Proc. of a Workshop on Logic from Computer Science, Y.N. Moschovakis, ed.,
Springer (Berlin, 1991), 359-372.

SPLITTING AND DENSITY FOR THE RECURSIVE SETS OF A FIXED TIME COMPLEXITY

WOLFGANG MAASS AND THEODORE A. SLAMAN

ABSTRACT. We analyze the fine structure of the time complexity classes induced by random access machines. Say that A and B have the same time complexity ($A +_C B$) if for all time constructible f $A \in DTIME_{RAM}(f) \iff B \in DTIME_{RAM}(f)$. The $=_C$-equivalence class of A is called it *complexity type*. We examine the set theoretic relationships between the sets in an arbitrary complexity type C. For example, every recursive set X can be partitioned into two sets A and B such that $A =_C B =_C X$. Additionally, the ordering of C under $\subseteq^*$ (inclusion modulo finite sets) is dense.

1. INTRODUCTION

There has been a persistent intuition that the computational complexity of a set A of strings is related to the structure of the distribution of the elements of A. For example, Mahaney's [Ma] solution to the Berman-Hartmanis conjecture or Martin's [Mar] characterization of the Turing degrees of the maximal recursively enumerable sets grew out of the desire to test this intuition.

Although a large portion of current research in theoretical computer science is concerned with the investigation of problems that lie in P, very little is known about the relationship between the computational complexity and set theoretic aspects of sets in P. The results of this paper may be viewed as one step in this direction, since they provide nontrivial information about sets in P (for example about the structure of the class of sets of quadratic time complexity). We work in the context of a very fine scale for complexity and study the set theoretic properties among the recursive sets which are equally complex. Our results are all in the direction that there is a

The first author was partially supported by NSF-Grant CCR 8903398. The second author was partially supported by NSF-Grant DMS-8601856 and Presidential Young Investigator Award DMS-8451748 .

generous set theoretic variety among the sets of any particular time complexity. We show that for every recursive set X there are two sets of the same time complexity as X which partition the elements of X into disjoint pieces (Splitting Theorem); that if X and Y have the same complexity, $Y \subset X$ and $X - Y$ is infinite then there is an A such that $Y \subset A \subset X$ and both of $A - Y$ and $X - A$ are infinite (Density Theorem); and the countable atomless Boolean algebra can be embedded in the structure consisting of the collection of sets of the same complexity of X ordered by inclusion modulo the ideal of finite sets. The last result implies that the existential theory of this partial order is decidable.

In evaluating time complexity, we will use the random access machine (RAM) with uniform cost criterion (see [CR], [AHU], [MY], [P]) as our model for computation. This model is frequently adopted in considering the design of algorithms. It is also very sensitive to time complexity distinctions and allows sophisticated diagonalization constructions. It does not matter for the following which of the common versions of the RAM-model with instructions for ADDITION and SUBTRACTION of integers is chosen (note that it is common to exclude MULTIPLICATION of integers from the instruction set in order to ensure that the computation time of the RAM is polynomially related to time on a Turing machine). In order to be specific we consider the RAM model as it was defined by Cook and Reckhow [CR] (we use the common "uniform cost criterion" [AHU], i.e. $l(n) = 1$ in the notation of [CR]). In this model, a machine consists of a finite program, an infinite array $X_0, X_1, \ldots$ of registers (each capable of holding an arbitrary integer), and separate one-way input- and output-tapes. The program consists of instructions for ADDITION and SUBTRACTION of two register contents, the conditional jump "TRA m if $X_i > 0$" which causes control to be transferred to line m of the program if the current content of register X_i is positive, instructions for the transfer of register contents with indirect addressing, instructions for storing a constant, and the instruction "READ X_i" (transfer the content of the next input cell on the input-tape to register X_i) and "PRINT X_i" (print the content of register X_i on the next cell of the output-tape).

