Index Sets and Presentations of Complexity Classes

(revised version)

Kenneth W. Regan[1]
State University of New York at Buffalo

[1]Author's current address: Computer Science Department, 226 Bell Hall, UB North Campus, Buffalo, NY 14260-2000. Email: *regan@cs.buffalo.edu*, tel.: (716) 645–3189, fax: (716) 645–3464.

Abstract

This paper draws close connections between the ease of presenting a given complexity class $\mathcal{C}$ and the position of the index sets $I_{\mathcal{C}} = \{ i : L(M_i) \in \mathcal{C} \}$ and $J_{\mathcal{C}} = \{ i : M_i \text{ is total } \wedge L(M_i) \notin \mathcal{C} \}$ in the arithmetical hierarchy. For virtually all classes $\mathcal{C}$ studied in the literature, the lowest levels attainable are $I_{\mathcal{C}} \in \sum_3^0$ and $J_{\mathcal{C}} \in \prod_2^0$; the first holds iff $\mathcal{C}$ is Δ_2^0-presentable, and the second iff $\mathcal{C}$ is recursively presentable. A general kind of priority argument is formulated, and it is shown that every property enforcible by it is not recursively presentable. It follows that the classes of P-immune and P-biimmune languages in exponential time are not recursively presentable. It is shown that for all $\mathcal{C}$ with $I_{\mathcal{C}} \notin \sum_3^0$, "many" members of $\mathcal{C}$ do not provably (in true Π_2-arithmetic) belong to $\mathcal{C}$. A class $\mathcal{H}$ is exhibited such that whether $I_{\mathcal{H}} \in \sum_3^0$ is open, and $I_{\mathcal{H}} \notin \sum_3^0$ implies that the polynomial hierarchy is infinite.

1 Introduction

This paper extends work by Hájek [Háj79], who analyzed complexity classes in terms of their definitions in the familiar first-order language $\mathcal{L}_A$ of arithmetic. Finite "names" for r.e. languages are obtained by fixing a standard effective enumeration $M_1, M_2, M_3, \ldots$ of Turing machine acceptors. For motivation, we look at one of Hájek's main examples: Consider the following ways of formalizing the class P (polynomial time) in terms of this enumeration:

1. $\phi_1(i) :=$ '$L(M_i)$ is acceptable in polynomial time.'

2. $\phi_2(i) :=$ 'M_i runs in polynomial time'

3. $\phi_3(i) :=$ 'M_i has an explicitly-defined time clock that shuts off operation after n^i steps.'

(Here and later the single quotes '...' mean that we have a specific formalization of the condition in mind, but use an informal rendering for better readability.) The first defines what various sources call a "property of languages" or an "extensional property." The second defines a subset I_2 of I_1 that is not extensional, since there are Turing machines that do not run in polynomial time but accept the same language as some machine that does. The third defines a proper subset I_3 of I_2, but still defines the class P insofar as $\mathrm{P} = \{\, L(M_i) : i \in I_3 \,\}$. We say that I_1, I_2, and I_3 are all *index sets* for P, and I_1 is the *full index set*.

The third definition has the following advantage from the viewpoint of formal systems such as *Peano Arithmetic* (PA): whenever $\phi_3(i)$ holds, there is a proof of $\phi_3(i)$ in PA. Hájek showed that I_2 and I_1, however, are not r.e. It follows that for any sound, recursively axiomatized (r.a.) formal system $\mathcal{F}$, including $\mathcal{F} =$ PA and much stronger systems, there are instances i of the formula ϕ_2 such that the sentence $\phi_2(i)$ is true but not provable in $\mathcal{F}$. The same goes for ϕ_1. In fact, Hájek proved that I_2 is many-one complete for the level $\sum_2^0$ of the *arithmetical hierarchy*, and I_1 is $\sum_3^0$-complete, giving a sense in which the formula ϕ_1 is more difficult to prove than ϕ_2. Nevertheless, we can say that membership in P is a provable property (cf. [Bak79]), on account of the *provable representation* ϕ_3.

This raises the question: *Which other complexity classes $\mathcal{C}$ correspond to provable properties?* For NP there is a similar trick: $\nu(i) :=$ 'M_i is transparently coded to simulate a nondeterministic Turing machine (NTM) that has an n_i clock.' The complexity class UP is standardly defined in terms of polynomial-time NTMs N such that for all inputs x, either there is exactly one nondeterministic sequence that leads N to accept x, or there is none. The direct way of formalizing this condition yields a partial index set for UP that is $\prod_1^0$-complete, hence not r.e., hence not a provable representation. However, it is possible to write a formula $\psi(i)$ expressing 'Either M_i simulates a clocked NTM that meets the above condition, or the language accepted by M_i is finite,' such that $\mathrm{UP} = \{\, L(M_i) : \psi(i) \,\}$ and every true instance of $\psi(i)$ is provable in PA. Moreover, PA proves that all finite sets belong to UP, so every language L named by an i such that $\psi(i)$ holds provably belongs to UP.

This author's earlier work [Reg88, Reg92] tied the above problems to the complexity of *universal languages U* that define *presentations* of $\mathcal{C}$. It showed that for every sound, r.a. formal system $\mathcal{F}$ that has at least the axioms of basic arithmetic (see $\mathcal{Q}$ in section 6), a class $\mathcal{C}$ has an $\mathcal{F}$-provable representation iff $\mathcal{C}$ has an r.e. universal language. It is natural to restrict attention to representations, such as ϕ_2 and ϕ_3, which imply that the machines they hold for are total. Then such a provable representation can be found iff $\mathcal{C}$ has a recursive universal language. Thus the question becomes: *Which classes are recursively presentable? Which are r.e.-presentable?* In terms of Machtey and Young [MY78], which classes have *effective programming systems*? What this paper does is supply new techniques for answering these questions, by going back to two particular index sets associated to $\mathcal{C}$: the full index set $I_{\mathcal{C}} = \{ i : L(M_i) \in \mathcal{C} \}$, and the "$J$ index set" $J_{\mathcal{C}} := \{ i : M_i \text{ is total and } L(M_i) \notin \mathcal{C} \}$. All this is accomplished via Hájek's suggestion of determining where index sets lie in the arithmetical hierarchy.

Section 2 gives formal definitions and background, and section 3 gives results on classes $\mathcal{C}$ with $I_{\mathcal{C}} \in \sum_3^0$. Section 4 proves that a class $\mathcal{C}$ is recursively presentable iff $J_{\mathcal{C}} \in \prod_2^0$ (subject to a mild condition on $\mathcal{C}$), and gives results connecting r.e.-presentability to both $I_{\mathcal{C}}$ and $J_{\mathcal{C}}$. These results show that whether $\mathcal{C}$ contains the finite sets or not makes a large difference. Section 5 presents a new and useful technique for proving that certain classes $\mathcal{C}$ are *not* recursively presentable. It defines a natural algorithmic mechanism for what we call "diagonalization by rote," and gives a detailed comparison to diagonalization methods formulated by Ambos-Spies, Fleischhack, and Huwig [AFH87, AFH88]. Both the mechanism and its relation to non-r.p. classes appear to have applications beyond the results in this paper. Section 6 considers provability in certain formal systems, studied by Lipton and DeMillo [Lip78, DL80] and Leivant [Lei82], that are *not* recursively axiomatizable. It concludes with the speculative possibility that certain proofs of $I_{\mathcal{C}} \notin \sum_3^0$ may help to solve major open problems in complexity theory.

Earlier versions of the results in sections 2–4 and 6 appeared in the conference paper [Reg84], and in the dissertation [Reg86]. The results in section 5 are new, and solve open problems in [BS85, Reg88].

2 Notation and Basic Results

Turing machines in this paper will use alphabet $\Sigma = \{ 0, 1 \}$. For each $x \in \Sigma^*$ define $num(x)$ to be the natural number having binary representation $1x$. Then num defines a 1-1 correspondence between Σ^* and $\mathbf{N}^+$, and we denote its inverse by $str(\cdot)$. The *empty string* λ corresponds to 1. The complement of a language A is denoted by $\sim A$. The *join* $A \oplus B$ of the sets $A, B \subseteq \Sigma^*$ is defined to be $\{ x0 : x \in A \} \cup \{ y1 : y \in B \}$. We write $A \equiv^f B$ if the symmetric difference $A \triangle B$ is finite. A^f denotes $\{ L : L \equiv^f A \}$, and for any class $\mathcal{C}$ of languages, $\mathcal{C}^f := \cup_{A \in \mathcal{C}} A^f$. If $\mathcal{C} = \mathcal{C}^f$, then $\mathcal{C}$ is *closed under finite variations* (cfv). $\mathcal{C}$ is *somewhere-cfv* (scfv) if there exists $A \in \mathcal{C}$ such that $\mathcal{C} \supseteq A^f$. The *empty class* $\emptyset$ is cfv but not scfv.

Our complexity-class notation is mostly standard. RE denotes the class of r.e. languages, REC the recursive languages. EXPTIME stands for DTIME$[2^{O(n)}]$, in contrast to EXP = DTIME$[2^{n^{O(1)}}]$. Without loss of generality, we suppose that each Turing machine M_i in the fixed enumeration $[M_i]_{i=1}^\infty$ is an *oracle* Turing machine (OTM). Then for any language A,

$\mathrm{RE}^A = \{\, L(M_i^A) : i \in \mathbf{N}^+ \,\}$, namely the class of languages accepted by OTMs with oracle set A, and $\mathrm{REC}^A = \{L(M_i^A) : M_i$ with oracle set A halts for all inputs$\}$. Classes $\mathcal{C}$ may be given as oracles, as typified by $\mathrm{RE}^{\mathcal{C}} = \cup_{A \in \mathcal{C}} \mathrm{RE}^A$. Absence of an oracle is formally the same as having the empty language as oracle.

A class $\mathcal{C}$ is *r.e.-presentable* if there is a total recursive function $\sigma : \mathbf{N}^+ \to \mathbf{N}^+$ such that $\mathcal{C} = \{\, L(M_{\sigma(i)}) : i \in \mathbf{N}^+ \,\}$. $\mathcal{C}$ is *recursively presentable* if it can be arranged in addition that each $M_{\sigma(i)}$ is total. Also say a class $\mathcal{C}$ of recursive languages is *bounded* if $\mathcal{C}$ is contained in a recursively presentable class, or equivalently, if $\mathcal{C} \subseteq \mathrm{DTIME}[t(n)]$ for some total recursive function t. The empty class is not r.p. or presentable in any sense at all, but it is bounded.

It is possible to define presentations in a completely machine-independent manner. Fix a polynomial-time computable *pairing function* $\langle \cdot, \cdot \rangle$, i.e., a bijection from $\mathbf{N}^+ \times \mathbf{N}^+$ to $\mathbf{N}^+$. For any language U and $k \in \mathbf{N}^+$ define the language $U_k := \{\, x : \langle x, k \rangle \in U \,\}$. Say U is *universal* for a class $\mathcal{C}$ if $\mathcal{C} = \{\, U_k : k \in \mathbf{N}^+ \,\}$. (We do not mandate that U itself belong to $\mathcal{C}$.) Then $\mathcal{C}$ is r.e.-presentable iff there exists an r.e. universal language for $\mathcal{C}$, and $\mathcal{C}$ is r.p. iff there is a recursive universal language. These notions and equivalences all "relativize"; e.g. $\mathcal{C}$ is REC^A-presentable iff $\mathcal{C}$ has a universal language in REC^A. The notation $\langle x, k, l \rangle$ stands for $\langle \langle x, k \rangle, l \rangle$, and further iterates of the pairing function are composed similarly. Thus a recursive language U gives rise to a recursive enumeration of *classes* $\mathcal{C}_1, \mathcal{C}_2, \ldots$, where for each k, U_k is universal for $\mathcal{C}_k$.

