Gamma^* \). When \( T_j \) runs in the
minimum $n + 1$ steps for all (inputs of length) $n$, we say $T_j$ runs in \textit{real time}. Some authors relax the definition to say that for some $c > 0$, there is no computation of $T_j$ having $c$ consecutive steps in which the input head fails to move right. We keep the latter notion separate, calling it operation in \textit{delay c}, and when $c$ is variable, operation in \textit{finite delay} or \textit{quasi-real time} (Book and Greibach, 1970). Every language acceptable in finite delay is acceptable in real time (cf. (Hartmanis and Stearns, 1965), but the same is not true of functions, if only because delay $c$ allows more time to write on the output tape. We split the difference between these stipulations, requiring the input head to advance at each step until the blank marking the end is encountered, but then allowing some delay before the TM must halt.

\textbf{DEFINITION 2.1.} (a) A TM $T$ (with oracle $A$) runs in \textit{real time plus c extra steps} if on any input $x$, $T$ (or $T^A$) reaches the blank at the end of $x$ after $|x|$ steps, and thereafter halts within $c$ steps.

(b) An OTM $T$ is an $\mathcal{AL}$-machine if for some $c > 0$ and all oracle sets $A$, $T^A$ runs in real time plus $c$ extra steps, and in addition $T^A$ runs in $c \cdot \log_2 n$ space.

The intent is to regard maps of the form $x \mapsto x \neq k$, where $k$ is fixed or bounded, as being real-time computable. We also consider reductions of this form where $k$ is unbounded. Then no fixed amount of "extra steps" suffices, and so we allow the number of extra steps to increase by a suitable slow-growing function of $n$.

\textbf{DEFINITION 2.2.} An OTM $T$ is a $\mathcal{L}$-machine if for some $c > 0$ and all oracle sets $A$, $T^A$ runs simultaneously in $c \cdot \log_2 n$ space and real time plus $c \cdot \log_2 n$ extra steps.

For any $A \subseteq \Gamma^*$, $\mathcal{AL}_A$ denotes the class of total functions from $\Gamma^*$ to $\Gamma^*$ computed by $\mathcal{AL}$-machines with oracle $A$, and $\mathcal{AL}^A$ denotes the analogous class of languages. $\mathcal{AL}^A_\neq$ and $\mathcal{AL}^A$ are defined similarly with reference to Definition 2.1. With reference to the standard "DTISP[$t(n), s(n)$]" notation for simultaneous time/space bounds, we have

\textbf{PROPOSITION 2.1.} For any oracle set $A \subseteq \Gamma^*$ and function $h: \Gamma^* \rightarrow \Gamma^*$,

(a) $\mathcal{AL}^A = DTISP^A[n + 1, \log_2 n]$.

(b) If $h \in \mathcal{AL}^A_\neq$ and $k \in \text{Ran}(h)$, then the language $h^{-1}(k)$ is in $\mathcal{AL}^A$.

The proof, which uses techniques from (Hartmanis and Stearns, 1965; Rosenberg, 1967), is left to the reader.
To justify our subsection heading, we note that any language which is acceptable on-line in \( o(\log n) \) space is regular (Hopcroft and Ullman, 1969), and so acceptable is simultaneous real time and constant (i.e., zero) space. Thus \( \mathcal{RL} \) stands at the minimum nontrivial level of complexity, at least in terms and space. Although this divide is not quite so crisp for the function classes \( \mathcal{RL}_\tau \) and \( \mathcal{2L}_\tau \), they carry the same intent of defining a "minimum-complexity reducibility relation."

**Definition 2.3.** For any languages \( A \) and \( B \), we write \( A \leq_{m}^{q} B \) if for some \( f \in \mathcal{2L}_\tau \) and all \( x, x \in A \Rightarrow f(x) \in B \). If \( f \) is 1–1, and if \( \mathcal{2L}_\tau \) contains a left inverse for \( f \), then we write \( A \leq_{l}^{q} B \). If in addition \( f \) is "length-increasing" on strings, i.e., \(|f(x)| > |x| \) for all \( x \), then we write \( A \leq_{l}^{q} B \).

The relations \( \leq_{m}^{l} \), \( \leq_{l}^{l} \), and \( \leq_{l}^{l} \) are defined similarly for \( \mathcal{RL}_\tau \). For any oracle \( X \), \( \leq_{m}^{X} \) denotes reducibility by a function in \( \mathcal{2L}_X \), and so on.

Note that \( A \leq_{l}^{l} B \Rightarrow A \leq_{l}^{l} B \Rightarrow A \leq_{m}^{l} B \), so that many of our results will hold with the more familiar polynomial-time reducibilities in place of the stronger real-time/log-space ones. We do not try to define a real-time analogue of polynomial-time Turing reducibility, \( \leq_{\tau}^{p} \). The standard recursion-theoretic many–one and Turing reducibilities are denoted by \( \leq_{m} \) and \( \leq_{\tau} \). For a proof of the next result, which says that our new reducibilities are fairly well-behaved, see (Regan, 1986b).

**Proposition 2.2.** For any oracle set \( X \), the classes \( \mathcal{RL}_X^X \) and \( \mathcal{2L}_X^X \) are closed under composition, so that the relations \( \leq_{m}^{X} \), \( \leq_{l}^{X} \), \( \leq_{l}^{X} \), \( \leq_{l}^{X} \) (etc.) are transitive.

**Note.** Previous papers by this author (Regan, 1986a, 1986b, 1988) have used the weaker definition of \( \mathcal{2L}_\tau \) as simultaneous log-space and quasi-real time plus \( O(\log n) \) extra steps. This owes to their use of a pairing function, i.e., a bijection from \( \Sigma^* \times \Sigma^* \) to \( \Sigma^* \), rather than the operation \((x, y) \mapsto x \# y \) used here. We do not know whether there is a pairing function \( \langle \cdot, \cdot \rangle \) such that both \( \langle \cdot, \cdot \rangle \) and \( \langle \cdot, \cdot \rangle^{-1} \) belong to \( \mathcal{2L}_\tau \) under the present stipulation. The pairing function used in the above papers is both computable and invertible in delay 2 and zero space. We have changed the definition of \( \mathcal{2L}_\tau \), and also that of "projection" in the next subsection, in order to preserve the statements of results under both the "\( \langle \cdot, \cdot \rangle \)" and the "\( \# \)" method of representing tuples.

### 2.2. Universal Languages and Recursive Presentations

For any language \( U \subseteq \Gamma^* \) and \( k \in \mathbb{N}^+ \), we can define the \( k \)-th projection of \( U \), written \( U_k \), to be \( \{x \in \Gamma^* \mid x \# k \in U \} \). Then \( \mathcal{P}_k[U] \) denotes the class \( \{U_k \mid k \in \mathbb{N}^+ \} \). If \( \mathcal{C} = \mathcal{P}_k[U] \), then we call \( U \) a universal language for \( \mathcal{C} \). Our use of this terms does not entail that \( U \) itself belongs to \( \mathcal{C} \). Any universal
language $U$ for $\mathcal{C}$ is $\leq_{n}^{U}$-hard for $\mathcal{C}$, since the map $x \mapsto x \neq k$ reduces $U_k$ to $U$ for any $k$. Thus classes such as $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{PSPACE}$ (etc.), which do not even have linear-time complete sets, do not contain any universal languages for themselves.

Given classes $\mathcal{C}$ and $\mathcal{D}$, we say $\mathcal{C}$ is $\mathcal{D}$-presentable if $\mathcal{D}$ contains a universal language for $\mathcal{C}$. This usage embraces the terms recursively presentable (r.p.) and r.e.-presentable, and also the term $X$-r.p. with reference to $\mathcal{D} := \mathcal{A} \subseteq \mathcal{C}^X$ (i.e., \{ $L | L \leq_T X$ \}). By the $S$–$m$–$n$ Theorem (in relativized form; see Rogers, 1967, or Soare, 1987), a class $\mathcal{C}$ is $X$-r.p. iff there exists a recursive enumeration $[Q_k]_{k=1}^{\infty}$ of $X$-total OTM's such that $\mathcal{C} = \{ L(Q_k^k) | k \in \mathbb{N}^+ \}$.

If we are given an enumeration $[A_k]_{k=1}^{\infty}$ of the languages in some class $\mathcal{A}$, then we refer to the particular universal language $A_\omega := \{ x \neq k | x \in A_k \}$. Then $A_k$ is the $k$th-projection of $A_\omega$, as defined above. We define the join of two languages $A$ and $B$ to be $A \oplus B := \{ x \neq 1 | x \in A \} \cup \{ x \neq 2 | x \in B \}$. Then $A_\omega$ may be regarded as a kind of "infinite join" of the languages $A_k$.

If $A_\omega$ is recursive, then we also call $[A_k]_{k=1}^{\infty}$ a recursive presentation of $\mathcal{A}$. We deal similarly with the concept of a recursive presentation $[\mathcal{C}_k]_{k=1}^{\infty}$ of classes, leaving the formal definition to the appropriate context. In cases where we do not specify whether the enumeration $[A_k]$ is finite or infinite, we denote it by $[A_k]_{k \in K}$, where $K$ is chosen to be either $\mathbb{N}^+$ or $\{1, \ldots, m \}$ for some $m \in \mathbb{N}^+$. (Our notation and results allow more general choices of "$K,"$ but we do not explore them in this paper.)

2.3. The Cantor-Set Topology on Languages

For any language $A \subseteq \Gamma^*$, we define $\chi[A]$ to be the graph of its characteristic function, which under the standard enumeration of $\Gamma^*$ becomes an infinite vector of 0's and 1's. This yields a 1–1 correspondence between $\mathcal{P}(\Gamma^*)$ and $\{0, 1\}^{\omega}$; each language may also be regarded as an infinite branch of the full binary tree $\mathcal{B}$. Using the standard notation $\alpha \subseteq \beta$ to mean $\alpha$ is an initial substring of $\beta$, we write $\alpha \subseteq A$ if $\alpha \subseteq \chi[A]$. For all $\alpha \in \{0, 1\}^{*}$ and $\chi \in \Gamma^*$, we write $\xi \preceq \chi$ if $|\xi| \leq \text{num}(\chi)$.

For every $\alpha \in \{0, 1\}^{*}$, define the class $\mathcal{O}_\alpha := \{ A \subseteq \Gamma^* | A \ni \alpha \}$. The collection $\{\mathcal{O}_\alpha\}$ forms an open basis for a topology $\mathcal{I}$ on $\mathcal{P}(\Gamma^*)$. This $\mathcal{I}$ is variously called the Cantor-set topology (Rogers, 1967) or the positive-information topology (Cutland, 1980). Thus a class $\mathcal{C}$ is open in $\mathcal{I}$ if for every $A \in \mathcal{C}$ there exists $\alpha \subseteq A$ such that $\mathcal{O}_\alpha \subseteq \mathcal{C}$. $\mathcal{C}$ is closed iff its complement in $\mathcal{P}(\Gamma^*)$ is open. Every singleton class $\mathcal{C} = \{A\}$ is closed in $\mathcal{I}$. We can restrict the topology $\mathcal{I}$ to any $\mathcal{D} \subseteq \mathcal{P}(\Gamma^*)$, saying, e.g., that $\mathcal{C}$ is closed in $(\mathcal{D}, \mathcal{I})$ if $\mathcal{C} = \mathcal{B} \cap \mathcal{D}$ for some closed class $\mathcal{B}$. For references exploiting this topology in complexity theory, see (Mehlhorn, 1973; Kozen and Machtey, 1980; Bennett and Gill, 1981; Dowd, 1982; Lutz, 1987).
It is worth noting that $\mathcal{I}$ is generated by the metric $\rho$ defined for all $A, B \subseteq \Gamma^*$ by $\rho(A, B) := 2^{-n}$, where $\text{str}_I(n)$ is the least string in $A \triangle B$. (If $A = B$, then $\rho(A, B) = 0$.) Owing to König's Lemma, the space $(\mathcal{P}(\Gamma^*), \mathcal{I})$ is compact, which implies that all closed classes $\mathcal{C} \subseteq \mathcal{P}(\Gamma^*)$ are also compact and hence complete. The latter term means that if a sequence $B_1, B_2, B_3, \ldots$ of languages in $\mathcal{C}$ converges in the metric to a language $B$, then $B$ also belongs to $\mathcal{C}$.

The motivation for $\mathcal{I}$ is that if $\mathcal{C}$ is open and $A \in \mathcal{C}$, then the membership of $A$ in $\mathcal{C}$ is "witnessed" by a finite amount of "positive information" about $A$, namely an initial segment $\alpha$ such that $\mathcal{C}_\alpha \subseteq \mathcal{C}$. For any class $\mathcal{C}$ we write $\text{Con}(\mathcal{C})$ for $\{\alpha \in \{0, 1\}^* | (\exists A \subset \Gamma^*)[\alpha \sqsubseteq A \land A \in \mathcal{C}]\}$, namely those bits of positive information which are consistent with membership in $\mathcal{C}$. The closure of $\mathcal{C}$, written $\text{cl}(\mathcal{C})$, is the smallest closed class containing $\mathcal{C}$, which is well-defined as the intersection of all closed classes containing $\mathcal{C}$.

It also equals $\{B \subseteq \Gamma^* | (\forall \beta)[\beta \subseteq B \Rightarrow \beta \in \text{Con}(\mathcal{C})]\}$, i.e., the largest class $\mathcal{B}$ such that $\text{Con}(\mathcal{B}) = \text{Con}(\mathcal{C})$. Thus $\mathcal{C}$ is dense in $\mathcal{P}(\Gamma^*)$, meaning that $\text{cl}(\mathcal{C})$ equals $\mathcal{P}(\Gamma^*)$, iff $\text{Con}(\mathcal{C}) = \{0, 1\}^*$. $\mathcal{C}$ is nowhere dense iff its closure contains no nonempty open class, i.e., $\text{Con}(\sim \text{cl}(\mathcal{C})) = \{0, 1\}^*$, or equivalently

$$
(\forall \alpha \in \{0, 1\}^*)(\exists \beta \in \{0, 1\}^*)[\beta \sqsupseteq \alpha \wedge \mathcal{C}_\beta \cap \mathcal{C} = \emptyset]. \quad (2.1)
$$

Thus $\mathcal{C}$ is nowhere dense iff no previous work $\alpha$ in building up a language prevents one from extending $\alpha$ to a finite witness for nonmembership in $\mathcal{C}$.

