Theoretical Computer Science 38 (1985) 143–156
North-Holland

THE RECURSION-THEORETIC STRUCTURE OF COMPLEXITY CLASSES

Diana SCHMIDT

Institut für Informatik I, Universität Karlsruhe, Postfach 6830, D-7500 Karlsruhe 1, Fed. Rep. Germany

Communicated by R.V. Book
Received April 1984
Revised December 1984

Abstract. We prove easy recursion-theoretic results which have as corollaries generalizations of existing diagonalization theorems on complexity classes: roughly speaking, almost no 'reasonable' (time, space or even abstract) complexity class can be expressed as the (non-trivial) union of two recursively presentable classes which are closed under finite variations (e.g. unless NP = P, NP ≠ P ∪ {NP-complete languages}); and, consequently, the non-trivial complement of one complexity class in another (e.g. (NP\P), provided NP ≠ P) is almost never recursively presentable.

1. Introduction

In [8], [3] and [12], a number of closely related results appeared: first, Theorem 23 of Landweber et al. [8].

Theorem 1.1. *If* $P \subsetneq NP$, *then neither* (NP\P) *nor the class of non-complete languages in* (NP\P) *is recursively presentable.*

Then Theorem 6 of Chew and Machtey [3], which may be stated as follows.

Theorem 1.2. *Let* C_1 *be a recursively presentable list of infinite r.e. languages,* C_2 *a recursively presented list of recursive languages which is closed under finite variations, and* B *a recursive language. Then*

$$B \notin C_2 \Rightarrow \{B \cap C \mid C \in P\} \not\subseteq (C_1 \cup C_2).$$

Now, putting $C_2 = P$ (respectively $C_2 = P \cup \{\text{complete sets in NP}\}$) and $C_1 = (NP \setminus C_2)$, C_2 is recursively presentable and closed under finite variations, and if $NP \neq P$ then C_1 is nonempty (for $C_2 = P \cup \{\text{complete sets in NP}\}$ this follows, for example, from Ladner's work in [6]) and contains only infinite sets. But for any $B \in C_1$, $\{B \cap C \mid C \in P\} \subseteq NP = (C_1 \cup C_2)$; hence, by Chew and Machtey's theorem, C_1 cannot be recursively presentable even by r.e. indices. Thus Chew and Machtey's

result implies (a sharpening of) that of Landweber et al. and a host of similar ones obtained by replacing NP by any other class which is closed under intersection with languages in P.

Last among these related results is that of Schöning [12], which is stated as the existence of a certain diagonal language $A \notin (\boldsymbol{C}_1 \cup \boldsymbol{C}_2)$ given languages $A_1 \notin \boldsymbol{C}_1$, $A_2 \notin \boldsymbol{C}_2$, but can be reformulated as follows.

Theorem 1.3. *Let $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ both be recursively presentable classes of recursive languages which are closed under finite variations, and let A be a recursive language. Then, putting*

$$(\leqslant A) = \{B \mid B \leqslant_{\mathrm{m}}^{\mathrm{P}} A\},$$

$$(\leqslant A) \not\subseteq \boldsymbol{C}_1 \,\&\, (\leqslant A) \not\subseteq \boldsymbol{C}_2 \Rightarrow (\leqslant A) \not\subseteq \boldsymbol{C}_1 \cup \boldsymbol{C}_2.$$

(In fact, this formulation is slightly stronger than Schöning's; it is equivalent to the theorem obtained by replacing Schöning's assertion "(e) if $A_1 \in \mathrm{P}$ and $A_2 \notin \{\varphi, \Gamma^*\}$, then $A \leqslant_{\mathrm{m}}^{\mathrm{P}} A_2$" by the somewhat stronger assertion "(e') $A \leqslant_{\mathrm{m}}^{\mathrm{P}} A_1 \oplus A_2$", which follows from Schöning's proof anyway. We leave the proof of this equivalence to the reader.)

The last two theorems can be used to deduce easily results like:

(1) $\mathrm{NP} \neq \mathrm{P} \Rightarrow \mathrm{NP} \neq \mathrm{P} \cup \{A \mid A \text{ is NP-complete under } \leqslant_{\mathrm{T}}^{\mathrm{P}}\}$;

(2) $\mathrm{NP} \neq \mathrm{P} \Rightarrow (\mathrm{NP} \setminus \mathrm{P})$ is not recursively presentable;

(3) the class of infinite subsets of Γ^* decidable in polynomial time is not recursively presentable.

But the above theorems do not, as they stand, yield corollaries of the form '$\boldsymbol{C} \not\subseteq (\boldsymbol{C}_1 \cup \boldsymbol{C}_2)$' or '$(\boldsymbol{C} \setminus \boldsymbol{C}_1)$ is not recursively presentable' in which $\boldsymbol{C}$ is not closed under $\leqslant_{\mathrm{m}}^{\mathrm{P}}$ or at least under intersection with languages in P. The purpose of this paper is to strengthen the theorems so that they do yield such corollaries: for almost all 'reasonable' complexity classes $\boldsymbol{C}$, and even more classes $\boldsymbol{C}_1$, $\boldsymbol{C}_2$ which arise naturally in complexity theory,

$$\boldsymbol{C} \not\subseteq \boldsymbol{C}_1 \;\&\; \boldsymbol{C} \not\subseteq \boldsymbol{C}_2 \;\Rightarrow\; \boldsymbol{C} \not\subseteq (\boldsymbol{C}_1 \cup \boldsymbol{C}_2).$$

In particular, $\boldsymbol{C} = \boldsymbol{C}_1 \cup \boldsymbol{C}_2 \Rightarrow \boldsymbol{C} = \boldsymbol{C}_1$ or $\boldsymbol{C} = \boldsymbol{C}_2$ ('most reasonable complexity classes cannot be the non-trivial union of two others'); and

$$\boldsymbol{C} \cap \boldsymbol{C}_2 \neq \emptyset \;\Rightarrow\; (\boldsymbol{C} \setminus \boldsymbol{C}_2) \text{ is not recursively presentable}$$

('the non-trivial complement of one complexity class in another is mostly hard to generate').