The relationship between computation time on RAM's and Turing machines is discussed in [CR] (Theorem 2), [AHU] (section 1.7), and [P] (chapter 3). It is obvious that a multi-tape Turing machine of time complexity $t(n)$ can be simulated by a RAM of time complexity $O(t(n))$. With a lit-

tle bit more work (see [P]) one can construct a simulating RAM of time complexity $O(n + (t(n)/\log t(n)))$ (assuming that the output has length $O(n + (t(n)/\log t(n)))$). In addition, Cook and Reckhow demonstrate the existence of a universal RAM M^* such that for any RAM M there is a constant m such that M^* can simulate k steps in the execution of M using only $m \cdot k$ steps of its own. By a diagonal argument, they conclude a fine time hierarchy theorem for RAM's.

Say that a recursive function f is *time constructible* on a RAM if there is a RAM which can compute the function $1^n \mapsto 1^{f(n)}$ in $O(f(n))$ many steps. Let T be the set of recursive functions from $\mathbb{N}$ to $\mathbb{N}$ that are non-decreasing and time constructible on a RAM. We adopt T as our scale for measuring time complexity. Typically, when one calculates either an upper or lower bound on the running time of a computational procedure that bound is an element of T.

Suppose that f is in T. Let $DTIME(f)$ be the collection of recursive sets $A \subseteq \{0,1\}^*$ such that A can be computed by a RAM of time complexity $O(f)$.

Definition. (1) Say that A *has the same deterministic time complexity as* B (written $A =_C B$) if for all $f \in T$, A is in $DTIME(f)$ if and only if B is in $DTIME(f)$.

(2) A *complexity type* is an $=_C$-equivalence class.

We write $\mathbf{0}$ for $DTIME(n)$, which is the least complexity type. Note, for every complexity type $\mathcal{C}$ and every f in T either $\mathcal{C} \subseteq DTIME(f)$ or $\mathcal{C} \cap DTIME(f) = \emptyset$. We will investigate some basic properties of the partial order

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

where $\mathcal{C}$ is an arbitrary complexity type and $\subseteq^*$ denotes inclusion modulo finite sets (i.e. $X \subseteq^* Y$ if and only if $X - Y$ is finite).

This paper is an expanded version of the expanded abstract [MS1] For a number of basic results about complexity types (and a list of open research problems) we refer to [MS2].

2. Splitting, Density and Embedding Theorems

We recall the following definition and results from [MS2], where it is

shown that for each recursive set A there is a normal form for the presentation of the running times of the RAM's which compute A. To fix some notation, let $\{f\}$ denote the recursive function whose program on a RAM is coded by f.

Definition. $(t_i)_{i\in\mathbb{N}} \subseteq \mathbb{N}$ is called a *characteristic sequence* if $f : i \mapsto t_i$ is recursive and

(1) $(\forall i \in \mathbb{N})[\{t_i\} \in T$ and the program t_i is a witness for the time-constructibility of $\{t_i\}]$;
(2) $(\forall i, n \in \mathbb{N})[\{t_{i+1}\}(n) \leq \{t_i\}(n)]$.

Definition. Let A be a recursive subset of $\{0,1\}^*$ and let $\mathcal{C}$ be a complexity type. Then, $(t_i)_{i\in\mathbb{N}}$ is *characteristic for* A if $(t_i)_{i\in\mathbb{N}}$ is a characteristic sequence and

$$(\forall f \in T)\,[A \in DTIME(f) \iff (\exists i \in \mathbb{N})\,(f(n) = \Omega(\{t_i\}(n)))]\,.$$

Similarly, $(t_i)_{i\in\mathbb{N}}$ is *characteristic for* $\mathcal{C}$ if $(t_i)_{i\in\mathbb{N}}$ is characteristic for some $A \in \mathcal{C}$ (or equivalently, for all $A \in \mathcal{C}$).

In [MS2], it is shown that for every recursive set A there is a sequence $(t_i)_{i\in\mathbb{N}}$ that is characteristic for A and for every characteristic sequence there is a recursive set for which it is characteristic.

Theorem. (Splitting Theorem) *For every recursive set X, there are two disjoint recursive sets A and B such that $A \cup B = X$ and $A =_{\mathcal{C}} B =_{\mathcal{C}} X$.*

Proof. Let X be recursive, let $\mathcal{C}$ be the complexity type of X and let $(t_i)_{i\in\mathbb{N}}$ be a characteristic sequence for X. We build sets A and B so that the following conditions hold.