$\mathcal{C}$ is effectively nowhere dense if there is a recursive function $g: \{0, 1\}^* \rightarrow \{0, 1\}^*$ giving $\beta$ in terms of $\alpha$. $\mathcal{C}$ is nowhere dense in $\mathcal{D}$ iff $\mathcal{C} \cap \mathcal{D}$ is nowhere dense in the space $(\mathcal{D}, \mathcal{I})$, and this is equivalent to

$$
(\forall \alpha \in \text{Con}(\mathcal{D}))(\exists \beta \in \text{Con}(\mathcal{D}))[\beta \sqsupseteq \alpha \wedge \mathcal{C}_\beta \cap \mathcal{C} \cap \mathcal{D} = \emptyset]. \quad (2.2)
$$

For any $\alpha \in \{0, 1\}^*$, we write $D_\alpha$ for the finite set $\{x | x_{\text{num}(\alpha)} \neq 1\}$. Given languages $A, B, C$ we define the splice of $(A, B)$ by $C$ to be $(A \cap C) \cup (B \cap \overline{C})$. Given 0–1 vectors $\alpha, \beta$ we define the splice $\alpha/\beta$ to be $\alpha$ if $|\alpha| \geq |\beta|$, and the vector obtained by replacing the first $|\alpha|$ places of $\beta$ by $\alpha$, otherwise. Note that $\alpha/(\beta/(\alpha/\beta))$ always equals $\alpha/\beta$. Given $\alpha \in \{0, 1\}^*$ and $B \subseteq \Gamma^*$, we write $\alpha/B$ for the language corresponding to $\alpha/\beta[B]$, and observe that this equals the splice of $(D_\alpha, B)$ by $\{\lambda, \ldots, \text{str}(|\alpha|)\}$. A class $\mathcal{D}$ is closed under finite splices if for all $\alpha \in \text{Con}(\mathcal{D})$ and $B \in \mathcal{D}$, $\alpha/B \in \mathcal{D}$.

The following (probably known) result supplies some motivation for our considering classes of the form $\mathcal{P}(Y, N) := \{L \subseteq \Gamma^* | L \supseteq Y \wedge L \supseteq N\}$, where $Y, N \subseteq \Gamma^*$. Note that $\mathcal{P}(Y, N) \neq \emptyset$ iff $Y$ and $N$ are disjoint, and $\mathcal{P}(Y, N)$ is dense iff $Y = N = \emptyset$.

**Proposition 2.3** (Regan, 1984, 1985). A class $\mathcal{D}$ is closed in $\mathcal{I}$ and closed under finite splices iff $\mathcal{D} = \mathcal{P}(Y, N)$ for some $Y, N \subseteq \Gamma^*$. 
The proof, which defines \( Y := \{ x \mid (\forall \alpha)[z_{num(\alpha)} = "0" \Rightarrow \alpha \notin \text{Con}(\mathcal{D})] \}, \) \( N := \{ x \mid (\forall \alpha)[z_{num(\alpha)} = "1" \Rightarrow \alpha \notin \text{Con}(\mathcal{D})] \}, \) and uses the completeness of \((\mathcal{D}, \rho), \) is left to the reader.

Last, for any \( A, B \subseteq \Gamma^* \) we write \( A \equiv B \) if \( A \triangle B \) is finite, and \( A' \) for \( \{ B \mid A \equiv B \}. \) Similarly we write \( \not\exists \) for \( \bigcup \{ A \subseteq \Gamma^* \}, \) and say \( \not\exists \) is closed under finite variations (cfv.) if \( \not\exists \subseteq \not\exists \). Note that every cfv. class other than \( \not\exists \) is dense in \( \mathcal{P}(\Gamma^*) \), so that the topology \( \mathcal{I} \) does not by itself offer any distinctions among familiar classes such as \( \mathcal{P}, \mathcal{N}, \mathcal{D}, \mathcal{D}_j, \mathcal{F}, \) and even \( \not\exists \). On the other hand, the following result shows some of the usefulness of nowhere dense classes. The first statement actually follows from the classical Baire Category Theorem, which states that no complete metric space (here, \( \mathcal{D} \) with \( \rho \)) can be a countable union of nowhere dense subspaces (here, the union of \( \mathcal{C}_x := \{ C \triangle \mathcal{D} \mid C \in \mathcal{C} \} \) over \( x \in \{0,1\}^* \). The second statement could be made to follow from an effective form of Baire's theorem, but adapting one of the formulations in (Mehlhorn, 1973; Kozen and Machtley, 1980; Dowd, 1982; Lutz, 1987) would appear less suited to the present paper than does our exhibiting the following standard diagonalization argument.

**Lemma 2.4.** Let \( \mathcal{D} := \mathcal{P}(Y, N) \) for some disjoint \( Y, N \subseteq \Gamma^* \), and let \( \mathcal{C} \) be closed in \( \mathcal{D} \). Then \( \mathcal{C} \) is nowhere dense in \( \mathcal{D} \) iff there exists \( A \in \mathcal{D} \) such that \( A' \cap \mathcal{C} = \emptyset \). Moreover, if \( Y, N \) are recursive and \( \mathcal{C} \) is effectively nowhere dense in \( \mathcal{D} \), then \( A \) can be chosen recursive.

**Proof.** First let \( A \) be given, and let \( \alpha \in \text{Con}(\mathcal{D}) \). Put \( A' := \alpha/A \). Since \( A' \notin \mathcal{C} \) and \( \mathcal{C} \) is closed, there exists \( \beta \subseteq A' \) such that \( \alpha \beta \cap \mathcal{C} = \emptyset \). By the closure under finite splices, \( A' \in \mathcal{D} \), and so \( \beta \in \text{Con}(\mathcal{D}) \). Hence (2.2) is met.

Conversely, let \( \mathcal{C} \) be nowhere-dense in \( \mathcal{D} \). Let \([\gamma_j]_{j=0}^\infty \) be the enumeration of \( \text{Con}(\mathcal{D}) \) obtained by striking out consistent prefixes from the standard enumeration of \( \{0,1\}^* \); this is recursive if \( Y \) and \( N \) are. Now we define \( A \) in stages.

**Stage 1.** Put \( \delta_1 := \lambda \).

**Stage \( j \).** Put \( \alpha_j := \gamma_j/\delta_j, \) Apply (2.2) with \( \alpha := \alpha_j \) to obtain \( \beta_j \in \text{Con}(\mathcal{D}) \) such that \( \alpha \beta_j \cap \mathcal{C} \cap \mathcal{D} = \emptyset \). Then define \( \delta_j := \delta_j, \beta_j). \)

Clearly \( \delta_1 \in \text{Con}(\mathcal{D}) \). At stage \( j \), the induction assumption, the hypotheses, the closure of \( \mathcal{D} \) under finite splices, and (2.2) respectively place \( \delta_{j-1}, \gamma_j, \alpha_j, \) and \( \beta_j \) in \( \text{Con}(\mathcal{D}), \) so \( \delta_j \in \text{Con}(\mathcal{D}). \) There is a unique language \( A \) such that \( \delta_j \subseteq A \) for all \( j \), and since \( \mathcal{D} \) is complete, \( A \in \mathcal{D}. \) Also note that if \( \beta_j \) is given by \( g(\alpha_j) \) for some recursive \( g(\cdot), \) then the sequence \([\delta_j]_{j=1}^\infty \) depends recursively on \([\gamma_j]_{j=1}^\infty \), and since \( \delta_j \subseteq \delta_{j+1} \subseteq \ldots \), so does the language \( A. \)
Now consider any $A' \in \mathcal{D}$ such that $A' \equiv A$. Then $A' = \gamma/A$ for some $\gamma \in \text{Con}(\mathcal{D})$, and so there exists $j$ such that $\gamma = \gamma_j$. Consider the string $\beta_j$ found at stage $j$. We claim $\beta_j = \gamma_j/\delta_j$. By (2.2), $\beta_j = (\gamma_j/\delta_{j-1})\beta$ for some string $\beta$. Then $\delta_j = (\delta_{j-1}/(\gamma_j/\delta_{j-1}))\beta$, and $\gamma_j/\delta_j = (\gamma_j/(\delta_{j-1}/(\delta_{j-1})))\beta$. As noted above, this simplifies to $\gamma_j/\delta_j = (\gamma_j/\delta_{j-1})\beta = \beta_j$, proving the claim.

Since $\delta_j \subseteq A$, the claim gives us $\beta_j \subseteq A'$, so $A' \in \mathcal{C}_\beta$, and (2.2) gives us $A' \notin \mathcal{C}$.]

### 3. Witness Functions and Diagonalization

The heart of this paper is an "automatic" technique for constructing languages $E$ of relatively low complexity which meet infinitely many requirements $R_j$, where each requirement has a particularly simple form. That is, the class $R_j$ of languages meeting $R_j$ is nonempty and open in the topology $\mathcal{T}$, so that $R_j$ can be satisfied by a finite initial segment of $E$. If in addition the complement of each $R_j$ is nowhere dense in $\mathcal{T}$, so that no finite amount of "previous work" could prevent $R_j$ from being satisfied, then all the requirements $[R_j]_{j=1}^\infty$ can be met by building $E$ in the most straightforward manner. We say the technique is "automatic" because it does not require attending to individual requirements, but only to checking that certain witness functions are total.

In common parlance, a witness to the assertion "$A \neq B$" is a string $y \in A \triangle B$. If $B$ is fixed, and $R_j(A) \leftrightarrow "A \neq B"$ for all $A \subseteq \Gamma^*$, then the class $R_j$, being the complement of $\{B\}$, is open in $\mathcal{T}$. Then we take the witness to "$A \in R_j$" to be the whole initial segment $\alpha$ of $A$ up to $y$ (i.e., such that $|\alpha| = \text{num}(y)$), or any other suitable initial segment of $A$.

When it comes to meeting $R_j$ in practice, the language $E$ will have been built up to some string $x$ quite differently from $A$. In the case $R_j := \sim \{B\}$, we need a witness $y \in A \triangle B$ with $y \geq x$. Such a $y$ exists for all $x$ if and only if $A \neq \emptyset B$, or put another way, iff $A/\emptyset B$ is not another way, iff $A' \cap \{B\} = \emptyset$. In particular we can define the "next witness" function $f_{A,B}(x)$ for all $x$ to be the least $y \geq x$ giving $y \in A \triangle B$. The general case requires a more elaborate definition, which is suggested by the proof of Lemma 2.4.

**Definition 3.1.** Let $\mathcal{B}$ be closed in $\mathcal{T}$, let $A \subseteq \Gamma^*$ be such that $A' \cap \mathcal{B} = \emptyset$, and let $g: \{0, 1\}^* \rightarrow \{0, 1\}^*$. Then we say $g(\cdot)$ witnesses the nowhere-denseness of $\mathcal{B}$ with $A$ if for all $\gamma \in \{0, 1\}^*$, $g(\gamma) \in A$ and $C_{\gamma/g(\gamma)} \cap \mathcal{B} = \emptyset$. We associate to $g$ the later-witness function $f_{g}: \Gamma^* \rightarrow \Gamma^*$ for the assertion "$A' \cap \mathcal{B} = \emptyset$," which is defined for all $x$ by

$$f_g(x) := \text{str}(\max\{|g(\gamma)| \gamma < x\}).$$

(3.1)

The later-witness functions provide a measure of how far one must
search to fulfill the next step of the construction, which is guaranteed never to stall by the nowhere-denseness of $\mathcal{B}$.

Among these functions there is a least element with respect to both strict and a.e. majorization. It is given for all $x$ by $f_{A,x} := \min\{ y \mid (\forall \gamma < x) (\exists \beta \subseteq A)[\gamma \cap \beta = \emptyset \land \beta \subseteq y]\}$. We call this the next-witness function for “$A' \cap \mathcal{B} = \emptyset$.” When $\mathcal{B} = \{ B \}$, $f_{A,x}$ is the same as the $f_{A,x}$ given above.

In the most general situation we consider, there will be an indexed collection $[A_k]_{k \in K}$ of languages, where $K$ is either $\mathbb{N}^+$ or $\{1, \ldots, m\}$ for some $m \in \mathbb{N}^+$, and for each $k$, there will be countably many classes $\mathcal{B}_{kl}$ such that $A_k \cap \mathcal{B}_{kl} = \emptyset$ for all $l$. This replaces the “$R_j$” notation, where we regard $\sim$ as the class “$R_j$.” Since we shall not reap any profit from having only finitely many $\mathcal{B}_{kl}$ for a given $k$, we may as well suppose $l$ always runs over $\mathbb{N}^+$, with classes repeated if necessary. Then we write $f_{kl}$ to denote some later-witness function for “$A_k \cap \mathcal{B}_{kl} = \emptyset$,” for all $k \in K$ and $l \in \mathbb{N}^+$.

The following noneffective lemma isolates the main combinatorial twist in our diagonalization mechanism.

**Lemma 3.1.** Suppose a later-witness function $f_{kl}$ for “$A_k \cap \mathcal{B}_{kl} = \emptyset$” exists for all $k \in K$, $l \in \mathbb{N}^+$. Let $h: \mathbb{N}^* \rightarrow K$ be any function such that for all $k, l$ there exists an $x$ for which $h(\cdot)$ takes the constant value $k$ on the interval $[x, f_{kl}(x)]$. Then the language

$$E := \bigcup_{k \in K} (A_k \cap h^{-1}(k))$$

is not in $\mathcal{B}_{kl}$ for any $k \in K$, $l \in \mathbb{N}^+$.

**Proof.** Given $k$ and $l$, let $x$ be such that $h(x) = k$ for all $x \leq f_{kl}(x)$. Let $\gamma$ be the unique member of $\{0, 1\}^*$ such that $\gamma \subseteq E$ and $|\gamma| = \text{num}(x) - 1$. Then there exists $\beta \subseteq A_k$ with $\beta \subseteq f_{kl}(x)$ such that $\gamma \cap \mathcal{B}_{kl} = \emptyset$. Since $[x, \text{str}(|\beta|)] \subseteq h^{-1}(k)$, $\gamma / \beta \subseteq E$. Hence $E \notin \mathcal{B}_{kl}$. $\blacksquare$

For the result in the case where each $\mathcal{B}_{kl}$ is a singleton class $\{ C_{kl} \}$, see (Regan, 1986b, 1988).

We focus attention onto a single function $f$ in lieu of the collection $\{ f_{kl} \}$ by defining.

**Definition 3.2.** A function $f: \mathbb{N}^* \rightarrow \mathbb{N}^*$ is a witness-ranging function for the assertion “$(\forall k, l)[A_k \notin \mathcal{B}_{kl}]$” if for all $x$, all $k$, $l \leq x$, and all $\gamma \leq x$, there exists $\beta \subseteq A_k$ such that $\beta \subseteq f(x)$ and $\gamma \cap \mathcal{B}_{kl} = \emptyset$.

For any such $f$ to exist, the assertion “$(\forall k, l)[A_k \cap \mathcal{B}_{kl} = \emptyset]$” must of course hold. Then, as with the individual later-witness functions $f_{kl}$, there is a least witness-ranging function.