Regan's 'Uniform Diagonalization Theorem' [11] has a certain resemblance with Theorems 3.1 and 3.2 below, but it is an adaptation of Schöning's theorem to the subclass of $(\leqslant A)$ obtained by considering only length increasing invertible reductions, so it is a strengthening of that theorem for a special case and not a generalization.

Section 2 deals with notation and definitions; we prove the main theorems in Section 3; in Section 4 we show which $\boldsymbol{C}$ fulfil the conditions of the main theorems;

in Section 5 we do the same for $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$; and in Section 6 we indicate how to apply our theorems to yield a result of Breidbart [2].

2. Notation and definitions

Notation

In the following, we suppose that all languages are over some fixed alphabet Γ such that $|\Gamma| \geq 2$. We use lower-case letters u, v, w to denote words (over Γ), upper-case letters A, B, C, G, X, Y to denote languages (over Γ), bold-face $\boldsymbol{C}$ to denote classes of languages and $\boldsymbol{F}$ to denote classes of functions from $\mathbb{N}$ into $\mathbb{N}$.

Definitions

P and NP denote, as usual, the classes of languages which can be recognized in polynomial time by some deterministic (resp. nondeterministic) Turing machine.

$A \triangle B = (A \setminus B) \cup (B \setminus A)$; $\boldsymbol{C}$ is *closed under finite variations* if, for each A, B such that $A \triangle B$ is finite, $A \in \boldsymbol{C} \Rightarrow B \in \boldsymbol{C}$.

$\leqslant_{\mathrm{T}}^{\mathrm{P}}$, $\leqslant_{\mathrm{m}}^{\mathrm{P}}$ are the polynomial time-bounded versions of Turing and many-one reducibility.

A class $\boldsymbol{C}$ of sets is *recursively presentable* (*by recursive indices*) if there is an effective enumeration of Turing machines $M_1, M_2, \ldots$ all of which halt on all inputs, such that $\boldsymbol{C} = \{L(M_i) \mid i \in \mathbb{N}\}$. $\boldsymbol{C}$ is merely *recursively presentable by r.e. indices* if the Turing machines are not required to halt on all inputs, i.e. if they merely recognize the sets of $\boldsymbol{C}$ rather than deciding them. A class of functions from $\mathbb{N}$ into $\mathbb{N}$ is *recursively presentable* if there is an effective enumeration of Turing machines which compute the functions.

If A is any language, the *intervals* of A are the maximal subsets of A of the form $\{w \mid m \leqslant |w| \leqslant n\}$ $(m \leqslant n \in \mathbb{N})$, and the *gaps* of A are the intervals of $\bar{A}$.

A is a *gap language* if A is the union of all its intervals, i.e. if for all u, v such that $|u| = |v|$, $u \in A \Leftrightarrow v \in A$. Let A be a gap language, B be any language. B *majorizes* A if for all $n \in \mathbb{N}$, $A \cap \{w \mid |w| \geq n\}$ has an interval which is contained in B and a gap which is contained in $\bar{B}$.

A class $\boldsymbol{C}$ of languages is said to be *recursive gap closed* if, for every recursive gap language G_0 with infinitely many gaps and intervals, there is a language G which majorizes G_0 s.t. $G, \bar{G} \in \boldsymbol{C}$. Clearly, if $\boldsymbol{C} \subseteq \boldsymbol{C}'$ and $\boldsymbol{C}$ is recursive gap closed, then so is $\boldsymbol{C}'$.

3. The main theorems

The two theorems which follow are of a very general nature and have, on the face of it, nothing to do with complexity theory. But, as we shall see in Sections 4 and 5, they are applicable to many classes which arise naturally in complexity theory.

Theorem 3.1. *Let C be a class of recursive languages which is recursive gap closed and closed under union and intersection, and let C_1, C_2 be recursively presentable classes which are closed under finite variations. Then*

$$C \not\subseteq C_1 \;\&\; C \not\subseteq C_2 \Rightarrow C \not\subseteq C_1 \cup C_2.$$

Proof. Let $P_1, P_2, \ldots$ and $Q_1, Q_2, \ldots$ be effective enumerations of Turing machines which present C_1 and C_2 respectively. Now if $C \not\subseteq C_1$ and $C \not\subseteq C_2$ then there is an $A_1 \in C \setminus C_1$ and an $A_2 \in C \setminus C_2$. Define

$$r_1(n) = \max\{(\mu m > n)(\exists z \in L(P_i) \triangle A_1)(|z| = m) \mid i < n\},$$

$$r_2(n) = \max\{(\mu m > n)(\exists z \in L(Q_i) \triangle A_2)(|z| = m) \mid i < n\} \quad \text{for all } n \in \mathbb{N}.$$

r_1, r_2 are total (because $A_1 \notin C_1$, $A_2 \notin C_2$ and C_1, C_2 are closed under finite variations) and recursive (because A_1, A_2 are recursive and each P_i, Q_i halts on all inputs). Now define $r(n) = \max\{r_1(n), r_2(n)\}$ for all $n \in \mathbb{N}$; then r is also recursive and the set G_r defined by

$$G_r = \{w \mid |w| = 0 \text{ or } (\exists n \in \mathbb{N})(r^{2n}(0) < |w| \leq r^{2n+1}(0))\}$$

is a recursive gap language with infinitely many gaps and intervals. By hypothesis, there is a language G which majorizes G_r such that $G, \bar{G} \in C$. Define $A = (G \cap A_1) \cup (\bar{G} \cap A_2)$. Then, since C is closed under intersection and union, $A \in C$.