(1) A and B are disjoint and their union is equal to X.
(2) For every i, A and B are both elements of $DTIME(\{t_i\})$.
(3) If g is an element of T and one of A or B is an element of $DTIME(g)$ then X is an element of $DTIME(g)$.

The first condition states that A and B split X. The second and third conditions together imply that both A and B are elements of $\mathcal{C}$.

Condition (1) imposes a simple constraint on the construction: A and B must be constructed by dividing the elements of X into two disjoint sets. The further actions we take during the construction operate within this constraint.

There are three ingredients to the proof: the strategies to ensure conditions (2) and (3) and the mechanism by which the strategies are combined. The strategies for (2) and the global organization of the construction are taken from [MS2], where they were used to show that every characteristic sequence is characteristic for some recursive set.

We will eventually organize our construction as a stage by stage priority construction. We will assign the strategies a priority ranking. During stage s, we will decide the values of $A(\sigma)$ and $B(\sigma)$ for each string σ of length s by executing finitely many steps of finitely strategies with input σ. We adopt the values for $A(\sigma)$ and $B(\sigma)$ that satisfy the constraints imposed by the strategies of highest priority. If σ and τ are distinct strings of length s, it is reasonable to think of the values of $A(\sigma)$ and $B(\sigma)$ as being determined in parallel during stage s.

Time Control Strategies. The time control strategies are used to ensure that A and B are no more complex than X.

Suppose that f is in T. The time control strategy C_f associated with f ensures that A and B are in $DTIME(f)$. C_f will have a simulation constant m_f, fixed throughout the construction. It takes m_f many steps in the universal RAM to simulate 1 step in the execution of the RAM which computes f. C_f limits the total number of steps taken in the execution of strategies of lower priority during stage s to $O(f(s))$.

C_f uses the following mechanism to impose this constraint. For each string σ, C_f divides the execution of the strategies of lower priority into blocks of size b many steps, where b is less than $|\sigma|$. Each time that the construction executes b many steps for the sake of lower priority, C_f requires the construction to run $m_f \cdot b$ many steps in the simulation of the evaluation of of f at $|\sigma|$. If the computation of f at $|\sigma|$ converges then C_f constrains the strategies of lower priority to decide the values of A and B at σ immediately. Thus, within a constant factor, the attention of the global construction is equally shared between C_f and all of the strategies of lower priority. Since, for all σ, C_f calls a halt to the computations of $A(\sigma)$ and $B(\sigma)$ when it sees $f(|\sigma|)$ converge, the function mapping $|\sigma|$ to the total number of steps in the construction devoted to evaluating strategies of priority less than or equal to C_f on input σ is $O(f(|\sigma|))$.

C_f will ensure that A and B are in $DTIME(f)$ provided that there are only finitely many strings for which the strategies of higher priority than C_f choose values for A and B which disagree with those chosen by C_f and

the strategies of lower priority.

The Complexity Strategies. The complexity strategies are used to ensure that A and B are at least as complex as X.

We describe the strategy R_M^A to ensure that if the RAM M computes A with run time function g, where g is in T, then X is in $DTIME(g)$. The strategy R_M^B is similar. Our approach is less direct than the one used in [MS2], where given $(t_i)_{i\in N}$ we could directly construct a set so that $(t_i)_{i\in N}$ was characteristic for that set. Here, we are additionally constrained by the fact that the only strings that we can put into A are those that already belong to X. Thus, if σ is not in X then we cannot diagonalize against a computation predicting that σ is not in A.

Let M and g be a RAM and a function as above. Given an input string σ, R_M^A acts as follows.

(1) First, R_M^A takes $|\sigma|$ many steps to look back and see whether an inequality between A and the set computed by M has already been established. If so then R_M^A halts its activity and does not impose any constraint on the values of $A(\sigma)$ or $B(\sigma)$. If not then R_M^A goes to step (2).

(2) R_M^A simulates the execution of M on input σ. If M halts then R_M^A goes to (3).

