Constructive Complexity [1]

Karl Abrahamson[1] Michael R. Fellows[2]
Michael A. Langston[3] Bernard M.E. Moret[4]

[1]Department of Computer Science, Washington State University, Pullman, WA 99164
[2]Department of Computer Science, University of Victoria, Victoria V8W 3P6, B.C. (Canada)
[3]Department of Computer Science, University of Tennessee, Knoxville, TN 37996
[4]Department of Computer Science, University of New Mexico, Albuquerque, NM 87131

Abstract

Powerful and widely applicable, yet inherently non-constructive, tools have recently become available for classifying decision problems as solvable in polynomial time, as a result of the work of Robertson and Seymour. These developments challenge the established view that equates tractability with polynomial-time solvability, since the existence of an inaccessible algorithm is of very little help in solving a problem. In this paper, we attempt to provide the foundations for a constructive theory of complexity, in which membership of a problem in some complexity class indeed implies that we can find out how to solve that problem within the stated bounds. Our approach is based on relations, rather than on sets; we make much use of self-reducibility and oracle machines, both conventional and "blind," to derive a series of results which establish a structure similar to that of classical complexity theory, but in which we are in fact able to prove results which remain conjectural within the classical theory.

1 Introduction

Powerful and widely applicable, yet inherently non-constructive, tools have recently become available for classifying decision problems as solvable in polynomial time, as a result of the work of Robertson and Seymour [RS1, RS2] (see also [Jo]). When applicable, the combinatorial finite-basis theorems at the core of these developments are non-constructive on two distinct levels. First, a polynomial-time algorithm is only shown *to exist*: no effective procedure for finding the algorithm is established. Secondly, even if known, the algorithm only *decides*: it uncovers nothing like "natural evidence." Examples of the latter had been rare; one of the few such is primality testing, where the existence of a factor can be established without, however, yielding a method for finding said factor. Moreover, there had been a tendency to assume that there is no significant distinction between decision and construction for sequential polynomial time [KUW]. However, there is now a flood of

[1]This research is supported in part by the Washington State Technology Center, by the National Science Foundation under grant MIP–8603879, by the Office of Naval Research under contracts N00014–88–K–0343 and N00014–88–K–0456, and by the NSERC of Canada.

polynomial-time algorithms based on well-partial-order theory that produce no natural evidence [FL1, FL2, NT]. For example, determining whether a graph has a knotless embedding in 3-space (i.e., one in which no cycle traces out a nontrivial knot) is decidable in cubic time [FL2], although no *recursive* method is known for producing such an embedding even when one is known to exist. What is worse (from a computer scientist's point of view), the non-constructive character of these theorems is inherent, as they are independent of standard theories of arithmetic [FRS]; thus attempts at "constructivizing" these results [FL4] may yield practical algorithms in some cases, but cannot succeed over the entire range of application.

These developments cast a shadow on the traditional view that equates the tractability of a problem with the existence of a polynomial-time algorithm to solve the problem. This view was satisfactory as long as existence of such algorithms was demonstrated constructively; however, the use of the existential quantifier is finally "catching up" with the algorithm community. The second level of non-constructiveness—the lack of natural evidence—has been troubling theoreticians for some time: how does one trust an algorithm that only provides one-bit answers? Even when the answer is uniformly "yes", natural evidence may be hard to come by. (For instance, we know that all planar graphs are 4-colorable and can check planarity in linear time, but are as yet unable to find a 4-coloring in linear time.) Robertson and Seymour's results further emphasize the distinction between deciding a problem and obtaining natural evidence for it.

All of this encourages us to consider ways in which some of the foundations of complexity theory might be alternatively formulated from a *constructive* point of view. (We intend the term "constructive" to convey an informal idea of our goals; this is not to be confused with a *constructivist* approach [Be], which would certainly answer our requirements, but may well prove too constrictive. However, our intent is certainly constructivist, in that we intend to substitute for simple existence certain acceptable styles of proof based on "natural" evidence.)

We base our development on the relationships between the three aspects of a decision problem: evidence checking, decision, and searching (or evidence construction). The heart of our constructive formulation is to provide for each problem to be equipped with its own set of *allowable* proofs, in much the same manner as the usual definition of $\mathcal{NP}$-membership [GJ]; moreover, in deterministic classes, the proof itself should be constructible. This leads us to a formalization based on evidence generators and evidence checkers—exactly as in classical complexity theory, but where the checker is given as part of the problem specification rather than discovered as part of the solution. In other words, we come to the "Algorithm Shop" prepared to accept only certain types of evidence.

