EASY CONSTRUCTIONS IN COMPLEXITY THEORY: GAP AND SPEED-UP THEOREMS

PAUL YOUNG¹

ABSTRACT. Perhaps the two most basic phenomena discovered by the recent application of recursion theoretic methods to the developing theories of computational complexity have been Blum's speed-up phenomena, with its extension to operator speed-up by Meyer and Fischer, and the Borodin gap phenomena, with its extension to operator gaps by Constable. In this paper we present a proof of the operator gap theorem which is much simpler than Constable's proof. We also present an improved proof of the Blum speed-up theorem which has a straightforward generalization to obtain operator speed-ups. The proofs of this paper are new; the results are not. The proofs themselves are entirely elementary: we have eliminated all priority mechanisms and all but the most transparent appeals to the recursion theorem. Even these latter appeals can be eliminated in some "reasonable" complexity measures. *Implicit* in the proofs is what we believe to be a new method for viewing the construction of "complexity sequences."

Unspecified notation follows Rogers [12]. $\lambda i\phi_i$ is any standard indexing of the partial recursive functions. N is the set of all nonnegative integers. λiD_i is a canonical indexing of all finite subsets of N : from i we can list D_i and know when the listing is completed. Similarly, λiF_i is a canonical indexing of all finite functions defined (exactly) on some initial segment $\{0, 1, 2, \dots, n\}$. $\lambda i\Phi_i$ is any Blum measure of computational complexity or resource. Specifically, for all i , $\text{domain } \Phi_i = \text{domain } \phi_i$, and the ternary relation $\Phi_i(x) \leq y$ is decidable (recursive). Intuitively, $\Phi_i(x)$ might be the amount of time or space used by Turing machine i or the i th ALGOL

Received by the editors August 15, 1971 and, in revised form, May 8, 1972.

AMS (MOS) subject classifications (1970). Primary 68A20, 02-02; Secondary 02F20, 94A30, 94-02.

Key words and phrases. Recursive function theory, computational complexity, speed-up theorems, gap theorems, Blum theory.

¹ This research was supported by NSF Grant GJ-27127, Purdue University, and NSF Grant GJ 596, University of Hawaii. The author thanks David Pager for some useful criticisms. These results were first presented at a conference on computational complexity in October 1971, at the Courant Institute for Mathematical Sciences. An abstract will appear in the Proceedings of that conference.

program on input x . If f is a function and S a set, $f|S$ denotes the restriction of f to S .

By a total effective operator Ψ we understand an operator mapping partial recursive functions to partial recursive functions whose domain includes all total recursive functions, each of which is mapped via Ψ to a total recursive function. Furthermore, Ψ is given by a total recursive function g such that for all $\phi_i \in \text{domain } \Psi$, $\Psi(\phi_i) = \phi_{g(i)}$. A trivial variation of a basic result due to Kriesel, Lacombe, and Schoenfield asserts that if Ψ is total effective, there is a total recursive ψ such that, for all total ϕ_i ,

$$\Psi(\phi_i) = \bigcup F_{\psi(v)} \mid F_v \subseteq \phi_i.^2$$

This is the formulation we will use in Theorems 3 and 4. The phrase "a.e." is an abbreviation for "for all but finitely many arguments." By convention, we take the maximum of the empty set of integers to be zero. S is a fixed total recursive function such that $\phi_i(x, y) = \phi_{S(i, x)}(y)$. ρ is a fixed total recursive function such that $\phi_{\rho(i, v)}(x) = F_v(x)$ if $x \in \text{domain } F_v$ while $\phi_{\rho(i, v)}(x) = \phi_i(x)$ if $x \notin \text{domain } F_v$.

We begin with a proof of the Blum speed-up theorem. This proof, which we believe to be simpler than the original proof in [2] or the later proof in [6], may be viewed as a simplification of Blum's proof in [2]. It is well known that, because all measures are recursively related [2], if the result holds for one measure it is easily shown to hold for all measures. We begin by considering any measure satisfying

(α_1) $\Phi_{S(i, x)}(y) \leq \Phi_i(x, y)$, and

(α_2) if $x \notin \text{domain } F_v$, $\Phi_{\rho(i, v)}(x) \leq \Phi_i(x)$.³

For example, Turing machine tape is such a measure. We wish to prove

THEOREM 1 (BLUM SPEED-UP THEOREM [2]). *For every total recursive function $r(x, y)$ we can effectively find a total recursive function $f(x)$ such that $\phi_i = f$ implies there exists j such that $\phi_j = f$ and $\Phi_i(x) > r(x, \Phi_j(x))$ a.e.*

