Extensional properties of sets of time bounded complexity (extended abstract)

WOLFGANG MAASS* AND THEODORE A. SLAMAN**

Abstract. We analyze the fine structure of time complexity classes for RAM's, in particular the equivalence relation $A =_C B$ ("A and B have the same time complexity") $\Leftrightarrow$ (for all time constructible f : $A \in DTIME_{RAM}(f) \Leftrightarrow B \in DTIME_{RAM}(f)$). The $=_C$-equivalence class of A is called its *complexity type*. We prove that every set X can be partitioned into two sets A and B such that $X =_C A =_C B$, and that the partial order of sets in an arbitrary complexity type under $\subseteq^*$ (inclusion modulo finite sets) is dense. The proofs employ a new strategy for finite injury priority arguments.

We consider the following set of time bounds:

$$T := \{f : \mathbf{N} \to \mathbf{N} \mid f(n) \geq n \text{ and } f \text{ is time constructible on a RAM}\},$$

where f is called time constructible on a RAM if some RAM can compute the function $1^n \mapsto 1^{f(n)}$ in $O(f(n))$ steps. We do not allow arbitrary recursive functions as time bounds in our approach in order to avoid pathological phenomena (e.g. gap theorems [HU], [HH]). In this way we can focus on those aspects of complexity classes that are relevant for concrete complexity (note that all functions that are actually used as time bounds in the analysis of algorithms are time constructible). We use the random access machine (RAM) with uniform cost criterion as machine model (e.g. as defined in [CR]; see also [AHU], [MY]) because this is the most frequently considered model in algorithm design, and because a RAM allows more sophisticated diagonalization - constructions than a Turing machine. One defines $DTIME_{RAM}(f) := \{A \subseteq \{0,1\}^* |$ there is a *RAM*

*Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, IL. 60680. Written under partial support by NSF-Grant CCR 8703889.

**Department of Mathematics, University of Chicago, Chicago, IL. 60637. Written under partial support by Presidential Young Investigator Award DMS-8451748 and NSF-Grant DMS-8601856.

of time complexity $O(f)$ that computes $A\}$. We write $DTIME(f)$ for $DTIME_{RAM}(f)$ in the following.

For sets $A, B \subseteq \{0,1\}^*$ we define

$$A =_C B(\text{“}A \text{ has the same det. time complexity as } B\text{”})$$
$$:\Leftrightarrow \forall f \in T(A \in DTIME(f) \Leftrightarrow B \in DTIME(f)).$$

A *complexity type* is an equivalence class of this equivalence relation $=_C$.

We write $\mathbf{0}$ for $DTIME(n)$, which is the “minimal” complexity type. Note that for every complexity type $\mathcal{C}$ and every $f \in T$ one has either $\mathcal{C} \subseteq DTIME(f)$ or $\mathcal{C} \cap DTIME(f) = \emptyset$.

In this paper we investigate some basic properties of the partial order

$$PO(\mathcal{C}) := \langle \{X | X \in \mathcal{C}\}, \subseteq^* \rangle,$$

where C is an arbitrary complexity type and $\subseteq^*$ denotes inclusion modulo finite sets (i.e. $X \subseteq^* Y :\Leftrightarrow X - Y$ is finite).

This work is part of the long range project to study the relationship between extensional properties of a set and its computational complexity. Among other work in this direction we would like to mention in particular the study of the complexity of sparse sets (see e.g. [Ma]), and the investigation of the relationship between properties of recursively enumerable sets under $\subseteq^*$ and their degree of computability (see e.g. Chapter XI in [So]). Our approach differs from this preceding work insofar as it also applies to “actually computable” sets (i.e. sets in P). Therefore it provides an opportunity to develop finer construction tools that can be used to examine also the structure of sets of small complexity. In this paper we introduce a new strategy for a finite injury priority argument that allows us to prove splitting and density theorems for sets of arbitrarily given complexity type $\mathcal{C}$ (for example sets of time complexity $\Theta(n^2)$). Further results about the structure of complexity types can be found in [MS].