$A \notin C_1$: suppose $A \in C_1$. Then there is a j such that $A = L(P_j)$. Now for any n such that $j < r^{2n}(0)$, by the definition of r_1 there is a $w \in L(P_j) \triangle A_1 = A \triangle A_1$ such that

$$r^{2n}(0) < |w| \leq r_1(r^{2n}(0)) \leq r(r^{2n}(0)) = r^{2n+1}(0).$$

Thus each interval of G_r which is contained in $\{w \mid |w| > j+1\}$ contains some $w \in A \triangle A_1$. But, since G majorizes G_r, one such interval is contained in G and hence G contains some $w \in A \triangle A_1$. This contradicts $A = (G \cap A_1) \cup (\bar{G} \cap A_2)$.

$A \notin C_2$: Analogous, using odd n and gaps instead of intervals. □

This proof is basically the same as the proof of the main theorem in Schöning's [12] and Theorem 3.1 can, with the help of Theorem 4.1 below, be seen to imply that theorem (put $C = \{B \mid B \leq_m^P A\}$ to obtain the reformulation in Section 1).

Now it turns out that if we assume that C_1 contains only infinite languages (which is quite a strong assumption for a recursively presentable class; for, as we shall see below, the class of *all* infinite languages in C is not recursively presentable even by r.e. indices) then we can considerably weaken the other assumptions on C and C_1 and still obtain the conclusion $C \not\subseteq C_1 \cup C_2$.

Theorem 3.2. *Let C be a class of recursive languages which is recursive gap closed and closed under intersection, C_1 be a recursively presentable class of infinite r.e. sets and C_2 a recursively presentable class of recursive sets which is closed under finite variations.*

Then

$$C \not\subseteq C_2 \Rightarrow C \not\subseteq C_1 \cup C_2.$$

Proof. Define r_2 as in the proof of Theorem 3.1 and r_1 by

$$r_1(n) = \max\{(\mu m)(\exists w)(n < |w| \leq m \text{ and } P_i \text{ accepts } w \text{ in at most } m \text{ steps}) \mid i < n\}.$$

r_1 is total (because, by hypothesis, each $L(P_i)$ is infinite) and recursive (because, although the P_i do not in general halt on all inputs, each P_i is simulated for only finitely many steps in the computation of $r_1(n)$).

Now define $r(n) = \max\{r_1(n), r_2(n)\}$ for all $n \in \mathbb{N}$ and $G_r = \{w \mid |w| = 0 \text{ or } (\exists n \in \mathbb{N})(r^{2n}(0) < |w| \leq r^{2n+1}(0))\}$ as before. Again, there is a language $G \in C$ which majorizes G_r. This time, define $A = G \cap A_2$. $A \notin C_2$ follows as before and $A \in C$ because C is closed under intersection.

$A \notin C_1$: if $A \in C_1$, then there is a j such that $A = L(P_j)$. Now for any n such that $j < r^{2n+1}(0)$, by the definition of r_1 there is a $w \in L(P_j) = A$ such that

$$r^{2n+1}(0) < |w| \leq r_1(r^{2n+1}(0)) \leq r(r^{2n+1}(0)) = r^{2n+2}(0).$$

Thus each gap of G_r which is contained in $\{w \mid |w| > j+1\}$ contains some $w \in A$. But, since G majorizes G_r, one such gap is contained in $\bar{G}$ and hence $\bar{G}$ contains some $w \in A$. This contradicts $A = G \cap A_2$. □

This proof is basically the same as that of Chew and Machtey [3, Theorem 6] and Theorem 3.2 can, with the help of Theorem 4.1 below, be seen to imply that theorem (put $C = \{B \cap C \mid C \in \mathrm{P}\}$). Note that, in Theorem 3.2, the additional assumption that C_1 contains only infinite sets is strong enough to counterbalance several weakenings of the premise (closure of C under union is no longer required; C_1 is required to be recursively presentable only by r.e. indices, may contain nonrecursive sets and is not required to be closed under finite variations; but $C \not\subseteq C_1$, though no longer explicitly assumed, now follows from the other assumptions by Corollary 3.3 below) while still yielding the conclusion $C \not\subseteq C_1 \cup C_2$. In fact, Theorem 3.2 immediately yields a corollary which shows how rare such C_1 as in Theorem 3.2 are.

Corollary 3.3. *Let C be a class of recursive languages which is recursive gap closed and closed under intersection. Then for any recursively presentable class C_1 of infinite r.e. sets*

$$\{A \in C \mid A \text{ is infinite}\} \not\subseteq C_1.$$

Proof. Put $C_2 = \{A \mid A \text{ is a finite language}\}$. C_2 is recursively presentable and closed under finite variations. $C \not\subseteq C_2$ since C is recursive gap closed and therefore certainly contains infinite languages. Hence, by Theorem 3.2, $C \not\subseteq C_1 \cup C_2$; the assertion follows immediately. □

This is not to say that there are no classes C_1 satisfying the conditions of Theorem 3.2: examples are classes of complete languages in some class with respect to some reduction (e.g. EXPSPACE-complete languages; or NP-complete languages if P$\neq$ NP) and any recursively presentable class of languages which all contain one fixed, infinite language (e.g. the class of all those r.e. languages which contain $\{0\}^*$). If C_1 satisfies the conditions of Theorem 3.2, Theorem 3.2 is usually quicker to apply than Theorem 3.1, whereas if both C_1 and C_2 contain finite languages, Theorem 3.1 must be applied. But the applications in which Theorem 3.2 really has the edge over Theorem 3.1 are those in which it is shown by contraposition that some class is not recursively presentable. A typical example is the following.