Up to Section 6, it suffices for our purposes to work with the informal notion of a "predicate" P with free variables $x_1, \ldots, x_m$, without attention to how P is formalized over $\mathcal{L}_A$ or some other logical language. The language L_P defined by P equals $\{\, \langle x_1, \ldots, x_m \rangle : P(x_1, \ldots, x_m) \,\}$. Two predicates P and R are *equivalent* if $L_P = L_R$. In quantifying predicates we often combine adjacent like quantifiers into one *block*; for instance, $(\exists y_1)(\exists y_2)R$ becomes $(\exists y_1, y_2)R$.

Definition 2.1. Given $k \geq 0$, a predicate P is a $\sum_k^0$*-predicate* if there is a decidable predicate R such that P is equivalent to

$$(\exists y_1, \ldots, y_{i_1})(\forall y_{i_1+1}, \ldots, y_{i_2}) \cdots (Q_k y_{i_{k-1}+1}, \ldots, y_{i_k})\, R, \tag{1}$$

where Q_k is '$\forall$' if k is even, '$\exists$' if k is odd. A $\prod_k^0$*-predicate* is defined analogously beginning with $(\forall y_1, \ldots, y_{i_1})$.

A $\sum_0^0$-predicate or $\prod_0^0$-predicate is the same as a decidable predicate. For every $\sum_k^0$-predicate P, its negation $\neg P$ is a $\prod_k^0$-predicate, and $L_{\neg P} = \sim L_P$.

Definition 2.2. (after Kleene [Kle43]): For all $k \geq 0$, $\sum_k^0$ denotes the class of languages defined by $\sum_k^0$-predicates, $\prod_k^0$ the class of languages defined by $\prod_k^0$-predicates, and Δ_k^0 stands for $\sum_k^0 \cap \prod_k^0$. $\sum_0^0$, $\prod_0^0$, and Δ_0^0 are all synonyms for REC. Taken together, these classes form the *levels* of the *arithmetical hierarchy*, and we set $\mathrm{AH} = \cup_{k=0}^{\infty} \sum_k^0$.

What is sometimes called the *weak hierarchy theorem* of Kleene [Kle43] shows that the above classes defined by quantifiers are the same as those defined by oracles: for all $k \geq 1$, $\sum_k^0 = \mathrm{RE}^{\sum_{k-1}^0}$ and $\Delta_k^0 = \mathrm{REC}^{\sum_{k-1}^0}$. In particular, Δ_1^0 also equals REC, $\sum_1^0 = \mathrm{RE}$, and $\sum_2^0 = \mathrm{RE}^{\mathrm{RE}}$. Kleene's so-called *strong hierarchy theorem* shows that the levels of the arithmetical hierarchy are all distinct: for all $k \geq 1$,

$$\Delta_k^0 \subset \sum_k^0 \neq \prod_k^0 \subset \Delta_{k+1}^0.$$

For the *polynomial hierarchy* there is a corresponding weak theorem showing that classes defined via alternations of *polynomial-length bounded quantifiers* in front of polynomial-time decidable predicates are the same as classes defined by oracles; viz. $\sum_1^p = \mathrm{NP}$, $\sum_2^p = \mathrm{NP}^{\mathrm{NP}}$, and so on (see [Sto77, Wra77]). However, whether a corresponding strong theorem holds subsumes 'P =? NP' and many other major open questions in complexity theory.

In this treatment we have skirted around Kleene's main point that the predicates defining members of AH are exactly those that can be formalized over the language $\mathcal{L}_A$ of arithmetic. Kleene's main ingredient was the formalization of his decidable *T-predicate* $T(x, i, c)$, which expresses that c is a halting computation of M_i on input x. For greater readability we use related predicates $\mathit{Halt}(x, i, n)$ and $\mathit{Acc}(x, i, n)$, which are defined to hold iff M_i on input x respectively halts or accepts within n time steps. Then a TM M_i is total iff $(\forall x)(\exists n)\mathit{Halt}(x, i, n)$, and this $\prod_2^0$-predicate defines the language TOT.

Definition 2.3. Let $\mathcal{C}$ be any class of r.e. languages. A set $I \subseteq \mathbf{N}^+$ is an *index set* (or *partial index set*) for $\mathcal{C}$ if $\mathcal{C} = \{\, L(M_i) : i \in I \,\}$. The largest such set is the *full index set* $I_{\mathcal{C}} := \{\, i : L(M_i) \in \mathcal{C} \,\}$.

For instance, TOT is an index set for REC, but it is not equal to I_{REC}. Two full index sets that we refer to frequently are

- $I_{\mathrm{FIN}} = \{\, i : L(M_i) \text{ is finite} \,\}$, and
- $I_{\mathrm{COFIN}} = \{\, i : L(M_i) \text{ is co-finite} \,\}$.

For classes $\mathcal{C} \subseteq \mathrm{REC}$, we also use the following, which is actually an index set for $\mathrm{REC} \setminus \mathcal{C}$.

Definition 2.4. $J_{\mathcal{C}} := \{\, i : M_i \text{ is total and } L(M_i) \notin \mathcal{C} \,\}$.

The main theme of this paper is classifying index sets associated with complexity classes in the arithmetical hierarchy, and then obtaining complexity-theoretic consequences from these results. The familiar *Rice's Theorem* states that the only recursive full index sets are $I_{\mathrm{RE}} = \mathbf{N}^+$ and $I_{\emptyset} = \emptyset$. The *Rice-Shapiro Theorem* (see [Rog67]) shows that the only other classes $\mathcal{C}$ with r.e. full index sets are precisely those of the form $\mathcal{C} = \cup_{k=1}^{\infty} \{\, L \in \mathrm{RE} : U_k \subseteq L \,\}$, where U is a recursive language such that U_k is finite for each k. To pinpoint index sets at higher levels, we use the standard idea of showing completeness under recursive many-one reductions ($\leq_m$). The following example collects helpful known results that map out a lot of the territory; it also gives some useful ways of building predicates.

Example 2.1. The following index sets belong to the indicated level of the arithmetical hierarchy, and with the exception of $I_{\{\{\lambda\}\}}$, are complete for that level under $\leq_m$. (The classes Δ_k^0 for $k \geq 2$ are known to have no complete sets under $\leq_m$ at all.)

$\sum_1^0$: $I_{\{L:\lambda\in L\}} = \{\, i : (\exists m) Acc(\lambda, i, m)\,\}$.

$\prod_1^0$: $I_{\{\emptyset\}} = \{\, i : (\forall x, m) \neg Acc(x, i, m)\,\}$.

Δ_2^0 $I_{\{\{\lambda\}\}} = \{\, i : (\exists m) Acc(\lambda, i, m) \;\wedge\; (\forall x, m)\, [x > \lambda \rightarrow \neg Acc(x, i, m)]\,\}$.

$\sum_2^0$: $I_{\mathrm{FIN}} = \{\, i : (\exists y)(\forall x, m)\, [x > y \rightarrow \neg Acc(x, i, m)]\,\}$.

$\prod_2^0$: $TOT = \{\, i : (\forall x)(\exists m) Halt(x, i, m)\,\}$.

$\sum_3^0$: $I_{\mathrm{COFIN}} = \{\, i : (\exists y)(\forall x)(\exists m)\, [x > y \rightarrow Acc(x, i, m)]\,\}$.

$I_{\mathrm{REC}} = \{\, i : (\exists j)\, [j \in TOT \;\wedge\; (\forall x)\, [x \in L(M_i) \leftrightarrow x \in L(M_j)]\,]\,\}$.

To extend the completeness results for $\sum_3^0$, and show a sense in which the presence of I_{FIN} in $\sum_2^0$ is exceptional, we use the following general construction.

Theorem 2.1 ([Rog58, Rog67]) *Given any set $X \in \sum_3^0$, one can find a recursive function $\tau : \mathbf{N}^+ \rightarrow \mathbf{N}^+$ such that for all i,*

(a) $i \in X \Longrightarrow L(M_{\tau(i)})$ is co-finite,

(b) $i \notin X \Longrightarrow L(M_{\tau(i)})$ is not recursive.

Corollary 2.2 *Suppose $\mathcal{C} \subseteq$ REC, and $\mathcal{C} \supseteq A^f$ for some infinite language A. Then $I_{\mathcal{C}}$ is $\sum_3^0$-hard under $\leq_m$.*

Proof. Since A is recursive and infinite there exists a recursive function $g : \Sigma^* \rightarrow A$ that is bijectively onto A. For all i, let $\sigma(i)$ be the index of a TM that on input x automatically rejects if $x \notin A$, and otherwise accepts x iff M_i accepts $g^{-1}(x)$. Then $L(M_{\sigma(i)}) = g(L(M_i))$. Moreover, if $L(M_i)$ is not recursive, then $L(M_{\sigma(i)})$ is not recursive. Then, with reference to the statement of Theorem 2.1, for any given set $X \in \sum_3^0$:

$$i \in X \Longrightarrow L(M_{\tau(i)}) \in \mathrm{COFIN} \Longrightarrow L(M_{\sigma(\tau(i))}) \in A^f,$$

$$i \notin X \Longrightarrow L(M_{\tau(i)}) \notin \mathrm{REC} \Longrightarrow L(M_{\sigma(\tau(i))}) \notin \mathrm{REC} \Longrightarrow L(M_{\sigma\circ\tau(i)}) \notin \mathcal{C}.$$

So $\sigma \circ \tau$ reduces X to $I_{\mathcal{C}}$. □

In particular, I_{P} is $\sum_3^0$-complete [Háj79], as are the full index sets of NP, PSPACE, EXPTIME, and so on. The condition $\mathcal{C} \subseteq$ REC is needed because e.g. taking $\mathcal{C} = \mathrm{RE} \setminus \mathrm{P}$ gives $I_{\mathcal{C}} \in \prod_3^0$, so $I_{\mathcal{C}}$ cannot be $\sum_3^0$-hard with respect to $\leq_m$. Just about all complexity classes $\mathcal{C} \subseteq$ REC studied in the literature contain some infinite language together with all its finite variations, and so we read Corollary 2.2 as saying that $\sum_3^0$-completeness is the lowest level that $I_{\mathcal{C}}$ can have for "any class" (except for FIN). We show similarly that $\prod_2^0$-completeness is the lowest level for $J_{\mathcal{C}}$.

Proposition 2.3 *Suppose $\mathcal{C} \subset$ REC. Then $J_{\mathcal{C}}$ is $\prod_2^0$-hard under $\leq_m$.*

Proof. We many-one reduce TOT to $J_{\mathcal{C}}$. Let A be in REC $\setminus \mathcal{C}$. For all $i \in \mathbf{N}^+$, let $\sigma(i)$ be the index of a TM that on any input $x \in \Sigma^*$ does the following:

(i) For all $y \leq x$, simulate M_i on input y.

(ii) Accept x if and only if $x \in A$.

Step (ii) is reached if and only if M_i halts on all inputs y. If M_i is total, then $M_{\sigma(i)}$ is total and $L(M_{\sigma(i)}) = A$, so $\sigma(i) \in J_{\mathcal{C}}$. If M_i is not total, then $M_{\sigma(i)}$ is not total, and so $i \notin J_{\mathcal{C}}$. The function σ can be defined recursively, so $J_{\mathcal{C}}$ is $\prod_2^0$-hard. $\square$

In the next two sections we look at classes that meet these lower bounds, making them intuitively the easiest to define.

3 $\sum_3^0$-Classes

Call $\mathcal{C}$ a $\sum_3^0$-*class* if $I_{\mathcal{C}} \in \sum_3^0$. The main result of this section is a nearly-full characterization of $\sum_3^0$-classes by universal languages. First we give a lemma that treats an important special case.