Theorem 1. Every set X can be split into two sets A, B of the same complexity type as X (i.e. $X = A \cup B$, $A \cap B = \emptyset$, $X =_C A =_C B$).

In order to *prove* this result one needs a technique for controlling the complexity type of the constructed sets A, B. This is less difficult if X has an "optimal" time bound $f_X \in T$ for which $\{f \in T | X \in DTIME(f)\} = \{f \in T | f = \Omega(f_X)\}$ (in this case we say that X is of *principal* complexity type). However Blum's speed-up theorem [B] asserts that there are for example sets $X \in P$ such that

$$\{f \in T \mid X \in DTIME(f)\} = \left\{f \in T \mid \exists i \in \mathbf{N}\left(f(n) = \Omega\left(\frac{n^2}{(\log n)^i}\right)\right)\right\}.$$

Note that this effect occurs even if one is only interested in time constructible time bounds (and sets X of "low" complexity).

In order to prove Theorem 1 also for sets X whose complexity type is non-principal, we show that in some sense the situation of Blum's speed up theorem (where we can characterize the functions f with $X \in DTIME(f)$ with the help of a "cofinal" sequence of functions) is already the worst that can happen (unfortunately this is not quite true, since we cannot always get a cofinal sequence of functions f_i where $f_{i+1}(n) = O\left(\frac{f_i(n)}{g(n)}\right)$ for a fixed function g with $g(n) \to \infty$ for $n \to \infty$, as required for the proof of the speed-up theorem).

Definition. $(t_i)_{i \in \mathbf{N}} \subseteq \mathbf{N}$ is called a characteristic T-sequence if $t : i \mapsto t_i$ is recursive and

a) $\forall i \in \mathbf{N}(\{t_i\} \in T$ and program t_i is a witness for the time-constructibility of $\{t_i\})$

b) $\forall i, n \in \mathbf{N}(\{t_{i+1}\}(n) \leq \{t_i\}(n))$.

Lemma 1. ("inverse of the speed-up theorem"). For every recursive set A there exists a characteristic T-sequence $(t_i)_{i \in \mathbf{N}}$ such that $(t_i)_{i \in \mathbf{N}}$ is characteristic for A (i.e. $\forall f \in T(A \in DTIME(f) \Leftrightarrow \exists i \in \mathbf{N}(f(n) = \Omega(\{t_i\}(n)))))$.

One can also prove the *converse of Lemma 1* and construct for any given characteristic T-sequence $(t_i)_{i \in \mathbf{N}}$ a set A such that

$$\forall f \in T(A \in DTIME(f) \Leftrightarrow \exists i \in \mathbf{N}\ (f(n) = \Omega(\{t_i\}(n)))).$$

This construction. which is a refinement of the proof of Blum's speed-up theorem, is one component in the priority-constructions of Theorems 1 and 2. . It is more delicate

because in our more general situation it is not guaranteed that there is a function g with $g(n) \to \infty$ (or at least $g(n) \geq 2$) such that $\forall i(\{t_{i+1}\} \leq \{t_i\}/g)$ (the existence of such uniform "gap" g is used in the customary proof of Blum's speed-up theorem). Some further details of the proof are described in [MS].

Remark. The idea of characterizing the complexity of an arbitrary recursive set by a sequence of "cofinal" complexity bounds is rather old (see e.g. [MF], [L], [LY], [SS], [MW]). However none of these results provide the here needed characterization of the time complexity of an arbitrary recursive set in terms of a uniform cofinal sequence of time constructible time bounds. [L] and [MW] give corresponding results for space complexity of Turing-machines. These results exploit the linear speed-up theorem for space complexity on Turing-machines, which is not available for time complexity on RAM's. Time complexity on RAM's has been considered in [SS], but only sufficient conditions are given for the cofinal sequence of time bounds (these conditions are stronger than ours, and they are probably not necessary). The more general results on complexity sequences in axiomatic complexity theory ([MF], [SS]) involve "overhead functions", or deal with nonuniform sequences, which makes the specialization to the notions considered here impossible. It is an open problem whether one can characterize the time complexity of any recursive set on Turing-machines in the same way as it is done here for RAM's (because of the lack of a fine time hierarchy theorem for multi-tape Turing-machines).