Now we may replace the condition on $h$ in Lemma 3.1 by
DEFINITION 3.3. (a) Let $f: \Gamma^* \rightarrow \Gamma^*$ be monotone increasing, and let $h$ be defined on $\Gamma^*$. Say $h$ outruns $f$ if for each $k \in \text{Ran}(h)$, there are infinitely many $x$ such that for all $y$ with $x < y \leq f(x)$, $h(y) = k$.

(b) A language $L$ outruns $f$ if its characteristic function $\chi_L$ outruns $f$.

Equivalently, each of the functions $h_k: x \mapsto \mu y (y > x \land h(y) \neq k)$ is infinitely often greater than $f$, where $k \in \text{Ran}(h)$. Technically this equivalence holds even when $|\text{Ran}(h)| = 1$, and allows us to extend the notion for functions $f$ which are not monotone increasing, but we do not devote special attention to either possibility. We view $h$ as a "coloring" of $\Gamma^*$ which produces long bands or "gaps" of each color.

LEMMA 3.2. Let $f: \Gamma^* \rightarrow \Gamma^*$ range witnesses to the assertion "$(\forall k \in K) (\forall l \in \mathbb{N}^+)[A_k \notin \mathcal{B}_{kl}]$," and suppose $h: \Gamma^* \rightarrow K$ outruns $f$. Then the language $E := \bigcup_{k \in K} (A_k \cap h^{-1}(k))$ is not in $\mathcal{L}_{KUT}$.

Proof. The clause "there are infinitely many $x$..." in Definition 3.3 serves only to guarantee that for any $k \in K$ and $l \in \mathbb{N}^+$ there exists $x > k, l$ such that $[x, f(x)] \subseteq h^{-1}(k)$. Taking the next-witness function $f_{kl}$ for any $k, l$ gives us $[x, f_{kl}(x)] \subseteq h^{-1}(k)$, and the rest follows via Lemma 3.1.

The force of our diagonalization mechanism comes from a uniform method for computing a function $h$ which outruns a given recursive function $f$, where the complexity of $h$ is intuitively minimal and independent of $f$. Here we distinguish between the cases $K := \mathbb{N}^+$ and $K := \{1, \ldots, m\}$, relating them to $2\mathcal{L}_x$ and $\mathcal{A}\mathcal{L}_x$. This is also the first place where we distinguish among simple, relativized, and uniformly relativized or "constructive" forms of results; while the first may be easier to state and use, we actually prove the last.

THEOREM 3.3a (Simple Form). Let $f: \Gamma^* \rightarrow \Gamma^*$ be a recursive function. Then there is a function $h \in 2\mathcal{L}_x$ from $\Gamma^*$ onto $\mathbb{N}^+$ which outruns $f$. Also, for any $m \in \mathbb{N}^+$, there is a function $h \in \mathcal{A}\mathcal{L}_x$ from $\Gamma^*$ to $\{1, \ldots, m\}$ which outruns $f$.

This result is well-behaved under relativizations. We say that a class $\mathcal{H}$ of functions outruns another function class $\mathcal{F}$ if $(\forall f \in \mathcal{F})(\exists h \in \mathcal{H})[h$ outruns $f]$. Writing $\mathcal{H}[\infty]$ or $\mathcal{H}[m]$ to single out those functions in $\mathcal{H}$ which have range $\mathbb{N}^+$ or range $\{1, \ldots, m\}$, respectively, lets us state

THEOREM 3.3b (Relativized Form). For any oracle set $A \subseteq \Gamma^*$, $2\mathcal{L}_x^{\infty}[A]$ outruns $\mathcal{F}\mathcal{A}\mathcal{C}^A$. For any $m \in \mathbb{N}^+$, $\mathcal{A}\mathcal{L}_x^{\infty}[m]$ outruns $\mathcal{F}\mathcal{A}\mathcal{C}^A$.

Still stronger is the statement that the function $h$ can be obtained from $f$ uniformly and effectively, i.e., by a single recursive procedure which takes
f as a parameter, and that moreover the procedure itself works relative to any oracle set. We state it in detail for the infinite case only, adding that h is rapidly decreasing:

**Theorem 3.3c (Constructive Form).** There is a primitive recursive function \( \sigma: \mathbb{N}^+ \to \mathbb{N}^+ \) such that for all \( i \in \mathbb{N}^+ \) and oracle sets \( A \); if \( T^A_i \) is total, then \( T^A_{\sigma(i)} \) computes a function from \( I^* \) onto \( \mathbb{N}^+ \) which outruns \( T^A_i(\cdot) \). Moreover, \( T^A_{\sigma(i)} \) runs in \( \log n \) space and real time + \( \log n \) extra steps on inputs \( x \in I^* \) of length \( n \), and \( |T^A_{\sigma(i)}(x)| \leq \log_2(|x| + 2) \) for all \( x \).

The extra steps are used solely to copy the value \( k := T^{A_{\sigma(i)}}(x) \) from a worktape to the output tape. In the analogous finite case, where the possible values are restricted to \( \{1, \ldots, m\} \), this means that no more than \( \log_2 m \) “extra steps” are needed to copy the value. Then \( T^{A_{\sigma(i)}} \) becomes an \( \mathbb{R}^L \)-machine.

The strategy this engenders is as follows: Given a specification of the requirements \( \{R_j\} \), where hypotheses or some “digging” as in Lemma 2.4 may provide the appropriate languages \( A_k \) and classes \( \mathbb{R}_k \), find a witness-ranging function \( f \). Show that \( f \) is recursive (or recursive in \( X \)). Use Theorem 3.3 to construct a relatively “easy” function \( h \) which outruns \( f \). Then Lemma 3.2 guarantees that the language \( E \) defined in terms of \( h \) and the \( \{A_k\} \) will meet all the requirements, while the reduction from \( E \) to \( A_{\omega} \), given by \( x \mapsto x \neq h(x) \) ensures that \( E \) is not overly complex.

The next section proves Theorem 3.3c in an even more detailed form. The reader who already accepts it may skip to Section 5.

---

### 4. The Delay Construction

In the following reformulation of Theorem 3.3c, we rename the transducer \( T_{\sigma(i)} \) under construction to “\( H \).”

**Theorem 4.1.** Given any oracle transducer \( T \), one can uniformly and effectively find a total oracle transducer \( H \) such that for any oracle set \( X \),

1. \( |H^X(x)| \leq \log_2(|x| + 2) \) for all \( x \in I^* \).
2. \( H^X(\cdot) \) runs in \( \log_2 n \) space and real time + \( \log_2 n \) extra steps. The extra steps are used only to copy \( H^X(x) \) from a worktape to the output tape.
3. If \( T^X \) is total, then \( H^X(\cdot) \) is onto \( \mathbb{N}^+ \) and outruns \( T^X(\cdot) \).
4. If \( T^X \) is not total, then \( H^X(\cdot) \) is a.e. constant.

Before presenting the proof, which also yields Theorems 3.3a and 3.3b for the finite case, we discuss its strategy. The general object is to get \( H \) to
change its value infinitely often, but to delay changing it for as long as needed, which is "until" \( T^X \) on some input \( x \) has halted. Let \( f \) be the function computed by \( T^X \). Then the first task is to keep \( H(\cdot) \) constant (say equal to \( k \)) on the interval \([x, f(x)]\).

For \( H \) on some input \( y \) (say \( x \approx y \leq f(x) \)) to "look ahead" to compute \( f(x) \) would be computationally excessive, since we desire \( H(y) \) to use as little time and space as possible. It does us no harm, however, to keep \( H(y) = k \) constant on some interval \([x, z] \) with \( z > f(x) \). In particular, let \( y_0 \) be the first input which is long enough to allow the whole computation of \( f(x) \) to finish within the time that \( y_0 \) is read. Then for inputs \( y \geq y_0 \), \( H \) has enough "time" to verify that the computation of \( f(x) \) has terminated, and to change its value accordingly. This idea often appeared in the literature under the sobriquet "looking back."

The second task is to save a string \( x' > f(x) \) with which to begin a new cycle, where \( H(x') \) must be different from \( k \). We can take \( x' := y_0 \) since \( H \) "knows" that \( y_0 > f(x) \), and since \( H(y_0) \neq k \). The problem is that after \( |y_0| \) steps on inputs \( y > y_0 \), \( H \) has only \( \log_2 |y_0| \) tape cells available, i.e., not enough to write \( |y_0| \) down. To save \( y_0 \) on tape for some future "time" \( y \), we introduce the idea of running a clock backwards, where the clock regulates the space allowed to \( H \) as a function of time. (Author's note: I do not know of a similar idea in the literature.)

The delay constructions of S. Breidtbart (1978) and D. Schmidt (1985) handle these tasks by having a TM acceptor \( M \) on input \( y \) "look back" at its own computations on strings in the interval \([x, f(x)]\), to ensure that it has kept a constant value (here, "accept" or "reject") throughout it. This self-reference is formally justified by an appeal to the Recursion Theorem, explicitly so in (Breidtbart, 1978). The difference from our work is intuitively that the machine \( M \) "knows" that it has stayed constant on \([x, f(x)]\) and changed to another value on \([x', f(x')]\), whereas \( H \) is constructed merely to bring this about. Especially since we shall derive consequences which are analogous to the Recursion Theorem, we consider it noteworthy that technique avoids such an appeal.

The clock itself is a routine for pushing a marker as slowly as possible down a semi-finite tape. A function \( s: \mathbb{N} \rightarrow \mathbb{N} \) is fully space constructible (Hopcroft and Ullman, 1979) if there is a TM \( M \) which uses \( s(n) \) tape cells (on a designated worktape) on all inputs of length \( n \). We add the qualifier "on-line" if \( M \) is required to run on-line, and so on.

**Lemma 4.2.** For any \( c > 0 \), there is a function \( s(n) \) which is fully space-constructible in real time, such that \( s(n) \leq c \cdot \log_2(n + 2) \) for all \( n \in \mathbb{N} \).

**Proof of Lemma 4.2.** We need only show that a fairly standard way of using a Turing machine \( L \) to count in binary has the desired properties.
Our $M$ has alphabet \{ $\wedge$, $\$, $\$, 0, 1, 2 \}, a single worktape which initially contains only "$\wedge$" in its leftmost cell, only one state $q$, and tuples of the form $(q, \text{read, write, move, } q)$ given by

$$M := \{( \wedge, \wedge, R), (\$, 0, R), (\$, L, 0, 1, L), (1, 2, L), (2, 0, R)\}.$$  

In the single infinite computation of $M$ on input "$\wedge$$\$", the "$\$" sign appears at the second step, and always occupies the rightmost nonblank tape cell, except when it is changed to "0" just prior to being moved rightward one cell. We leave the reader to check that at any step $n \geq 2$, the "$\$" occupies cell $r(n) := \max\{m \mid 2^m - m \leq n\}$. Using the standard linear tape-saving methods of (Hopcroft and Ullman, 1979), which involve expanding the tape alphabet, one can construct an appropriate $s(n) := c \cdot r(n)$ for any $c > 0$.

The growth rate of $\Theta[\log, n]$ is the best possible; a result of (Hopcroft and Ullman, 1969) implies that any function $s(\cdot)$ which is fully space-constructible on-line is either bounded above by a constant, or is bounded below by $c \cdot \log, n$ for some $c > 0$.

Armed with this for $c := \frac{1}{2}$, we use it to restrict a 5-worktape TM to $\log, n$ space.

**Proof of Theorem 4.1.** Given the oracle transducer $T$, we construct $H$ as follows. Besides the input and output tapes, $H$ has five worktapes: a clock tape, a backward clock tape, a function tape, the oracle tape, and a value tape. The alphabet for all five tapes contains the special endmarker symbol "$\$" $\notin \Gamma$, which $H$ prints on each at the first step. No worktape head is ever allowed to move to the right of its respective endmarker. This restricts the space available to $H$. Some states in the finite control of $H$ are labeled "value intended"; the rest are "value unintended."

$H$ reads one symbol of its input $y$ at each step until reaching the end. If the current state is labeled "value unintended," $H$ halts immediately, and so the value $H(y)$ is the empty string, which stands for 1. If it is "value intended," $H$ enters a routine which copies the string (over $\Sigma^*$) formed by the nonblank contents of the value tape onto the output tape, and then $H$ halts. Granting that $H$ is logspace-bounded, these stipulations ensure that $H$ runs in real time plus $\log, (|y| + 2)$ extra steps.

The input $y$ itself has no other significance; only its length matters. Thus we can view the actions of $H$ as a single infinite computation which is interrupted only by reaching the end of the input. For all $n \in \mathbb{N}^+$, we call the $n$th step of this computation "time $n$." Although $H$ is limited to $\log, (n + 2)$ space at any time $n$, in the long run $H$ has unbounded storage space.

The single computation is composed of infinitely many cycles $C_1, C_2, \ldots$.
each of which "represents" a different value $k_i \in \mathbb{N}^*$. During cycle $C_i$, we ensure that for a certain $x_i$, $H(y)$ takes the constant value $k_i$ on $[x_i, f(x_i)]$. It is not necessarily the case that $x_{i+1} = f(x_i)$, and the values taken by $H$ on $(f(x_i), x_{i+1})$ are immaterial. All we need is that every natural number appear infinitely often in the sequence $k_1, k_2, \ldots$

Each cycle begins in a special Ready configuration, and is composed of "tasks" in the sequence

$$\text{Ready} \rightarrow \text{Record-Time} \rightarrow \text{Evaluate-f} \rightarrow \text{Change-Value} \rightarrow \text{Prepare} \rightarrow \text{Ready},$$

beginning the next cycle. These tasks can be "interrupted" at any time by three routines: two called Update-Markers and Wait, and the one which prints the value when the end of the input is reached. Running in the background at all times is the process which space-constructs $L_{c \cdot r(n)}$ (with $c := \frac{1}{2}$, that is) in real time on the clock tape. We describe the interrupts first.

Update-Markers is invoked every time to clock tape moves its marker. The other worktape heads then mark their current positions and move rightward to their respective endmarkers. They move them one cell to the right, and then return to the marked positions, ending the task. Thus if each "$\$$" occupies cell $m$ at the beginning, then each "$\$$" occupies cell $m + 1$ at the end. Since the clock tape moves its "$\$$" with exponentially decreasing frequency, there is time for Update-Markers to finish before its next invocation after the first few steps of $H$.

The Wait routine is invoked every time the evaluation or oracle tape head moves onto its endmarker outside of Update-Markers. Until the clock tape head triggers another run of Update-Markers, the other heads remain stationary. Intuitively speaking, the Wait routine answers all requests for "more space" in the main tasks, freeing another cell before they proceed.

There are two "copies" of Update-Markers and Wait in $H$'s finite control, each for "intended" and "unintended" values. All of the above tasks are superseded whenever $H$ encounters the blank at the end of the input, whereupon the appropriate output routine is invoked.