Corollary 3.4. *Let C be a class of recursive languages which is recursive gap closed and closed under finite variations, union and intersection, and let C_2 be a recursively presentable class which is closed under finite variations.*

Then

$$C \cap C_2 \neq \emptyset \Rightarrow (C \setminus C_2) \text{ is not recursively presentable}$$

(and, if C_2 contains all finite languages, $(C \setminus C_2)$ is not even recursively presentable by r.e. indices).

Proof. If $(C \setminus C_2)$ were recursively presentable, then, putting $C_1 = (C \setminus C_2)$, C, C_1 and C_2 would fulfil the conditions of Theorem 3.1 ($C \not\subseteq C_2$ because otherwise we should have $(C \setminus C_2) = \emptyset$, which is not recursively presentable), yielding $C \not\subseteq (C \setminus C_2) \cup C_2$, a contradiction. If C_2 contains all finite languages, then merely assuming that $(C \setminus C_2)$ is recursively presentable by r.e. indices makes Theorem 3.2 applicable to yield the same contradiction. □

We shall see in Sections 4 and 5 that this corollary implies, roughly speaking, that the complement of one complexity class in another is almost never recursively presentable. Corollary 3.5 below implies that almost no 'reasonable' complexity class can be expressed nontrivially as the union of two others.

Corollary 3.5. *Let C, C_1 and C_2 be as in Theorem* 3.1 *or Theorem* 3.2. *Then*

$$C = C_1 \cup C_2 \Rightarrow C = C_1 \text{ or } C = C_2.$$

Proof. $C = C_1 \cup C_2 \Rightarrow C \supseteq C_1$ and $C \supseteq C_2$. Moreover, by Theorem 3.1 respectively Theorem 3.2,

$$C = C_1 \cup C_2 \Rightarrow C \subseteq C_1 \quad \text{or} \quad C \subseteq C_2 \Rightarrow C = C_1 \text{ or } C = C_2. \qquad \square$$

4. Classes which are recursive gap closed

In this section, we show that many classes which arise naturally in complexity theory have the properties required of C in the main theorems.

We refer the reader to [12] for any unexplained notation. We shall use the following notation for (relativized and unrelativized) Turing machine complexity classes:

$$\mathrm{DTIME}^A(\boldsymbol{F}) = \{X \mid \text{there is an } f \in \boldsymbol{F} \text{ and a deterministic } f\text{-time-bounded Turing machine with oracle } A \text{ which decides } X\};$$
$$\mathrm{NTIME}^A(\boldsymbol{F}) = \{X \mid \text{there is an } f \in \boldsymbol{F} \text{ and a nondeterministic } f\text{-time-bounded Turing machine with oracle } A \text{ which recognizes } X\};$$
$$\mathrm{DTIME}(\boldsymbol{F}) = \mathrm{DTIME}^{\emptyset}(\boldsymbol{F}); \qquad \mathrm{NTIME}(\boldsymbol{F}) = \mathrm{NTIME}^{\emptyset}(\boldsymbol{F});$$

and, for any complexity class $\boldsymbol{C}$,

$$\text{co-}\boldsymbol{C} = \{\Gamma^* \setminus X \mid X \in \boldsymbol{C}\}.$$

$$O(\boldsymbol{F}) = \{g \mid (\exists f \in \boldsymbol{F}, k \in \mathbb{N})(g(n) \leqslant kf(n) \text{ for all } n \in \mathbb{N})\}.$$

The Turing machine space complexity classes $\mathrm{DSPACE}^A(\boldsymbol{F})$, $\mathrm{NSPACE}^A(\boldsymbol{F})$, etc. are defined analogously, except that the suffices 'on-line' and 'off-line' are added to indicate whether the input must be read from left to right only or not (this distinction is only important if the space bounds in $\boldsymbol{F}$ are less than linear); we refer the reader to [5], [9] and [14] for a comparison of the power of on-line and off-line Turing machines.

Theorem 4.1. $\mathrm{DTIME}(n)$ *is recursive gap closed; hence, so is any complexity class* $\mathrm{DTIME}(\boldsymbol{F})$ *or* $\mathrm{NTIME}(\boldsymbol{F})$ *such that* $n \in O(\boldsymbol{F})$.

Proof. Let G_0 be a recursive gap language with infinitely many gaps and intervals. We define a gap language $G \in \mathrm{DTIME}(n)$ (whence also $\bar{G} \in \mathrm{DTIME}(n)$) which majorizes G_0 as follows:

Since G_0 is recursive, there is some Turing machine T which decides G_0. We define a Turing machine M which on input w proceeds as follows:

M reads w from left to right. Simultaneously, on its work tapes M computes $M(\langle\ \rangle)$, $T(\langle\ \rangle)$, $M(0)$, $T(0), \ldots, M(0^p)$, $T(0^p)$ for as long as it can (until w is exhausted), meanwhile keeping a note of whether, in the initial segment $\{\langle\ \rangle, 0, \ldots, 0^p\}$ which M can supervise, $L(M)$ contains a whole interval of $L(T)$ $(= G_0)$ (we call this 'case I') or $\overline{L(M)}$ contains a whole gap of $L(T)$ (we call this 'case G') and, if so, which case occurred last. If neither case I nor case G occurred or if case G occurred last, then M accepts w; and if case I occurred last, then M rejects w.