Idea of the Proof of Theorem 1. Associate with the given set X a characteristic T-sequence $(t_i)_{i \in N}$ as in Lemma 1. For every $e, n \in \mathbf{N}$ and $x \in \{0,1\}^*$ define

$$TIME(e,x) := (\text{number of steps in the computation of } \{e\} \text{ on input } x)$$

and

$$MAXTIME(e,n) := \max\{TIME(e,x) \mid |x| = n\}.$$

It is sufficient to partition X into sets A and B in such a way that for every $e \in \mathbf{N}$ the following requirements R_e^A, R_e^B, S_e^A, S_e^B are satisfied:

$$R_e^A :\Leftrightarrow (A = \{e\} \Rightarrow \forall f \in T(\forall n(MAXTIME(e,n) \leq f(n))$$
$$\Rightarrow \exists j \in \mathbf{N}(f(n) = \Omega(\{t_j\}(n)))))$$
$$S_e^A :\Leftrightarrow A \in DTIME(\{t_e\}(n)).$$

R_e^B, S_e^B are defined analogously.

Note that it is not possible to satisfy R_e^A by simply setting $A(x) := 1 - \{e\}(x)$ for some x: in order to achieve that $A \subseteq X$ we can only place x into A if $x \in X$.

Instead, we adopt the following strategy to satisfy R_e^A (the strategy for R_e^B is analogous): For input $x \in \{0,1\}^*$ compute $\{e\}(x)$.

Case I. If $\{e\}(x) = 0$, then this strategy issues the constraint "$x \in A \Leftrightarrow x \in X$",

Case II. If $\{e\}(x) = 1$, then this strategy issues the constraint "$x \notin A$" (which forces x into B if $x \in X$).

In the case of a conflict for some input x between strategies for different requirements one lets the requirement with the highest priority (i.e. the smallest index e) succeed (this causes in general an "injury" to the other competing requirements).

The interaction between the described strategies is further complicated by the fact that in the case where R_e^A is never satisfied via Case II, or via Case I for some $x \in X$, we have to be sure that Case I issues a constraint *for almost every input* x (provided that the simulation of $\{e\}(x)$ is not prematurely halted by some requirement S_i^A with $i \leq e$, see below). Consequently the number of requirements whose strategies act on the same input x grows with $|x|$ (only those R_i^A, R_i^B with $i < |x|$ can be ignored where one can see by "looking back" for $|x|$ steps that they are already satisfied).

The strategy for requirement $S_e^A(S_e^B)$ is as follows: it issues the constraint that for all inputs x with $|x| \geq e$ the *sum* of all steps that are spent on simulations for the sake of requirements $R_i^A, R_i^B, S_i^A, S_i^B$ with $i \geq e$ has to be bounded by $O(\{t_e\}(|x|))$. One can prove that in this way $S_e^A(S_e^B)$ becomes satisfied (because only finitely many inputs are placed into A or B for the sake of requirements of higher priority). One also has to prove that the constraint of S_e^A does not hamper the requirements of lower priority in a serious manner.

This part of the construction is more difficult than its counterpart in Blum's speed-up-theorem [B], because it need not be the case that $\{t_{i+1}\} = o(\{t_i\})$. A further complication is caused by the fact that although there are constants K_i, K_{i+1} such that $\{t_i\}(n)$ converges in $\leq K_i \cdot \{t_i\}(n)$ steps and $\{t_{i+1}\}(n)$ converges in $\leq K_{i+1} \cdot \{t_{i+1}\}(n)$ steps, we may have that $K_i \ll K_{i+1}$ (and therefore $K_i \cdot \{t_i\}(n) \ll K_{i+1} \cdot \{t_{i+1}\}(n)$). Therefore the requirements S_j^A with $j > i$ are not able to "take over" the job of S_i^A, and *all* computations $\{t_i\}(|x|)$, $i \leq |x|$, have to be simulated simultaneously for each input x.

