On the Problem of Finding
Natural Computational
Complexity Measures*

J. Hartmanis

TR 73-175

June 1973

Department of Computer Science
Cornell University
Ithaca, New York 14850

* This research has been supported in part by National Science Foundation Grant GJ-33171X.
On the Problem of Finding Natural Computational Complexity Measures

J. Hartmanis

Abstract:

To develop an abstract theory which deals with the quantitative aspects of computing we need a deeper understanding of how to define "natural" computational complexity measures axiomatically. To this end, this paper summarizes the principal properties which hold for some natural complexity measures but not for all measures and which have been proposed as desirable properties of natural measures. The paper discusses the nature of these properties, studies their interrelations and their possible values towards defining natural computational complexity measures. A number of open problems are discussed and directions for further research are suggested.
On the Problem of Finding Natural Computational Complexity Measures

J. Hartmanis

1. INTRODUCTION

A successful theory of computing must deal realistically with the quantitative aspects of computing and it must develop a general theory which deals with the measuring of computational complexity and with the properties of these measures. Such a theory, clearly, has to be a rigorous mathematical theory but it furthermore has to reflect the main aspects of real computing to justify its existence by contributing to a deeper understanding of computing and the general development of computer science.

During the last ten years considerable progress has been made in the study of the quantitative aspects of computing and by now we can justly speak of a theory of computational complexity [1,6,10,11]. As a matter of fact, we believe that the study of quantitative aspects of computing is a central theme of theoretical computer science and that eventually computational complexity will emerge as a principal part of the theory of computing. We also believe that so far the most surprising and possibly deepest results in theoretical computer science have been obtained in computational complexity and that work in this area, including the analyses of algorithms, is quite likely to change our paradigm of computing quite fundamentally. It has also turned out that some of the hardest (unsolved) problems have been encountered in this area of research.
The current work in computation complexity can be divided into two major research areas:

1. The study of computational complexity measures, their properties, the relations between different complexity measures, and the properties of complexity classes defined by these measures. An overview of this part of the theory of computational complexity can be found in [6,11].

2. The study of the computational complexity of specific problems or specific subclasses of problems in a given complexity measure. This area of research is often referred to as analysis of algorithms and its main concern is with developing good algorithms for specific problems and with establishing tight upper and lower computational complexity bounds for these problems. For typical results in this area see [10].

In this paper we are primarily concerned with the first problem area in which we can discern two major currents of research:

a) A thorough investigation of specific computational complexity measures and the properties of complexity classes as defined by these measures. For example, measures defined by the number of steps taken or the number of tape squares used by a Turing machine have received considerable attention and have yielded some interesting results and problems [6,7,10,11].

b) The study of properties common to all computational complexity measures satisfying the two Blum complexity axioms [1,6,11].

Considerable success has been achieved in both research areas and, clearly, they are interconnected. On the other hand, there is a strong feeling that the axiomatic approach admits too many com-
-plexity measures, many of which appear pathological and violate our intuitive ideas about how natural complexity measures should behave [6,8,9]. At the same time, there is a feeling that in the study of specific measures the inessential peculiarities of the measures are obscuring the deeper properties of the computational process. Thus, we believe that our task is to define abstractly what is meant by a "natural" computational complexity measure or to abstract the major properties of specific measures and define classes of natural measures reflecting different aspects of practical computing.

We are convinced that in this area, as in many other research areas before, the deeper understanding and clarity will come from an abstract approach. Thus we see the abstract definition and study of natural complexity measures as an urgent and important task.

In this paper we summarize the main properties which hold for some specific computational complexity measures but which do not hold for all measures. Some of these properties have been proposed as properties which should hold for any natural measure. We will discuss these properties, study their interrelations and investigate their value towards defining natural measures. We will also prove some new results which seem to shed further light on how to proceed towards the definition of natural measures. Furthermore, we will state several open problems whose solutions would improve our understanding of specific computational complexity measures and indicate some hopeful research directions.
2. **ENUMERABILITY CONSIDERATIONS**

In this and the following sections we list a number of properties which hold for some specific natural complexity measures but do not hold for all measures. We study the relations between these properties and discuss their desirability. We also derive some results which suggest what additional properties should be thought and how the scope of complexity theory should be enlarged to reflect additional aspects of real computing.