Clearly, M can be designed to operate in real time. Moreover, by the construction of M, M goes on accepting everything until it notices that it has accepted a whole interval of G_0; after that, it goes on rejecting everything until it notices that it has rejected a whole gap of G_0; and so on, ensuring that the language G accepted by M majorizes G_0. This is a typical application of the 'looking-back' method as introduced by Ladner [6]. □

Theorem 4.2. $\mathrm{DSPACE}_{\text{on-line}}(\log n)$ *is recursive gap closed; hence, so is any complexity class* $\mathrm{DSPACE}(F)$ *or* $\mathrm{NSPACE}(F)$ *such that* $\log n \in O(F)$.

Proof. Let G_0 be a recursive gap set. We define a gap set $G \in \mathrm{DSPACE}_{\text{on-line}}(\log n)$ (whence also $\bar{G} \in \mathrm{DSPACE}_{\text{on-line}}(\log n)$) which majorizes G_0 as follows:

Let T be a Turing machine which decides G_0. We define an on-line logspace-bounded Turing machine M which on input w proceeds as follows:

First, M reads w from left to right, keeping a binary counter of the length of the input on a work tape. When the whole of w has been read, the portion of the tape now occupied by the counter is marked off and only this portion is used in the following computations, thus ensuring that M is logspace-bounded. Now M proceeds exactly like the machine M in the proof of Theorem 4.1, except that it is now limited not by a time bound but by the space bound. Just as in the proof of Theorem 4.1, the language G accepted by M majorizes G_0. This is another typical application of Ladner's 'looking back' method. □

Note that the two essential ingredients of the proof of Theorem 4.2 as far as the space-bounding function $\lfloor \log n \rfloor$ is concerned are the facts that it is 'uniformly tape constructible' (i.e. there is a Turing machine—on-line, in this case—which uses precisely $\lfloor \log n \rfloor$ squares of its work tape on every input of length n) and that $\lim_{n \to \infty} \lfloor \log n \rfloor = \infty$; the proof would work just as well, even for off-line machines, for any other space-bounding function with these two properties. The fact (Corollary 4.7 below) that the theorem is false even for off-line space complexity classes with more slowly-growing space bounds implies that no space bound which grows more slowly than $\log n$ has both these properties (this latter fact is Theorem 4(i) of [5]).

In fact, Theorems 4.1 and 4.2 can be merged into one.

Theorem 4.3

$$\{X \mid \textit{there is a deterministic on-line } n\textit{-time-bounded and } \log(n)\textit{-space-bounded Turing machine which decides } X\},$$

is recursive gap closed; hence, so is any class of languages containing this class.

Proof. Define a Turing machine M which on input w works just like the machine in the proof of Theorem 4.1 (and hence is on-line and real time) except that it is adapted in the following way to ensure that it is also logspace-bounded: before starting to compute $M(\langle\ \rangle), T(\langle\ \rangle), \ldots, M$ marks off zero squares of available work space on its work tape and, on another part of the work tape, sets up a binary counter with initial value zero. Only the marked-off work space may be used in computing $M(\langle\ \rangle), T(\langle\ \rangle), \ldots$. Whenever M needs more work space for this computation, it calls a subroutine which increases the binary counter by one repeatedly until the length of the counter increases; then M may extend the marked-off work space by one square and proceed with the computation of $M(\langle\ \rangle), T(\langle\ \rangle), \ldots$. Clearly, M uses $O(\log |w|)$ squares of its work tape and is hence logspace bounded; and, as before, the language accepted by M majorizes G_0. □

The time-and-space-limiting mechanism of M is almost identical to that used in Breidbart's proof of Theorem 1 in [2]; in Section 6 we shall show how to infer Breidbart's result from Theorem 4.3.

In fact, even most abstract complexity classes are recursive gap closed. (We refer the reader to [1, Chapter 9] for an introduction to abstract complexity theory.)

Theorem 4.4. *Let $\mathscr{C}$ be an abstract complexity measure. Then there is a recursive function f such that, for any function g such that $f(n) \leq g(n)$ almost everywhere, $\mathscr{R}_g^{\mathscr{C}}$ is recursive gap closed. ($\mathscr{R}_g^{\mathscr{C}}$ is the complexity class defined by resource bound g.)*

Proof. Use either Theorem 4.1 or Theorem 4.2 and the fact that any two abstract complexity measures are recursively related. □

Thus, Theorems 3.1 and 3.2 are applicable to most abstract complexity classes, provided they are closed under union and intersection. In the rest of this section we shall consider (Chomsky and complexity) classes to which Theorems 3.1 and 3.2 do not apply.

By [5, Theorem 3], any on-line space complexity class whose space bound does not grow as fast as $\log n$ is identical to the class of regular languages, which is not recursive gap closed.