In order to show that a single RAM R can carry out simultaneously all of the described strategies, one exploits in particular that a RAM can dovetail an unbounded number of simulations in such a way that the number n_e of steps that it has to spend in order to simulate a single step of a simulated program $\{e\}$ does not grow with the number of simulated programs (the precise construction of R is rather complex).

In order to verify that this construction succeeds, one has to show that each requirement R_e^A, R_e^B is "injured" at most finitely often. This is not obvious, because we may have for example that R_{e-1}^B (which has higher priority) issues overriding constraints for infinitely many arguments x according to Case I. However in this case we know that only finitely many of these x are elements of X (otherwise R_{e-1}^B would have been seen to be satisfied from some point of the construction on), and all of its other constraints are "compatible" with the strategies of lower priority (since we make $A, B \subseteq X$).

Finally we verify that each requirement $R_e^A(R_e^B)$ is satisfied. This is obvious if Case II occurs in the strategy for R_e^A for some input x where R_e^A is no longer injured; or if Case I occurs for such input x with $x \in X$ (in both cases we can make $A \neq \{e\}$). However it is also possible that $x \notin X$ for each such x (and that $\{e\} = A$), in which case R_e^A becomes satisfied for a different reason. In this case we have $\{e\}(x) = 0 = X(x)$ for each such x. Therefore we can use $\{e\}$ to design a new algorithm for X that is (for every input) at least as fast as the algorithm $\{e\}$ for A (it uses $\{e\}$ for those inputs where $\{e\}$ is faster than the "old" algorithm for X of time complexity $\{t_e\}$). Therefore one can prove that $X \in DTIME(f)$ for every $f \in T$ that bounds the running time of algorithm $\{e\}$ for A. This implies that $f(n) = \Omega(\{t_j\}(n))$ for some $j \in \mathbf{N}$ (by construction of the characteristic T-sequence $(t_i)_{i \in \mathbf{N}}$). □

Idea of the proof of Theorem 2. Let $(t_i)_{i\in\mathbf{N}}$ be a characteristic T-sequence for X. It is sufficient to construct A such that $Y \subseteq A \subseteq X$ and for all $e \in \mathbf{N}$ the requirements R_e, S_e, T_e, U_e are satisfied, where R_e, S_e are identical with the requirements R_e^A, S_e^A in the proof of Theorem 1 (together they ensure that $A =_C X$) and

$$T_e: \quad |A - Y| \geq e$$

$$U_e: \quad |X - A| \geq e.$$

The strategy to satisfy R_e is similar to the strategy for satisfying R_e^A. However in Case II (where $\{e\}(x) = 1$), unlike in the splitting theorem, R_e does not have the power to keep x out of A (even if R_e has the highest priority) because x may later enter Y. Instead, R_e issues in Case II the constraint "$x \notin A \Leftrightarrow x \notin Y$" (i.e. R_e wants to keep x out of A if it turns out that $x \notin Y$).

It is easy to see that R_e becomes satisfied if Case I occurs for some $x \in X$, or if Case II occurs for some $x \notin Y$ (provided that R_e is not "injured" at x by requirements of higher priority). If neither of these events occurs, then we can conclude that $\{e\}(x) = X(x)$ whenever the simulation of $\{e\}(x)$ can be finished before it is halted for the sake of some requirement S_i with $i \leq e$. This information can be used (as in the proof of Theorem 1) to design an algorithm for X that converges for every input x "at least as fast" as the computation $\{e\}(x)$. □

Corollary 3. For every complexity type $\mathcal{C}$ the partial order $PO(\mathcal{C})$ is dense.