PROOF. The idea of the proof is to build a program t which computes a function $\phi_t(u, x)$ such that, for each u , $\lambda x \phi_t(u, x) = \lambda x \phi_t(0, x)$. We take $f = \lambda x \phi_t(0, x)$ and in the construction use the first parameter u in a diagonalization over all possible programs to guarantee that if $\phi_i = f$ then $\Phi_i(x) \geq r(x, \Phi_{i+1}(x))$ a.e. In view of equation (α_1) this would suffice to prove

² In writing $\subseteq$ and $\bigcup$, we recall that every function is a set of ordered pairs.

³ We will also assume that it is legitimate to define a function recursively, not just from its earlier values, but also from its earlier run times. Intuitively, this amounts to assuming that if we used a program to calculate the value of a function at an early argument, we can know the resources used in the computation even if we do not explicitly know the entire program used for computing the function. Formally of course, one must use the recursion theorem or some other means to validate such an argument.

the theorem. Unfortunately it can be proven [3] that in general there is no way to do this and still have $\lambda x \phi_i(u, x) = \lambda x \phi_i(0, x)$ for all u . However, in view of equation (α_2) , it would suffice to have $\lambda x \phi_i(u, x) = \lambda x \phi_i(0, x)$ a.e. for each u , because we can always "patch" $\lambda x \phi_i(u, x)$ on finitely many arguments without increasing the basic complexity. This suggests that we first prove the following weakened version of Theorem 1:

LEMMA 1 (EFFECTIVE PSEUDO-SPEED-UPS; BLUM [3]). *Let Φ be a measure satisfying (α_1) and the conditions of footnote 3. Then for every total recursive function $r(x, y)$ we can effectively find a total recursive function f such that given any i for which $\phi_i = f$ we can effectively find a j for which $\phi_j = f$ a.e. and $\Phi_i(x) > r(x, \Phi_j(x))$ a.e.*

PROOF. We will define a program t computing a partial recursive function $\phi_t(u, x)$ of two variables. f will be $\lambda x \phi_t(0, x)$ and we shall show that ϕ_t is total. $\phi_t(u, x)$ is defined recursively from $\phi_t(u, 0)$, $\phi_t(u, 1)$, $\dots$, $\phi_t(u, x-1)$, and also, if $u < x$, recursively from $\phi_t(x, x)$, $\phi_t(x-1, x)$, $\phi_t(x-2, x)$, $\dots$, $\phi_t(u+1, x)$. Specifically, at Stage x we define

$$C_{u,x} = \left\{ i \mid u \leq i < x \text{ and } i \notin \bigcup_{y < x} C_{u,y} \text{ and } \Phi_i(x) \leq r(x, \Phi_i(i+1, x)) \right\}.$$

We say that $C_{u,x}$ is the set of programs *cancelled* at Stage x in the computation of $\phi_t(u, x)$. And we define

$$\phi_t(u, x) = 1 + \max\{\phi_i(x) \mid i \in C_{u,x}\}.$$

I.e., $\phi_t(u, x) = 1 + \max\{\phi_i(x) \mid u \leq i < x \text{ and } \Phi_i(x) \leq r(x, \Phi_i(i+1, x)) \text{ and } i \text{ was not cancelled earlier in the computation of } \lambda x \phi_t(u, x)\}$.

It is immediate from this definition that $\phi_t(u, 0) = 1$ for all u and $\phi_t(u, x) = 1$ whenever $u \geq x$. Furthermore for any $u < x$, in order to have $\phi_t(u, x)$ defined it suffices to have $\Phi_t(x, x)$, $\Phi_t(x-1, x)$, $\Phi_t(x-2, x)$, $\dots$, and $\Phi_t(u+1, x)$ all defined and $\phi_t(u, 0)$, $\phi_t(u, 1)$, $\dots$, and $\phi_t(u, x-1)$ all defined. Hence it suffices to have $\phi_t(x, x)$, $\phi_t(x-1, x)$, $\dots$, $\phi_t(u+1, x)$; $\phi_t(u, 0)$, $\phi_t(u, 1)$, $\dots$, and $\phi_t(u, x-1)$ all defined.