Theorem 4.5. *The class of context-free languages, and hence also that of regular languages, is not recursive gap closed, and moreover Theorems* 3.1 *and* 3.2 *fail for both these classes. (The class of context-sensitive languages is recursive gap closed.)*

Proof. The last assertion follows from Theorem 4.2, since the class of context-sensitive languages is just NSPACE(n). The class of context-free languages is not recursive gap closed because, by the '*uvwxy*'-theorem, every context-free language is either finite or has gaps of bounded length, whereas there are certainly infinite recursive gap languages with gaps of unbounded length. Moreover, Theorems 3.1 and 3.2 fail for both the context-free languages and the regular languages because finiteness is decidable in (standard recursive presentations of) both these classes, so in both these classes both the subclass of finite sets and that of infinite sets are recursively presentable. □

Unlike the on-line space complexity classes, the non-trivial off-line space complexity classes do not stop at space bound $\log n$ but go down to space bound $\log \log n$ (see [5, Theorem 2] and [9, Theorem 1]). But, as we shall see, below $\log n$, even if we throw in nondeterminism, the off-line space complexity classes cease to be recursive gap closed. First, we need the following lemma.

Lemma 4.6. *Given $s: \mathbb{N} \to \mathbb{N}$, if M is any $s(n)$-space-bounded off-line nondeterministic*

Turing machine with tape alphabet A_M and set of states S_M, then for any n such that

$$\log n > (\log |A_M| + \log |S_M| + 1)s(n),$$

and, for any $a \in A_M$,

$$a^n \in L(M) \Rightarrow a^{n+kn!} \in L(M) \quad \textit{for each } k \in \mathbb{N}.$$

Proof. By a *generalized state* we shall mean a triple

(contents of M's work tape, position of M's read/write head on work tape, state of M)

—i.e. what is usually called a configuration, except that we disregard M's input tape. Clearly, M can take on at most $|A_M|^{s(n)} \cdot s(n) \cdot |S_M|$ different generalized states in a computation on input a^n, and if

$$\log n > (\log |A_M| + \log |S_M| + 1)s(n),$$

then $n > |A_M|^{s(n)} \cdot s(n) \cdot |S_M|$ so M cannot take on n different generalized states in any computation on input a^n.

Now define an *excursion* as a part of a computation on input a^n in which the read/write head on the input tape starts and ends on a blank square and scans only a's otherwise. A *short excursion* is one which starts and ends at the same end of the input and a *long excursion* is one whose beginning and end are at opposite ends of the input. Now in a long excursion the whole input is eventually traversed, say from left to right. So, since M takes on fewer than n different generalized states, if we consider the generalized states assumed by M as the read/write head on the input tape crosses each square of the input for the first time (in a given long excursion), some generalized state must occur twice in this list, say at positions i and j $(i<j)$. But then, on any input $a^{n+k(j-1)}$, there is a computation in which this generalized state goes on recurring at position $i+l(j-i)$ for each $l \leqslant k$ and is then followed by the same sequence of generalized states as in the rest of the long excursion on input a^n, thus yielding a long excursion on this new input which begins and ends with the same generalized states as before. In particular, since $(j-i) \mid n!$, for any long excursion on input a^n there is a long excursion on input $a^{n+kn!}$ $(k \in \mathbb{N})$ which begins and ends with the same generalized states. But this is trivially true of short excursions anyway. Hence, since any accepting computation of M on a^n may be regarded as a sequence of excursions, possibly followed by a computation in which only the a's are scanned, it follows that for any accepting computation on a^n there is also an accepting computation on $a^{n+kn!}$ for each $k \in \mathbb{N}$. □

This proof is basically the same as that of [9, Theorem 2(1)] and similar proofs occur elsewhere in the literature, but we include it here in full to make it clear that this particular version does also apply to nondeterministic Turing machines.

Corollary 4.7. *If* $s:\mathbb{N}\to\mathbb{N}$ *is such that*

$$\lim_{n\to\infty}\frac{s(n)}{\log n}=0,$$

then $\mathrm{NSPACE}_{\text{off-line}}(s(n))$ (*and, hence, also* $\mathrm{DSPACE}_{\text{off-line}}(s(n))$) *is not recursive gap closed.*

Proof. If M is an $s(n)$-space-bounded off-line nondeterministic Turing machine with tape alphabet A_M and set of states S_M, then, since $\lim_{n\to\infty} s(n)/\log n = 0$, there is an $n_0\in\mathbb{N}$ such that, for all $n\geq n_0$, $\log n > (\log|A_M|+\log|S_M|+1)s(n)$. But then, by Lemma 4.6, for each $n\geq n_0$ and $a\in A_M$, either $a^n\notin L(M)$ or, if $a^n\in L(M)$, then $L(M)$ contains no gap of length $>n!$ because $a^{n+kn!}\in L(M)$ for each $k\in\mathbb{N}$. Thus either $a^n\notin L(M)$ for all $n\geq n_0$, in which case $L(M)$ has only finitely many intervals, or $a^n\in L(M)$ for some $n\geq n_0$, in which case all gaps in $L(M)$ are of bounded length. Since there are infinite recursive gap languages with gaps of unbounded length, it follows that $\mathrm{NSPACE}_{\text{off-line}}(s(n))$ cannot be recursive gap closed. □

Remark. Let us consider four related properties of certain classes $\boldsymbol{C}$ of languages:

(1) $\boldsymbol{C}$ is recursive gap closed;

(2) there are no recursively presentable classes $\boldsymbol{C}_1$, $\boldsymbol{C}_2$, each closed under finite variations, such that

$$\boldsymbol{C}\subseteq\boldsymbol{C}_1\cup\boldsymbol{C}_2 \quad\text{but}\quad \boldsymbol{C}\not\subseteq\boldsymbol{C}_1 \;\&\; \boldsymbol{C}\not\subseteq\boldsymbol{C}_2;$$

(3) there are no recursively presentable classes $\boldsymbol{C}_1$, $\boldsymbol{C}_2$ closed under finite variations such that $\boldsymbol{C}=\boldsymbol{C}_1\cup\boldsymbol{C}_2$ and $\boldsymbol{C}_1\cap\boldsymbol{C}_2=\emptyset$;

(4) there is no recursive presentation of $\boldsymbol{C}$ in which finiteness is decidable.