Lemma 3.1 *If $I_{\mathcal{C}} \in \sum_3^0$ and $\mathcal{C} \supseteq$ FIN, then $\mathcal{C}$ is r.e.-presentable.*

Proof. There is a recursive 4-place predicate $R_{\mathcal{C}}$ that defines $I_{\mathcal{C}}$ in the sense that for all i, $i \in I_{\mathcal{C}} \iff (\exists a)(\forall b)(\exists c) R_{\mathcal{C}}(i,a,b,c)$. Now define

$$U = \{ \langle x, \langle i,a \rangle \rangle : (\forall b \leq x)(\exists c,m)\,[R_{\mathcal{C}}(i,a,b,c) \ \wedge \ Acc(x,i,m)]\,\}. \tag{2}$$

Since '$(\forall b \leq x)$' is a bounded quantifier, U is r.e. For any i, a, suppose $(\forall b)(\exists c) R_{\mathcal{C}}(i,a,b,c)$. Then $L(M_i) \in \mathcal{C}$, and for all x, $\langle x, \langle i,a \rangle \rangle \in U \iff (\exists m) Acc(x,i,m)$, so $L(M_i) = U_{\langle i,a \rangle}$. If $\neg(\forall b)(\exists c) R_{\mathcal{C}}(i,a,b,c)$, then $(\forall c) \neg R_{\mathcal{C}}(i,a,b_0,c)$ for some b_0, and so for all $x \geq b_0$, $\langle x, \langle i,a \rangle \rangle \notin U_{\mathcal{C}}$. Then $U_{\langle i,a \rangle}$ is finite, but this still means $U_{\langle i,a \rangle} \in \mathcal{C}$. So $\{ U_k \} \subseteq \mathcal{C}$. Last, if $L \in \mathcal{C}$ then for any M_i accepting L, there exists $a_0 \in \Sigma^*$ satisfying $(\forall b)(\exists c) R_{\mathcal{C}}(i,a_0,b,c)$, and then $L = U_{\langle i,a_0 \rangle}$. So U is universal for $\mathcal{C}$. $\square$

This is used to prove part (b) of the next theorem.

Theorem 3.2 *For any class $\mathcal{C} \subseteq$ RE:*

(a) If $\mathcal{C}$ is Δ_2^0-presentable, then $I_{\mathcal{C}} \in \sum_3^0$.

(b) If $I_{\mathcal{C}} \in \sum_3^0$ and $\mathcal{C}$ is scfv, then $\mathcal{C}$ is Δ_2^0-presentable.

This states that having a $\sum_3^0$ index set is practically equivalent to having a universal language that is recursive in the Halting Problem. Since the classes Δ_2^0 and RE are quite different, Theorem 3.2 makes an interesting contrast with Lemma 3.1: for classes $\mathcal{C}$ containing all finite sets there is a "gap" between r.e.-presentability and Δ_2^0-presentability.

Proof. (a) Let $U \in \Delta_2^0$ be universal for $\mathcal{C}$. Since $\Delta_2^0 = \prod_2^0 \cap \sum_2^0$, there exist recursive 3-place predicates R_1 and R_2 such that for all $z \in \Sigma^*$, $z \in U \iff (\forall a)(\exists b) R_1(z, a, b) \iff (\exists c)(\forall d) R_2(z, c, d)$. Then for all i,

$$L(M_i) \in \mathcal{C} \iff (\exists k)(\forall x)\,[(\exists m) Acc(x, i, m) \leftrightarrow \langle x, k\rangle \in U]$$

$$\begin{aligned}\iff (\exists k)(\forall x)\colon\ &[(\exists m) Acc(x, i, m) \ \wedge\ (\forall a)(\exists b) R_1(\langle x, k\rangle, a, b)] \ \vee \\ &[(\forall n)\neg Acc(x, i, n) \ \wedge\ (\forall c)(\exists d)\neg R_2(\langle x, k\rangle, c, d)]\end{aligned}$$

$$\begin{aligned}\iff (\exists k)(\forall x)\colon\ &[(\forall a)(\exists m)(\exists b)\,(Acc(x, i, m) \ \wedge\ R_1(\langle x, k\rangle, a, b))] \ \vee \\ &[(\forall n)(\forall c)(\exists d)\,(\neg Acc(x, i, n) \ \wedge\ \neg R_2(\langle x, k\rangle, c, d))]\end{aligned}$$

$$\begin{aligned}\iff (\exists k)(\forall x)\colon\ [(\forall n)(\forall a)(\forall c)(\exists m)(\exists b)(\exists d)\colon &(Acc(x, i, m) \ \wedge\ R_1(\langle x, k\rangle, a, b)) \vee \\ &(\neg Acc(x, i, n) \ \wedge\ \neg R_2(\langle x, k\rangle, c, d))]\end{aligned}$$

$$\iff (\exists k)(\forall x, n, a, c)(\exists m, b, d) \text{ [some recursive predicate]}.$$

This finishes the proof of (a).

(b) Let $\mathcal{C} \supseteq E^f$ for some $E \in \mathrm{RE}$, and let e be the index of a TM accepting E. Define $\mathcal{D} = \{\, L \triangle E : L \in \mathcal{C}\,\}$. Then $\mathcal{D} \supseteq \mathrm{FIN}$. For all i, $i \in I_{\mathcal{D}} \iff$

$$(\exists \ell)\,[\ell \in I_{\mathcal{C}} \ \wedge\ (\forall x)\,[(\exists m) Acc(x, i, m) \leftrightarrow ((\exists n) Acc(x, \ell, n) \oplus (\exists n') Acc(x, e, n'))]]\,,$$

where $\oplus$ here denotes exclusive-or. The abstract schema is $\exists[\exists\forall\exists \ \wedge\ \forall[\exists \leftrightarrow (\exists \oplus \exists)]]$. Since '$\exists \leftrightarrow (\exists \oplus \exists)$' can be rewritten as a disjunct of conjuncts of '$\forall$' and '$\exists$' sentences, the schema $\forall[\exists \leftrightarrow (\exists \oplus \exists)]$ reduces to $\forall\exists$. Since $\exists[\exists\forall\exists \ \wedge\ \forall\exists]$ reduces to $\exists\forall\exists$, $I_{\mathcal{D}} \in \sum_3^0$. By Lemma 3.1, $\mathcal{D}$ is r.e.-presentable. Take $R_{\mathcal{D}}$ and U from the proof of Lemma 3.1, and define:

$$V = \{\, \langle x, \langle i, a\rangle\rangle : \langle x, \langle i, a\rangle\rangle \in U \oplus (\exists n) Acc(x, e, n)\,\}. \tag{3}$$

Also abbreviate '$(\forall b \leq x)(\exists c, m) : R_{\mathcal{C}}(i, a, b, c) \wedge Acc(x, i, m)$' to $(\exists c, m) U(x, i, a, c, m)$. Then the condition in (3) is equivalent to

$$[(\exists c, m) U(x, i, a, c, m) \ \wedge\ (\forall n')\neg Acc(x, e, n')] \ \vee\ [(\forall c', m')\neg U(x, i, a, c', m') \ \wedge\ (\exists n) Acc(x, e, n)]$$

$$\iff (\exists c, m, n)(\forall c', m', n') : [U(x, i, a, c, m) \wedge \neg Acc(x, e, n')] \ \vee\ [\neg U(x, i, a, c', m') \ \wedge\ Acc(x, e, n)]$$

$$\iff (\forall c', m', n')(\exists c, m, n) : [U(x, i, a, c, m) \wedge \neg Acc(x, e, n')] \ \vee\ [\neg U(x, i, a, c', m') \ \wedge\ Acc(x, e, n)].$$

That is, the $\forall$ and $\exists$ quantifier blocks are interchangeable between the last two lines, since none of the four terms contains variables in both $\{\, c, m, n\,\}$ and $\{\, c', m', n'\,\}$. Thus V is definable both by a $\sum_2^0$-predicate and by a $\prod_2^0$-predicate, which implies that $V \in \Delta_2^0$. $\square$

Except for the fact that $\emptyset$ is a $\sum_3^0$-class that is trivially not Δ_2^0-presentable, we do not know how far the condition that $\mathcal{C}$ be scfv in Theorem 3.2(b) can be weakened. A particular consequence of Theorem 3.2(a) is:

Corollary 3.3 *If $\mathcal{C}$ is r.p. or r.e.-presentable, then $I_{\mathcal{C}} \in \sum_3^0$.* $\square$

For future reference, we note also that the collection of $\sum_3^0$-classes is closed under finite unions and intersections, and under the operation $\mathcal{C} \mapsto$ co-$\mathcal{C}$ for $\mathcal{C} \subseteq$ REC. In fact, the collection is closed under recursively presented countable unions.

4 Recursively Presentable Classes and the J Index Set

Our first result says that whether a class $\mathcal{C}$ is recursively presentable essentially depends on the index set $J_{\mathcal{C}}$.

Theorem 4.1

(a) If $\mathcal{C}$ is recursively presentable, then $J_{\mathcal{C}} \in \prod_2^0$.

(b) If $J_{\mathcal{C}} \in \prod_2^0$, and $\mathcal{C}$ is bounded and scfv, then $\mathcal{C}$ is recursively presentable.

Proof. (a) Let $\mathcal{C} = \{ U_k \}$ for some $U \in$ REC. By the *S-m-n Theorem* (see [Rog67]) there is a recursive function $\sigma : \mathbf{N}^+ \to \mathbf{N}^+$ such that for all k, $M_{\sigma(k)}$ is total and accepts U_k. Then for all i, $i \in J_{\mathcal{C}} \iff i \in \mathit{TOT} \ \wedge\ L(M_i) \notin \mathcal{C} \iff i \in \mathit{TOT} \ \wedge\ (\forall k)[L(M_i) \neq L(M_{\sigma(k)})]$. Because $M_{\sigma(k)}$ is always total, the definition of '$i \in J_{\mathcal{C}}$' expands to:

$$(\forall x, k)(\exists m, y, n_1, n_2)[\mathit{Halt}(x, i, m) \ \wedge\ \mathit{Halt}(y, i, n_1) \ \wedge\ \mathit{Halt}(y, \sigma(k), n_2) \ \wedge \\ (\mathit{Acc}(y, i, n_1) \oplus \mathit{Acc}(y, \sigma(k), n_2))].$$

Hence $J_{\mathcal{C}} \in \prod_2^0$.

(b) Let V be a recursive language such that $\mathcal{C} \subseteq \{ V_k \}$, and let A be such that $\mathcal{C} \supseteq A^f$. Let $\sigma : \mathbf{N}^+ \to \mathbf{N}^+$ be such that for all k, $M_{\sigma(k)}$ is total and accepts V_k, as in (a). By $J_{\mathcal{C}} \in \prod_2^0$, there is a recursive predicate $R(\cdot,\cdot,\cdot)$ such that for all i, (M_i total $\wedge\ L(M_i) \notin \mathcal{C}) \iff (\forall a)(\exists b)R(i, a, b)$. For all $i, a \in \mathbf{N}^+$ and $x \in \Sigma^*$, define $S(x, i, a)$ to hold iff $(\forall b \leq x)\neg R(i, a, b)$. Since the lone quantifier is bounded, $S(\cdot,\cdot,\cdot)$ is recursive. Now define

$$U = \{ \langle x, \langle k, a \rangle \rangle : [S(x, \sigma(k), a) \ \wedge\ x \in V_k] \ \vee\ [\neg S(x, \sigma(k), a) \ \wedge\ x \in A] \}.$$

Clearly U is recursive, and we claim that U is a universal language for $\mathcal{C}$. First suppose $L \in \mathcal{C}$. Then there is a k such that $L = V_k$. Since $M_{\sigma(k)}$ accepts L, we have $\sigma(k) \notin J_{\mathcal{C}}$, so $(\exists a)(\forall b)\neg R(\sigma(k), a, b)$. Thus for some a, $S(x, \sigma(k), a)$ holds for all x, and so $L = U_{\langle k,a \rangle}$.