Proceeding inductively, if we assume that $\lambda u \phi_t(u, x')$ is total for all $x' < x$, we observe next that $\phi_t(u, x) = 1$ for all $u \geq x$ so $\phi_t(x-1, x)$ is defined. Consequently by the same reasoning $\phi_t(x-2, x)$ is defined. Proceeding recursively down to $\phi_t(0, x)$, we see that $\lambda u \phi_t(u, x)$ is total, completing an inductive proof that ϕ_t is total.

Clearly $C_{0,x} - \{0, 1, \dots, u-1\} = C_{u,x}$. Furthermore it is obvious that for each u there exists n_u such that if $i < u$ and $i \in \bigcup_v C_{0,v}$ then

$$i \in \bigcup_{v \leq n_u} C_{0,v}.$$

Thus no $i < u$ belongs to $C_{0,x}$ for $x > n_u$. It is thus immediate that for $x > n_u$, $C_{0,x} = C_{u,x}$ so $\phi_i(0, x) = \phi_i(u, x)$ for $x > n_u$.

Finally, if $\phi_i = \lambda x \phi(0, x)$ we must have

$$(\beta) \quad \phi_i(x) > r(x, \Phi_i(i+1, x)) \quad \text{for all } x > i,$$

for if not, in the calculation of $\lambda x \phi_i(0, x)$ we must *cancel* i for the first such $x > i$, forcing the contradiction $\phi_i(0, x) \neq \phi_i(x)$. The proof is now essentially complete:

We may assume without loss of generality that r is monotone in its second variable. Since from (β) and (α_1) , we have

$$(\gamma) \quad \Phi_i(x) > r(x, \Phi_i(i+1, x)) \geq r(x, \Phi_{S(t, i+1)}(x)) \quad \text{a.e.}$$

This completes the proof of Lemma 1.

Since for each i , $\lambda x \phi_i(0, x) = \phi_{\rho(S(t, i+1), v)}(x)$ for some v , in measures satisfying (α_2) we obtain, from (γ) ,

$$(\delta) \quad \Phi_i(x) \geq r(x, \Phi_{\rho(S(t, i+1), v)}(x)) \quad \text{a.e.,}$$

proving Theorem 1 for such measures.

As remarked earlier, Theorem 1 for arbitrary measures now follows by recursive relatedness, but a direct proof for arbitrary measures is easily obtained by modifying the above proof. To accomplish this, one simply defines ϕ_t by an application of the recursion theorem, modifying the definition of $C_{u,x}$ to

$$C_{u,x} = \left\{ i \mid u \leq i < x \text{ and } i \notin \bigcup_{y < x} C_{u,y} \text{ and } \Phi_i(x) \leq \max_{v < x} r(x, \Phi_{\rho(S(t, i+1), v)}(x)) \right\}.$$

The preceding proof now goes through virtually verbatim, except that the proof of (β) translates directly to a proof of the desired conclusion, namely (δ) .

The speed-up theorem of [2] is actually slightly stronger than Theorem 1 because, in [2], f is taken to be zero-one valued. We leave the modifications of our proof necessary to make f zero-one valued as an exercise for the reader.

Our next proof is of the Borodin gap-theorem. The only advantage this proof has over the original proof [4] is that this proof seems to be more easily generalized to obtain an operator gap-theorem. Intuitively, gap theorems assert that there is no uniform method to increase bounds on computational resources in a way which guarantees that the larger bound

⁴ It is the proof of *this* relation which makes other speed-up proofs more difficult than ours. Our *explicit* construction of this relation avoids the more elaborate appeals to the recursion theorem or the Turing machine simulations of earlier proofs.

always enables one to perform some calculation which could not be carried out within the smaller bound. A gap-theorem not stated in this paper may be found in [1]. Deep results which enable one to partially overcome gap phenomena by selecting well-behaved bounding functions are given in [9] and [11] and exploited in [1].

THEOREM 2 [4]. *For all total recursive functions $a(x)$ and $r(x, y)$ (with $r(x, y) \geq y$) we can effectively find a total recursive function t such that $t \geq a$ a.e. and $\Phi_j(x) \geq t(x)$ implies $\Phi_j(x) > r(x, t(x))$ provided $x > j$.*

PROOF. To define $t(x)$ set $t_{x+1} = a(x)$, and for $0 < i \leq x+1$, $t_{i-1} = r(x, t_i) + 1$. Thus $t_{x+1} < t_x < t_{x-1} < \dots < t_1 < t_0$. Of the $x+1$ intervals $[t_i, t_{i-1})$, we can find at least one, $[t_{i_0}, t_{i_0-1})$ such that for no $j < x$ do we have

$$t_{i_0} \leq \Phi_j(x) \leq r(x, t_{i_0}) < t_{i_0-1}.$$

The theorem is thus obtained by setting $t(x) = t_{i_0}$.