A Ready configuration occurs when all five worktape heads are scanning the "$\$$" endmarkers on their respective tapes. This occurs for the first time at the second step. Now suppose Ready has occurred at some time $n_i$. As part of the supposition, the value tape currently holds the value $k_i$, and the current state labels this an "intended" value. The first cycle has $n_1 = 2$, $k_1 = \lambda$, and the appropriate state. The value $k_i$ remains on the tape and is "intended" until the end of the Evaluate-f task.
The Record-Time task is entered immediately after the (brief) run of Update-Markers which is triggered in Ready. Its objective is to transfer the string \( x_i := O^n \) to the evaluation tape. At the first step of Record-Time, the evaluation tape head is scanning a newly-freed cell; it then prints a "\( \wedge \)" sign there and moves right onto its endmarker again. Meanwhile the head on the backward-clock tape prints a "\( \ell \)" in the cell it has freed, which equals cell \( r(n_i) \) numbered from the left. The intent is to count up to \( n_i \) by imitating the actions of the clock tape to push the "\( \ell \)" backwards to the first cell.

This counting need not be done in real time; rather, it advances one step only when there is a freshly freed cell on the evaluation tape. The sequence runs \( \text{Wait} \rightarrow \text{Update-Markers} \rightarrow \{ \text{carry out one step of the clock routine on the backward-clock tape, print "0" on the evaluation tape, move head right to trigger Wait again} \} \). It repeats until the "\( \ell \)" reaches the leftmost cell and the head visits it a second time, when the clock routine would call for pushing the marker another cell. At this point there are exactly \( n_i \) 0's following the last "\( \wedge \)" on the evaluation tape, because the configuration at time \( n_i \) (namely Ready) called for the clock tape to push the "\( \$ \)" rightward. Except for the constant running of the clock tape and the induced moving of the "\( \$ \)" markers, nothing else has happened on any of the tapes. Then Record-Time finishes.

\( H \) then sets about the task of evaluating \( f(O^n) \), i.e., \( T^X(O^n) \). By well-known means one may effectively transform the OTM \( T \) into \( T' \) which simulates \( T \) using only one tape besides the oracle query tape. For \( T' \) we suppose that the input \( x \) initially appears left-justified on the former tape, and also that all halting computations end with just the value \( T'^X(x) \) on this tape. Oracle queries are written and handled just as on the query tape of \( T \). Then we may place the instructions for \( T' \) directly into the finite control of \( H \) (with "value intended" labels and other slight modifications). Whenever \( T' \) asks for a tape cell which \( H \) cannot yet provide, the corresponding tape head moves onto a "\( \$ \)" endmarker, triggering the Wait/Update-Markers routine. This freezes the simulation until the tape cell becomes available. Note that \( T, T', \) and \( H \) all make the same oracle queries for any given input \( x \) and oracle set \( X \). Granting that \( T^X(\cdot) \) is total, this task always halts.

Since we have made \( H \) physically write \( f(x_i) = f(O^n) \) down, the time \( n' \) at which the task finishes is greater than \( |f(O^n)| \) (in fact, \( n' \) is much greater than this). Hence \( 0^n > f(O^n) \). Since we have left the contents of the value tape unchanged, we have ensured that \( H \) takes the constant value \( k_i \) on \( [x_i, f(x_i)] \), i.e., on \( [O^n, f(O^n)] \). Since \( n_{i+1} \) will be \( > n' \), this also ensures that taking \( x_{i+1} := 0^{n+1} \) for the next cycle gives \( f(x_i) < x_{i+1} \).

Upon the termination of Evaluate-\( f \), control passes into states labeled, "value unintended" for the Change-Value and Prepare tasks. The result
$H(y)$ is actually immaterial when $f(x_i) < y < x_{i+1}$, and since there is no upper bound on the choice of $x_{i+1}$, $H$ is figuratively welcome to take as long as it likes before generating the next value $k_{i+1}$ on the value tape. If writing $k_{i+1}$ down takes more space than currently available, $H$ simply waits.

Whenever $k_{i+1}$ is ready, $H$ invokes Prepare. In Prepare, $H$ re-blanks the backward-clock tape and oracle tape, while moving the heads on all but the clock tape to their right endmarkers. There they wait until the clock tape head again reaches its marker. This last action sends $H$ into a “value-intended” state and produces another Ready configuration. Let $n_{i+1}$ be the “time” at which this occurs; then the next cycle $C_i+1$ begins with $x_{i+1} := 0^{n_{i+1}}$.

By the construction, and since every $k \in \mathbb{N}^+$ appears infinitely often in $[k_i]_{i=1}^{\infty}$, $H^X(\cdot)$ outruns $T^X(\cdot)$, for any $X$ making $T^X$ total. If $T^X(x)$ diverges, then the Evaluate-f task on argument $x$ never ends, making $H(\cdot)$ a.e.-constant. To ensure that $H(\cdot)$ is a.e.-constant whenever $T^X$ is not total, have $H$ run Evaluate-f on all $y < x$ as well. Clearly the code of $H$ depends uniformly and effectively on that for $T$.

This also finishes the proof of Theorems 3.3(a–c).

**Corollary 4.3.** There is a single OTM $H$ such that for any total function $f: \Sigma^* \to \Sigma^*$, there is an oracle $A \subseteq \Sigma^*$ such that $H^A$ is a $2\mathcal{L}$-machine, and $H^A(\cdot)$ is onto $\mathbb{N}^+$ and outruns $f$.

**Proof.** It is straightforward to design an OTM $T$ which computes any total function $f$ with oracle set $A_f := \{ <x, w> | w \subseteq f(x) \}$, where $<\cdot, \cdot>$ is some recursive pairing function. Then apply the above construction to $T$.

Although we shall show that this construction gives rise to a uniform, general treatment of many diagonalization results in the literature, it is not in any sense the ultimate refinement of the technique. In the rest of this section we discuss three possible avenues to more powerful results.

### 4.1. Technical Improvements

It is possible to bound both the ratio $|H(x)|/|x|$ and the number of “extra” steps taken by $H$ by some function $s(n)$ which is strictly less than $O(\log n)$. In fact, $s(\cdot)$ can be any unbounded nondecreasing recursive function. This is done by reworking the construction so that Update-Markers leaves the value tape endmarker unmoved, adding a special routine for moving it whenever the value tape head “requests” more space, and using a very slowly growing recursive enumeration $[k_i]_{i=1}^{\infty}$ of $\mathbb{N}^+$. Although $H$
runs on-line, and the function \( r(n) \) giving the number of value tape cells used is strictly \( o(\log n) \), this does not contradict the note following Lemma 4.2, because log space is needed on some other tape to enumerate the values \( k_j \). It is not possible to bound the space used by \( H \) by an \( o(\log n) \) function.

With reference to the notation in Section 3, we have taken \( k := \mathbb{N}^+ \) in Theorem 4.1c. The construction works for any r.e. set \( K \), where for various technical reasons we should assume that \( K \) contains 1 (i.e., \( \lambda \in K \)). To modify it, we can use the evaluation tape to construct a recursive enumeration \( \{k_i\}_{i=1}^\infty \) of \( K \) in which each element appears infinitely often, introducing the next value \( k_{i+1} \) when space becomes available for it. In fact, \( K \) need only be r.e. in \( X \), and we can effectively parametrize Theorem 4.1c in two variables: \( i \) for \( T_i \), and \( k \) for \( K := L(M_f^X) \cup \{\lambda\} \).

We can also reduce the number of tapes needed for \( H \) by storing the value on a “track” of the evaluation tape, and by merging the clock tape and the backward-clock tape. The latter action is possible because the clock tape head visits every cell in between movements of its endmarker, so that it can carry out one step of the backward clock on each pass during the Record-Time task. We conjecture, however, that any TM filling the function of our \( H \) needs two worktapes (in addition to the oracle tape).

4.2. Extensions

There is considerable slack in the delay construction, represented by the intervals between successive cycles. It may be possible to use these to satisfy other “requirements” besides the diagonalization objectives. Recalling the definition of \( E \) by \( \bigcup_{i=1}^\infty (h^{-1}(k) \cap A_k) \) in Lemma 3.1, we have for a simple instance

**Proposition 4.4.** With reference to Lemma 3.1, let \( \{B_1, ..., B_r\} \) be any finite collection of languages which does not contain all of \( \{A_k \mid k \in K\} \). Then in we can arrange that \( E \neq B_l \) for all \( l, 1 \leq l \leq r \).

**Proof.** At least one of the languages \( A_k \) is different from each \( B_l \). Then \( (\exists n)(\forall l)(\exists x)[|x| \leq n \land x \in A_k \land x \notin B_l] \). In constructing \( H \), make the “intended” value equal to \( k \) for at least the first \( n \) steps. This can be done in \( H \)'s finite control rather than using the value tape. Then start the first cycle from the first Ready configuration after step \( n \), rather than at step 2.  

We use this trick in Theorem 7.1. Note that while \( E \) is still \((X-)\)recursively, the construction may no longer be effective if some particular \( k \) giving \( A_k \neq B_l \) for all \( l \) cannot be computed in advance.

We speculate on a possible deeper use of the intervals between cycles in
the case of diagonalization over two classes, when \( h: \Gamma^* \to \{1, 2\} \) is essentially a language. Suppose \( A \) and \( B \) are \( \mathcal{NP} \)-complete and \( f: \Sigma^* \to \Sigma^* \) is given. Can one construct a language \( D \) which outruns \( f \) so that the "splice" language \( E := (A \cap D) \cup (B \cap \overline{D}) \) is also \( \mathcal{NP} \)-complete? We ask the analogous question when \( A, B \) are \( p \)-isomorphic to SAT, and envision using the intervals between cycles to tie "loose ends" in patching the isomorphisms \( p_A, p_B \) sending \( A, B \) onto SAT into a single polynomial isomorphism sending the derived language \( E \) onto SAT. By analogy with the numerical expression \( e = ad + b(1 - d) \), positive answers would tell us that the \( \mathcal{NP} \)-complete sets and the sets which are \( p \)-isomorphic to SAT are "convex" in terms of complexity, and simplify results in (Regan, 1988).

It may also be possible to tighten the construction by a more delicate clocking mechanism, leaving qualitatively less slack. This may render it applicable in cases where specific ranges \([x, f(x)]\) have to be colored a certain color, replacing the proviso that \([x, f(x)]\) receives the color for infinitely many (hence arbitrarily large) \( x \).

4.3. Non-machine Formulations

Consider the case of out-running by languages (i.e., with \( K := \{1, 2\} \)), in which we have shown that \( \mathsf{RL} \) outruns \( \mathsf{FA}^\mathsf{SC} \). Now let \( C \) be a class of languages which is closed under splices by sets in \( \mathsf{RL} \). Then every non-trivial \( \equiv \ell \)-invariant property of languages in \( C \) is undecidable, regardless of the strength of the sound, i.e., formal system \( \mathcal{F} \) in question. By contrast, finiteness is decidable for context-free languages. (See, respectively Regan, 1988, and Hopcroft and Ullman, 1979). Thus we say that \( \mathsf{RL} \), which is basically the minimum class requiring unbounded amounts of time and space, is past the "threshold" for all nontrivial properties of member languages to be undecidable.

Is there a more acute formulation of this "threshold," one which does not refer to particular complexity measures, and which better captures the significance of outrunning all recursive function? We envision a characterization which is either purely "language-theoretic", or which involves first-order definability. Regarding the latter, the class \( \mathbf{C} \) introduced in (Immerman, 1983) and extended in (Immerman, 1987, 1989) seems a plausible candidate, since it is properly contained in \( \mathbf{LOGSPACE} \), but we have not been able to show that it outruns \( \mathbf{FA}^\mathsf{SC} \).

5. Uniform Diagonalization Theorems

In Section 3 we observed that a witness-ranging function \( f \) for \( [A_k] \) with respect to \( [\mathcal{B}_{kl}] \), where each \( \mathcal{B}_{kl} \) is closed in \( \mathcal{I} \), exists if and only if
A_k \cap \mathscr{B}_{kl} = \emptyset \text{ for all } k \text{ and } l. \text{ In Section 4 we gave a relativizable technique for outrunning a given recursive function } f. \text{ It remains to find conditions on } [A_k] \text{ and } [\mathscr{B}_{kl}] \text{ which allow one to compute a witness-ranging function } f \text{ recursively.}

It would be nice to say that the effectiveness of } f \text{ depends only on the effectiveness of the presentation of the languages } A_k \text{ and the classes } \mathscr{B}_{kl}. \text{ However, the dependence is actually on the functions } g: \{0, 1\}^* \to \{0, 1\}^* \text{ from Definition 3.1, which witness the nowhere-denseness of each } \mathscr{B}_{kl} \text{ with } A_k. \text{ Not only might there be no recursive } g(\cdot) \text{ for some } A_k \text{ and } \mathscr{B}_{kl}—\text{ indeed, the nowhere-denseness of } \mathscr{B}_{kl} \text{ might not be effective at all—but even if some such } g_{kl} \text{ exists for all } k \text{ and } l, \text{ the collection } \{g_{kl}\} \text{ may not be uniform in the same way that the enumerations } [A_k] \text{ and } [\mathscr{B}_{kl}] \text{ are.}

Nevertheless, these potential obstacles vanish when each } \mathscr{B}_{kl} \text{ consists of a single language } C_{kl}. \text{ We devote the rest of this section to this special case. Using the projection notation of Section 2.2, we suppose the } \{C_{kl}\} \text{ are presented via a single language } C \subseteq \Gamma^* \text{ such that for all } k \in K \text{ and } l \in \mathbb{N}^+, \text{ } C_{kl} = (C_k)_l = \{x | x \neq l \in C_k\} = \{x | x \neq l \neq k \in C\}. \text{ Then we define } \mathcal{C}_k := \{C_{kl} | l \in \mathbb{N}^+\} \text{ for each } k. \text{ In the other direction, we say an indexing } [\mathcal{C}_k]_{k \in K} \text{ is } (X-) \text{ recursively presented if we have an } (X-) \text{ recursive language } C \text{ such that } C_k = \mathcal{P}_x[C_k] \text{ for all } k, \text{ and if } K \subseteq X. \text{ The latter holds when } K := \mathbb{N}^+ \text{ or } K := \{1, \ldots, m\}, \text{ of course; and for simplicity we take } K := \mathbb{N}^+ \text{ in what follows.}

**Lemma 5.1a (Simple Form).** Let } A \text{ and } C \text{ be recursive, and suppose that for all } k \in \mathbb{N}^+, \text{ } A_k \cap C_k = \emptyset, \text{ where } \mathcal{C}_k := \mathcal{P}_x[C_k] \text{ as above. Then the least witness-ranging function } f \text{ for the assertion "}(\forall k)[A_k \notin \mathcal{C}_k]\text{" is recursive.}