Now consider any $U_{\langle k,a \rangle}$. If $S(x, \sigma(k), a)$ holds for all x, then $U_{\langle k,a \rangle} = V_k = L(M_{\sigma(k)})$, and $(\forall b)\neg R(\sigma(k), a, b)$. Hence $\sigma(k) \notin J_{\mathcal{C}}$, and since $M_{\sigma(k)}$ is total, this means $L(M_{\sigma(k)}) \in \mathcal{C}$, so $U_{\langle k,a \rangle} \in \mathcal{C}$. If $S(x, \sigma(k), a)$ fails for some x, then it fails for all $y \geq x$, and so $U_{\langle k,a \rangle}$ is a finite variation of A. But by hypothesis, $U_{\langle k,a \rangle}$ belongs to $\mathcal{C}$ anyway. This proves the claim, giving $\mathcal{C} = \{ U_\ell \}$. $\square$

While every r.p. class is bounded, there do exist nontrivial unbounded classes $\mathcal{C}$ that have $J_{\mathcal{C}} \in \prod_2^0$. Two examples, the former shown in [Háj79] and the latter in [Reg88], are

$$\mathrm{EQ} := \{ A \in \mathrm{REC} : \mathrm{NP}^A = \mathrm{P}^A \},$$

and the "upper cone" $\{ L \in \mathrm{REC} : A \leq_m^p L \}$ of any fixed recursive language A, where $\leq_m^p$ stands for polynomial-time many-one reducibility.

We do not know how far the other condition, namely that $\mathcal{C}$ be scfv, may be weakened. Landweber and Robertson [LR72] showed that it is needed in another situation: whereas $\mathcal{C} \cap \mathcal{D}$ is r.p. whenever $\mathcal{C}$ and $\mathcal{D}$ are r.p. and $\mathcal{C} \cap \mathcal{D}$ is scfv, they give r.p. classes $\mathcal{C}, \mathcal{D}$ such that $\mathcal{C} \cap \mathcal{D}$ is nonempty and not even r.e.-presentable. (Note: their use of the term "recursively presentable" equals ours of "r.e.-presentable," but the observation still holds.) We remark also that the collections of r.p. and r.e.-presentable classes, like that of $\sum_3^0$-classes, are closed under recursively presented countable unions.

In the next section we use Theorem 4.1(a) to show that certain bounded classes are *not* recursively presentable. The rest of this section presents results combining the properties $I_{\mathcal{C}} \in \sum_3^0$ and $J_{\mathcal{C}} \in \prod_2^0$.

Theorem 4.2 *For any class $\mathcal{C} \subseteq \mathrm{REC}$, if $J_{\mathcal{C}} \in \prod_2^0$, then $I_{\mathcal{C}} \in \sum_3^0$.*

Proof. For all i, $L(M_i) \in \mathcal{C} \iff (\exists j)\,[M_j \text{ is total } \wedge L(M_j) = L(M_i) \wedge j \notin J_{\mathcal{C}}]$, since $\mathcal{C} \subseteq \mathrm{REC}$. As we have seen, each of the three conjuncts can be written either as a $\prod_2^0$-predicate or as a $\sum_2^0$-predicate, so the definition of $I_{\mathcal{C}}$ can be placed into $\sum_3^0$ form. □

The collection of $\sum_3^0$-classes $\mathcal{C} \subseteq \mathrm{RE}$ is not closed under difference or symmetric difference, e.g. because $I_{\mathrm{RE}\setminus\mathrm{P}}$ is the complement of a $\sum_3^0$-complete set, and so is not in $\sum_3^0$. Nevertheless, we have the special case:

Theorem 4.3 *For any $\mathcal{C}, \mathcal{D} \subseteq \mathrm{REC}$, if $I_{\mathcal{D}} \in \sum_3^0$ and $J_{\mathcal{C}} \in \prod_2^0$, then $I_{\mathcal{D}\setminus\mathcal{C}} \in \sum_3^0$.*

Proof. For all i, $L(M_i) \in \mathcal{C} \iff (\exists j)[M_j \text{ is total } \wedge L(M_j) = L(M_i) \wedge j \in J_{\mathcal{C}}]$. This is different in only the last conjunct from the definition in the proof of Theorem 4.2, and likewise can be placed into $\sum_3^0$ form. □

Corollary 4.4 *(a) The difference or symmetric difference of any two classes whose 'J' index sets belong to $\prod_2^0$ is a $\sum_3^0$-class. In particular, the difference of any two r.p. classes, such as* NP *and* P*, is a (possibly empty) $\sum_3^0$-class.*

(b) If NP $\neq$ P*, then there is a recursive permutation π of $\mathbf{N}^+$ such that for all i, $L(M_i) \in \mathrm{P} \iff L(M_{\pi(i)}) \in \mathrm{NP} \setminus \mathrm{P}$.*

Proof. (a) This follows from Theorems 4.2 and 4.3. (b) If NP $\neq$ P, then NP $\setminus$ P is "scfv with an infinite set," and so by Corollary 2.2, $I_{\mathrm{NP}\setminus\mathrm{P}}$ is $\sum_3^0$-hard under $\leq_m$. By (a), $I_{\mathrm{NP}\setminus\mathrm{P}}$ is $\sum_3^0$-complete.

By the generalization of *Myhill's Theorem* to higher levels of the arithmetical hierarchy (see [Rog67]), all $\sum_3^0$-complete sets are recursively isomorphic, and so $I_{\mathrm{NP}\setminus\mathrm{P}}$ is mapped onto I_{P} by some recursive permutation of $\mathbf{N}^+$. $\square$

This gives a sense in which NP $\setminus$ P is "recursively isomorphic" to P, if and only if NP $\neq$ P. The same obtains for REC $\setminus$ P and EXPTIME $\setminus$ P, without needing any assumptions.

While the 'I' index set cannot distinguish between P and NP $\setminus$ P, the 'J' index set can. The following is obtained using the technique of "delayed diagonalization."

Theorem 4.5 ([Reg88], after [Sch82]) *Let $\mathcal{D}$ be closed downward under many-one reductions computable by Turing machines that run in linear time and log space. Let $\mathcal{C}_1, \mathcal{C}_2$ be such that $\mathcal{D} \subseteq \mathcal{C}_1 \cup \mathcal{C}_2$ and both $\mathcal{D} \setminus \mathcal{C}_1$ and $\mathcal{D} \setminus \mathcal{C}_2$ are scfv. Then at least one of $\mathcal{C}_1, \mathcal{C}_2$ is not recursively presentable. If $\mathcal{C}_1$ is r.p. and* FIN $\subseteq \mathcal{D} \setminus \mathcal{C}_2$*, then $\mathcal{C}_2$ is not r.e.-presentable either.*

Corollary 4.6 ([LLR81, CM81]) *If* NP $\neq$ P, *then* NP $\setminus$ P *is Δ_2^0-presentable but not r.p. or r.e.-presentable. In particular, $J_{\mathrm{NP}\setminus\mathrm{P}} \notin \prod_2^0$.*

The same goes for REC $\setminus$ P and EXPTIME $\setminus$ P. We remark that every $\sum_3^0$-class is either r.e.-presentable or the difference of an r.e.-presentable class with the r.p. class FIN. To complete the picture, we observe:

Corollary 4.7 *Let $\mathcal{C}_1 = (\mathrm{EXPTIME} \setminus \mathrm{P}) \cup \mathrm{FIN}$. Then $\mathcal{C}_1$ is an example of a bounded scfv. class that is r.e.-presentable but not recursively presentable.*

Proof. $\mathcal{C}_1$ is r.e.-presentable by Lemma 3.1. Were $\mathcal{C}_1$ recursively presentable, then with $\mathcal{C}_2 = \mathrm{P}$ we could write EXPTIME as the nontrivial union of r.p. (s)cfv classes, a possibility ruled out by Theorem 4.5. $\square$

The next section presents a technique for showing classes not to be r.p. in cases where we do not see a way to apply Theorem 4.5 directly.

5 Priority Arguments and Presentations

Diagonalization arguments in the literature generally concern the construction of a language L in *stages* to meet infinitely many *requirements* $R_1, R_2, R_3, \ldots$ At any given stage s, finitely many requirements have been *satisfied*, finitely many are *active*, and the infinitely many remaining ones are *not yet considered*. Of the *active* requirements, zero or more may be satisfiable at stage s, and if more than one is satisfiable, the one with *highest priority* (usually the one with lowest index) *receives attention* and is satisfied. The objective is to ensure that enough requirements are satisfied to enforce that the constructed language L has the desired property. (Whether all are satisfied or not sometimes depends on how requirements are phrased.) The argument is "zero-injury" if every requirement satisfied at some stage s remains satisfied at all stages $s' > s$.

We define a particular kind of argument that we call "diagonalization by rote." This is related to, but different from, the formulations of *P-1*, *P-2*, and *P-3 diagonalizations* by Ambos-Spies, Fleischhack, and Huwig [AFH88]. The argument is divided into *stages* so that at the end of each stage s, the membership or non-membership in L of each of the first s strings in Σ^* is determined. Equivalently, stage s begins with the *characteristic prefix* $\alpha_{s-1} = L(\lambda)\cdots L(str(s\text{–}1))$, and the output α_s of the stage is either $\alpha_{s-1}0$ or $\alpha_{s-1}1$ according to whether $str(s)$ is placed into L or into $\sim L$. If $x = str(s)$, we also write $\alpha_{L:x}$ for α_s. In place of an enumeration $[R_e]_{e=1}^{\infty}$ of requirements, we use a *satisfaction relation* $S \subseteq \mathbf{N}^+ \times \{0,1\}^*$. Then $S(e,\alpha)$ informally means "requirement R_e is satisfied by the characteristic prefix α." The argument can be regarded as "zero-injury" if S has the following *extension property*: for all $e \in \mathbf{N}^+$ and $\alpha, \beta \in \{0,1\}^*$, $S(e,\alpha) \Longrightarrow S(e,\alpha\beta)$. However, we can accommodate "injuries" as well by allowing S not to have this property. The other ingredients of our scheme are a function h such that $h(s)$ tells how many requirements are made *active* at stage s, and a "defeat function" d that tells what to do if there are no active, satisfiable requirements to work on at a given stage.

Definition 5.1. A *rote priority argument* consists of a decidable satisfaction relation $S(\cdot,\cdot)$ and computable functions $h : \mathbf{N}^+ \to \mathbf{N}^+$ and $d : \{0,1\}^* \to \{0,1\}$. It defines a unique recursive language $L(S,h,d)$ by the following infinite process:

$\alpha := \lambda$, $s := 1$
next_stage: /* Invariant: $|\alpha| = s\text{–}1$ */
  for $e := 1$ to $h(s)$ do
    if $\neg S(e,\alpha) \wedge S(e,\alpha 0)$ then do
      $\alpha := \alpha 0$, $s := s{+}1$, goto next_stage, endif;
    if $\neg S(e,\alpha) \wedge S(e,\alpha 1)$ then do
      $\alpha := \alpha 1$, $s := s{+}1$, goto next_stage, endif;
  next_e
  $\alpha := \alpha \cdot d(\alpha)$, $s := s{+}1$,
goto next_stage.

This formalism is more general than it appears at first sight: In any priority argument that effectively constructs a recursive language L, one can define "stage s" to end at the point where one has enough information to run the algorithm on the first s strings, and then define $S(e,\alpha_s)$ according to which requirements are currently deemed "satisfied" at that stage. If S has the extension property and $S(e,\alpha_s)$ holds then R_e is intuitively "cancelled," but the formalism also permits R_e to become unsatisfied (i.e., "injured") at later stages. Numbering requirements in a single sequence, assigning higher priority to lower-numbered requirements, and making the active requirements an initial sequence $R_1,\ldots,R_{h(s)}$, may lose some generality (see remarks following Proposition 5.2), but simplifies our presentation.

Our scheme focuses attention on the bounding function h, which governs the rate at which new requirements are activated, and on the complexity of the relation S. The key stipulation is that the success of the argument should not depend on this rate, so long as each requirement eventually becomes active. In zero-injury settings this is the familiar idea of arbitrarily slowing down a diagonalization. Say a function $h : \mathbf{N}^+ \to \mathbf{N}^+$ is *monotone* if for all $r, s \in \mathbf{N}^+$, $s \geq r \implies h(s) \geq h(r)$.

Definition 5.2. A property Π of recursive languages is *enforcible by rote-priority* if there is a decidable satisfaction relation $S \subseteq \mathbf{N}^+ \times \{0,1\}^*$ and a computable function $d : \{0,1\}^* \to \{0,1\}$ such that for all computable monotone functions $h : \mathbf{N}^+ \to \mathbf{N}^+$:

$$\begin{aligned} h \text{ unbounded} \quad &\implies \quad L(S,h,d) \text{ has property } \Pi, \\ h \text{ bounded} \quad &\implies \quad L(S,h,d) \text{ does not have property } \Pi. \end{aligned}$$

The property Π is enforcible by *relaxed* rote priority if this condition holds for all monotone functions h that are computable by Turing machines that run in linear time and log space.