The result of this approach is a formulation based on *relations*, rather than on sets as in the classical formulation. This formulation provides a natural perspective on self-reducibility and oracle complexity [Ba, Ko, Sn, So, Va], concepts which have recently received renewed scrutiny. We define oracle mechanisms through which we can ask interesting questions about the value of proofs and the nature of non-constructiveness; we manage to answer some of these questions, including one which remains a conjecture in its classical setting.

2 Problems and Complexity Classes

In classical complexity theory, a decision problem is a language (or set) over some alphabet, $L \subseteq \Sigma^*$; the main question concerning a language is the *decision* problem: given some string x, is it an element of the language L? However, since a simple "yes" or "no" answer clearly lacks credibility, one may require that the algorithm also produce a proof for its answer, within some prespecified acceptability criteria. Such a proof—or sketch of one—is termed *evidence* and the problem of deciding membership as well as producing evidence is called a *search* (or certificate construction) problem. Finally, in order for such a proof to be of use, it must be concise and easily checked; the problem of verifying a proof for some given string is the *checking* problem. In a relational setting, all three versions admit a particularly simple formulation; for a fixed relation $\mathcal{R} \subseteq \Sigma^* \times \Sigma^*$:

- *Checking*: given (x, y), does $(x, y) \in \mathcal{R}$?
- *Deciding*: given x, does there exist a y such that $(x, y) \in \mathcal{R}$?
- *Searching*: given x, find a y such that $(x, y) \in \mathcal{R}$.

For the most part, the decision and search versions of a problem have been held to be of comparable complexity, as such is indeed the case for many problems (witness the notion of $\mathcal{NP}$-completeness and $\mathcal{NP}$-equivalence); as to checking, classical complexity theory has only considered it in the case of non-deterministic classes. Schnorr [Sn] has studied the relationship between the decision and search versions of a problem and attempted to characterized this relationship for decision problems within $\mathcal{NP}$; Schöning [So] has proposed a model of computation (*robust* oracle machines) under which decision automatically incorporates checking; other authors have also investigated the search version of decision problems and proposed mechanisms for its characterization [Ba, Ko, Le, etc.]. We go one step further and (essentially) ignore the decision version, concentrating instead on the checking and search versions and their relationship.

Definition 1: A decision problem is a pair $\Pi = (I, M)$, where M is a checker and I is the set of "yes" instances. $\square$

The checker M defines a relation between yes instances and acceptable evidence for them, providing a concise and explicit description of the relation. Moreover, this formulation only allows problems for which a checker can be specified, thereby avoiding existence problems. However, in the following, we shall generally use the relational formalism explicitly; in that formalism, given relation $\mathcal{R}$, the set of acceptable instances of the problem is the domain of the relation, which we denote $D(\mathcal{R})$.

This definition makes a problem into a subjective affair: two different computer scientists may come to the "Algorithm Shop" with different checkers and thus require different evidence-generating algorithms—to the point where one may be satisfied with a constant-time generator and the other may require an exponential-time one. Many classical problems (such as the known $\mathcal{NP}$-complete problems) have "obvious" evidence—what we shall call *natural evidence*; for instance, the natural evidence for the satisfiability problem is a satisfying truth assignment.

Solving such a problem entails two steps: generating suitable evidence and then checking the answer with the help of the evidence; this sequence of steps is our constructive version of the classical decision problem. (Indeed, to reduce our version to the classical one, just reduce the checker to a trivial one which simply echoes the first bit of the evidence string.) Thus the complexity of such problems is simply the complexity of their search and checking components, which motivates our definition of complexity classes.