First, we establish some notation and define the main concepts used in this paper.

Let

$$\phi_1, \phi_2, \phi_3, \ldots$$

be an admissible enumeration of the partial recursive functions (of one variable) and let

$$\phi_1', \phi_2', \phi_3', \ldots$$

be an enumeration of a subset of the partial recursive functions (the step counting functions).

We say that $\phi = \{(\phi_i', \phi_i)\}$ is a computational complexity measure with step counting functions $\phi_i'$, iff

1. $(\forall x, i) [\phi_i'(x) \text{ converges } \Leftrightarrow \phi_i(x) \text{ converges}]$
2. $(\exists M(\ , \ , \ \text{recursive}) (\forall i,m,x) [M(i,m,x) = 1 \text{ if } \phi_i'(x) = m \text{ else } 0].$

Thus a computational complexity measure assigns to every algorithm $\phi_i$ (in our enumeration of algorithms for the partial recursive functions) a step counting function $\phi_i'$, which
a) converges at the value $x$ iff the algorithm converges, and
b) for which we can test recursively whether its value at $x$ is $m$.

Almost any natural measure we choose to measure the complexity of computations satisfies the two axioms (or can easily be modified to do so) and there is strong agreement that these two axioms must be satisfied by any computational complexity measure \([1,6,8,9]\). It is also very impressive to see how rich a theory has already been developed using only these two axioms for complexity measures \([1,6,8,9,11]\). Thus we are justified to refer to any set of step counting functions satisfying the two axioms as a computational complexity measure.

On the other hand, we will show that these axioms permit some very strange computational complexity measures which seem to be pathological and which do not satisfy our intuitive notions about natural computational complexity measures.

One could compare the situation here with the definition of a pointset topology by means of a family of open sets closed under finite intersections and arbitrary unions. These topologies are clearly too general and their study becomes more interesting and relevant if we learn how to define compactness, separation properties, etc. We believe that in the abstract theory of quantitative computing we are still missing these additional concepts which make this theory directly relevant and applicable to computer science.

Before proceeding with the discussion of additional properties
for complexity measures we introduce some notation.

For any computational complexity measure $\phi$ and recursive function $t$ the set of functions

$$C_t = \{ f \mid f \text{ recursive and } (\exists i) \left[ \phi_i = f \text{ and } \phi_i(x) \leq t(x) \text{ a.e.} \right] \}$$

is a complexity class of the measure $\phi$.

Thus the complexity class $C_t$ (or $C_t^\phi$ to emphasize the measure used) consists of all recursive functions whose complexity is bounded by $t$ almost everywhere.

We say that a set of functions $C$ is recursively enumerable (or has a r.e. presentation) iff there exists a recursively enumerable set $A$ such that

$$C = \{ \phi_i \mid i \in A \}.$$

It can easily be shown that the complexity classes of time and tape bounded Turing machine computations are all recursively enumerable [6,7] and we have not yet found a specific natural measure with non-enumerable complexity classes. As a matter of fact, in any measure all the complexity classes are recursively enumerable for sufficiently large complexity bounds, as stated below.

**Lemma:** For every complexity measure $\phi$ there exists a recursive function $t_\circ$ such that $C_t^\phi$ is recursively enumerable, provided $t(x) \geq t_\circ(x)$ a.e.

Actually more can be said [9].

**Lemma:** For any two computational complexity measures $\phi$ and $\hat{\phi}$ there exists a recursive function $t_\circ$ such that for any recursive $t_\bot$ and
$t_2 \geq t_0$, the (index sets of the) complexity classes $C_{t_1}^\phi$ and $C_{t_2}^\phi$ are recursively isomorphic.

From this result we see that for sufficiently large resource bounds the complexity classes of any measure are indistinguishable by recursively invariant properties.

It is also important to note that from a practical point of view we are not concerned in computing with the highly complex computations. We are primarily interested in the functions contained in the lower complexity classes. Thus it came as quite a shock when Landweber and Robertson [8] and F. D. Lewis [9], independently, discovered that there exist computational complexity measures with non-enumerable complexity classes. We construct one such computational complexity measure to illustrate how easily pathological measures can be obtained.

**THEOREM:** There exist complexity measures $\phi$ and recursive $t$ such that $C_t^\phi$ is not recursively enumerable.