We also say that S and d enforce Π. Note that the same S and d can enforce many different properties. The clause for bounded h says that if new requirements are not supplied, the function d takes over and *defeats* the property Π. The mechanism is not intended to say anything about the complexity of all languages that have property Π, only about the languages $L(S,h,d)$ for certain functions h. The following is essentially proved in the course of Theorem 5.4 below.

Lemma 5.1 *Given any unbounded monotone computable function f, one can construct an unbounded, monotone h in linear time and log space such that for all $s \in \mathbf{N}^+$, $h(s) \leq f(s)$.* □

The closest comparison is with the notion of *P-2 diagonalization* in [AFH88]. A *P-2* diagonalization is defined by a recursive language U that presents some subclass of P (i.e. for all e, $U_e \in$ P). It *enforces* the property Π if for all languages L such that the implications

$$(\exists^\infty x)\,[\alpha_{L:x}0 \in U_e \;\vee\; \alpha_{L:x}1 \in U_e] \implies (\exists y)\,\alpha_{L:y} \in U_e \tag{4}$$

hold for all e, L has property Π. The "$(\exists^\infty x)$" quantifier corresponds to the "for all unbounded $h\ldots$" clause in Definition 5.2, as shown below, but there is no direct counterpart to the clause for bounded h and the defeat strategy d. Indeed, the property shared by all languages is trivially enforcible by *P-2* diagonalization, but not by rote-priority. However, we show:

Proposition 5.2 *Let Π be a property enforcible by P-2 diagonalization whose complement in the recursive languages is scfv. Then Π is enforcible by rote priority, with zero injuries.*

Proof. Let A be a recursive language such that no member of A^f has property Π, and let U define the diagonalization. For all e and α define $S(e,\alpha)$ to hold iff for some i, $1 \leq i \leq |\alpha|$, $\alpha_1 \cdots \alpha_i \in U_e$. Then S has the extension property. Define $d_A(\alpha) = 1$ if $str(|\alpha|{+}1) \in A$ and $d_A(\alpha) = 0$ if not. This satisfies the clause for bounded h. Now let h be any unbounded monotone function. That $L = L(S,h,d)$ has property Π follows if we can show for all e that either

(a) for some x, $\alpha_{L:x} \in U_e$, or

(b) the set $\{ x : \alpha_{L:x}0 \in U_e \ \vee \ \alpha_{L:x}1 \in U_e \}$ is finite.

Since S has the extension property, we can call e *cancelled* at any stage s in the process of Definition 5.1 in which $S(e, \alpha_s)$ holds. If the given e is eventually cancelled, let s be the least such stage; then with $x = str(s)$, α_s is the same as $\alpha_{L:x}$, and so (a) holds. Now suppose e is never cancelled. There is a finite stage s_0 such that for all $e' < e$, either e' is already cancelled or e' is never cancelled in the process at all. Since h is unbounded and monotone, there exists $s_1 \geq s_0$ such that for all $s \geq s_1$, $h(s) \geq e$. If the set in (b) contains $str(s)$ for some $s \geq s_1$, then requirement e is the earliest active, uncancelled, satisfiable requirement at stage s, and the algorithm cancels it in going to stage s+1. This contradiction shows that the set in (b) is finite, finishing the proof. $\square$

The proof shows something stronger: the same S can be used with any defeat strategy d_A built around such an A, and the enforcement holds even when h is not computable. It also works for any h mapping $\mathbf{N}^+$ to finite subsets of $\mathbf{N}^+$ that is monotone in the sense that for all $e, s, s' \in \mathbf{N}^+$, $e \in h(s) \ \wedge \ s' \geq s \Longrightarrow e \in h(s')$, with unboundedness meaning $\cup_s h(s) = \mathbf{N}^+$. Such an h can make the active requirements not an initial sequence. We have attempted to obtain a converse to Proposition 5.2 assuming all of these provisions. Namely, let Π and S such that S has the extension property and for all d_A where $A^f \cap \Pi = \emptyset$, S and d_A enforce Π by rote-priority. For all e and $\alpha \in \{0,1\}^+$, writing α' for α with the last bit removed, define $U_e(\alpha) := S(e, \alpha) \ \wedge \ \neg S(e, \alpha')$. Let L satisfy the implication (4) for all e; then the object is to produce h and d such that $L = L(S, h, d)$. It appears that $h(s)$ must be defined so that

(a′) If there exists an e such that (a) holds with $x = str(s)$, then the set $h(s)$ includes the least such e, and

(b′) If e is such that only (b) holds, and some member of the finite set in (b) is $\geq str(s)$, then the set $h(s)$ does not include e.

This can be done with h unbounded and monotone, although even when S is decidable the least e in part (a′) may be so large that h is an uncomputable "busy-beaver" function. The defeat strategy d must be defined so that $L(S, h, d) = L$, but we do not see how to arrange that the property is defeated whenever h is bounded.

The point of these remarks is that rote-priority is more-widely applicable than *P-2* diagonalization, and whereas the implication (4) is ineffective, Definition 5.1 brings out the algorithmic content. *P-1* diagonalization in [AFH88] maps into the special case of rote-priority where $S(e, \alpha)$ has the extension property and depends on only one bit of α. More precisely, it involves relations $U_e \subseteq \mathbf{N}^+ \times \{0,1\}$, and yields $S(e, \alpha) \leftrightarrow$ for some i, $1 \leq i \leq |\alpha|$, $\langle i, \alpha_i \rangle \in U_e$. The issue of "dependent bits" comes up in the examples below. Properties enforcible by *P-3* diagonalizations are defined as for *P-2*, except that (4) is replaced by:

$$(\exists^k)(\exists^\infty x)(\exists \beta : |\beta| \leq |x|^k)\, \alpha_{L:x}\beta \in U_e \Longrightarrow (\exists y)\, \alpha_{L:y} \in U_e. \tag{5}$$

To model this, we would replace the "if" clauses in Definition 5.1 on the two single-bit extensions $\alpha 0$ and $\alpha 1$ by a search over all extensions β of length polynomial in $\log(s)$. For simplicity we do not do so here (but cf. [BT94]). Our formalism seems well suited for analyzing problems about "martingale" functions defined on characteristic prefixes and "word-decreasing self reducible sets" raised in recent work; see [Lut92, Lut93, AS94].

Our scheme also makes it easier to analyze time bounds. The best time bound for deciding $L(S,h,d)$ that one can guarantee from the proof of Proposition 5.2 includes a term $|\alpha|^{k(e)} = 2^{|x|k(e)}$, where $k(e)$ is the exponent of the polynomial runtime $n^{k(e)}$ of U_e. The problem is that $k(e)$ need not be bounded by any fixed constant, and indeed [AFH88] shows that whenever U presents all of P no language with the *P-2*-enforced property Π belongs to EXPTIME. In the examples that follow, however, not all of the bits of α need to be examined, and we can meet the following time bounds. The bounds are not intended to be optimal (indeed we have already indicated linear time for h), but are chosen just to make everything work out conveniently in EXPTIME.

Lemma 5.3 *Suppose there are constants a, b, c such that $S(e,\alpha)$ is computable in time $|\alpha|^a \cdot (\log|\alpha|)^e$, $d(\alpha)$ in time $|\alpha|^b$, and $h(s)$ in time $2^{c|s|}$. Further suppose that $h(s) \leq \log s/\operatorname{loglog} s$ for all s. Then $L(S,h,d) \in$ EXPTIME.*

Proof. Given routines for S, h, and d meeting the specified bounds, let M be the TM that on any input x runs the procedure of Definition 5.1 through stage $s = num(x)$, and then accepts iff the last bit of α_s is a '1.' We can bound the time taken by M grossly by multiplying s by the time taken for stage s. At stage s, M evaluates up to $3 \cdot h(s)$ instances of $S(e,\alpha)$, where $|\alpha| \leq s$ and $e \leq h(s)$. The time taken per instance is at most $s^a \cdot (\log s)^{h(s)}$. Since $\log s$ to the power $\log s/\operatorname{loglog} s$ equals s, the time per instance is at most s^{a+1}. Thus the total time for stage s is at most $2^{c|s|} + 3h(s)s^{a+1} + s^b$. Writing this in terms of $n = |x|$ via $s \leq 2^{n+1}$ (since $s = num(x)$) and $h(s) \leq \log s \leq 2n$ (when $s \geq 4$, since the logs are to base 2) gives a bound on the time for stage s of $2^{cn} + 6n(2^{n+1})^{a+1} + (2^{n+1})^b$. Multiplying this by $s = 2^{n+1}$ stages still leaves the overall time bounded by 2^{kn} for some constant k and all n. Hence $L(S,h,d) \in$ EXPTIME. $\square$

5.1 Some examples

Example 5.1. *Membership in* EXPTIME $\setminus$ P*:* This is enforced using a universal language U for P by defining $S(e,\alpha)$ to hold if for some $m \leq |\alpha|$, $\alpha_m = 1 \iff str(m) \notin U_e$, and by taking $d(\alpha) := 0$ for all α. If h is unbounded, the diagonalization succeeds; if h is bounded, then $L(S,h,d)$ is finite, and so belongs to P.

If we assume that U_e is accepted in time n^e, then $S(e,\alpha)$ is computed in time $|\alpha| \cdot n^e$, where n is the length of the longest string x indexed by α. Since $num(x) = |\alpha|$, $|x| = \lfloor \log|\alpha| \rfloor$, so $S(e,\alpha)$ is computed within time $|\alpha| \cdot (\log|\alpha|)^e$. Hence Lemma 5.3 applies, and so $L(S,h,d) \in$ EXPTIME. (Of course, one can diagonalize out of P into smaller time classes than EXPTIME. This is doable by choosing h to grow more slowly, and by keeping explicit track of which e have been "cancelled" at previous stages.)

Example 5.2. P-*immunity:* A language A is P-*immune* if A is infinite, but no infinite subset of A belongs to P. Take U to be universal for P as above, and define for all e, α :

$$\begin{aligned} S(2e\text{–}1, \alpha) &\leftrightarrow \text{ for some } n \le |\alpha|,\ str(n) \in U_e\ \wedge\ \alpha_n = 0, \\ S(2e, \alpha) &\leftrightarrow \text{ for some } n \text{ such that } e \le n \le |\alpha|,\ \alpha_n = 1. \end{aligned}$$

Also define $d(\alpha) = 1$ for all α.

As in all of these examples, S has the extension property. If U_e is infinite, then there are infinitely-many m such that for all $\alpha \in \{0,1\}^m$, $S(2e\text{–}1, \alpha 0)$ holds. Hence the corresponding requirement R_{2e-1} := '$U_e \not\subseteq L$' eventually receives highest priority and is satisfied. The even-numbered requirements all get satisfied, and collectively ensure that $L(S, h, d)$ is infinite. This happens so long as h is unbounded. If h is bounded, then $L(S, h, d)$ is co-finite, and hence not P-immune. Note that this also works for the class P-IMMUNE $\cup$ FIN.

Actually, since we can assume that h is $o(n)$, we can dispense with the even-numbered requirements. It is impossible that all but finitely many stages s bring the satisfaction of a new requirement, so our choice of $d(\cdot)$ ensures that $L(S, h, d)$ is infinite. Again presuming that U_e is computable in time n^e, $S(2e\text{–}1, \alpha)$ is computable within time $|\alpha| \cdot (\log |\alpha|)^e$, so $L \in$ EXPTIME.