It is easy to see that $PO(\mathcal{C})$ is isomorphic to the countable atomless Boolean algebra AB if $\mathcal{C} = \mathbf{0}$. Furthermore it was shown that AB can be embedded into $PO_{0,1}(\mathcal{C})$ for every complexity type $\mathcal{C}$. However the following corollary suggests that the structure of the partial order $PO(\mathcal{C})$ is substantially more complicated than that of AB if $\mathcal{C} \neq \mathbf{0}$. Obviously any complexity type $\mathcal{C} \neq \mathbf{0}$ is closed under complementation, but not under union or intersection. However, it could still be the case that any two sets $A, B \in \mathcal{C}$ have a least upper bound in the partial order $PO(\mathcal{C})$. This is ruled out by the following result.

Corollary 4. Consider an arbitrary complexity type $\mathcal{C} \neq \mathbf{0}$. Then any two sets $A, B \in \mathcal{C}$ have a least upper bound in the partial order $PO(\mathcal{C})$ if and only if $A \cup B \in \mathcal{C}$. In particular one can define with a first order formula over $PO(\mathcal{C})$ whether $A \cup B \in \mathcal{C}$ (respectively $A \cap B \in \mathcal{C}$) for $A, B \in \mathcal{C}$.

Proof. Assume that $A, B \in \mathcal{C}$, $A \cup B \notin \mathcal{C}$, $A \cup B \subseteq D$ and $D \in \mathcal{C}$. Then $(A \cup B) \leq_{\mathcal{C}} D$ and $(A \cup B) \underset{\infty}{\subset} D$. Thus there exists by Theorem 2 a set $D' \in \mathcal{C}$ with $(A \cup B) \underset{\infty}{\subset} D' \underset{\infty}{\subset} D$. Therefore D is not a least upper bound for A and B in $PO(\mathcal{C})$. □

Remark. This result suggests that the first order theory of the partial order $PO(\mathcal{C})$ is nontrivial for $\mathcal{C} \neq \mathbf{0}$.

Acknowledgement. We would like to thank Joel Berman for helpful comments.

REFERENCES

[AHU] A.V. AHO, J.E. HOPCROFT, J.D. ULLMAN, The Design and Analysis of Computer Algorithms, Addison-Wesley (Reading, 1974).

[B] M. BLUM, A machine-independent theory of the complexity of recursive functions, *J. ACM,* **14**(1967), 322-336.

[CR] S.A. COOK, R.A. RECKHOW, Time-bounded random access machines, *J. Comp. Syst. Sc.,* **7**(1973), 354-375.

[HH] J. HARTMANIS, J.E. HOPCROFT, An overview of the theory of computational complexity, *J. ACM,* **18**(1971), 444-475.

[L] L.A. LEVIN, On storage capacity for algorithms, *Soviet Math. Dokl.,* **14**(1973), 1464-1466.

[Ly] N. LYNCH, Helping: several formalizations, *J. of Symbolic Logic,* **40**(1975), 555-566.

[MY] M. MACHTEY, P. YOUNG, An Introduction to the General Theory of Algorithms, North-Holland (Amsterdam, 1978).

[Ma] S. MAHANEY, Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis, *J. Comp. Syst. Sc.,* **25**(1982), 130-143.

[MF] A.R. MEYER, P.C. FISCHER, Computational speed-up by effective operators, *J. of Symbolic Logic,* **37**(1972), 55-68.

[MW] A.R. MEYER, K. WINKLMANN, The fundamental theorem of complexity theory, *Math. Centre Tracts,* **108**(1979), 97-112.

[MS] W. MAASS, T.A. SLAMAN, On the complexity types of computable sets (extended abstract), to appear in : *Proc. of the Structure in Complexity Theory Conference* 1989.

[SS] C.P. SCHNORR, G. STUMPF, A characterization of complexity sequences, *Zeitschr. f. math. Logik u. Grundlagen d. Math.,* **21**(1975), 47-56.

[So] R.I. SOARE Recursively Enumerable Sets and Degrees, *Springer* (Berlin, 1987).