Definition 2: A constructive complexity class is a pair of classical complexity classes, (C_1, C_2), where C_1 denotes the resource bounds within which the evidence generator must run and C_2 the bounds for the checker. Resource bounds are defined with respect to the classical statement of the problem, i.e., with respect to the size of the domain elements; for nondeterministic classes, C_1 is omitted, thereby denoting that the evidence generator may simply guess the evidence. $\Box$

For instance, we shall define the class $\mathcal{P}_c$ to be the pair $(\mathcal{P}, \mathcal{P})$, thus requiring both generator and checker to run in polynomial time; in other words, $\mathcal{P}_c$ is the class of all $\mathcal{P}$-checkable and $\mathcal{P}$-searchable relations. In contrast, we shall define $\mathcal{NP}_c$ simply as the class of all $\mathcal{P}$-checkable relations, placing no constraints on the generation of evidence. (But note that, since the complexity of checking is defined with respect to the domain of the relation, the polynomial-time bound on checking translates into a polynomial-time bound on the length of acceptable evidence.)

Definition 3: A problem (I, M) belongs to a class (C_1, C_2) if and only if the relation defined by M on I is both C_1-searchable and C_2-checkable. $\Box$

These general definitions only serve as guidelines in defining interesting constructive complexity classes. Counterparts of some classical complexity classes immediately suggest themselves: since all non-deterministic classes are based on the existence of checkable evidence, they fit very naturally within our framework. Thus, for instance, we define

- $\mathcal{NLOGSPACE}_c = (\text{—}, \mathcal{LOGSPACE})$
- $\mathcal{NP}_c = (\text{—}, \mathcal{P})$
- $\mathcal{NEXP}_c = (\text{—}, \mathcal{EXP})$

(Note that, even if $\mathcal{NEXP} = \mathcal{EXP}$—i.e, $\mathcal{NEXP}$ is $\mathcal{EXP}$-decidable—, it does not follow that $\mathcal{NEXP}$ is $\mathcal{EXP}$-searchable; indeed, there exists a relativization where the first statement is true but the second false [IT]. In contrast, $\mathcal{NP}$ is $\mathcal{P}$-decidable if and only if it is $\mathcal{P}$-searchable. This contrast—an aspect of upward separation—adds interest to our definitions of $\mathcal{NP}_c$ and $\mathcal{NEXP}_c$.) Deterministic classes, on the other hand, may be characterized by giving generator and checker the same resource bounds:

- $\mathcal{LOGSPACE}_c = (\mathcal{LOGSPACE}, \mathcal{LOGSPACE})$
- $\mathcal{P}_c = (\mathcal{P}, \mathcal{P})$
- $\mathcal{PSPACE}_c = (\mathcal{PSPACE}, \mathcal{PSPACE})$

The principal tool in the study of complexity classes is the reduction. Many-one reductions between decision problems are particularly simple in the classical framework: one simply maps instances of one problem into instances of the other, respecting the partition into yes and no instances, so that the original instance is accepted if and only if the transformed one is. However, in our context, such reductions are both insufficient and too stringent: we require evidence to go along with the "answer," but can also use this evidence to "correct any error" made during the forward transformation. In essence, the goal of a constructive reduction is to recover evidence for the problem at hand from the evidence gleaned for the transformed instance. Thus a many-one reduction between two problems is given by a *pair* of maps; similar mechanisms have been proposed by Levin [Le] for transformations among search problems and by Krentel [Kr] (who calls his "metric reductions") for transformations among optimization problems, where the solution is the value of the optimal solution. Note that the strict preservation of the partition into yes and no instances is no longer required: we must continue to map yes instances into yes instances, but can also afford to map no instances into yes instances, as we shall be able to invalidate the apparent "yes" answer when attempting to check the evidence.

Definition 4: A constructive (many-one) transformation from problem (relation) $\mathcal{R}_1$ to problem $\mathcal{R}_2$ is a pair of functions (f, g), such that

1. $x \in D(\mathcal{R}_1) \implies f(x) \in D(\mathcal{R}_2)$; and
2. $x \in D(\mathcal{R}_1) \vee (f(x), y') \in \mathcal{R}_2 \implies (x, g(x, y')) \in \mathcal{R}_1$.

$\square$

(Note that, for obvious reasons, the evidence-transforming map, g, must take the original instance as argument as well as the evidence for the transformed instance.) With each complexity class we associate a suitable class of transformations by bounding the resources available for the computation of the two maps: for instance, transformations within $\mathcal{NP}_c$ must run in polynomial time and transformations within $\mathcal{P}_c$ or $\mathcal{NLOGSPACE}_c$ in logarithmic space.