**Proof:** Let $\phi_{i_1}, \phi_{i_2}, \ldots$ be a recursive enumeration of the constant functions such that $\phi_{i_j}(n) = j$. Define the measure $\phi$ as follows: for all $k \neq i_j, j = 1, 2, \ldots$, let $\phi_k(n) \geq n$, and let

$$\phi_{i_j}(n) = \begin{cases} 0 & \text{if } M_j(j) \text{ does not halt in } n \text{ steps}, \\ n & \text{otherwise}. \end{cases}$$

Thus $C_0$ consists of all those constant functions $\phi_{i_j}(n) = j$ for which the $j^{th}$ Turing machine $M_j$ does not halt on input $j$. Therefore, if $\phi_{k_1}, \phi_{k_2}, \phi_{k_3}, \ldots$ is an enumeration of $C_0$, then $\phi_{k_1}(1), \phi_{k_2}(1), \phi_{k_3}(1), \ldots$ is an enumeration of $\{j | M_j(j) \text{ does not halt}\}$.
This is a contradiction, since this set is known (and can easily be shown) not to be recursively enumerable.

We described the non-enumerability of complexity classes first because historically it played an important role by showing that not all complexity measures have "nice mathematical" properties. Many researchers accepted the fact that additional properties would have to be postulated for complexity measures to reflect more realistically real computing situations but the discovery of non-enumerable complexity classes showed that even from a purely esthetic mathematical point of view additional properties should be demanded.

Since recursive enumerability still may not be sufficient to insure that all complexity classes of a measure have "nice" properties, F. D. Lewis [9] has proposed a somewhat stronger property.

CONFORMITY: A measure $\phi$ has the **conformity property** (or **conforms**) iff the index sets for any two (non-trivial) complexity classes are recursively isomorphic.

At the present time we do not know what simple additional axioms will guarantee that a complexity measure will conform or have recursively enumerable complexity classes. At the same time, an inspection of the proof of the previous theorem shows that the construction of the pathological measure relied heavily on the fact that in this measure a finite change in $\phi_i$ could cause an unbounded change in the corresponding step counting function. In most natural measures this is not the case and thus we are led to
another desirable property.

FINITE INVARIANCE: A measure \( \phi \) is said to be \textit{finitely invariant} iff for all recursive functions \( \phi_i \) and \( \phi_j \),

\[
\phi_i \equiv \phi_j \text{ a.e.}
\]

implies that \( \phi_i \) and \( \phi_j \) are in the same complexity classes, i.e. 
\[(\forall \text{ recursive } t^\ast) [\phi_i \in C_t \iff \phi_j \in C_t].\]

One can easily prove the next result [9].

Corollary: Let \( \phi \) be a finitely invariant measure. Then the measure \( \phi \) conforms and all complexity classes of \( \phi \) are recursively enumerable.

We do not believe that conformity or finite invariance should be used directly as additional "axioms", since they are not in the spirit of the two fundamental complexity axioms [8]. On the other hand, we believe that any natural measure should have these properties. Thus we are led to one of our problems.

PROBLEM: Find simple additional axioms which guarantee that measures satisfying these axioms have recursively enumerable complexity classes (or that they conform). See also comments on this problem in the last section.

It is not clear what form these axioms should have and our next result shows that any axioms which will imply conformity of a measure \( \phi \) must be such that they do not hold for all sub-measures of \( \phi \).
Given a complexity measure $\phi = \{(\phi_i, \phi_i)\}$. Then $\hat{\phi} = \{(\phi_i^j, \phi_i^j)\}$ is a submeasure of $\phi$ iff $\phi_i^j$, $j = 1, 2, \ldots$, is an admissible enumeration of the partial recursive functions.

**Theorem:** Every complexity measure $\phi$ contains a submeasure $\hat{\phi}$ which has non-enumerable complexity classes.

**Proof:** The basic idea of the proof is that there exists a recursive function $t$, a sub-enumeration $\phi_i^j$ and an enumeration of the constant functions $\phi_i^j_k$, $\phi_i^j_k(n) = k$ with the property that $C_t$ contains only constant functions and $\phi_i^j_k$ is in $C_t$ iff $M_k(k)$ does not converge. Thus, as in the previous non-enumerability proof, the assumption that $C_t$ is recursively enumerable leads to a contradiction.