It is an instructive exercise to compare the way the preceding proof is generalized in the following proof to the way Borodin's proof naturally generalizes in [5].

THEOREM 3 (OPERATOR GAP; CONSTABLE [5]). *For every total recursive function a and every total effective operator Ψ (with $\Psi(f) \geq f$) we can effectively find a monotonic total recursive function t such that $t \geq a$ and*

$$\{j \mid \Phi_j \leq t \text{ a.e.}\} = \{j \mid \Phi_j \leq \Psi(t) \text{ a.e.}\}^5$$

PROOF. Proceeding inductively, assume that we are at Stage x in a construction of t and that, for some $d_x \geq x$, t has been defined for all $z < d_x$. We denote this finite initial segment of t by $t^{(x)}$ and define $x+1$ total extensions of $t^{(x)}$ (called $t_x, \dots, t_0$) by

$$t_x(z) = \max\{a(z), t_x(z-1) + 1\} \quad \text{for all } z \geq d_x,$$

and

$$t_{i-1}(z) = \max\{t_i(z) + 1, t_{i-1}(z-1), \Psi(t_i)(z)\}^6$$

(The reader will find it useful to draw a picture of the t_i and of their initial segments F'_i .) We next define finite initial segments $F'_0, F'_1, \dots, F'_x$ of $t_0, t_1, \dots, t_x$ by

$$F'_0 = t_0 / \{0, 1, \dots, d_x\}, \quad \text{and} \quad F'_{i+1} = t_{i+1} / \{0, 1, \dots, A_{i+1}\}$$

where

$$A_0 = d_x \quad \text{and} \quad A_{i+1} = (\mu z)[z \geq A_i \text{ and } F'_i \subseteq_{F_v \subset t_{i+1}/\{0, \dots, z\}} F_{\Psi(v)}].$$

⁵ A stronger extension of Theorem 2 would also assert that $\Phi_j \geq t$ infinitely often implies $\Phi_j \geq \Psi(t)$ infinitely often. However, this is false [5].

⁶ By convention, for any function t' , $t'(-1) = -1$.

The relevant information is that if t' is *any* extension of such an F'_i for $i \leq x$, then $\Psi(t')(z) \leq t_{i-1}(z)$ for any z such that $d_x \leq z \leq A_{i-1}$.

Now for each $j < x$, we define F'_i to be *unsafe* for j if

(α) $F'_i(z) < \Phi_j(z)$ for some z such that $d_x \leq z \leq A_i$, but

(β) $\Phi_j(z') \leq F'_{i-1}(z')$ for all z' such that $d_x \leq z' \leq A_{i-1}$,

(where (β) is assumed vacuously true if $i=0$). Note that if F'_i is unsafe for j , then for any $i' < i$, since $A_{i'} \leq A_i$, clause (β) guarantees that $\Phi_j(z) \leq F'_{i'}(z)$ for all $d_x \leq z \leq A_{i'}$; similarly we cannot have $\Phi_j(z') \leq F'_{i-1}(z')$ for all z' such that $d_x \leq z' \leq A_{i'-1}$ for any $i' > i$. Thus for any $j < x$, at most one F'_i is unsafe for j . It follows that at least one of $F'_x, F'_{x-1}, \dots, F'_0$ must be safe for every $j < x$. Since we can effectively test whether F'_i is safe for j , we may now extend $t^{(x)}$ to F'_{i_0} where F'_{i_0} is safe for every $j < x$. We thus go on to Stage $x+1$ with $d_{x+1} = A_{i_0} + 1$ and $t^{(x+1)} = F'_{i_0}$.

Clearly we have defined t to be a monotonic total recursive function, and, for every j and for every $x > j$, when we extend t at Stage x ; the extension is safe for j , guaranteeing either that, for every z' in the domain of the extension, $\Phi_j(z') \leq t(z')$ or else that, for some z' is the domain of the extension, $\Psi(t)(z') < \Phi_j(z')$. It follows that if $t(z') < \Phi_j(z')$ infinitely often then $\Psi(t)(z') < \Phi_j(z')$ infinitely often, proving the theorem.