(3) If M returns value 1, then R_M^A constrains the construction from putting σ into A. If in fact $\sigma \in X$ then the constraint imposed by R_M^A implies that σ must be put into B. *(In this case, R_M^A has diagonalized A against M.)*

If M returns value 0, then R_M^A constrains the construction to put σ into A if σ is in X. *(In this case, either R_M^A has diagonalized A against M or we can infer that σ is not in X.)*

Suppose that f is a nonlinear element of T, c is a constant and M^X is a RAM that can be used to compute X in time $c \cdot f$. Suppose that we execute R_M^A within the constraint imposed by C_f. As noted earlier, there is a constant k_f such that the effect of C_f on R_M^A is to limit the execution of R_M^A on input σ to $k_f \cdot f(|\sigma|)$ many steps.

If A is not equal to the set computed by M, then there is a constant c_{finite} such that for all sufficiently long strings σ the evaluation of R_M^A takes only c_{finite} many steps after reading σ. Since f is not linear, $k_f \cdot f$ eventually dominates the number of steps it takes for R_M^A to look back and halt. Thus, C_f is essentially invisible to R_M^A in this case.

Now assume that M does compute A. In this case, if we reach (3) for a string σ it must be the case that M has output answer 0 and σ is not in X. Thus, the action of R_M^A is providing a subset of the complement of X that is computed in time less than or equal to $k_f \cdot f$. In fact, we show that X is in $DTIME(g)$. Suppose that σ is given. If $M(\sigma)$ converges in less than $k_f \cdot f(|\sigma|)$ many steps and σ is an element of X then we could keep σ out of A and diagonalize. Thus, if $g(|\sigma|)$ is less than $k_f \cdot f(|\sigma|)$ then $\sigma \notin X$. On the other hand, if $g(|\sigma|)$ is greater than $k_f \cdot f(|\sigma|)$ then we can evaluate $X(\sigma)$ in less than or equal to $c \cdot 1/k_f \cdot g(|\sigma|)$ many steps using M^X. Thus, there is a constant factor k such that the value of $X(\sigma)$ can be determined in $k \cdot g(|\sigma|)$ many steps, using whichever case occurs first.

Hence, if R_M^A is executed in the time control environment imposed by a function in the characteristic sequence for C then A will satisfy the associated complexity requirement.

Compatibility Between Complexity Strategies. We have already shown that R_M^A will satisfy its requirement if it is executed in a time control environment imposed by C_f and $X \in DTIME(f)$. In this context, we must show that the effect of R_M^A on the strategies of lower priority is essentially finite. If M does not agree with A then R_M^A's effect is explicitly finite, as established by the look back in step (1). Otherwise, there may be infinitely many strings σ such that R_M^A constrains the construction so that if $\sigma \in X$ then $\sigma \in A$. However, none of these strings can belong to X. Thus, any constraint imposed by R_M^A on σ is vacuous. Since any constraint imposed by a strategy $R_{M'}^A$ or $R_{M'}^B$ on σ will also be vacuous when $\sigma \notin X$, these constraints are compatible. Although the effect of R_M^A is not finite it can only contribute finitely many conflicts with the other strategies appearing in our construction.

Simultaneous Execution. The time complexity strategies impose a constraint on the way that we may distribute the computation steps in our construction. In particular, C_f requires that the total number of steps devoted to all the strategies of lower priority must be of the same order as the number of steps devoted to the analysis of f. We adapt a scheme from [MS2] by which we work within this constraint and still introduce infinitely many strategies of lower priority over the course of the construction. We give the reasoning behind the proof that this scheme works without reproducing all of the details found in [MS2].

Let M^* be a universal RAM. By the Cook-Reckhow theorem, for any

strategy Q there is a constant q such that M^* can simulate n many steps in the execution of Q using only $q \cdot n$ many of its own steps. Further, by using distinct parts of memory, given a finite sequence of strategies $Q_1, \ldots, Q_k$ and the constants $q_1, \ldots, q_k$ associated with simulating these strategies and given numbers $n_1, \ldots, n_k$, M^* can simulate n_i many steps in each Q_i using only $\sum_{i=1}^{k} q_i \cdot n_i$ many of its own steps.