3. **Other Provincial Properties of Measures**

Next we turn to two other properties of complexity measures which have some interesting implications about "diagonalization" properties.

The first property holds for some specific natural measures we have considered and we believe that it should hold for all natural measures.

**Properness:** A measure $\phi$ is said to be proper iff for all $i$ $\phi_i$ is in $C_{\phi_i}$.

Thus in proper measures the step counting functions are computable in their own bounds. Intuitively, this seems a desirable
property since to determine how long an algorithm runs should not be more complicated than running the algorithm.

The next property has been proposed by Landweber and Robertson [8].

PARALLELISM: A measure \( \phi \) has the parallel computation property iff there exists a recursive function \( h(\ ,\ ) \) such that for all \( i \) and \( j \)

\[
\phi_h(i,j)(x) = \phi_i(x) \text{ if } \phi_i(x) \leq \phi_j(x) \text{ else } \phi_j(x)
\]

and

\[
\phi_h(i,j)(x) = \min \{ \phi_i(x), \phi_j(x) \}
\]

This property holds for tape bounded Turing machine computations as well as for time bounded computations of multi-tape Turing machines. We conjecture that it does not hold for time bounded computations on one-tape Turing machines. Thus we do not believe that all natural measures must have this property.

PROBLEM: Let \( \phi_1, \phi_2, \ldots \) be the enumeration of all one-tape Turing machines and let \( \phi_1, \phi_2, \ldots \) be the corresponding step counting functions. Does this measure have the parallel computation property?

Next we recall a result from complexity theory [1,6] to motivate another definition.

Lemma: For every computational complexity measure \( \phi \) there exists a recursive function \( h(\ ,\ ) \) such that for all \( i \)
\[ C_{\phi_i}(x) \not\subseteq C_\text{h}[x, \phi_i(x)] \]

This result can be proven by a straightforward diagonalization argument and we refer to this, following Constable [3], as upward diagonalization in contrast to the concept defined below.

**Downward Diagonalization**: A complexity measure \( \phi \) has the downward diagonalization property iff there exists a monotonically increasing, unbounded recursive function \( h() \), such that for all \( i \) and \( t(x) \leq x \)

\[ h \circ t(x) \leq \phi_i(x) \text{ a.e.} \]

implies that

\[ C_t \not\subseteq C_{\phi_i}. \]

This property asserts that for every step counting function \( \phi_i \) we can find a computation which runs in this many steps and whose running time can not be improved by more than the function \( h \). Upward and downward diagonalization are discussed by Constable [3], where it is also shown that not all complexity measures possess the downward diagonalization property.

**Theorem**: Let \( \phi \) be any measure which is proper and has the parallel computation property. Then \( \phi \) has the downward diagonalization property.

**Proof**: The proof follows the general method of diagonalization used in [6,7].

We conjecture that the measure, defined by the number of steps
taken by one-tape Turing machines, does not have the parallel computation property but does have the downward diagonalization property. We conjecture the same for time measures defined by RASP machines [5].

It is also interesting to note [8] that there exist measures \( \phi \) which are proper and have the parallel computation property but still have non-enumerable complexity classes.

In the study of specific complexity measures it becomes clear that the tape-bounded Turing machine computations have some very nice properties. Most of these properties can be traced back to the unbounded reusability of tape without increasing the step count. This is in a sense a very strong form of parallelism and we do not expect that all natural measures should have this property. At the same time, we believe that it is a property of an interesting subclass of natural measures and it would be very interesting to define it abstractly. It would be particularly interesting if this would lead to an abstract differentiation between tape and time based complexity measures.

We know that the next property holds for tape measures.

**DENSENESS:** A complexity measure \( \phi \) is dense iff for all recursive \( t_1 \) and \( t_2 \) such that

\[
C_{t_1} \neq C_{t_2}
\]

there exists a recursive \( t \) for which

\[
C_{t_1} \neq C_{t} \neq C_{t_2}.
\]

**THEOREM:** The complexity measure defined by the amount of tape used by a Turing machine is dense and there exist other measures
which are not dense.

Proof: See [2].

PROBLEM: Is the measure defined by the number of steps taken by multi-tape Turing machines a dense measure? How about one-tape Turing machines and RASP? We conjecture that the answer is positive in all three cases.