In [10], Meyer and Fischer give a proof of the following generalization of Theorem 1:

THEOREM 4. *For every total effective operator Ψ , and every total recursive function a , there exists a zero-one valued total recursive function f such that $\phi_i = f$ implies $\Phi_i \geq a$ a.e. and $\phi_i = f$ implies there exists a j such that $\phi_j = f$ and $\Phi_i \geq \Psi(\Phi_j)$ a.e.*

Their proof, which is elegant and short, has its computational details obscured by clever use of the recursion theorem. Our method of proving Theorem 1 has an extension to a proof of Theorem 4. Although our proof is a little longer and less elegant than the Meyer-Fischer proof, it is straightforward, avoiding all appeals to the recursion theorem in "reasonable" measures and making only straightforward appeals in any case.

As in the proof of Theorem 1, we first prove a weaker lemma, assuming that the measure satisfies the conditions of inequalities (α_1), (α_2), and footnote 3. In proving this lemma, we explicitly introduce a "complexity sequence" λp_i of total recursive functions which roughly satisfy $p_i \geq \Psi(p_{i+1})$ a.e. This achieves speed-up by the operator Ψ because we will make the sequence λp_i cofinal with the run times, Φ_j , of the function f which we are defining. In the proof of Lemma 1, our complexity sequence was (implicitly) given by $p_i(x) = r(x, \Phi_i(i+1, x))$. Just as in the construction of the sequences λt_i of the proofs of Theorems 2 and 3, our point

of view here is that to *explicitly* construct a sequence λp_i satisfying $p_i \gg p_{i+1}$ a.e. one must start "somewhere in the middle" and, e.g., before defining $p_0(x)$, first define $p_x(x)$ and then inductively work up to $p_0(x)$ by defining p_u from p_{u+1} for $0 \leq u < x$. (For a full discussion of the role of "complexity sequences" for speed-ups, see [9].)

LEMMA 2 (EFFECTIVE OPERATOR PSEUDO-SPEED-UPS). *Let Φ be any measure satisfying (α_1) and the conditions of footnote 3. Then for every total effective operator Ψ and total recursive function a , we can effectively find a total recursive function f such that: (i) For every i such that $\phi_i = f$, $\Phi_i \geq a$ a.e., and (ii) for every i such that $\phi_i = f$ we can effectively find a j such that $\phi_j = f$ a.e. and $\Phi_i > \Psi(\Phi_j)$ a.e.*

OUTLINE OF PROOF. We define a complexity sequence λp_i of total recursive functions in stages. At Stage x , for all $u \geq x$ and all $y = 0, 1, 2, 3, \dots$, we begin assigning the values $p_u(y) = a(y)$ unless $p_u(y)$ was defined at an earlier stage. Simultaneously, for every u and every y (including $y < x$), whenever $p_{u+1}(y), p_{u+2}(y), \dots, p_y(y)$ are all defined and $\phi_i(u, 0), \dots, \phi_i(u, y-1)$ are all defined, if $\phi_i(u, y)$ was not defined at an earlier stage, define

$$(\beta) \quad \phi_i(u, y) = 1 + \max\{\phi_i(y) \mid u \leq i < y \text{ and } \Phi_i(y) \leq p_{i+1}(y) \text{ and } i \text{ was not cancelled while computing } \phi_i(u, 0), \phi_i(u, 1), \dots, \phi_i(u, y-1)\}.$$

Cancel all such i . Also during Stage x , for all $u < x$ and for all y such that $\phi_i(u, y)$ was not defined at an earlier stage, whenever enough of $\lambda z \phi_i(u+1, z)$ (and hence of $\lambda z \Phi_i(u+1, z)$) is defined to make

$$\Psi(\lambda z \Phi_i(u+1, z))(y)$$

defined, define

$$(\gamma) \quad p_u(y) = \Psi(\lambda z \Phi_i(u+1, z))(y).$$

Stage x ends when at least all of $p_u(x)$ for $u \leq x$ and $p_x(y)$ for $y \leq x$ have been defined.