Consider the following pattern for dividing time resources between the strategies acting on the same input string σ. Let $Q_1, \ldots, Q_k$ be strategies and let $q_1, \ldots, q_k$ be their simulation constants as above. Let a *sweep* denote one implementation of the following recursion, beginning with Step(k) with s_k equal to 1.

Step(i) Execute s_i many steps of Q_i at σ. *Note that this takes $q_i \cdot s_i$ many steps for M^*.*

(a) If i is greater than 1 then let s_{i-1} equal $q_i \cdot s_i + s_i$ and go to Step($i-1$). *s_{i-1} is equal to the total number of steps that it would take M^* to run the sweep at σ through the execution of the strategies $Q_k, \ldots, Q_i$.*

(b) Otherwise, end.

There is a fixed number of steps to each sweep, depending only on the sequence $\vec{Q}$. In particular, the number of steps to a sweep does not depend on the length of the string which is taken as input for the strategies. Further, for each i, the number of steps assigned to Q_i during a sweep is exactly as many steps as are needed to simulate the total activity of all strategies of lower priority, independently of the number $k - i$ of strategies of lower priority that occur in the considered sweep. The first property ensures that for each sequence of strategies $\vec{Q}$ there is an s such that one sweep through the sequence $\vec{Q}$ takes less than s steps to execute. The second property shows that we can combine strategies in a global construction and respect the constraints imposed by the time control strategies.

The Global Construction. We build A and B by combining the strategies $C_{\{t_i\}}$ for t_i in $(t_i)_{i \in \mathbf{N}}$, the characteristic sequence for C; the strategies $R^A_{M_i}$ for M_i a RAM, to ensure that if M_i computes A in time g_i then $X \in DTIME(g_i)$; and the symmetric strategies $R^B_{M_i}$. We give these strate-

gies the following priority ordering.

$$C_{\{t_1\}}, R^A_{M_1}, R^B_{M_1}, C_{\{t_2\}}, R^A_{M_2}, R^B_{M_2}, \ldots$$

We construct A and B by stages. During stage s we decide the values of A and B on all strings σ of length s. Letting σ be a string of length s, the construction operates on σ as follows.

We alternate between two activities: executing sweeps through the sequence of strategies and updating sequence of active strategies and the tentative values to A and B at σ.

We let the initial sequence of active strategies be the initial segment of the priority ordering $\vec{Q}_0$ of length ℓ. We determine ℓ by executing the first s steps of the following iteration.

(1) Begin with ℓ equal to 1 and $\vec{Q}$ be $\langle C_{\{t_1\}} \rangle$.
(2) Given ℓ and $\vec{Q}$ of length ℓ, execute one sweep through $\vec{Q}$ at argument σ. Go to (3).
(3) Add 1 to ℓ and let $\vec{Q}$ be the initial segment of the priority list of length ℓ. Go to (2).

Let ℓ be the largest value for which the iteration completed a sweep before the $|\sigma|$ steps were completed. Given that $\vec{Q}$ is always taken to be an initial segment of the priority list, the number of steps needed to complete a sweep is determined solely by the length of $\vec{Q}$ and not by the argument σ. Thus the length of $\vec{Q}_0$ depends only on $|\sigma|$ and is a non-decreasing function of $|\sigma|$ with infinite limit.

Having determined $\vec{Q}_0$, we execute the following iteration, beginning with $\vec{Q}$ equal to $\vec{Q}_0$, $A(\sigma)$ equal to $X(\sigma)$ and $B(\sigma)$ equal to 0.