We note that if any witness-ranging function for "}(\forall k)[A_k \notin \mathcal{C}_k]\text{" is recursive, then so is the least one, because it is decidable for any given } k, \text{ } l, \text{ and } y \text{ whether } y \in A_k \triangle C_{kl}. \text{ In the general case, referring to Definitions 3.1 and 3.2, it is not decidable whether } \mathcal{C}_{y/b} \cap \mathscr{B}_{kl} = \emptyset \text{ even when } k, \text{ } l, \text{ } y, \text{ and } \beta \text{ are fixed, and even when recursive witness function } g_{kl} \text{ are provided. After stating the relativized form, we prove a stronger statement.}

**Lemma 5.1b (Relativized Form).** For any oracle set } X, \text{ Lemma 5.1a holds if "recursive" is replaced by "X-recursive" throughout.}

**Lemma 5.1c (Constructive Form).** There is a primitive recursive function } \tau: \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+ \text{ such that for all } a, c \in \mathbb{N}^+ \text{ and oracle sets } X \subseteq \Gamma^*, \text{ if } M^X_a \text{ and } M^X_c \text{ are total, and taking } A := L(M^X_a), \text{ } C := L(M^X_c) \text{ satisfies } A_k \cap \mathcal{C}_k = \emptyset \text{ for each } k, \text{ then } T^X_{(a,c)} \text{ is total and computes the least witness-ranging function } f \text{ for the assertion "}(\forall k)[A_k \notin \mathcal{C}_k]\text{."}
Proof. Given \( a \) and \( c \), let \( \tau(a, c) \) be the index of a transducer \( T \) which does the following with any oracle set \( X \), on any input \( x \):

\[
\begin{align*}
z &:= \lambda \\
&\text{for } k := 1 \text{ to } \text{num}(x) \text{ do:} \\
&\quad \text{for } l := 1 \text{ to } \text{num}(x) \text{ do:} \\
&\qquad \text{for } y := x \text{ to } \infty \text{ do:} \\
&\qquad \quad \text{Simulate } M^X_a(y \neq k) \text{ and } M^X_c(y \neq l \neq k); \\
&\qquad \quad \text{If (one machine accepts and the other rejects) and } y > z \\
&\qquad \quad \text{then do: let } z := y, \text{ exit } y\text{-loop, :od; } \\
&\qquad \quad \text{else goto next } y; \\
&\quad \text{endif,} \\
&\text{next } y. \\
&\text{next } l. \\
&\text{next } k. \\
&\text{output}(z).
\end{align*}
\]

By the totality of \( M_a \) and \( M_c \) with oracle \( X \), the "simulate" step always terminates, and since the assumptions imply that \( |A_k \triangle C_k| \) is infinite for all \( k, l \in \mathbb{N}^+ \), so does the unbounded loop on \( y \). Hence \( T^X \) is total. For all \( x \in \Gamma^* \), the interval \([x, T^X(x)]\) contains witnesses to "\( A_k \neq C_k \)" for all \( k, l \leq x \). Let \( k, l \) be the indices for which the value of \( z \) was last changed; then \( T^X(x) = z \) is the least string \( \geq x \) such that \( z \in A_k \triangle C_k \). Hence \( T^X \) computes the least witness-ranging function.

The code of \( T \) strings together the codes of \( M_a \) and \( M_c \) in an elementary way, from which one can see that \( \tau(\cdot, \cdot) \) is primitive recursive.

The result we have been building up to extends the main theorem of U. Schöning (1982), sometimes referred to as "the uniform diagonalization theorem." Schöning's theorem proper has \( K := \{1, 2\} \), uses no oracle \( X \), supposes also that \( A_1 \in \mathcal{P} \) and that \( A_2 \) is neither \( \varnothing \) nor \( \Gamma^* \), and concludes \( E \leq_m A_2 \). This follows directly from our statement because \( \mathcal{L}_x \subseteq \mathcal{P} \mathcal{P} \).

Recall that \( A_{\omega} := \{x \neq k \mid x \in A_k\} \).

**Theorem 5.2(a) (Simple Form).** Let \( [A_k]_{k=1}^c \) and \( [C_k]_{k=1}^c \) be recursive presentations of recursive languages and r.p. c.f.v. classes, respectively. Suppose \( \forall k, A_k \neq C_k \). Then there exists a recursive language \( E \) such that \( E \notin \bigcup_{k=1}^c C_k \), and yet \( E \leq_n A_{\omega} \).

**Theorem 5.2(b) (Relativized Form).** Let \( A \) and \( C \) be recursive in a given oracle set \( X \). Suppose that \( A_k \cap C_k = \varnothing \) for each \( k \), where \( C_k := \mathcal{P}_1[C_k] \) as above. Then there exists an \( X \)-recursive language \( E \) such that \( E \notin \bigcup_{k=1}^c C_k \), and yet \( E \leq_{n}^{x} A_{\omega} \).
Again we prove a stronger statement involving the codes of oracle Turing machines.

**Theorem 5.2(c) (Constructive Form).** There is a primitive recursive function \( e: \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+ \) such that for all oracles \( X \) and \( a, c \in \mathbb{N}^+ \), \( M_{i(a,c)}^X \) is total, and if \( M_{i(a,c)}^X \) and \( M_{i(a,c)}^X \) are total, and taking \( A := L(M_{i(a,c)}^X) \), \( C := L(M_{i(a,c)}^X) \) makes \( A_{i(a,c)} \cap \mathbb{N}^+ = \emptyset \) for all \( k \), then \( M_{i(a,c)}^X \) accepts a language \( E \) such that \( E \leq \mathbb{N}_i^{i(a,c)} A \).

**Proof.** Given \((a, c)\), and with reference to the functions \( \sigma \) and \( \tau \) used in Theorem 3.3c and Lemma 5.1c, let \( d := \sigma(\tau(a, c)) \). By Theorem 3.3(c), \( T_d \) is total as an OTM. Now define \( e(a, c) \) to be the index of an OTM which with any oracle set \( X \) and input \( x \) first simulates \( T_{i(a,c)}^X(x) \). If \( k \) is the value returned, \( M_{i(a,c)}^X(x) \) then simulates \( M_{i(a,c)}^X(x \neq k) \), accepting iff this accepts. Since \( \sigma \) and \( \tau \) are primitive recursive, so is the function \( e(\cdot, \cdot) \).

By the hypotheses and the foregoing results, \( T_{i(a,c)}^X(x) \) computes a total function \( h \) which outruns the least witness-ranging function for \((\forall k \in \mathbb{N}^+) [A_k \notin \mathbb{N}^+] \)." Given that \( M_{i(a,c)}^X \) is also total, the language accepted by \( M_{i(a,c)}^X \) equals \( \bigcup_{k=1}^{2^l} (h^{-1}(k) \cap L(M_{i(a,c)}^X)) \). This is the same as \( E \) in Lemma 3.2, from which it follows that \( E \leq C_{kl} \) for all \( k \) and \( l \), so \( E \leq \mathbb{N}_i^{i(a,c)} \).

It remains to show that \( E \leq \mathbb{N}_i^{i(a,c)} L(M_{i(a,c)}^X) = A \). The mapping \( g: x \mapsto x \neq h(x) \) is clearly \( 1-1 \) and accomplishes the reduction. By Theorem 3.3(c) (or Theorem 4.1), not only is \( h \in 2L_\mathbb{N}^X \) with range \( \mathbb{N}^+ \), but also \( |h(x)| \leq \log_2(|x| + 2) \) for all \( x \in \mathbb{N}^+ \). Hence \( g(\cdot) \) is computable in log space and real time + log-many extra steps. To compute \( g^{-1}(y) \), where \( y \) arises as \( x \neq h(x) \) for some \( x \), is not so straightforward for a \( 2L_\mathbb{N} \)-machine \( T \) because \( x \) itself may contain "#" signs. However, upon reading a "#" at any step \( n \), \( T \) can buffer the next \( \log_2(n + 2) \)-many bits which follow on a worktape. If the blank endmarker appears, signaling that these bits are the "\( h(x) \)" part be dropped, \( T \) just halts. Else, \( T \) copies the buffer to the output and continues copying the input (via the buffer) until encountering the next "#." (Note that this inversion process does not require using the oracle \( X \).) Last, \( g(\cdot) \) is also strictly length-increasing, so \( E \leq \mathbb{N}_i^{i(a,c)} A \).

As indicated in Theorem 4.2(d), one can make \( T_{i(a,c)}^X(\cdot) \) a.e.-constant whenever the hypotheses of Theorem 5.2 fail for the given oracle set \( X \). Then \( L(M_{i(a,c)}^X) \) becomes a finite variation of one of the languages \( A_k \), though it is not possible in general to determine which one recursively. There is some connection to the open problem which will be raised in Section 7 as to whether certain "a.e. fixed points" can be computed. We do not use this extra feature in any results in this paper.

Pursuing remarks in Subsection 4.1, one can do this for choices of \( K \) other than \( K := \mathbb{N}^+ \), and even incorporate \( K \) as a parameter in the
construction, making $\varepsilon$ a function of $a$, $b$, and $c$ where $K := L(M_k)$. In the finite case $K := \{1, \ldots, m\}$, one obtain an $\mathcal{RL}_\varphi$-reduction rather than a $\mathcal{DL}_\varphi$-reduction. Here we only state the simple form.

**Theorem 5.3.** Let $\mathcal{C}_1, \ldots, \mathcal{C}_m$ be r.p. cfv. classes, and let $A_1, \ldots, A_m$ be recursive languages such that $A_k \notin \mathcal{C}_k$ for each $k$, $1 \leq k \leq m$. Then there is a language $E$ such that $E \notin \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_m$ and yet $E \leq^*_{\mathcal{NL}} \{ x \neq k \mid x \in A_k \}$.

**Proof.** With $K := \{1, \ldots, m\}$, Theorem 3.3a gives us a function $h \in \mathcal{RL}_\varphi$ from $\Gamma^*$ onto $K$ which outruns the least witness-ranging function $f$ for $(\forall k) [A_k \notin \mathcal{C}_k]$, whose recursiveness is assured by Lemma 5.1a. Since $\text{Ran}(h)$ is finite, the map $x \mapsto x \neq h(x)$ is in $\mathcal{RL}_\varphi$, and by arguments similar to the last proof, it is $\mathcal{RL}_\varphi$-invertible as well as length-increasing. The language $E := \bigcup_{k=1}^m (A_k \cap h^{-1}(k))$ reduces to $(x \neq k \mid x \in A_k)$ by this map, and lies outside each $\mathcal{C}_k$ by Lemma 3.2.

By some finite recoding one can obtain $E \leq^*_{\mathcal{NL}} (A_1 \oplus \cdots \oplus A_m)$ under some rule for defining iterated joins. This is so even if, supposing $\Gamma \supseteq \{0, 1\}$, one redefines the join $A \oplus B$ to be $\{ x0 \mid x \in A \} \cup \{ y1 \mid y \in B \}$. The only important point here is that the "decision bit" 0 or 1 must be placed at the end, not in front, for the reduction to be possible in real time.

**5.1. Applications**

We begin with some illustrations for the finite case. The first is well-known and is also cited in (Schöning, 1982), where one may find demonstrations that a number of the classes below are recursively presentable. Others are left to the reader, with the general idea being that $\mathcal{C}$ is r.p. iff $\mathcal{C}$ has a $\Sigma_2$ (i.e., "\exists^*\forall") definition in arithmetic; for precise characterizations, see (Regan, 1988).

**Ladner's Theorem** (Ladner, 1975). If $\mathcal{NP} \neq \mathcal{P}$, then there exist languages $E \in \mathcal{NP}$ which are neither $\mathcal{NP}$-complete nor in $\mathcal{P}$.

**Proof.** Take $\mathcal{C}_1 := \mathcal{P}$, $A_1 := \text{SAT}$, $\mathcal{C}_2 := \{ \mathcal{NP}\text{-complete sets}\}$, $A_2 := \emptyset$. Assuming $\mathcal{NP} \neq \mathcal{P}$ these choices satisfy the hypotheses of Theorem 5.3, which gives us $E \notin \mathcal{C}_1 \cup \mathcal{C}_2$ such that $E \leq^*_{\mathcal{NL}} \text{SAT} \oplus \emptyset$, so $E \leq^*_{\mathcal{P}} \text{SAT}$, so $E \in \mathcal{NP}$.

Perhaps not so well known is the uniform, relativized enhancement of this which Theorem 5.2c provides. It requires the fact (see Baker, Gill, and Solovay, 1975) that there is a single total OTM $Z$ such that $L(Z^X)$ is $\mathcal{NP}^X$-complete under $\leq^*_{\mathcal{P}}$ for any oracle $X$. To make the result stronger, we relativize the reducibility in defining $\mathcal{NP}^X := \{ A \in \mathcal{NP}^X \mid L(Z^X) \leq^*_{\mathcal{P}} A \}$ for all $X \in \Gamma^*$. 
Theorem 5.4. There is a single total OTM $M_a$ such that whenever $B$ is an oracle making $\mathcal{N}^B \neq \mathcal{P}^B$, $L(M^B_a) \in \mathcal{N}^B \setminus (\mathcal{P}^B \cup \mathcal{NP}^B)$.

Proof. There are total OTM's $M_1$ and $M_2$ such that for all oracles $X$, $\mathcal{P}[L(M_1^X)] = \mathcal{P}^X$, and $\mathcal{P}[L(M_2^X)] = \mathcal{NP}^X$; the latter is obtainable from the OTM $Z$ given above. (That is, $\mathcal{P}^X$, $\mathcal{NP}^X$, and $\mathcal{NP}^X$ are all recursively presentable as relativized classes.) Taking $K := \{1, 2\}$ and putting $M_1$ and $M_2$ together gives the OTM $M_r$ required in Theorem 5.2c, and $M_r$ can be a similar amalgam of the OTM $Z$ and an OTM which always accepts the empty set. All conditions in Theorem 5.2c are then met, and the required machine $M_e$ comes out. $\blacksquare$

The next result is a slight variation of one in (Schöning, 1982). A relativized formulation akin the Theorem 5.4 is possible; we leave it to the reader.

Theorem 5.5. Suppose the polynomial hierarchy $\mathcal{PH}$ neither collapses to $\mathcal{P}$ nor is equal to $\mathcal{P} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E}$. Then there are languages $E$ in $\mathcal{P} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E}$ which are neither $\mathcal{NP}^E$-hard nor in the polynomial hierarchy.