It is straightforward to show that for each x , Stage x must end, and hence that $p_u(x)$ is defined for all u and x as is $\phi(u, x)$. Furthermore, for any u at any Stage x with $x \geq u$, $p_u(y)$ can only be defined by equation (γ) . Hence $p_u = \Psi(\lambda z \Phi_i(u+1, z))$ a.e. Clearly if $\phi_i = \lambda z \phi_i(0, z)$, then we must have

$$\Phi_i > p_{i+1} = \Psi(\lambda z \Phi_i(i+1, z)) \text{ a.e.}$$

for otherwise i gets cancelled. Furthermore from equation (β) just as in the proof of Lemma 1, for any u , $\lambda z \phi_i(0, z) = \lambda z \phi_i(u, z)$ a.e.

This essentially completes the proof: Assuming the operator Ψ is monotone, since from (α_1) we have $\Phi_t(i+1, z) \geq \Phi_{S(t, i+1)}(z)$, we can obtain $\Phi_i > \Psi(\lambda z \Phi_t(i+1, z)) \geq \Psi(\Phi_{S(t, i+1)})$ a.e. completing the proof of Lemma 2. To prove Theorem 4 (still without the requirement that f be zero-one valued) if Ψ is monotone, we obtain from (α_2) that $\Psi(\Phi_{S(t, i+1)}) \geq \Psi(\Phi_{\rho(S(t, i+1, v))})$ a.e. for some v such that $\phi_{\rho(S(t, i+1, v))} \lambda z \phi_t(0, z)$.

Unfortunately, although every *general* recursive operator Ψ is bounded by a monotone operator, not every total effective operator is so bounded, [7]. In the event that Ψ cannot be bounded by a monotone operator, the above construction becomes much more complicated unless one uses an appeal to the recursion theorem to allow changing equation (γ) to

$$(\gamma') \quad p_u(y) = \max\{\Psi(\Phi_{\rho(S(t, u+1, v))})(y) \mid v \leq y\}$$

with the obvious changes in the sentence preceding (γ) to be sure that enough of ϕ_t has been defined to make $p_u(y)$ defined. We leave these changes and the changes necessary to make f zero-one valued to the reader.

In closing we would like to remark that not all results of recursion theoretic complexity theory seem accessible to such simple diagonalization and "dove-tailing of computations" arguments as those employed here. For examples of apparently more difficult results as well as some open questions, the reader might consult [3], [8], [10], [11], and [13].

REFERENCES

1. L. Bass and P. Young, *Hierarchies based on computational complexity and irregularities of class determining measured sets*, J. Assoc. Comput. Mach. **20** (1973).
2. M. Blum, *A machine-independent theory of the complexity of recursive functions*, J. Assoc. Comput. Mach. **14** (1967), 322–336. MR **38** #4213.
3. ———, *On effective procedures for speeding up algorithms*, J. Assoc. Comput. Mach. **18** (1971), 290–305.
4. A. Borodin, *Computational complexity and the existence of complexity gaps*, J. Assoc. Comput. Mach. **19** (1972), 158–174.
5. R. Constable, *The operator gap*, J. Assoc. Comput. Mach. **19** (1972), 175–183.
6. J. Hartmanis and J. Hopcroft, *An overview of the theory of computational complexity*, J. Assoc. Comput. Mach. **18** (1971), 444–475.
7. John Helm, *On effectively computable operators*, Z. Math. Logik Grundlagen Math. **17** (1971), 231–244.
8. J. Helm and P. Young, *On size vs. efficiency for programs admitting speed-ups*, J. Symbolic Logic **36** (1971), 21–27.
9. E. McCreight and A. Meyer, *Classes of computable functions defined by bounds on computation*, Proc. First Annual ACM Sympos. on Theory of Computing, pp. 79–88.
10. A. Meyer and P. Fischer, *Computational speed-up by effective operators*, Project MAC Report, Summer 1970; J. Symbolic Logic **37** (1972), 48–68.
11. A. Meyer and R. Moll, *Honest bounds for complexity classes of recursive functions*, Project MAC Report, April 1972.

12. Hartley Rogers, Jr., *Theory of recursive functions and effective computability*, McGraw-Hill, New York, 1967. MR 37 #61.

13. P. Young, *Speed-ups by changing the order in which sets are enumerated*, Math. Systems Theory 5 (1971), 148–156; minor correction, *ibid.* 6 (1972).

DEPARTMENT OF MATHEMATICS, PURDUE UNIVERSITY, WEST LAFAYETTE, INDIANA 47907

Current address: Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720