Given the sequence $\vec{Q}$ of currently active strategies, we iterate the execution of sweeps (continuing our simulations of $\{t_i\}(|\sigma|)$ and of $M_i(\sigma)$) through $\vec{Q}$ until a time control strategy $C_{\{t_i\}}$ computes the value of $\{t_i\}(|\sigma|)$ and calls a halt for the strategies of lower priority. We form the updated sequence of active strategies by omitting $C_{\{t_i\}}$ and all strategies of lower priority and retaining the others. If during the iteration of the sweeps, a strategy $R^A_{M_i}$ or $R^B_{M_i}$ imposed a constraint on the value of $A(\sigma)$ or $B(\sigma)$ then we adopt the updated values imposed by the strategy of highest priority as our tentative values. If i is greater than 1 then we continue as above with the truncated sequence of active strategies. Otherwise, we end the

evaluation of the construction on σ when $C_{\{t_i\}}$ calls a halt, i.e. when our simulation of the evaluation of $\{t_i\}(|\sigma|)$ converges. We extend the definitions of A and B to σ by adopting the values current when $C_{\{t_i\}}$ calls a halt.

The time expended for the sake of determining A and B at σ is either spent in the initial setting up of $\vec{Q}_0$, in the constant number of steps spent in reading off the constraints imposed during the earlier iteration of sweeps and going to the next iteration or in executing a sweep. The first two involve only $O(|\sigma|)$ many steps in operational overhead.

By the observation that the length of a sweep does not depend on the stage when it is executed, we see that for every strategy S there is an s such that S is in $\vec{Q}_0$ as computed by every string of length greater than or equal to s. We have already argued that the complexity strategies only act in a way that is incompatible with the actions of lower priority strategies finitely often. This finite injury does not effect the ability of the lower priority strategies to satisfy their requirements. By operating with sweeps and an additional overhead of size $O(|\sigma|)$ and halting the actions of strategies of lower priority upon request, we have ensured that except for the finitely many exceptional strings mentioned above the construction respects every $C_{\{t_i\}}$. Thus, for all i A and B are in $DTIME(\{t_i\})$. Similarly, for each complexity strategy $R^A_{M_i}$ or $R^B_{M_i}$ and each argument σ of sufficient length, that strategy receives $O(\{t_i\}(|\sigma|))$ many steps during the series of sweeps at argument σ. Further, except for finitely many strings, all of its constraints are respected by the construction, either because it stops issuing constraints due to having found an inequality or because all of its constraints are vacuous. Thus, all of the complexity requirements are also satisfied. □

We can draw some corollaries from the splitting theorem and its proof.

Theorem. *For every non-trivial complexity type $\mathcal{C}$ the partial order $PO(\mathcal{C})$ of sets in $\mathcal{C}$ ordered by inclusion mod finite ($\subseteq^*$) has neither maximal nor minimal elements.*

Let $PO_{0,1}(\mathcal{C})$ be the partial order

$$\langle \{X \mid X \in \mathcal{C} \vee X = \{0,1\}^* \vee X = \emptyset\}, \subseteq^* \rangle .$$

$PO_{0,1}(\mathcal{C})$ is the result of adjoining a greatest and a least element to $PO(\mathcal{C})$.

Corollary. *For every complexity type $\mathcal{C}$ there is an embedding E from the countable atomless Boolean algebra (CBA), regarded as a partial order,*

into $PO_{0,1}(\mathcal{C})$. Further, this embedding associates the Boolean operations in CBA to the usual set theoretic ones.

Proof. This corollary does not follow directly from the splitting theorem but rather from a modestly stronger version, which has almost the same proof. We first describe how to obtain the embedding E. Then, we read off the needed strong splitting theorem. Finally, we indicate how to alter the proof of the original splitting theorem to prove the stronger form.

There is a standard scheme to embed CBA into a partial order that satisfies the splitting theorem. We start with b_0 an intermediate element of CBA and X_0 an element of $\mathcal{C}$. We set $E(b_0)$ equal to X_0. Then we extend E to the Boolean closure of $\{b_0\}$ by mapping $\neg b_0$ to the complement of X_0, 0 to $\emptyset$ and 1 to $\{0,1\}^*$.