Proof. Letting QBF denote the known $\mathcal{P} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E}$-complete language of quantified Boolean formulas, take

$$\begin{align*}
\mathcal{C}_1 & := \{ L \in \mathcal{P} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E} \mid L \text{ is } \mathcal{NP} \text{-hard} \} \\
\mathcal{C}_2 & := \mathcal{PH} \\
A_1 & := \emptyset \\
A_2 & := \text{QBF}.
\end{align*}$$

The assumptions allow Theorem 5.3 to operate, yielding $E \notin \mathcal{C}_1 \cup \mathcal{C}_2$ such that $E \leq^p \text{QBF}$. Thus $E \in \mathcal{P} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E}$, and since $E \notin \mathcal{C}_1$, $E$ is not $\mathcal{NP}^E$-hard. $\blacksquare$

S. Breidbart's (1978) original delay construction was dedicated to the following result. It is also used as an illustration in (Schmidt, 1985).

Breidbart's Splitting Theorem (Breidbart, 1978). Let $A$ be any recursive language such that $A$ and $\overline{A}$ are both infinite. Then there exists $E \in DTISP[n + 1, \log_2 n]$ such that $(A \cap E)$, $(\overline{A} \cap E)$, $(A \cap \overline{E})$, and $(\overline{A} \cap \overline{E})$ are all infinite.

Proof. Define

$$\begin{align*}
\mathcal{C}_1 & := \{ B \in \mathcal{RL} \mid (\exists n \in \mathbb{N}) : |A \cap B| \leq n \} \\
\mathcal{C}_2 & := \{ B \in \mathcal{RL} \mid (\exists n \in \mathbb{N}) : |A \cap \overline{B}| \leq n \} \\
\mathcal{C}_3 & := \{ B \in \mathcal{RL} \mid (\exists n \in \mathbb{N}) : |\overline{A} \cap B| \leq n \} \\
\mathcal{C}_4 & := \{ B \in \mathcal{RL} \mid (\exists n \in \mathbb{N}) : |\overline{A} \cap \overline{B}| \leq n \} \\
A_1 & := \mathcal{I}^* \\
A_2 & := \emptyset \\
A_3 & := \mathcal{I}^* \\
A_4 & := \emptyset.
\end{align*}$$
Since $RL$ is r.p. (enumerate $RL$-machines), each $C_k$ is r.p. as well as $cfv$. That $A$ is neither finite nor co-finite gives $A_k \notin C_k$ for each $k$. We obtain $E \leq_{h}^{*} (\emptyset \oplus I^*)$, so $E \in RL$, and $E \notin \bigcup_{k=1}^{n} C_k$ does the rest.

We show a further use for the real-time refinement by improving a result of S. Even, T. Long, and Y. Yacobi (1982). The authors only obtained that $A$ and $B$ below were polynomially related in deterministic complexity, because they used a pairing function which is not known to be computable in linear time. Although "the" complexity of a language is not generally well-defined, owing to Blum's Speed-Up Theorem (Blum, 1967), one can define the relative notion by saying that for any algorithm $M_a$ accepting $A$ there is an $M_b$ for $B$ which has the same running time up to a (linear, polynomial) factor, and vice-versa.

Theorem 5.6. Suppose $NP \neq co-NP$. Then there are languages $A$ and $B$ in $NP$ whose deterministic time complexities are linearly related, but whose nondeterministic time complexities are not even polynomial related.

The result is not just a matter of taking $A := SAT$, say, and $B := \overline{A}$, because the authors consider the nondeterministic complexities of $A$ and $\overline{A}$ to be the same. Nor can one take $B := A \oplus \overline{A}$ and observe that $A \notin \Delta \cap \overline{\Delta}$, because an instance $x$ to $A$ corresponds to instances $x0, x1$ (or $x \neq 0, x \neq 1$) to $B$ which do not have the same length. Though an algorithm $M_a$ for $A$ yields an algorithm $M_b$ for $B$ such that $t_a(n) = t_b(n+1)$ for all $n$, it is possible that the functions $\lambda n \cdot t_a(n)$ and $\lambda n \cdot t_b(n)$ are not even polynomially related (as pointed out to this author by T. Long). What is needed is a language $B$ whose instances are not compound, but which have the same "type" as those to $A$.

Proof. One can take $A$ to be any language in $NP \cap co-NP$. Then take $C_1 := NP, A_1 := \sim A, C_2 := co-NP, A_2 := A$. The resulting language $E$ equals $(A \cap D) \cup (\overline{A} \cap \overline{D})$ for some $D \in RL$, where $D$ is obtained as $h^{-1}(1)$ for some $h \in RL$ and that $D \in RL$ follows by Proposition 2.1. Then $E$ and $A$ are deterministically linear-time equivalent. Since $E \leq_{m}^{*} (A \oplus \overline{A}), E \in RL \setminus (NP \cup co-NP)$. Then $A$ and $B := E$ are the required languages.

We end with one illustration of the infinite case ($K := \mathbb{N}^+$), leaving the main application to Section 7. It shows how incomparabilities "proliferate" in reducibility structures. It holds for any effective reducibility $\leq$, such that $\equiv$, $\sqsupseteq$, $\equiv$ in place of $\leq_{m}^{*}$. Recall $D_{o} := \{x \neq k \mid x \in D_{k}\}$.

Theorem 5.7. Let $[C_k]_{k=1}^{\infty}$ and $[D_k]_{k=1}^{\infty}$ be recursive enumerations of recursive languages such that for each $k$, $C_k \equiv_{m} D_k$. Then there is a language $E$ such that $E \leq_{m}^{*} D_{o}$ (so $E \leq_{m}^{*} D_{o}$) and $C_k \equiv_{m} E$ for all $k$. 
Proof. Referring to Theorem 5.2a with $K := N^+$, take $A_k := D_k$ and $\mathcal{C}_k := \{L | L \leq_m^P C_k \vee C_k \leq_m^P L\} \cap \{L | L \leq_m^P D_k\}$ for each $k$. The last intersection serves only to make $\mathcal{C}_k$ a bounded class (while keeping it large enough), so that $[\mathcal{C}_k]_{\kappa - 1}$ is a recursively presented sequence of r.p. classes. Since $\equiv_m^P \supseteq \equiv^f$, no finite variation of $A_k$ is in $\mathcal{C}_k$, for any $k$. Hence Theorem 5.2a applies. We obtain $F$ such that $F \leq_m^P D_w$ and $F \not\in \bigcup_{\kappa - 1} \mathcal{C}_k$. Since $\leq_m^P \supseteq \leq_m^q$, $E \leq_m^P D_w$, and so for all $k$, $E \leq_m^P C_k$ and $C_k \leq_m^P E$.

If we also throw in $\mathcal{C}_k := \{L \leq_m^P C_k | L \sim_m^P D_k\}$ and $A_k := C_k$ for all $k$, then we obtain a language $E \leq_m^P (C_w \oplus D_w)$ which is incomparable with all of the languages $C_k$ and $D_k$. Looking in the abstract, from $(\forall C_k)(\exists D_k)[C_k$ and $D_k$ are incomparable] we have inferred $(\exists E)(\forall C_k, D_k)[E$ is incomparable with both $C_k$ and $D_k]$, where the complexity of $E$ is not too great.

The general phenomenon of being able to replace "$\forall \exists$" quantifiers by "$\exists \forall$" is often called uniformity. The next section presents an application in which the closed classes $B$ being diagonalized over are not all singletons, and which resembles the uniform boundedness theorem of classical analysis.

6. A Uniform Reduction Theorem

J. Grollmann and A. Selman (1984) proved that if a promise problem is $\mathcal{NP}$-hard, then it is uniformly $\mathcal{NP}$-hard. They define a promise problem to have the general form "Given $x \in Q$, is $x \in R$?", where $Q \subseteq \Gamma^*$ is the promise set and $R \subseteq \Gamma^*$ is the property set. The promise problem $(Q, R)$ is $\mathcal{P}$-solvable if there is a polynomial-time algorithm which decides $R$ correctly on inputs from $Q$, or equivalently, if there is some language $L \in \mathcal{P}$ such that $L \supseteq Q \cap R$ and $L \supseteq Q \setminus R$. For example, a theorem of (Grötschel, Lorasz, and Schrijver, 1981) shows that the promise problem $(Q := \{\text{perfect graphs}\}, R := \{3\text{-colorable graphs}\})$ is $\mathcal{P}$-solvable, even though $R$ is $\mathcal{NP}$-complete and $Q$ is not known to be in $\mathcal{P}$.

We change notation slightly by defining $Y := Q \cap R$ and $N := Q \setminus R$. Then the solution space of the promise problem $\Pi(Y, N)$ is the class $\mathcal{I}(Y, N)$ defined in Section 2.3. Taking "$\mathcal{NP}$-hard" to refer to polynomial-time Turing reducibility, and SAT as a representative $\mathcal{NP}$-complete language, we define

DEFINITION 6.1. The promise problem $\Pi(Y, N)$ is

(a) $\mathcal{NP}$-hard if every language $A$ in $\mathcal{I}(Y, N)$ is $\mathcal{NP}$-hard,

(b) uniformly $\mathcal{NP}$-hard if there is a single poly-time bounded OTM $M$ such that for all $A \in \mathcal{I}(Y, N)$, $L(M^A) = \text{SAT}$.

We show that (a) and (b) are equivalent for any $Y, N \subseteq \Gamma^*$, as a consequence of a somewhat more general theorem.
The definitions corresponding to (a) and (b) in Grollman and Selman (1984, 1988) restrict attention to recursive languages $A$. Hence there are actually two forms of the assertion "(a) $\iff$ (b)", an effective and a noneffective one. Although neither form implies the other directly, we obtain both as ramifications of the same technique of diagonalizing over countably many classes $C_k$. In the notation of Section 3, we have $B_k = B_{k+1} = \cdots = B_k = C_k$ for all $k \in \mathbb{N}^+$ (and all $k \in \mathbb{N}^+$), where unlike the setting of Section 5, each $C_k$ may have uncountable many members. The fact that each $C_k$ is (effectively) nowhere dense and closed in $\mathcal{I}$ will pull the diagonalization through.

Any OTM $M$ defines a mapping from $\mathcal{P}(\Gamma^*)$ to $\mathcal{P}(\Gamma^*)$ by $A \mapsto L(M^A)$. More generally, we can suppose that a portion $W \subseteq \Gamma^*$ of the oracle is already present and fixed, and consider maps of the form $A \mapsto L(M^W \oplus A)$. We say $M^W \oplus (-)$ is total if for all $A \subseteq \Gamma^*$, $M^W \oplus A$ is total.

**Lemma 6.1.** Given $W \subseteq \Gamma^*$ and an OTM $M$, suppose $W \oplus (-)$ is total. Then for any class $B$ which is closed in $\mathcal{I}$, the class $\mathcal{A} := \{ A \supseteq \Gamma^* \mid L(M^W \oplus A) \in B \}$ is closed in $\mathcal{I}$.

**Proof.** Suppose $A \notin \mathcal{A}$. Then $L(M^W \oplus A) \notin B$. Since $B$ is closed, there exists $\beta \subseteq L(M^W \oplus A)$ such that $C_\beta \cap B = \emptyset$. Since $M^W \oplus A$ is total, the set $\{ \gamma \in \Gamma^* \mid \text{on some input } x \leq \beta, \text{ queries } z \text{ in the course of its computation} \}$ is finite, and so has a largest element $Z$. Take any $\gamma \subseteq W \oplus A$ such that $\gamma \geq z$; then for any $C \supseteq \gamma$, $\beta \subseteq L(M^C)$. Then we can take $x \subseteq A$ such that for any $A' \supseteq x$, $\gamma \subseteq W \oplus A'$. This means that for any $A' \supseteq x$, $\beta \subseteq L(M^W \oplus A')$, so $L(M^W \oplus A') \notin B$, so $A' \notin \mathcal{A}$. Hence $C_\gamma \cap \mathcal{A} = \emptyset$, and so $\mathcal{A}$ is closed in $\mathcal{I}$. 

This is actually half of a characterization of those mappings $\Phi: \mathcal{P}(\Gamma^*) \to \mathcal{P}(\Gamma^*)$ which are continuous over $\mathcal{I}$, meaning that $\Phi^\mathcal{I} (B)$ is closed for every closed class $B$. (This is dual to the familiar condition that the inverse image of an open set under $\Phi$ be open.) For the other half, one can define

$$W := \{ x \neq \beta \mid (\forall A \subseteq \Gamma^*)[x \subseteq A \Rightarrow \beta \subseteq \Phi(A)] \}. \quad (6.1)$$

Then it is straightforward to construct an OTM such that $M^W \oplus (-)$ is total and $\Phi(A) = L(M^W \oplus A)$ for all $A$. (It is not generally possible to make $M$ itself total for all oracles.) For simplicity, we continue to refer to total OTM's rather than "continuous function(al)s."

**Definition 6.2.** A family $\mathcal{F}$ of mappings from $\mathcal{P}(\Gamma^*)$ to $\mathcal{P}(\Gamma^*)$ is finitely patchable if for all $\Phi \in \mathcal{F}$ and $\gamma \in \{0, 1\}^*$, the mapping $\Phi^\gamma : A \mapsto \Phi(\gamma/A)$ is also in $\mathcal{F}$. 


An example of a finitely patchable family, consisting of countably many continuous mappings over I, is the collection of polynomial-time bounded OTMs, which collectively define \( \leq \). Many other familiar reducibilities can be defined in terms of such families of OTMs.

**Theorem 6.2.** Let \( \mathcal{F} \) be a finitely-patchable family of maps of the form \( \Phi: A \to L(M^{W \oplus A}) \), where \( W \) is fixed and \( M^{W \oplus (\cdot)} \) is total. Let \( B, Y, N \subseteq \Gamma^* \) with \( Y, N \) disjoint. With \( \mathcal{D} := \mathcal{F}(Y, N) \), suppose

\[
(\forall \Phi \in \mathcal{F})(\exists A \in \mathcal{D}): \Phi(A) \neq B.
\]

Then

\[
(\exists A \in \mathcal{D})(\forall \Phi \in \mathcal{F}): \Phi(A) \neq B.
\]

**Proof.** Since \( \mathcal{F} \) is defined by OTMs and \( W \) is fixed, \( \mathcal{F} \) is countable. Let \( [\Phi_k]_{k=1}^{\infty} \) enumerate \( \mathcal{F} \), and for each \( k \) put \( \mathcal{C}_k := \{ A \in \mathcal{D} | \Phi_k(A) = B \} \). By Lemma 6.1, each \( \mathcal{C}_k \) is closed, hence closed in \( \mathcal{D} \). We claim that each \( \mathcal{C}_k \) is nowhere dense in \( \mathcal{D} \). If some \( \mathcal{C}_k \) is not nowhere dense, then \( \mathcal{C}_k \) contains a nonempty open subset of \( \mathcal{D} \) (since \( \mathcal{C}_k \) is already closed), and so \( (\mathcal{C}_\gamma \cap \mathcal{D}) \subseteq \mathcal{C}_k \) for some \( \gamma \in \text{Con}(\mathcal{D}) \). By the finite patchability, there exists \( l \) such that \( \Phi_l(A) = \Phi_k(\gamma / A) \) for all \( A \in \Gamma^* \). Since \( \mathcal{D} \) is closed under finite splices, \( \gamma / A \in \mathcal{D} \) for all \( A \in \mathcal{D} \), and then \( \gamma / A \in \mathcal{C}_k \). Hence for all \( A \in \mathcal{D} \), \( \Phi_l(A) = \Phi_k(\gamma / A) \) - \( B \). But this contradicts (6.2), proving the claim.