Example 5.3. *Bi-immune sets for* P*:* A language A is P-*biimmune* if both A and $\sim\!A$ are P-immune. This time we define:

$$\begin{aligned} S(2e\text{–}1, \alpha) &\leftrightarrow \text{ for some } n \le |\alpha|,\ str(n) \in U_e\ \wedge\ \alpha_n = 0, \\ S(2e, \alpha) &\leftrightarrow \text{ for some } n \le |\alpha|,\ str(n) \in U_e\ \wedge\ \alpha_n = 1. \end{aligned}$$

Again take $d(\alpha) := 1$ for all α. By much the same reasoning as in the last example, if U_e is infinite then both requirements corresponding to U_e are eventually satisfied. By Lemma 5.3, the property of P-biimunity within EXPTIME is also enforcible by rote zero-injury priority.

Example 5.4. *Strong* P-*biimmunity:* A language A is *strongly* P-*biimmune* ([BS85]; see also [BH77] and "incompressible" in [Lut93]) if for every polynomial-time computable function f, either f is 1-1 a.e. or there exist strings x, y such that $x \in A$, $y \notin A$, and $f(x) = f(y)$. In consequence, no function f that is not 1-1 a.e. can be a correct $\le_m^p$-reduction from A (to any other language B). Any strongly P-biimmune language A is P-biimmune, while $A \oplus A$ is P-biimmune but not strongly so. Given a standard enumeration $[f_e]_{e=1}^{\infty}$ of FP, define

$$S(e, \alpha) \leftrightarrow \text{ for some } m, n \le |\alpha|,\ \alpha_m = \text{‘}0\text{’}\ \wedge\ \alpha_n = \text{‘}1\text{’}\ \wedge\ f_e(m) \ne f_e(n),$$

where the arguments to f_e are technically $str(m)$ and $str(n)$. Again take $d(\cdot)$ to be the "when in doubt, reject" strategy. If f_e is not 1-1 a.e., then there are arbitrarily large y such that for some $x \le y$, $f(x) = f(y)$. This means that we *still* have the property $(\exists^\infty n)(\forall \alpha \in \{0,1\}^n)\,[S(e, \alpha 0) \vee S(e, \alpha 1)]$ in this zero-injury setting, so that the requirement for f_e eventually receives attention and is satisfied.

Strong P-biimmunity is not enforcible by a *P-1* diagonalization, since P-biimmunity is shown to be the strongest such property in [AFH88]. Here, computing $S(e, \alpha)$ requires examining at least two bits of α rather than one, and the time is $|\alpha|^2 \cdot (\log |\alpha|)^e$. Lemma 5.3 still applies, so the "rote" strongly P-biimmune sets do belong to EXPTIME. However, [AFH88] shows that there is a strongest property Π that is enforcible by *P-2* diagonalization, and that no language in EXPTIME has this Π.

5.2 Rote-priority classes are not r.p.

What drives our main theorem is that the languages $L(S, h, d)$ can absorb arbitrarily large doses of the "defeat strategy" when h is slow-growing but unbounded, and yet still enjoy property Π.

Theorem 5.4 *Let $\mathcal{C}$ be a class of recursive languages that is enforcible by relaxed rote-priority. Then there is a recursive function $\sigma : \mathbf{N}^+ \to \mathbf{N}^+$ such that for all i, $M_{\sigma(i)}$ is total and:*

$$\begin{aligned} L(M_i) \text{ infinite} &\implies L(M_{\sigma(i)}) \in \mathcal{C}, \\ L(M_i) \text{ finite} &\implies L(M_{\sigma(i)}) \notin \mathcal{C}. \end{aligned}$$

Thus $\sigma(\cdot)$ many-one reduces I_{FIN} to $J_{\mathcal{C}}$. Hence $J_{\mathcal{C}} \notin \prod_2^0$, so $\mathcal{C}$ is not recursively presentable.

Proof. For each i we design a Turing machine H_i that takes a numerical input m of length $n = |str(m)|$ and does the following: H_i first marks off $l = \lceil \log n \rceil$ cells on a worktape. This can be done in n steps. Then H_i begins a "dovetail" simulation of M_i (from the fixed enumeration of TMs) on inputs $\lambda, 0, 1, 00, \ldots$, doing as much as it can within the l cells for n steps. Here "dovetail" means that H_i simulates one step of M_i on λ, then one step on 0, then one more step on λ, one on 0, then one on 1, then back to λ, and so on. Finally, H_i outputs the number of inputs that M_i accepted during its simulation of M_i. Then H_i runs in real time and log space, and the function h_i it computes is monotone, and unbounded iff $L(M_i)$ is infinite.

Given the recursive predicate S and function d from Definition 5.2, define $\sigma(i)$ to be the code of the TM that simulates the algorithm of Definition 5.1 with S, d, and h_i. This σ is a primitive recursive function, and for each i, accepts $M_{\sigma(i)}$ is total and accepts $L(S, h_i, d)$. Then for any i, if $L(M_i)$ is finite then the boundedness of h_i makes $L(M_{\sigma(i)}) \notin \mathcal{C}$, and since $M_{\sigma(i)}$ is total, $i \in J_{\mathcal{C}}$. If $L(M_i)$ is infinite then h_i is unbounded, so $L(S, h_i, d) \in \mathcal{C}$, and so $i \notin J_{\mathcal{C}}$. ☐

Remark. A stronger construction that makes each h_i computable in *real time* (i.e., time n+1) and log space follows from the proof of Theorem 4.1(a) in [Reg92]. The same applies to h in Lemma 5.1.

Via Theorem 4.1 and Example 5.1 above, this yields another proof that EXPTIME $\setminus$ P is not recursively presentable; ditto (EXPTIME $\setminus$ P) $\cup$ FIN. Likewise, the class of P-immune sets is EXPTIME (or in any larger class) is not r.p. Most of interest to us,

Corollary 5.5 *The classes of languages in* EXPTIME *that are (a)* P*-immune or finite, (b)* P*-biimmune, or (c) strongly* P*-biimmune, are not recursively presentable.*

Proof. As shown above, each of these classes is obtainable by (relaxed) rote-priority. $\square$

Corollary 5.6 *For any sound, recursively axiomatized formal system $\mathcal{F}$, there are biimmune languages A in* EXPTIME *that are provably (in PA) infinite and co-infinite, but that are not provably (in $\mathcal{F}$) recursive and bi-immune.*

Proof. Let D be any language in P that PA can prove to be infinite and co-infinite, and adjust the defeat-strategy in the rote argument for P-biimmunity to read $d(\alpha) := 1$ if $str(|\alpha|+1) \in D$, $d(\alpha) := 0$ otherwise. PA can prove that all finite variations of D, and also all P-biimmune sets in EXPTIME, are infinite and co-infinite. In addition, PA can prove the correctness of our rote-priority argument for P-biimmunity in EXPTIME. Hence, with reference to the construction in Theorem 5.4,

$$\text{PA} \vdash (\forall i)\, \text{`}L(M_{\sigma(i)}) \text{ is infinite and co-infinite,'}$$

regardless of whether $L(M_i)$ is finite or not. By Theorem 5.4 the class $\mathcal{C} = \{\, L(M_{\sigma(i)}) : L(M_{\sigma(i)}) \text{ is P-biimmune} \,\}$ has $J_{\mathcal{C}}$ $\sum_2^0$-hard. If $\mathcal{C}$ had an $\mathcal{F}$-provable representation by total machines, then there would be a recursive universal language for $\mathcal{C}$, and Theorem 4.1(a) would give $J_{\mathcal{C}} \in \prod_2^0$, a contradiction. $\square$

More intuitively put, A can be taken to be a bi-immune set whose symmetric difference with D is not provably infinite. A is produced by rote whenever the function mapping $m \mapsto \min\{\, n : n$ steps suffices to find m strings in $L(M_i) \,\}$ is total but outgrows every function that $\mathcal{F}$ can prove to be total.

The desire to use obtain the above conclusion about unprovability without the tacit restriction to provable formulas of *total* TM indices motivates the following:

Open Problem 1. Are any of the classes in Corollary 5.5 r.e.-presentable?

We suspect not. However, this conclusion cannot follow by a general analysis of rote-priority, because $\mathcal{C} := (\text{EXPTIME} \setminus \text{P}) \cup \text{FIN}$ is an example of an r.e.-presentable class enforced by rote-priority. This $\mathcal{C}$ has $J_{\mathcal{C}} \in \prod_3^0$, so $J_{\mathcal{C}}$ is not $\sum_3^0$-hard under $\leq_m$. This prompts us to ask,

Open Problem 2. For any of the classes $\mathcal{C}$ in Examples 5.1–5.4, is $J_{\mathcal{C}}$ $\sum_3^0$-hard?

We are further interested in knowing just which properties can be brought within the compass of Definitions 5.1 and 5.2.

6 Non-$\sum_3^0$ Classes

We consider the question of whether all languages in a given complexity class $\mathcal{C}$ "provably belong to $\mathcal{C}$." In this section we deal with a particular formal system, called $\text{T}\Pi_2$, which is sound but not recursively axiomatizable. $\text{T}\Pi_2$ is a superset of several theories studied in the literature, including

Basic Number Theory as described by Lipton and DeMillo [Lip78, DL80]. This is interesting because of the contention by the authors of these papers, rebutted in [Lei82] and [JY85], that all constructive methods that computer scientists would ever use are formalizable in this theory. Before defining $T\Pi_2$, we note some technical aspects of formalizing predicates in arithmetic. The following should be contrasted with Definition 2.1.

Definition 6.1. A formula ϕ with free variables $x_1, \ldots, x_m$ is a *pure $\sum_k$-formula* over $\mathcal{L}_A$ if it is a string

$$`\phi\text{'} = (\exists y_1, \ldots, y_{i_1})(\forall y_{i_1+1}, \ldots, y_{i_2}) \cdots (Q_k y_{i_{k-1}+1}, \ldots, y_{i_k})\, \psi, \tag{6}$$

where Q_k is '$\forall$' if k is even, '$\exists$' if k is odd, and ψ is a formula over $\mathcal{L}_A$ with variables $x_1, \ldots, x_m, y_1, \ldots, y_{i_k}$ and *no quantifiers at all.*

If ψ is allowed to contain *bounded quantifiers* of the form $(\exists y < t)$ or $(\forall y < t)$ where t is an arithmetical term, then it is customary to call ψ a *Δ_0-formula* and ϕ a *$\sum_k^0$-formula*. The *Matijasevic-Robinson-Davis-Putnam Theorem* (see [Mat73, MR75, Smo91]) implies that for every r.e. language L there is a pure $\sum_1$-formula ϕ such that $L = L(\phi)$, in fact where ψ has the form $p(x, y_1, y_2, \ldots, y_{12}) = 0$ for some polynomial p in 13 variables. Thus for $k \geq 1$ all languages in $\sum_k^0$ are definable by *pure* $\sum_k$-formulas.

Definition 6.2. (a) For all $k \geq 0$, A_k denotes the set of pure $\prod_k$-sentences over $\mathcal{L}_A$ that are true in the standard model of arithmetic.

(b) $\mathcal{Q}$ is the theory over $\mathcal{L}_A$ whose axioms are those of Peano Arithmetic minus the induction schema, but including '$(\forall n)[n \neq 0 \rightarrow (\exists m)(n = m+1)]$' (see [BJ74]).

(c) $T\Pi_2$ is the theory obtained by adding A_2 to the axioms of $\mathcal{Q}$.

The analogous collections of true pure $\sum_k$-sentences are called 'V_k' in [Rog67]. Since the axioms of $\mathcal{Q}$ are themselves pure $\prod_2$-sentences, we could just have said that $T\Pi_2$ has A_2 as axiom set. The axiom set A_2 is *not* recursive; instead, it belongs to $\prod_2^0$ and is complete under many-one reduction (see chapter 14 of [Rog67]).