Now, for any class $\boldsymbol{C}$ which is closed under finite variations, intersection and union and properly contains the class of all finite languages, (1) ⇒ (2) ⇒ (3) ⇒ (4). ((1) ⇒ (2) by Theorem 3.1; (2) ⇒ (3) because any recursively presentable class is nonempty; and (3) ⇒ (4) because, if (4) fails to hold, putting $\boldsymbol{C}_1$ = class of finite languages and $\boldsymbol{C}_2=(\boldsymbol{C}\setminus\boldsymbol{C}_1)$ violates (3)).

Now our prime examples of classes which violate (1)—the context-free languages and the regular languages—also violate (4) (and, hence, (2) and (3)). We do not know whether the classes $\mathrm{NSPACE}_{\text{off-line}}(s(n))$ and $\mathrm{DSPACE}_{\text{off-line}}(s(n))$ which violate (1) satisfy (2), (3) and (4) or not. Indeed, we do not know whether some or all of (1)-(4) are equivalent for classes $\boldsymbol{C}$ with the above properties, maybe supplemented by one or two further natural closure properties. The question whether (1) and (4) are equivalent was raised by W. Menzel.

5. Recursively presentable classes

Since many classes are shown in [12] to be recursively presentable and the proofs of the generalizations here are similar to the proofs there, we shall just give a summary.

Definition 5.1. Let C be any class of languages, $\leqslant$ any reduction relation. We use $(C\leqslant)$ to denote $\{X|(\exists Y\in C)(Y\leqslant X)\}$ and $(\leqslant C)$ to denote $\{X|(\exists Y\in C)(X\leqslant Y)\}$.

Note that if X is $\leqslant$-complete for C, then

$$\{\leqslant\text{-complete sets for } C\}=C\cap(\{X\}\leqslant).$$

Theorem 5.2. *The following classes of languages are recursively presentable (by recursive indices) and closed under finite variations:*

(a) $\mathscr{R}_F^{\mathscr{C}}$ *if* $\mathscr{C}$ *is any abstract complexity measure and* F *any recursively presentable set of unary recursive functions from* $\mathbb{N}$ *into* $\mathbb{N}$, *provided* $\mathscr{R}_F^{\mathscr{C}}$ *is closed under finite variations* [1, Th. 9.18]. *Particular instances of such abstract complexity classes are, for example,* $\mathrm{DTIME}(F)$, $\mathrm{NTIME}(F)$, $\mathrm{DSPACE}_{\text{off-line}}(F)$, $\mathrm{NSPACE}_{\text{off-line}}(F)$, $\mathrm{DSPACE}_{\text{on-line}}(F)$, $\mathrm{NSPACE}_{\text{on-line}}(F)$, *where* $F=\{n^k|k\in\mathbb{N}\}$, $F=\{2^{kn}|k\in\mathbb{N}\}$, $F=\{\lambda n\cdot n\}$, *etc, and the relativized versions of any of these classes with any recursive oracle* A (*e.g.* $\mathrm{DTIME}^A(F)$).

(b) co-C_1, $(C_1\cap C_2)$ (*provided* $C_1\cap C_2\neq\emptyset$) *and* $(C_1\cup C_2)$, *whenever* C_1 *and* C_2 *are themselves recursively presentable and closed under finite variations.*

(c) $(\leqslant C)$ *whenever* C *is recursively presentable,* $C\not\subseteq\{\emptyset,\Gamma^*\}$, F *is a recursively presentable set of unary recursive functions and* $\leqslant$ *is either the* F*-time-bounded or the* F*-space-bounded restriction of* $\leqslant_{\mathrm{T}}$, $\leqslant_{\mathrm{m}}$, $\leqslant_{\mathrm{tt}}$ *or any of the restrictions of truth-table reducibility* (*see* [7]), *of the nondeterministic* (*see* [7, *Section* 4]) *or strongly nondeterministic* (*see* [8, *Section* 3]) *variant of any of these reduction relations, or of* $\leqslant_{\mathrm{R}}$ (*see* [10]).

(d) $C_1\cap(C_2\leqslant)$ *whenever* $\leqslant$ *is as in* (c), C_1 *and* C_2 *are recursively presentable,* $C_1\cap(C_2\leqslant)\neq\emptyset$ *and* C_1 *is closed under finite variations and* $(\forall X\in C_2)(\forall$ *finite* $Y\subseteq\Gamma^*(X\not\leqslant Y\ \&\ X\not\leqslant\Gamma^*\setminus Y))$.

Remarks. By Theorem 5.2, the following classes are recursively presentable and closed under finite variations:

- $\mathrm{DTIME}(n)$, $\mathrm{DTIME}(n^2)$, $\mathrm{DTIME}(n^3),\ldots,$ P, NP, co-NP,
- all classes Σ_k^{P}, Π_k^{P} and Δ_k^{P} in the polynomial hierarchy, PH,
- EXPTIME, NEXPTIME, etc; $\mathrm{DSPACE}_{\text{on-line}}(\log n)$,
- $\mathrm{DSPACE}_{\text{off-line}}(\log\log n)$, $\mathrm{DSPACE}_{\text{off-line}}(\log n)$, $\mathrm{DSPACE}(n)$,
- $\mathrm{NSPACE}(n)$, $\mathrm{DSPACE}(n^2),\ldots,$ PSPACE, EXPSPACE, etc;
  $\{\leqslant$-complete sets for $C\}$,

whenever $\leqslant$ is as in Theorem 5.2(c) and C is recursively presentable and closed under finite variations and contains a language which is complete under $\leqslant$ (for then either $\{\leqslant$-complete languages for $C\}=C$ or, for any X which is $\leqslant$-complete for C, $\{\leqslant$-complete languages for $C\}=C\cap(\{X\}\leqslant)$ fulfils the conditions of Theorem 5.2(d). Thus, by Theorem 5.2, all 'reasonable' complexity classes and many more classes which arise naturally in complexity theory are recursively presentable and closed under finite variations.