So we may apply Lemma 2.4 for each \( k \), obtaining \( A_k \in \mathcal{D} \) such that \( A_k \cap \mathcal{C}_k = \emptyset \). Then there exists a witness-ranging function \( f \) for \( [A_k]_{k=1}^{\infty} \), with respect to \( [\mathcal{C}_k]_{k=1}^{\infty} \), and \( f \) can be outrun by some function \( h: \Gamma^* \to \mathbb{N}^+ \). Define \( E := \bigcup_{k=1}^{\infty} (A_k \cap h^{-1}(k)) \) as usual. By Lemma 3.2, \( E \notin \mathcal{C}_k \) for all \( k \). Since each \( A_k \) contains \( Y \), \( E \) contains \( Y \), and similarly \( E \) contains \( N \), so \( E \in \mathcal{D} \). Hence (6.3) is satisfied with \( A := E \).

**Corollary 6.3.** A promise problem \( \Pi(Y, N) \) is \( \mathcal{NP} \)-hard if and only if it is uniformly \( \mathcal{NP} \)-hard.

**Proof.** Let \( \mathcal{F} \) be the family of polynomial-time bounded OTMs, and take \( B := \text{SAT} \). Then \( \neg(6.3) \) says that \( \Pi(Y, N) \) is \( \mathcal{NP} \)-hard, while \( \neg(6.2) \) says that \( \Pi(Y, N) \) is uniformly \( \mathcal{NP} \)-hard. Since \( \neg(6.3) \rightarrow \neg(6.2) \) is a logical truth, Theorem 6.2 gives the desired equivalence.

For the effective form of the result, we look at the mechanics of the proof a little more closely.

**Theorem 6.4.** Given an oracle set \( X \), suppose \( [Q_k]_{k=1}^{\infty} \) is an \( X \)-recursive enumeration of OTMs such that for some \( W \subseteq \Gamma^* \), each \( Q_k^{W \oplus (\cdot)} \) is total and...
the family of maps \( \Phi_k: A \rightarrow L(Q_k^{W \oplus A}) \) is finitely patchable. Suppose that \( W, B, Y, \) and \( N \) are all recursive in \( X \). With \( \mathcal{D} := \mathcal{S}(Y, N) \), we have: if

\[
(\forall k)(\exists A \in \mathcal{D}): L(Q_k^{W \oplus A}) \neq B,
\]

then

\[
(\exists A \in \mathcal{D})(\forall k): L(Q_k^{W \oplus A}) \neq B,
\]

and \( A \) is \( X \)-recursive.

**Proof.** We need to show that the classes \( \mathcal{C}_k \) are uniformly and effectively (relative to \( X \)) nowhere dense in \( \mathcal{D} \). That is, we must exhibit an \( X \)-recursive function \( g: \{0, 1\}^* \times N^+ \rightarrow \{0, 1\}^* \) such that

\[
(\forall k)(\forall \beta \in \text{Con}(\mathcal{D}))[g(\beta, k) \in \text{Con}(\mathcal{D}) \land g(\beta, k) \supseteq \beta \land g_{\mathcal{C}_k} \subseteq \sim \mathcal{C}_k].
\]

Since \( Y \) and \( N \) are \( X \)-recursive and we take \( \mathcal{D} := \mathcal{S}(Y, N) \), \( \text{Con}(\mathcal{D}) \) is \( X \)-recursive. Hence given \( \beta \in \text{Con}(\mathcal{D}) \) we may \( X \)-recursively enumerate extensions \( \alpha \) of \( \beta \) with \( \alpha \in \text{Con}(\mathcal{D}) \) in nondecreasing order of length. For each \( \alpha \) we test whether there is an input \( x \), with \( \text{str}(x) \leq |\alpha| \), such that \( \alpha \) codes enough information about the "\( A \)-half" of the oracle to render \( x \in L(Q_k^{W \oplus A}) \cap B \) for all \( A \supseteq \alpha \). Since \( W \leq_T X \) and \( B \leq_T X \), the test is decidable with help from \( X \). By the fact that each \( \mathcal{C}_k \) is nowhere dense, some such \( \alpha \) is guaranteed to exist. Hence the function \( \gamma(\beta, k) := \alpha \) is \( X \)-recursive and satisfies (6.6).

With \( g(\cdot, \cdot) \) in hand, Lemma 2.4 gives us an \( X \)-recursive presentation \( \{A_k\}_{k=1}^\infty \) of languages such that \( A_k \cap \mathcal{C}_k = \emptyset \) for each \( k \). Each \( g_k: \beta \mapsto g(\beta, k) \) yields an \( X \)-recursive later-witness function \( f_k \) for "\( A_k \cap \mathcal{C}_k = \emptyset \)" by the recipe of Definition 3.1, and \( f: x \mapsto \max\{f_k(x) | k \leq x\} \) is an \( X \)-recursive witness-ranging function. By Theorem 3.3b there is a function \( h \in \mathcal{L}_\mathcal{D}^X \) which outruns \( f \) and is onto \( N^+ \). With \( E := \bigcup_{k=1}^\infty (A_k \cap h^{-1}(k)) \) we have \( E \subseteq g_{\mathcal{D}} A_{\omega} \) in addition to (6.5) with \( A := E \), and since \( A_{\omega} \) is recursive in \( X \), so is \( E \). 

Note that it is not necessary for the finite patchability of \( \mathcal{S} \) to be effective; i.e., there need not be an \( X \)-recursive function \( \sigma: \{0, 1\}^* \times N^+ \rightarrow N^+ \) such that for all \( \beta, k, \) and \( A \), \( L(Q_{\sigma(\beta, k)}^{W \oplus A}) = L(Q_k^{W \oplus \beta \oplus A}) \).

**Corollary 6.5** (Grollman and Selman, 1984). Under the Grollman–Selman restriction of attention to recursive sets, a promise problem \( \Pi(Y, N) \) is \( \mathcal{N} \)-hard if and only if it is uniformly \( \mathcal{N} \)-hard.

**Proof.** The restriction entails that for \( \Pi(Y, N) \) to be \( \mathcal{N} \)-hard, \( Y \) and \( N \) themselves must be recursive. So must \( B := \text{SAT} \) and the enumeration
\[ [Q_k]_{k=1}^{\infty} \] of polynomial-time OTMs. So we may take \( X := \emptyset \) in Theorem 6.4. It proof shows that if \( \Pi(Y, N) \) is not uniformly \( \mathcal{NP} \)-hard, then one obtains a recursive counterexample \( E \in \mathcal{S}(Y, N) \) to \( \mathcal{NP} \)-hardness. \[ \]

It is also possible to prove Theorem 6.4 and Corollary 6.5 by extending the argument of Lemma 2.4 itself. Essentially this leads to the original argument of Grollmann and Selman (1984, 1988). Pursuing the remarks following that lemma, one can likewise prove Theorem 6.2 by an appeal to the Baire Category Theorem, as originally done in Regan (1984, 1985, and see below). Effective forms of the Baire Category Theorem are discussed in Lutz (1987), and corresponding variants of Theorem 6.4 also follow from these. The present method isolates the role played by finite variations, and lends itself to cases where the classes \( \mathcal{C}_k \) might be nowhere dense (i.e., satisfy (6.6)) for some other reason besides the finite patchability of \( \mathcal{F} \). If one already has the languages \( A_k \) such that \( A_k \cap \mathcal{C}_k = \emptyset \) in hand, then it provides the extra information that \( E \leq^{q,k} A_\omega \). It also leaves some room for slack in choosing \( E \).

The only ready-made improvement to Theorem 6.2 which we can see comes from replacing \( \{ B \} \) by a general collection of classes closed in \( \mathcal{I} \). We only give the noneffective form of the result, also noting that the index \( l \) once again ranges over \( \mathbb{N}^+ \).

**Theorem 6.6 (Regan, 1985).** Let \( Y, N \subseteq \Gamma^* \) be disjoint. Let \( \mathcal{B} \) be a countable collection of classes which are closed in \( \mathcal{I} \). Let \( \mathcal{F} \) be a finitely patchable family of continuous mappings over \( \mathcal{I} \). Suppose

\[
(\forall A \in \mathcal{S}(Y, N))(\exists \mathcal{B} \in \mathcal{B})(\exists \Phi \in \mathcal{F}):(\Phi(A) \in \mathcal{B}). \tag{6.7}
\]

Then

\[
(\exists \mathcal{B} \in \mathcal{B})(\exists \Phi \in \mathcal{F})(\forall A \in \mathcal{S}(Y, N)): (\Phi(A) \in \mathcal{B}). \tag{6.8}
\]

**Proof.** Let \( [\Phi_k]_{k=1}^{\infty} \) enumerate \( \mathcal{F} \), and let \( [\mathcal{B}_l]_{l=1}^{\infty} \) enumerate \( \mathcal{B} \). For each \( k, l \) define \( \mathcal{C}_{kl} := \{ A \subseteq \Gamma^* | \Phi_k(A) \in \mathcal{B}_l \} \). Thereafter proceed as in the proof of Theorem 6.2. \[ \]

**Corollary 6.7.** Let disjoint \( Y, N \subseteq \Gamma^* \) be given, and let \( \mathcal{B} \) be a countable class of languages. Suppose that for every \( D \in \mathcal{S}(Y, N) \) there exists \( B \in \mathcal{B} \) such that \( B \leq^{q} D \). Then there is a single language \( B \in \mathcal{B} \) such that \( B \leq^{q} \mathcal{S}(Y, N) \) uniformly.

When the languages \( B \in \mathcal{B} \) are all pairwise incomparable under \( \leq^{q} \), it may seem a bit surprising that \( \mathcal{S}(Y, N) \) must sit completely over one of them.
Proof. In Theorem 6.6, let $\mathcal{B}$ be the countable collection of singleton classes $\{B\}$ for $B \in \mathcal{B}$, and let $\mathcal{F}$ be the family of poly-time bounded OTMs. 

Note the quantifier interchange between (6.7) and (6.8), which accounts for our reference to uniformity. It is also worthwhile to compare Theorem 6.6 with the Uniform Boundedness Principle of real analysis as stated by H. Royden (1968), where we have changed part of the notation:

**Theorem.** Let $\mathcal{F}$ be a family of real-valued continuous functions on a complete metric space $\mathcal{F}$, and suppose that for each $a \in \mathcal{F}$ there is a number $M_a$ such that $|f(a)| \leq M_a$ for all $f \in \mathcal{F}$. Then there is a nonempty open set $\mathcal{O} \subseteq \mathcal{F}$ and a constant $M$ such that $|f(a)| \leq M$ for all $f \in \mathcal{F}$ and all $a \in \mathcal{O}$.

In our analogy, $\mathcal{F}$ is the same as before, $\mathcal{F}(Y, N)$ corresponds to $\mathcal{F}$, and the condition "$\Phi(A) \in \mathcal{B}$" plays the role of "$|f(a)| \leq M_a$." Though $\mathcal{F}$ can be uncountable here, the ability to write $\mathcal{F}$ as the countable union of subsets $E_m := \{a \in \mathcal{F} | (\forall f \in \mathcal{F})[|f(a)| \leq m]\}$ makes up for it. The salient difference is that (6.8) holds for all $A \in \mathcal{F}(Y, N)$, instead of all $a \in \mathcal{O}$ where $\mathcal{O}$ is just a nonempty open subset of $\mathcal{F}$. Figuratively speaking, Theorem 6.6 uses the finite patchability of $\mathcal{F}$ and the closure of $\mathcal{F}(Y, N)$ under splices to "project" the uniform reduction in (6.8) onto the whole space $\mathcal{F}(Y, N)$.

The attempt to establish Theorem 6.2 for a general closed class $\mathcal{D}$, i.e., without the closure under finite splices that makes $\mathcal{D}$ equal $\mathcal{F}(Y, N)$ for some $Y, N \in \Gamma^*$, fails to a counterexample noted in (Regan, 1985, 1986b) involving a small class $\mathcal{D}$ and a very small finitely patchable $\mathcal{F}$. We have not been able to find a counterexample when $\mathcal{F}$ represents a reducibility relation which is used in practice, and we ask in particular,

**Open Question.** When $\mathcal{F}$ is given by the family of polynomial-time bounded OTMs, does Theorem 6.2 (or the more general Theorem 6.6) hold with $\mathcal{F}(Y, N)$ replaced by any class $\mathcal{D}$ which is closed in $\mathcal{F}$?

### 7. Fixed-Point Theorems

The culmination of our diagonalization techniques is a new fixed-point theorem for subrecursive classes. We give general statements first, then some applications, and finally compare it to classical fixed point results in recursion theory. For any language $U$ and function $\tau : \mathbb{N}^+ \rightarrow \mathbb{N}^+$, define the graph of $\tau$ on $U$ to be $U_{\tau} := \{x \neq k | x \in U_{\tau(k)}\}$.

**Theorem 7.1a (Simple Form).** Let $U$ be a recursive universal language for a class $\mathcal{C}$ which is closed downward under $\leq_m$. Then for any recursive
function $\tau: \mathbb{N}^+ \to \mathbb{N}^+$ such that $U_\tau \in \mathcal{C}$, there exist infinitely many $k$ such that $U_k \equiv f U_{\tau(k)}$.

We call such $k$ an a.e.-fixed point of $\tau$. It is an exact fixed point if $U_k = U_{\tau(k)}$. As before, we actually state and prove a stronger relativized form. Curiously, our techniques do not yield the analogous constructive form, and we discuss this problem at the end of this section.

**Theorem 7.1b (Relativized Form).** Given $X \subseteq \Gamma^*$, let $\mathcal{C}$ be a class closed downward under $\leq_m^{q_1,X}$, and let $U$ be an $X$-recursive universal language for $\mathcal{C}$. Let $\tau: \mathbb{N}^+ \to \mathbb{N}^+$ be any $X$-recursive function such that $U_\tau \in \mathcal{C}$. Then there exist infinitely many $k$ such that $U_k \equiv f U_{\tau(k)}$.