In fact, $T\Pi_2$ has all true $\sum_3$-sentences as theorems, since $(\exists n)\psi(n)$ logically follows from $\psi(n_0)$ for some fixed n_0 that makes $\psi(n_0)$ true. Homer and Reif [HR86] studied the analogously-defined theory with V_2 as axiom set; for similar reasons, they could just as well have taken A_1 as the axiom set, giving the same theory studied by O'Donnell et al. [O'D79, FOL83, KOR87]. $T\Pi_2$ is weaker than full Peano Arithmetic in that the PA induction schema

$$(\psi(0) \;\wedge\; (\forall n)[\psi(n) \rightarrow \psi(n+1)]) \rightarrow (\forall n)\psi(n)$$

is a true $\sum_3$-sentence generally only when $\psi(n)$ is a $\prod_2$-sentence. It is stronger in that its axiom set includes many statements not provable or known to be provable in PA or in set theory (ZF). In particular, the *Riemann Hypothesis* is provably (in ZF; see [Rog67]) equivalent to a $\prod_1$-statement over $\mathcal{L}_A$, and '$P \neq NP$' can be put into $\prod_2$-prenex form, so it is decided in $T\Pi_2$.

What matters for us is the observation that any $\sum_3^0$-class $\mathcal{C}$ can be defined by a pure $\sum_3^0$ formula with one free variable i, so that for all $i \in I_{\mathcal{C}}$, the sentence '$L(M_i) \in \mathcal{C}$' is a theorem of $\mathrm{T}\Pi_2$. We take this to mean that any $\sum_3^0$-class provably contains all of its members with respect to $\mathrm{T}\Pi_2$. The intuition that motivates the following theorem is that "natural" non-$\sum_3^0$ classes $\mathcal{C}$ do not provably contain *any* of their members. Technically this isn't so, since one can find examples where $\mathcal{C} = \mathcal{D} \cup \mathcal{E}$, $I_{\mathcal{C}} \notin \sum_3^0$, but $I_{\mathcal{D}} \in \sum_3^0$ and all members of $\mathcal{D}$ provably belong to $\mathcal{C}$ as well. But we prove that any non-$\sum_3^0$ class $\mathcal{C}$ can always be partitioned into a $\sum_3^0$-class $\mathcal{D}$ and a "jagged edge" $\mathcal{E}$ that does not provably contain any of its members. The gist of the proof is that the subclass $\mathcal{D}$ of languages A such that $\mathrm{T}\Pi_2 \vdash$ '$L(M_j) \in \mathcal{C}$' for *some* M_j accepting A is a $\sum_3^0$-class.

Theorem 6.1 *Suppose $\mathcal{C} \subseteq \mathrm{RE}$ with $I_{\mathcal{C}} \notin \sum_3^0$, and let $\psi_{\mathcal{C}}(\cdot)$ be any definition of $I_{\mathcal{C}}$ over $\mathcal{L}_A$. Then there exist classes $\mathcal{D}, \mathcal{E} \subseteq \mathrm{RE}$ such that*

(a) $\mathcal{D} \cup \mathcal{E} = \mathcal{C}$ and $\mathcal{D} \cap \mathcal{E} = \emptyset$,

(b) $I_{\mathcal{D}} \in \sum_3^0$ (therefore $I_{\mathcal{E}} \notin \sum_3^0$), and

(c) for all $i \in I_{\mathcal{E}}$, $\mathrm{T}\Pi_2 \nvdash \psi_{\mathcal{C}}(i)$.

Proof. First, if there is no definition of $I_{\mathcal{C}}$ over $\mathcal{L}_A$ then the conclusion follows trivially. Given such a definition $\psi_{\mathcal{C}}$, define $D = \{ j : \mathrm{T}\Pi_2 \vdash \psi_{\mathcal{C}}(j) \}$, and define $\mathcal{D} = \{ L(M_j) : j \in D \}$. It suffices to show that $I_{\mathcal{D}} \in \sum_3^0$; then setting $\mathcal{E} = \mathcal{C} \setminus \mathcal{D}$ completes the proof of the theorem. An informal definition for $I_{\mathcal{D}}$ is given for all i by

$$L(M_i) \in \mathcal{D} \iff (\exists j)\,[\text{`}L(M_i) = L(M_j)\text{'} \;\wedge\; \text{`}\mathrm{T}\Pi_2 \vdash \psi_{\mathcal{C}}(j)\text{'}]. \tag{7}$$

We have seen in Section 4 how to transcribe '$(\exists j)\,[L(M_i) = L(M_j)]$' into $\sum_3^0$ form. We do the same for '$(\exists j)\,[\mathrm{T}\Pi_2 \vdash \psi_{\mathcal{C}}(j)]$.' Conjoining these two clauses then gives a $\sum_3^0$-formula defining $I_{\mathcal{D}}$.

If a sentence ψ is derivable in $\mathrm{T}\Pi_2$, then there is a finite list $\theta_1, \ldots, \theta_k$ of true pure $\prod_2$ sentences over $\mathcal{L}_A$ such that the implication $(\theta_1 \wedge \ldots \wedge \theta_k) \to \psi$ is a theorem of $\mathcal{Q}$. Basically, $\theta_1, \ldots, \theta_k$ comprise the true $\prod_2$ statements used as axioms in the derivation of ψ in $\mathrm{T}\Pi_2$; all other axioms and rules of inference belong already to $\mathcal{Q}$. $\mathcal{Q}$ can also prove that the conjunction $\theta_1 \wedge \ldots \wedge \theta_k$ is equivalent to a single pure $\prod_2$-sentence θ over $\mathcal{L}_A$. $\mathcal{Q}$ is recursively (in fact, finitely) axiomatizable, and we can define over $\mathcal{L}_A$ a recursive predicate '$\mathcal{Q}Proof$' for it such that for all sentences ψ, $\mathrm{T}\Pi_2 \vdash \psi$ if and only if $(\exists d)\mathcal{Q}Proof(\text{`}\theta \to \psi,\text{'}\ d)$ holds for some $\theta \in A_2$. Regarding 'ψ' as '$\psi_{\mathcal{C}}(j)$' for any given j, we have,

$$\mathrm{T}\Pi_2 \vdash \psi_{\mathcal{C}}(j) \iff (\exists \theta, d)\,[\text{`}\theta \in A_2\text{'} \;\wedge\; \mathcal{Q}Proof(\text{`}\theta \to \psi_{\mathcal{C}}(j),\text{'}\ d)]. \tag{8}$$

By the fact that $A_2 \in \prod_2^0$ and the weak hierarchy theorem, there exists a recursive predicate $S(\cdot,\cdot\cdot)$ such that for all η, $\eta \in A_2 \iff (\forall a)(\exists b)S(\eta, a, b)$. Then for all $j \in \mathbf{N}^+$,

$$\mathrm{T}\Pi_2 \vdash \psi_{\mathcal{C}}(j) \iff (\exists \theta, d)(\forall a)(\exists b)\,[S(\theta, a, b) \;\wedge\; \mathcal{Q}Proof(\text{`}\theta \to \psi_{\mathcal{C}}(j),\text{'}\ d)]. \tag{9}$$

The predicate in $[\ldots]$ is recursive. Substituting (9) into (7) then gives the required $\sum_3^0$-definition of $I_{\mathcal{D}}$. $\square$

Remark. D actually equals $I_{\mathcal{D}}$ if one assumes that the definition $\psi_{\mathcal{C}}$ of $I_{\mathcal{C}}$ is "well-behaved" in the following sense: for all TM indices i and j, $\mathrm{T\Pi}_2 \vdash [\psi_{\mathcal{C}}(i) \wedge \text{`}L(M_i) = L(M_j)\text{'}] \rightarrow \psi_{\mathcal{C}}(j)$. Since '$L(M_i) = L(M_j)$' is expressible by a $\prod_2^0$ formula, this yields the implication: if $\mathrm{T\Pi}_2 \vdash \psi_{\mathcal{C}}(i)$, then $\mathrm{T\Pi}_2 \vdash \psi_{\mathcal{C}}(j)$. Not all definitions are so well-behaved, however: given any $\psi_{\mathcal{C}}(\cdot)$ as above, one can define $\phi_{\mathcal{C}}(i) = \psi_{\mathcal{C}}(i) \wedge$ 'if i is even then η,' where η is some true sentence over $\mathcal{L}_A$ that is not a theorem of $\mathrm{T\Pi}_2$.

We envisage applying Theorem 6.1 to complexity-theoretic properties studied in the literature whose natural definitions do not have any instances provable in $\mathrm{T\Pi}_2$. P-biimmunity is a prominent example. Take any recursive universal language U for P; then for all i, $L(M_i)$ is P-biimmune if and only if

$$(\forall k): \text{ `}U_k \text{ is finite' } \vee \ [\text{`}U_k \cap L(M_i) \neq \emptyset\text{'} \ \wedge \ \text{`}U_k \setminus L(M_i) \neq \emptyset\text{'}].$$

This can be expressed formally by

$$\begin{aligned} Bim(i) \ &:= \ (\forall k)(\exists w, y, z, m)(\forall x, n): [x \geq w \rightarrow x \notin U_k] \ \vee \\ &\qquad [(y \in U_k \ \wedge \ Acc(y, i, m)) \ \wedge \ (z \in U_k \ \wedge \ \neg Acc(z, i, n))], \end{aligned}$$

which is a $\prod_3^0$ formula defining the full index set of the class of P-biimmune languages. This does not contradict Corollary 2.2 because this class contains nonrecursive sets. Our interest focuses on the subclass RBM of recursive P-biimmune sets, and the relevant result about it is:

Theorem 6.2 ([Lei82]) $\mathrm{T\Pi}_2 \not\vdash \text{`}(\exists i)\,[Bim(i) \ \wedge \ i \in I_{\mathrm{REC}}]\text{.'}$

That is, $\mathrm{T\Pi}_2$ fails to prove the existence of a recursive P-biimmune set, so RBM does not provably contain any of its members *under* the definition given by $Bim(\cdot)$. Does this entail that $I_{\mathrm{RBM}} \notin \sum_3^0$? This is plausible because all true $\sum_3$-sentences are theorems of $\mathrm{T\Pi}_2$, but all we can conclude from Theorem 6.2 is that $\mathrm{T\Pi}_2$ cannot prove that I_{RBM} belongs to $\sum_3^0$. Put another way, there is no $\sum_3$-formula that $\mathrm{T\Pi}_2$ can prove equivalent to $Bim(\cdot)$. There may still exist, however, a characterization of bi-immunity that yields a $\sum_3^0$-definition of I_{RBM}, but whose equivalence to $Bim(\cdot)$ requires a much more powerful formal system to prove. We nevertheless consider Theorem 6.3 "good evidence" for a negative answer to the following

Open Problem 3. Is $I_{\mathrm{RBM}} \in \sum_3^0$? Is RBM r.e.-presentable?

These are related to the open questions in the previous section. It may be possible to resolve them by reducing a $\prod_3^0$-complete set to I_{RBM}, but we have not achieved this.

Pure counting arguments show that subrecursive non-$\sum_3^0$-classes exist. We offer as a second candidate of "natural" interest a class $\mathcal{H}$ which carries the implication that if $I_{\mathcal{H}} \notin \sum_3^0$, then all the levels of the polynomial hierarchy PH are distinct. Recall that $\mathrm{EQ} = \{\, A \in \mathrm{REC} : \mathrm{NP}^A = \mathrm{P}^A \,\}$.

Definition 6.3. $\mathcal{H} := \bigcap_{A \in \mathrm{EQ}} \mathrm{P}^A$.

That is, $\mathcal{H}$ consists of all languages that "collapse" to P^A whenever A is a recursive oracle making $\mathrm{NP}^A = \mathrm{P}^A$. We do not know whether the restriction that A be recursive matters in the definition.

Theorem 6.3 *(a)* $\mathrm{PH} \subseteq \mathcal{H} \subseteq \mathrm{PSPACE}$.

(b) If PH *collapses, i.e. if* $\mathrm{PH} = \sum_k^p$ *for some* k*, then* $\mathcal{H} = \mathrm{PH}$.

(c) $\mathrm{P}^{\mathcal{H}} = \mathrm{NP}^{\mathcal{H}} = \mathcal{H}$ *(i.e.,* $\cup_{L \in \mathcal{H}} \mathrm{P}^L = \cup_{L \in \mathcal{H}} \mathrm{NP}^L = \mathcal{H}$*).*

(d) $I_{\mathcal{H}} \in \prod_4^0 \setminus \prod_3^0$.