Theorem 1: Constructive many-one reductions are transitive. $\square$

For both resource bounds mentioned above, the proof is trivial.

Equipped with many-one reductions, we can proceed to define, in the obvious way, a notion of completeness for our classes. Note that a desirable consequence of our definitions would be that a problem complete for some class in the classical setting remains complete for the constructive version of the class when equipped with the natural checker, if available.

Since the notion of natural checker is rather vague, we establish this result for some specific classes.

Let *SAT/Nat* be the satisfiability problem (in conjunctive normal form) equipped with the natural checker which requires a truth assignment as evidence; similarly, let *CV/Nat* be the circuit value problem (see [La]) equipped with evidence consisting of the output of each gate and let *GR/Nat* be the graph reachability problem (see [Sa]) equipped with evidence consisting of the sequence of edges connecting the two endpoints.

Theorem 2:

1. *SAT/Nat* is $\mathcal{NP}_c$-complete.
2. *CV/Nat* is $\mathcal{P}_c$-complete.
3. *GR/Nat* is $\mathcal{NLOGSPACE}_c$-complete.

$\square$

Proof: We only sketch the proof of the first assertion; the others use a similar technique, taking advantage of the fact that the generic reductions used in the classical proofs are constructive.

That *SAT/Nat* is in $\mathcal{NP}_c$ is obvious. Denote by $\phi(M, x)$ the formula produced by Cook's transformation when run on a polynomial-time nondeterministic Turing machine M and input string x. Now let Π be some arbitrary problem in $\mathcal{NP}_c$; then there exists some evidence generator for Π, which can be given by a polynomial-time nondeterministic Turing machine M'. Let $f(x) = \phi(M', x)$ and let $g(x, y')$ be the output produced by M' in the computation described by the truth assignment y' for the formula $\phi(M', x)$. Then the pair (f, g) is easily seen to be a constructive, many-one, polynomial-time reduction from Π to *SAT/Nat*.

Q.E.D.

The polynomial-time reductions found in the $\mathcal{NP}$-completeness literature are generally constructive, in terms of natural evidence. Thus, for example, the vertex cover problem with natural evidence (namely, the vertex cover) is $\mathcal{NP}_c$-complete. Similar comments apply for $\mathcal{P}$-complete problems and $\mathcal{NLOGSPACE}$-complete problems. A natural question at this point is whether or not the following statement holds, for a classical complexity class $\mathcal{C}$ and its constructive counterpart $\mathcal{C}_c$:

$$\Pi \text{ is } \mathcal{C}_c\text{-complete if and only if } \Pi \in \mathcal{C}_c \text{ and } D(\Pi) \text{ is } \mathcal{C}\text{-complete.}$$

The only if part would obviously hold if we had required our constructive reductions to preserve the partition between yes and no instances. The if part can be interpreted as follows in the case of $\mathcal{C} = \mathcal{NP}$. We know that weak probabilistic evidence exists for all problems in $\mathcal{NP}$, as a form of zero-knowledge proof exists for all such sets [BC, GMW]; we also suspect that, for some sets in $\mathcal{NP}$ (such as the set of composite numbers), some forms of evidence (factors in our example) are harder to find than others (e.g., witnesses of the type used by Rabin [Ra]). Thus the if part would imply that there exists weak deterministic evidence for membership in an $\mathcal{NP}$-complete set.

3 Self-Reducibility and Oracle Complexity

Constructive complexity is closely tied to the issue of self-reducibility. Self-reducibility itself plays an important role in classical complexity theory (see, e.g. [Ba]), but lacks a natural definition in that framework (whence the large number of distinct definitions). In our constructive formulation, however, self-reducibility has a very natural definition, which, moreover, very neatly ties together the three facets of a decision problem.

Definition 5: A problem Π in some class (C_1, C_2) is (Turing) self-reducible if it is C_2-searchable with the help of an oracle for $D(\Pi)$. $\square$

In other words, a problem self-reduces if it can be searched as fast as it can be checked with the help of a decision oracle; all three facets of a decision problem indeed come together in this definition. In the case of $\mathcal{NP}_c$ problems, our definition simply states that such problems self-reduce if they can be solved in polynomial time with the help of an oracle for their decision version; such a definition coincides with the self-1-helpers of Ko [Ko] and the self-computable witnesses of Balcazar [Ba].