Notable exceptions are, by Corollary 3.4, classes which can be expressed non-trivially as the complement of one complexity class in another.

Remarks. In the light of Sections 4, 5, Corollary 3.4 may now be interpreted to imply that

> the (nontrivial) complement of one complexity class in another is almost never recursively presentable.

In view of the fact that all reasonable complexity classes are themselves recursively presentable, it follows that the (nontrivial) complement of one complexity class in another is (usually) harder to generate than any whole complexity class. (To put it somewhat imprecisely: instances of lower complexity bounds are harder to generate than instances of upper complexity bounds.) Thus, while there may be algorithms which generate large classes of instances of a given lower complexity bound (e.g. the generation of EXPTIME-complete languages as instances of languages not in P), there is no algorithm which can generate *all* such instances within a given complexity class.

Similarly, Corollary 3.5 may be interpreted to imply that almost no 'reasonable' complexity class is the union of two others (in a nontrivial way). For example, in the polynomial hierarchy, as is well known,

$$\Sigma_k^P \neq \Pi_k^P \Rightarrow \Sigma_k^P \cup \Pi_k^P \subsetneq \Delta_k^P.$$

6. Applications

Apart from the applications indicated in the remarks above, Theorems 3.1 and 3.2 can often be applied to prove the existence of languages with certain properties. Ref. [13], for example, contains an application to the structure of resource-bounded reducibilities. Here we indicate how to infer a result of Breidbart as an application.

Theorem 6.1 (Breidbart [2]). *If A is any infinite co-infinite recursive set and $C = \{B \mid B$ can be accepted in real time and log-space by a deterministic Turing machine$\}$, then there is a language $B \in C$ which splits A (i.e. $A \cap B$, $A \cap \bar{B}$, $\bar{A} \cap B$, $\bar{A} \cap \bar{B}$ are infinite).*

Sketch of proof. Put $C_1 = \{B \mid A \cap B$ is finite$\}$, $C_2 = \{B \mid A \cap \bar{B}$ is finite$\}$, $C_3 = \{B \mid \bar{A} \cap B$ is finite$\}$, $C_4 = \{B \mid \bar{A} \cap \bar{B}$ is finite$\}$.

For each i, C_i is recursively presentable and closed under finite variations and $C \not\subseteq C_i$. Hence, by Theorem 3.1 (applied three times), $C \not\subseteq C_1 \cup C_2 \cup C_3 \cup C_4$, i.e. there is a $B \in C$ which splits A.

Acknowledgment

I wish to thank an anonymous referee for many helpful suggestions as to the presentation of this paper, in particular for the reformulation of Schöning's theorem in the introduction, and for suggesting Theorem 4.3 as a possible connection with Breidbart's work.

References

[1] W.S. Brainerd and L.H. Landweber, *Theory of Computation* (Wiley, New York, 1974).
[2] S. Breidbart, On splitting recursive sets, *J. Comput. System Sci.* **17** (1978) 56–64.
[3] P. Chew and M. Machtey, A note on structure and looking back applied to the relative complexity of computable functions, *J. Comput. System Sci.* **22** (1981) 53–59.
[4] A.R. Freedman and R.E. Ladner, Space bounds for processing contentless inputs, *J. Comput. System Sci.* **11** (1975) 118–128.
[5] J.E. Hopcroft and J.D. Ullman, Some results on tape-bounded Turing machines, *J. ACM* **16** (1969) 168–177.
[6] R.E. Ladner, On the structure of polynomial time reducibility, *J. ACM* **22** (1975) 155–171.
[7] R.E. Ladner, N.A. Lynch and A.L. Selman, A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **1** (1975) 103–123.
[8] L.H. Landweber, R.J. Lipton and E.L. Robertson, On the structure of sets in NP and other complexity classes, *Theoret. Comput. Sci.* **15** (1981) 181–200.
[9] P.M. Lewis II, R.E. Stearns and J. Hartmanis, Memory bounds for recognition of context free and context sensitive languages, in: *IEEE Conf. Record on Switching Circuit Theory and Logical Design* (1965) 191–202.
[10] T.J. Long, Strong nondeterministic polynomial-time reducibilities, *Theoret. Comput. Sci.* **21** (1982) 1–25.
[11] K. Regan, On diagonalization methods and the structure of language classes, in: *Proc. FCT Conf.* (1983) 368–380.
[12] U. Schöning, A uniform approach to obtain diagonal sets in complexity classes, *Theoret. Comput. Sci.* **18** (1982) 95–103.
[13] D. Schmidt, On the complement of one complexity class in another, in: *Logic and Machines*, Lecture Notes in Computer Science **171** (Springer, Berlin, 1984) 77–87.
[14] R.E. Stearns, J. Hartmanis and P.M. Lewis II, Hierarchies of memory limited computations, in: *IEEE Conf. Record on Switching Circuit Theory and Logical Design* (1965) 179–190.