Proof. (a) For any A such that $\mathrm{NP}^A = \mathrm{P}^A$, we actually have $\mathrm{PH}^A = \mathrm{P}^A$, and since $\mathrm{PH} \subseteq \mathrm{PH}^A$ for all $A \subseteq \Sigma^*$, $\mathrm{PH} \subseteq \mathcal{H}$. Moreover, if A is chosen to be the known PSPACE-complete language QBF, then $\mathrm{P}^A = \mathrm{NP}^A = \mathrm{PSPACE}$, so $\mathcal{H} \subseteq \mathrm{PSPACE}$.

(b) If $\mathrm{PH} = \sum_k^p$, then using the $\sum_k^p$-complete sets B_k from [Wra77], we have $\mathrm{NP}^{B_k} = \mathrm{P}^{B_k} = \sum_k^p$, whence $\mathcal{H} = \sum_k^p$ by (a).

(c) Clearly $\mathcal{H} \subseteq \cup_{L \in \mathcal{H}} \mathrm{P}^L \subseteq \cup_{L \in \mathcal{H}} \mathrm{NP}^L$. Suppose $B \in \cup_{L \in \mathcal{H}} \mathrm{NP}^L$; then $B \in \mathrm{NP}^L$ for some $L \in \mathcal{H}$. For all $A \in \mathrm{EQ}$, $L \in \mathrm{P}^A$, and since $\mathrm{NP}(\mathrm{P}^A) = \mathrm{NP}^A$, $B \in \mathrm{NP}^A$. But $\mathrm{NP}^A = \mathrm{P}^A$ for all $A \in \mathrm{EQ}$, and so $B \in \mathrm{P}^A$ for all $A \in \mathrm{EQ}$. Hence $B \in \mathcal{H}$.

(d) Since $I_{\mathcal{H}}$ is $\sum_3^0$-hard by Corollary 2.2, $I_{\mathcal{H}} \notin \prod_3^0$. The condition '$L(M_i) \in \mathcal{H}$' can be expressed as $(\forall j) : j \in I_{\mathrm{EQ}} \to$ '$L(M_i) \in \mathrm{P}^{L(M_j)}$.' Since $J_{\mathrm{EQ}} \in \prod_2^0$, I_{EQ} is $\sum_3^0$-definable by Theorem 4.2. So is the implicand '$L(M_i) \in \mathrm{P}^{L(M_j)}$.' Hence the definition of $I_{\mathcal{H}}$ has the skeletal form $\forall[\exists\forall\exists \to \exists\forall\exists]$, which reduces to $\forall[\forall\exists\forall \vee \exists\forall\exists]$, and thence to $\forall\exists\forall\exists$. ☐

If the leading "for all" cannot be removed from the definition of $\mathcal{H}$, then $\mathcal{H} \neq \mathrm{PH}$, from which it follows that

$$\mathrm{P} \subset \mathrm{NP} \neq \mathrm{coNP} \subset \Delta_2^p \subset \textstyle\sum_2^p \subset \sum_3^p \subset \ldots \subset \mathrm{PH} \subset \mathrm{PSPACE}.$$

A proof of this may seem today a bit too much to ask for. We actually conjecture that $\mathcal{H} = \mathrm{PH}$, which is rendered more plausible but not proven by the result of Nisan and Wigderson [NW88] that for any $L \notin \mathrm{PH}$, the class of oracles A putting $L \in \mathrm{PH}^A$ has measure zero. A proof of $\mathcal{H} = \mathrm{PH}$ would show that on condition $\mathrm{PH} \neq \mathrm{PSPACE}$, there exists a recursive oracle set A such that $\mathrm{P}^A = \mathrm{NP}^A$ but NP^A does not contain PSPACE. Since $\mathrm{PSPACE} \subseteq \mathrm{PSPACE}^A$ for all A, this would cast more light on the construction by Ko [Ko89] of an oracle set A such that $\mathrm{P}^A = \mathrm{NP}^A \neq \mathrm{PSPACE}^A$.

7 Conclusion and Prospects

It is a truism to say that the easiest concepts to study are those with the simplest definitions. We equate the difficulty of studying a complexity class $\mathcal{C}$ with the positions of $I_{\mathcal{C}}$ and $J_{\mathcal{C}}$ in the arithmetical hierarchy. We have characterized those classes $\mathcal{C}$ that achieve the lowest possible levels, both in terms of whether $\mathcal{C}$ is a provable property of languages and whether $\mathcal{C}$ is recursively, r.e.-, or Δ_2^0-presentable. By showing that certain concepts such as bi-immunity correspond to non-r.p. classes ($J_{\mathcal{C}} \notin \prod_2^0$), we have provided some explanation of why they are often considered more abstruse than most of the other complexity classes and notions that researchers have devised. In view of the common opinion that more-sophisticated techniques than we currently know will be needed to resolve P =? NP and similar hard questions, however, we contend that such classes should be explored more. The class $\mathcal{H}$ in the last section is a rather blunt candidate. In [Soa87] one may find arguments of noticeably greater difficulty applied to show that certain full index sets $I_{\mathcal{C}}$ are not in $\sum_3^0$, and perhaps these arguments have the right level of sophistication.

Another important goal in complexity theory is to establish that methods of a certain level *cannot* solve these questions, or cannot prove that certain properties of languages hold. Section 6 may prompt further investigation of the theory $T\Pi_2$. Theorem 5.4 makes a contribution in this direction: no recursively presentable property can be enforced by a rote-priority argument. The rote-priority mechanism deserves attention in its own right, in light of recent interest in [AFH88] and concepts important to [Lut92, Lut93, AS94, BT94]. We look forward to further work on the formal difficulty of complexity-theoretic arguments.

Acknowledgments All of this research except Section 5 was conducted while I was at Merton College, Oxford University, as a student and then a Junior Fellow. I am grateful to Dominic Welsh, Angus Macintyre, and Uwe Schöning for their comments on this material, and to the Fellows of Merton for their support. The research in Section 5 was conducted in 1988 while I was a Post-Doctoral Visitor of the Mathematical Sciences Institute of Cornell University, and I thank them and the Cornell Computer Science Department for their support. James Royer gave me an example of a class $\mathcal{C}$ that is a recursively presented intersection of r.p. classes and has $J_{\mathcal{C}}$ $\sum_3^0$-complete (see [Reg88]), and this prompted me to do some of the new research in Section 5.

References

[AFH87] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial computable sets. *Theor. Comp. Sci.*, 51:177–204, 1987.

[AFH88] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizing over deterministic polynomial time. In *Proc. 1st Workshop on Computer Science Logic*, volume 329 of *Lect. Notes in Comp. Sci.*, pages 1–16. Springer Verlag, 1988. A full version is Forschungsbericht Nr. 1/88, Fachbereich Informatik der Universität Oldenburg.

[AS94] E. Allender and M. Strauss. Measure on small complexity classes, with applications for BPP. Technical Report DIMACS TR 94-18, Rutgers University and DIMACS, 1994. To appear in the proceedings of FOCS'94.

[Bak79] T. Baker. On 'provable' analogs of P and NP. *Math. Sys. Thy.*, 12:213–218, 1979.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM J. Comput.*, 6:305–321, 1977.

[BJ74] G. Boolos and R. Jeffrey. *Computability and Logic.* Cambridge University Press, 1974.

[BS85] J. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *Math. Sys. Thy.*, 18:1–10, 1985.

[BT94] H. Buhrman and L. Torenvliet. On the cutting edge of relativization: the resource-bounded injury method. In *Proc. 21st Annual International Conference on Automata, Languages, and Programming*, Lect. Notes in Comp. Sci. Springer Verlag, 1994. to appear.

[CM81] P. Chew and M. Machtey. A note on structure and looking-back applied to the relative complexity of computable functions. *J. Comp. Sys. Sci.*, 22:53–59, 1981.

[DL80] R. DeMillo and R. Lipton. The consistency of "P=NP" and related problems with fragments of number theory. In *Proc. 12th Annual ACM Symposium on the Theory of Computing*, pages 45–57, 1980.

[FOL83] S. Fortune, M. O'Donnell, and D. Leivant. The expressiveness of simple and second-order type structures. *J. ACM*, 30:151–185, 1983.

[Háj79] P. Hájek. Arithmetical hierarchy and complexity of computation. *Theor. Comp. Sci.*, 8:227–237, 1979.

[HR86] S. Homer and J. Reif. Arithmetic theories for computational complexity problems. *Inform. and Control*, 69:1–11, 1986.

[JY85] D. Joseph and P. Young. A survey of some recent results on computational complexity in weak theories of arithmetic. *Fundamenta Informatica*, 8:104–121, 1985.

[Kle43] S. Kleene. Recursive predicates and quantifiers. *Trans. AMS*, 53:41–73, 1943.

[Ko89] K. Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM J. Comput.*, 18:392–408, 1989.

[KOR87] S. Kurtz, M. O'Donnell, and J. Royer. How to prove representation-independent independence results. *Inf. Proc. Lett.*, 24:5–10, 1987.

[Lei82] D. Leivant. Unprovability of theorems of complexity theory in weak number theories. *Theor. Comp. Sci.*, 18:259–268, 1982.

[Lip78] R. Lipton. Model-theoretic aspects of computational complexity. In *Proc. 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 191–200, 1978.

[LLR81] L. Landweber, R. Lipton, and E. Robertson. On the structure of sets in NP and other complexity classes. *Theor. Comp. Sci.*, 15:181–200, 1981.

[LR72] L. Landweber and E. Robertson. Recursive properties of abstract complexity classes. *J. ACM*, 19:286–308, 1972.

[Lut92] J. Lutz. Almost everywhere high nonuniform complexity. *J. Comp. Sys. Sci.*, 44:220–258, 1992.

[Lut93] J. Lutz. The quantitative structure of exponential time. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 158–175, 1993.

[Mat73] Y. Matijasevic. On recursive unsolvability of Hilbert's tenth problem. In L. Henkin, A. Joja, G. Moisil, and P. Suppes, editors, *Logic, Methodology, and Philosophy of Science IV*. North-Holland, 1973.

[MR75] Y. Matijasevic and J. Robinson. Reduction of an arbitrary Diophantine equation to one in 13 unknowns. *Acta Arithmetica*, 27:521–553, 1975.

[MY78] M. Machtey and P. Young. *An Introduction to the General theory of Algorithms*. North-Holland, New York, 1978.

[NW88] N. Nisan and A. Wigderson. Hardness vs. randomness. In *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–11, 1988.

[O'D79] M. O'Donnell. A programming language theorem which is independent of Peano arithmetic. In *Proc. 11th Annual ACM Symposium on the Theory of Computing*, pages 176–188, 1979.

[Reg84] K. Regan. Arithmetical degrees of index sets for complexity classes. In *Proceedings, Logic and Machines, Münster, Germany, May 1983*, volume 171 of *Lect. Notes in Comp. Sci.*, pages 118–130. Springer Verlag, 1984.

[Reg86] K. Regan. On the Separation of Complexity Classes, 1986. Dissertation, Oxford University.

[Reg88] K. Regan. The topology of provability in complexity theory. *J. Comp. Sys. Sci.*, 3:384–432, 1988.

[Reg92] K. Regan. Diagonalization, uniformity, and fixed-point theorems. *Inform. and Comp.*, 98:1–40, 1992.

[Rog58] H. Rogers. Gödel-numberings of the partial recursive functions. *J. Symb. Logic*, 23:331–341, 1958.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted, MIT Press, 1987.

[Sch82] U. Schöning. A uniform approach to obtain diagonal sets in complexity classes. *Theor. Comp. Sci.*, 18:95–103, 1982.

[Smo91] C. Smoryński. *Logical Number Theory I: An Introduction*. Springer Verlag, 1991.

[Soa87] R. Soare. *Recursively Enumerable Sets and Degrees*. Springer Verlag, New York, 1987.

[Sto77] L. Stockmeyer. The polynomial time hierarchy. *Theor. Comp. Sci.*, 3:1–22, 1977.

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theor. Comp. Sci.*, 3:23–33, 1977.