During the recursion step of our construction, we start with an isomorphism E between two finite Boolean algebras $\mathcal{B}$ contained in CBA and $E(\mathcal{B})$ contained in $PO_{0,1}(\mathcal{C})$. Given a new element b of CBA, let $\mathcal{B}_b$ be the finite subalgebra of CBA generated by $\mathcal{B}$ and $\{b\}$. Let $\{a_1,\ldots,a_k\}$ be the atoms of $\mathcal{B}$. The non-zero elements in $\mathcal{B}_b$ of the form $b \wedge a_i$ or $\neg b \wedge a_i$ generate $\mathcal{B}_b$ under join. Thus, it is enough to extend E to these elements and let the union operation determine E's extension to all of $\mathcal{B}_b$. Naively, for each i such that $b \wedge a_i \neq a_i$ and $b \wedge a_i \neq 0$, we could find images for $b \wedge a_i$ and $\neg b \wedge a_i$ by splitting $E(a_i)$ in $PO_{0,1}(\mathcal{C})$ into two sets $A_{i,0}$ and $A_{i,1}$. However, to generate E by taking unions we need to ensure that these sets have a stronger splitting property: that all finite unions of the $A_{i,j}$ belong to $\mathcal{C}$.

This conclusion will follow once we know the following. Every set X in $\mathcal{C}$ can be split into two sets X_0 and X_1 such that for every f in T, if there is a set U in $DTIME(f)$ with either $X \cap U = X_0$ or $X \cap U = X_1$ then X is in $DTIME(f)$. Symbolically, X_0 and X_1 satisfy the condition

$$(*) \quad (\forall f \in T)\left[\begin{array}{c}(\exists U \in DTIME(f))\,(X \cap U = X_0 \vee X \cap U = X_1) \\ \Longrightarrow\ X \in DTIME(f)\end{array}\right].$$

Now suppose that for each i such that $b \wedge a_i \neq a_i$ and $b \wedge a_i \neq 0$, we produce sets $A_{i,0}$ and $A_{i,1}$ to split $E(a_i)$ and satisfy $(*)$ for $X = E(a_i)$, $X_0 = A_{i,0}$ and $X_1 = A_{i,1}$. If U is a finite union of the $A_{i,j}$ and U is not a element of $E(\mathcal{B})$ then there must be an i^* and a j^* such that $U \cap E(a_i) = A_{i^*,j^*}$. Applying $(*)$, for all $f \in T$ if $U \in DTIME(f)$ then $E(a_i) \in DTIME(f)$.

Thus, $E(a_i) \leq_C U$. Trivially, U is no more complex than $E(a_i)$ since U is a finite union of sets from $\mathcal{C}$. Consequently, U is in $\mathcal{C}$.

Thus, we have reduced the proof of the corollary to proving the strong splitting theorem. This form of the splitting theorem is proven using a slightly different version of the complexity strategies $R_M^{X_0}$ and $R_M^{X_1}$. Instead of simulating the output of M and attempting to diagonalize, we simulate the computation of U and attempt to differentiate $X \cap U$ from X_0 and X_1. Consider the case for X_0. If we are unable to complete the simulation evaluating $U(\sigma)$ then U is at least as complex as X at σ. If we see $\sigma \in U$, we issue the constraint that σ is not in X_0 and diagonalize between X_0 and U. (This is analogous to step (3) in R_M^A in step 3-case 1, when we ensured that M did not compute A.) When we see $\sigma \notin U$ we issue the constraint that $\sigma \in X_0$ if $\sigma \in X$ and ensure that if $X_0(\sigma) = U(\sigma)$ then U is at least as complicated as X at σ. (This is analogous to step 3-case 2 in R_M^A, when we ensured that either M did not compute A or we could compute X in order of the running time of M many steps.) If X_0 is produced by a construction that implements this strategy within a time constraint strategy C_f and U is equal to X_0 then, there is are constants c_1 and c_2 such that for every σ either the simulation of $U(\sigma)$ takes longer than $c_1 \cdot f(|\sigma|)$ many steps or we can compute that σ is not an element of X in c_2 times the running time to evaluate $U(\sigma)$. But then U is not a counter example to the strong splitting of X by X_0 and X_1.

These strong splitting strategies can be combined with the time control strategies as in the proof of the splitting theorem. □

Suppose that X and Y are subsets of $\{0,1\}^*$. Let $Y \underset{\infty}{\subset} X$ denote the condition that Y is a subset of X and that $X - Y$ is infinite.