It would be very interesting to find some other sets of properties of a measure which imply denseness, are implied by denseness and are equivalent to denseness.

4. FLOW CHART MEASURES

There is a strong feeling that we have a good understanding of flow chart computations and that the (number of steps or) number of assignments executed in flow chart computations is a natural complexity measure. The next result (due to K. Weihrauch in a somewhat different formulation) shows that any computational complexity measure can be realized as a submeasure of the flow chart measure. We give this result to further illustrate that natural measures have undesirable submeasures.

THEOREM: For every complexity measure $\phi = \{ (\phi_i, \phi_j) \}$, with $\phi_i(x) > 1$ for all $i$ and $x$, there exists a finite, recursive interpretation for flow charts such that a recursive subset of these flow charts realizes the measure $\phi$ (for the flow charts we are counting the number of assignment statements executed).

Proof: Let $\langle i, j \rangle$ be a fixed pairing function and consider the
following interpretation for flow charts.

TEST:

\[ T_0(x,y) = \text{TRUE} \iff y = \langle i,n \rangle \text{ and } \phi_i(x) = n + 1 \]

ASSIGNMENTS:

\[ y \rightarrow F_0(x,y), \text{ with } F_0[x, \langle i,0 \rangle] = \phi_i(x) \text{ if } \phi_i(x) = 1 \text{ else } 0; \]

\[ y \rightarrow F_1(x,y), \text{ with } F_1[x, \langle i,n \rangle] = \langle 0, \phi_i(x) \rangle \text{ if } \phi_i(x) = n + 1 \text{ else } \langle i, n+1 \rangle; \]

\[ y \rightarrow F_2(y), \text{ with } F_2[\langle j, \phi_i(x) \rangle] = \phi_i(x) \text{ if } j = 0 \text{ else } 0. \]

These tests and assignment functions are recursive and it is seen that the flow chart in Fig. 1, with two variables \( x \) and \( y \), when started with

\[ x \text{ and } y = \langle i,0 \rangle, \]

computes \( \phi_i(x) \) by executing exactly \( \phi_i(x) \) assignment statements. Thus we can pick a recursive subset of flow charts with the above interpretation and the number of assignments executed is identical to the corresponding \( \phi_i(x) \). Thus a submeasure of these flowcharts realizes the measure \( \phi \), as was to be shown.

Finally, we note that it would be very interesting to characterize abstractly the complexity measures generated by all flow charts for recursive (universal) interpretations. This is a class of complexity measures with many natural properties and this class should be understood better.
Figure 1.
5. **EXTENDED SCOPE OF COMPLEXITY THEORY**

Our previous result about flow chart measures shows that natural measures can contain all other measures as submeasures. Similarly, our results about non-enumerable complexity classes showed that natural measures have submeasures with non-enumerable complexity classes. This leads us to conclude that any axioms which will define a natural computational complexity measure must be such that they cannot hold for all submeasures of the measures they define. Finite invariance is one such property; unfortunately, as pointed out by Landweber and Robertson [8], finite invariance does not seem to be in the spirit of the other Blum axioms. This is not surprising, since any axiom which cannot hold for all submeasures must have a quantifier structure powerful enough to prevent it from holding for submeasures and this must make the axioms look more complicated than the original Blum axioms.

Actually we believe that we should not try to just pile on more and more axioms on complexity measures to insure that all algorithms with certain step counting properties are included. We are convinced that we should enlarge the scope of complexity theory and deal explicitly with the "programming language" and the fundamental steps of the computations. It seems much more natural to start with some primitives, prescribe how they can be combined into algorithms and how the steps should be counted in their execution. In this way, we automatically include all desired algorithms and no postulates have to be added to prevent us from selecting an unnatural submeasure.

So far in abstract complexity theory we have studied resource
bounds and complexity classes. Next we should deal explicitly with step counting and should introduce this as a primitive concept in the theory.

We believe also that an explicit concern with the structure of algorithms and their relation to complexity measures is likely to lead to a theory relevant to computing because it would model real computing problems more realistically. Some attempts in this direction have already been made [4].

At the same time, we also believe that different types of complexity measures can and should be distinguished by additional axioms and that such axioms should be studied and that particularly their interrelations should be investigated in depth.
6. REFERENCES