**Proof.** First we show that there is at least one a.e. fixed point $k$. Suppose no such $k$ exists. For all $k \in \mathbb{N}^+$, define $w := (U_k \in \mathcal{C})$, $A_k := U_k \in \mathcal{C}$. Then $[\mathcal{C}_k]_{\mathcal{Z}^+}^{\mathbb{N}^+}$ is an $X$-recursive presentation of r.p. (singleton) classes, and $[A_k]_{\mathcal{Z}^+}^{\mathbb{N}^+}$ is an $X$-recursive presentation of languages such that $(A_k) \cap \mathcal{C}_k = \emptyset$ for each $k$. The hypotheses of Theorem 5.2(b) are satisfied, and so there is a language $E$ such that $E \not\in \bigcup_{k=1}^{\mathbb{N}^+} \mathcal{C}_k$ (so $E \not\in \mathcal{C}$) and $E \leq_m^{q_1,X} A_m$. However, $A_m$ equals $U_m$, so $A_m \in \mathcal{C}$. Since $\mathcal{C}$ is closed downward under $\leq_m^{q_1,X}$, we have $E \in \mathcal{C}$, a contradiction.

To show that there are infinitely many, put $K := \{k \mid U_k \not\equiv f U_{\tau(k)}\}$. Supposing to the contrary that $\sim K$ is finite, define $\mathcal{C}_k$ and $A_k$ as above for all $k \in K$. By Proposition 4.4, we can arrange directly that $E \not\equiv U_k$ for the finitely many $k \in \sim K$. This preserves the contradiction caused by having $E \not\in \mathcal{C}$.

It is possible to show that there are infinitely many a.e. fixed points without using the trick of Proposition 4.4 when $\mathcal{C}$ is closed downward under $\leq_m^{q_1,X}$ as well as $\leq_m^{q_1,X}$, or when the sequence $U_1, U_2, U_3, \ldots$ repeats each language in $\mathcal{C}$ infinitely often.

The proof works equally well if $\tau: \mathbb{N}^+ \to \mathbb{N}^+$ is considered to be a mapping from a presentation $[Q_k]_{k=1}^{\mathbb{N}^+}$ of $\mathcal{C}$ (where each $Q_k$ has $X$ as oracle) to the global indexing $[M_k]_{k=1}^{\mathbb{N}^+}$ of OTM acceptors; the only important part is that the graph $\{x \neq k \mid x \in L(M_{\tau(k)}^X)\}$ of the mapping must still belong to $\mathcal{C}$. Or $\tau$ may map into some other enumeration $[B_k]_{k=1}^{\mathbb{N}^+}$ of languages. In particular, we have

**Theorem 7.2.** Let $\mathcal{C}$ be any r.p. class which is closed downward under $\leq_m^{q_1,X}$, and let $U$ be any recursive universal language for $\mathcal{C}$. Then for any language $B \in \mathcal{C}$ there are infinitely many $k$ such that $B_k \equiv f U_k$.
Proof. For each $k$, take $C_k := U_k$ and $A_k := B_k$. Denying the conclusion makes $A_k \cap C_k = \emptyset$ for all but finitely many $k$. Since $U$ and $B$ are recursive, $[C_k]_{k=1}^\infty$ and $[A_k]_{k=1}^{\infty}$ are recursively presented. The same reasoning as the last proof produces a language $E$ such that $E \notin \bigcup_{k=1}^{\infty} C_k$, so $E \notin C$, and $E \leq_f B_\omega$. However, $B_\omega = B$, and since $B \in C$ and $C$ is closed downward under $\leq_f$, we have the contradiction that $E \in C$. 

In other words, the identity function $\tau: \mathbb{N}^+ \to \mathbb{N}^+$, which maps $U_k$ to $B_k$, makes $U_k = \{x \neq k \mid x \in B_k\} = B \in \mathcal{C}$, so it has $\infty$ many a.e. fixed points $k$.

7.1. Two Applications.

In the first application, $B_k$ refers to the language of true Boolean $\Sigma_k$-sentences as defined in (Meyer and Stockmeyer, 1972). For each $k$, $B_k$ is complete for $\Sigma_k^p$ under $\leq^p_m$, and $B_\omega$ in our notation is essentially the known $\mathcal{P} \mathcal{P} \mathcal{P} \mathcal{N} \mathcal{C} \mathcal{E}$-complete language $QBF$. In the second, we represent the familiar $\mathcal{N} \mathcal{P}$-complete languages $\text{CLIQUE}$ as $C := \{G \neq k \mid G$ has a clique of size $k\}$, so that $k$-$\text{CLIQUE}$ equals $C_k$ in our notation.

**Corollary 7.3.** It is not possible to define a recursive presentation $[Q_k]_{k=1}^{\infty}$ of $\mathcal{P} \mathcal{P} \mathcal{P} \mathcal{N} \mathcal{C} \mathcal{E}$ without there being infinitely many $k$ such that $L(Q_k)$ is $\Sigma_k^p$-complete; in fact, such that $L(Q_k) \equiv_f B_k$.

**Corollary 7.4.** For every recursive presentation $[N_k]_{k=1}^{\infty}$ of $\mathcal{N} \mathcal{P}$ there are infinitely many $k$ such that $L(N_k)$ is a finite variation of the language $k$-$\text{CLIQUE}$.

The proofs are immediate. Note that the results make no unproven assumptions about $\mathcal{P}$, $\mathcal{N} \mathcal{P}$, $\mathcal{P} \mathcal{N}$, or $\mathcal{P} \mathcal{P} \mathcal{P} \mathcal{N} \mathcal{C} \mathcal{E}$. Now each language $k$-$\text{CLIQUE}$ actually belong to $\mathcal{P}$, so it is natural to ask:

**Open Question.** Does Corollary 7.4 hold for $\mathcal{P}$ in place of $\mathcal{N} \mathcal{P}$? Or negatively, is there a recursive presentation $[P_k]_{k=1}^{\infty}$ for $\mathcal{P}$ such that for all $k$, $L(P_k) \not\equiv_f k$-$\text{CLIQUE}$?

If such a presentation can be constructed, then $\mathcal{P} \neq \mathcal{N} \mathcal{P}$. (We do not know whether the converse of this last statement holds.) A simple sufficient condition for this is that there be an unbounded recursive function $r(k)$ such that for all $k$, $k$-$\text{CLIQUE}$ cannot be solved in deterministic time $O(n^{r(k)})$.

The same question may be asked for the $\mathcal{N} \mathcal{P}$-complete $\text{VERTEX COVER}$ problem in place of $\text{CLIQUE}$. Here, the associated $k$-$\text{VERTEX COVER}$ languages are known to be in deterministic time $O(n^2)$ independent of $k$ (Robertson and Seymour, 1986; Fellows and Langston, 1988). (See also the review article by D. Johnson 1987.) However, the known algorithms have constants or program size which grow exponentially in $k$. 

More detailed work on the connections among the fixed points, recursive presentations, and program size complexity may unlock some secrets of the \( \mathcal{P}^2 = \mathcal{N}^\mathcal{P} \) question.

7.2. Comparison to Other Fixed-Point Theorems.

Theorem 7.1 resembles two theorems from classical recursion theory, which we cite under names from Soare (1987). Here \( W := \{ x \neq i \mid T_i(x) \text{ halts} \} \) is the standard r.e. universal language for the r.e. sets.

**Recursion Theorem.** Let \( \tau: \mathbb{N}^+ \to \mathbb{N}^+ \) be a recursive function. Then there are infinitely many \( i \) such that \( W_i = W_{\tau(i)} \).

**Arslanov’s Fixed-Point Theorem.** Let \( \tau: \mathbb{N}^+ \to \mathbb{N}^+ \) be recursive in the Halting Problem (i.e., in \( W \)). Then there are infinitely many \( i \) such that \( W_i = W_{\tau(i)} \).

The analogy with Theorem 7.1b takes \( \mathcal{C} \) to be the class \( \mathcal{R} \mathcal{E} \) of r.e. sets and both \( U \) and \( X \) to be \( W \). The condition “\( U_i \in \mathcal{C} \)” then follows automatically if \( \tau \) is recursive, but need not hold when \( \tau \) is merely recursive in \( W \). Moreover, it is not the case that \( \mathcal{R} \mathcal{E} \) is closed under \( \preceq_{m}^{W} \) or \( \preceq_{t}^{W} \).

The chief difference from these results, however, is that they depend on special properties of the presentation \( W \), namely that \( \{ T_i \} \) be an acceptable programming system in the sense of (Rogers, 1967). One way to obtain an unacceptable r.e. universal language \( U \) for \( \mathcal{R} \mathcal{E} \) is to define \( U_{2i} := W_i \setminus \{ \lambda \} \), \( U_{2i+1} := W_i \cup \{ \lambda \} \) for all \( i \); then the Recursion Theorem fails with \( \tau(n) := n + 1 \). The Arslanov theorem still holds for this indexing, but M. Kummer (1989) has found an r.e. set \( U \) and a recursive function \( \tau \) for which it fails. By contrast, Theorem 7.1 needs no good behavior from \( U \). D. Kozen (1980) has presented what he calls a “weak recursion theorem” (namely, a subrecursive analogue of the “First” or “Kleene” Recursion Theorem, as labeled in Cutland, 1980, and Machtey and Young, 1981, respectively) under natural acceptability assumptions on programming systems for classes such as \( \mathcal{P} \mathcal{S} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E} \). His paper asserts that a corresponding “strong theorem” cannot exist because (e.g.) \( \mathcal{P} \mathcal{S} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E} \) is not \( \mathcal{P} \mathcal{S} \mathcal{P} \mathcal{A} \mathcal{C} \mathcal{E} \)-presentable. Theorem 7.1 uses the condition “\( U_i \in \mathcal{C} \)” to make up for this lack.

The Recursion Theorem provides exact fixed points. Applying the same \( U_{2i} := V_i \setminus \{ \lambda \} \)” trick as above to a given universal language \( V \) for \( \mathcal{C} \) shows that Theorem 7.1 cannot be improved to yield exact fixed points in general. We may still ask whether this is so when \( U \) is fairly well-behaved. However, we consider the Arlanov theorem, despite the differences, to be evidence that the answer is “usually not”; and we have not been able to
answer this even for the standard "clocked TM" presentations of $\mathcal{P}$, $\mathcal{NP}$, or $\mathcal{PSPACE}$.

Another noteworthy aspect of the Recursion Theorem is that one can actually compute a fixed point. The only proofs we know of Theorem 7.1 proceed by contradiction, and do not give rise to a constructive form as with most other results in this paper. A formal statement of the question mentioned in the Introduction, tailored for Theorem 7.2, runs:

**Open Question.** Does there exist a total recursive function $\sigma : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ such that for all $b, u \in \mathbb{N}^+$ with $M_u, M_b$ total: if taking $U := L(M_u), B := L(M_b)$ makes $\mathcal{C} := \mathcal{P}[U]$ closed downward under $\leq'$ and $B \in \mathcal{C}$, then $U_{\sigma(u,b)} \equiv \not B_{\sigma(u,b)}$?

This becomes rather more reasonable if we fix $U$ to be one of the standard presentations mentioned above. However, even here we do not know how to find the fixed points, and see no ready method even when also $B_k := \Sigma_k\text{-SAT}$ for all $k$.

8. Conclusions and Prospects

We have shown three facets of the same diagonalization technique: straightforward diagonalization constructions, a uniform reduction theorem whose contraposition yields a construction, and a fixed-point theorem for which we know of no construction at all. Among the first are many theorems in the literature which fall under the rubric of "delayed diagonalization," while the second is relatively new, and the third was previously unknown, at least to the author. Yet they reflect phenomena which arise in other areas of mathematics, for example uniform boundedness and (Brouwer-type) fixed-points. Can developments in these other areas be used to find parallels for new results in complexity theory? The potential is there.

Thus far we have only distinguished between effective and noneffective forms of the diagonalization theorems. Likewise, we have been satisfied with the fact that real-time/log-space is sufficient to outrun any recursive function. It is possible to envision a closer analysis of the growth rates and time complexities of recursive witness functions, and of the delay construction in Section 4. Such an analysis for resource-bounded notions of nowhere-denseness and meagerness, using complexity-theoretic analogues of the Baire Category Theorem and Banach-Mazur games, has already been initiated in (Lutz, 1987). Likewise, we ask whether $\mathcal{FP}$ or $\mathcal{FEXP}$ (in place of $\mathcal{FPRB}$) can be outrun by a class $\mathcal{F}$ which is substantially smaller than $\mathcal{RL}_x$ or $\mathcal{2L}_x$, where $\mathcal{F}$ might provide much stronger conclusions
than those of Theorem 5.2 in cases where the witness functions are computable in polynomial or exponential time.

Even if such improvements will be hard to come by, we can still find a useful task in determining just which types of diagonalization theorems are obtainable by the current technique, and which are not. The idea is that there "should be" a fairly natural formal system $\mathcal{F}$ which encompasses just those machinations with the Cantor-set topology and operations on oracle Turing machines which go into the relevant theorems. Such an $\mathcal{F}$ might help explain the difficulty of resolving questions such as $\mathcal{P} \neq \mathcal{NP}$ which are known not to relativize. The observation that the answer is "yes" for some oracle sets and "no" for others has been likened to a formal independence result from a certain theory, but the identity of the theory has yet to be found.

In sum, we have taken a technique which has already been called "widely applicable" (Schöning) and provided a flexible and general rendition which makes it even more so. We look forward to deeper results which exploit its full power.

AUTHOR'S NOTE AND ACKNOWLEDGMENTS

The results in Section 6 were contained in a paper (Regan, 1984) originally submitted to this journal (Information and Computation) in 1984, which I withdrew in favor of the present paper. I thank the referees for their helpful comments, some of which have influenced the present treatment. Earlier versions of these results formed part of my doctoral dissertations (Regan, 1986b). I also thank J. Royer and S. Mahaney for their comments on a companion paper (Regan, 1988), some of which helped me in writing this one. Professor Martin Kummer of Frankfurt solved a problem in Section 7.2 which was open in my original submission. L. Hemachandra prompted me to note a stronger feature of Theorem 4.2. J. Hartmanis, D. Kozen, A. Macintyre, and D. Welsh also gave me helpful suggestions. Finally, I am grateful to the referee of this paper for a careful review, and for comments which will also help guide further research.

RECEIVED September 1, 1988; FINAL MANUSCRIPT RECEIVED August 28, 1990

REFERENCES


Bennett, C., and Gill, J. (1981), Relative to a random oracle $A$, $\mathcal{P}(A) \neq \mathcal{NP}(A)$ with probability 1, SIAM J. Comput. 10, 96-113.


Regan, K. (1984), "A Uniform Reduction Theorem, with Applications to the Complexity of Inverting Easy-to-Compute Functions," draft paper, Oxford University.


Schöning, U. (1981), "Untersuchungen zur Struktur von \( \mathcal{P} \langle \phi \rangle \)," Ph.D. Dissertation, University of Stuttgart.