Definition. We say that X is *at least as complex as* Y if for all f in T, $X \in DTIME(f)$ implies that $Y \in DTIME(f)$. In this case, we write $Y \leq_C X$.

Theorem. (Density Theorem) *Assume that* $Y \underset{\infty}{\subset} X$ *and* $Y \leq_C X$. *Then there is a set* A *such that* $A =_C X$ *and* $Y \underset{\infty}{\subset} A \underset{\infty}{\subset} X$.

Proof. Most of the ingredients in the proof of the density theorem are the same as in the splitting theorem, so we abbreviate our discussion.

For the sake of $Y \underset{\infty}{\subset} A \underset{\infty}{\subset} X$, we must put very element of Y into A and restrict the elements of A to come from X. Thus we cannot use strategies which issue constraints of the form $\sigma \notin A$. For these, we must substitute

$\sigma \notin Y \implies \sigma \notin A$. We analyze the resulting complexity strategy R_M^A in its outcome when M computes A. Given a string σ, there are three cases: we infer that $\sigma \in Y$ (and hence in X) from $M(\sigma) = 1$; we infer $\sigma \notin X$ from $M(\sigma) = 0$; or we compute X in less time than it takes to evaluate $M(\sigma)$ using the RAM associated with the time control strategy that caused R_M^A to halt. Consequently, we can design an algorithm to compute X that converges at least as fast as the one associated with the time control strategies of higher priority. □

Corollary. *For every complexity class C, $PO_{0,1}(C)$ is dense.*

It is easy to see that $PO_{0,1}(C)$ is isomorphic to the countable atomless Boolean algebra if C is equal to **0**. Furthermore, for every complexity type, CBA can be embedded in $PO_{0,1}(C)$. However, the following corollary suggests that the structure of $PO(C)$ is substantially more complicated than CBA when C is non-trivial.

Obviously, every complexity type is closed under complementation and so not closed under union or intersection. However, it could still be the case that any two sets A and B with an upper bound in $PO(C)$ have a least upper bound in $PO(C)$. This is ruled out by the following result.

Corollary. *Suppose that C is not equal to **0** and A and B belong to C. Then, A and B have a least upper bound in $PO(C)$ if and only if $A \cup B \in C$.*

Proof. Assume that $A \cup B \notin C$ and that D is an upper bound both A and B in $PO(C)$. Then $A \cup B \subset^{\infty} D$ and $A \cup B \leq_C D$. By the density theorem there is a set D^* such that $A \cup B \subset^{\infty} D^* \subset^{\infty} D$ and $D^* =_C D$. Thus, D is not a least upper bound for $A \cup B$. □

A Question. We are left with an intriguing situation. Whether the union of two elements A and B from C is an element of C is a first order property of A and B in $PO(C)$. Thus, not all pairs from C are alike in $PO(C)$. Can this inhomogeneity in $PO(C)$ be used to show that the structure is complicated. In particular, is the first order theory of $PO(C)$ non-recursive?

ACKNOWLEDGMENT

We would like to thank Joel Berman for his acute and helpful comments.

REFERENCES

[AHU] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, 1974.

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

[MY] M. Machtey and 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.

[Mar] D. A. Martin, *Classes of recursively enumerable sets and degrees of unsolvability*, Z. Math. Logik Grundlag. Math. **12** (1966).

[MS1] W. Maass and T. A. Slaman, *Extensional properties of sets of time bounded complexity (extended abstract)*, Proc. of the 7th Int. Conference on Fundamentals of Computation Theory, Lecture Notes in Computer Science, vol. 380, Springer, Berlin, 1989, pp. 318-326.

[MS2] ———, *The complexity types of computable sets (extended abstract)*, Proc. of the Structure in Complexity Theory Conference, 1989.

[P] W.J. Paul, *Komplexitaetstheorie*, Teubner, Stuttgart, 1978.

DEPARTMENT OF MATHEMATICS, STATISTICS AND COMPUTER SCIENCE; UNIVERSITY OF ILLINOIS AT CHICAGO; CHICAGO, IL 60680

DEPARTMENT OF MATHEMATICS; THE UNIVERSITY OF CHICAGO; CHICAGO, IL 60637