A natural question to ask about reducibility is whether $D(\Pi)$ is really an appropriate oracle for Π: would not a more powerful oracle set make the search easier? We can show that such is not the case and that, in fact, our choice of oracle is in some sense optimal.

Definition 6: Π has *oracle complexity* at most $f(n)$ if there is a deterministic oracle algorithm, using any fixed oracle, that makes at most $f(n)$ oracle queries on inputs of length n and produces acceptable evidence for Π. $\square$

We restrict the oracle algorithm to run within appropriate resource bounds: such as polynomial time for problems in $\mathcal{NP}_c$ and logarithmic space for problems within $\mathcal{P}_c$.

The oracle complexity of some specific problems in $\mathcal{NP}_c$ has recently been investigated. Rivest and Shamir [RSh] show that the oracle complexity of the language of composite numbers with natural evidence (i.e., factors) is at most $n/3+O(1)$. Luks [Lu] has shown that graph isomorphism with natural evidence has oracle complexity $O(\sqrt{n})$. Using the canonical forms technique of Miller [Mi], the isomorphism problems for groups, Latin squares, and Steiner triple systems have oracle complexity $O(\log^2 n)$.

The following theorems summarize some properties of oracle complexity and demonstrate that our choice of oracle is optimal. We have restricted our purview to the three classes $\mathcal{NP}$, $\mathcal{P}$, and $\mathcal{LOGSPACE}$, as they are the most interesting from a practical standpoint and as they are also representative of the behavior of other complexity classes. The first two theorems have trivial proofs.

Theorem 3:
(1) If some $\mathcal{NP}_c$-complete problem has logarithmic oracle complexity, then $\mathcal{P} = \mathcal{NP}$.
(2) If some $\mathcal{P}_c$-complete problem has logarithmic oracle complexity, then $\mathcal{P} = \mathcal{LOGSPACE}$.
(3) If some $\mathcal{NLOGSPACE}_c$-complete problem has logarithmic oracle complexity, then $\mathcal{NLOGSPACE} = \mathcal{LOGSPACE}$. $\square$

Theorem 4: If some $\mathcal{NP}_c$-complete ($\mathcal{P}_c$-complete, $\mathcal{NLOGSPACE}_c$-complete) problem has polylogarithmic oracle complexity, then so do all problems in $\mathcal{NP}_c$ ($\mathcal{P}_c$, $\mathcal{NLOGSPACE}_c$). $\square$

The next theorem indicates that the best possible oracle need never be outside the complexity class in which the problem sits.

Theorem 5: If $\Pi \in \mathcal{NP}_c$ ($\mathcal{P}_c$, $\mathcal{NLOGSPACE}_c$) has oracle complexity less than or equal to $f(n)$ with some fixed oracle A, then oracle complexity no greater than $f(n)$ can be achieved for Π using an $\mathcal{NP}$-complete ($\mathcal{P}$-complete, $\mathcal{NLOGSPACE}$-complete) oracle. $\square$

Proof: Note that $f(n)$ must be $\mathcal{P}$-time constructible for $\Pi \in \mathcal{NP}_c$, with similar conditions for the other two classes. We only sketch the proof. If the oracle set A sits within the class, but is not complete for it, we can simply reduce A to a complete set within the class and thus replace each query to A by an equivalent query to the complete set. If the set A does not sit within the class, A can be replaced by a set A' consisting of prefixes of those computation records of the oracle algorithm that produce acceptable evidence.

Q.E.D.

We can pursue the consequences of this last result in terms of oracle choice. Our first corollary establishes that self-reduction is optimal for complete problems; its proof follows immediately from our last theorem.

Corollary 1: If a problem is complete for one of these three classes, then it self-reduces as efficiently as it reduces to any oracle at all. $\square$

Since it is easy to provide at least one self-reduction, it follows that complete problems for these three classes, in our constructive formalism, *always* self-reduce; such is not the case in the classical setting. Our second corollary shows that self-reduction can only be suboptimal for "incomplete" problems.

Corollary 2:
(1) If a problem in $\mathcal{NP}_c$ is found that self-reduces less efficiently than it does to a given oracle, then $\mathcal{P} \neq \mathcal{NP}$.
(2) If a problem in $\mathcal{P}_c$ is found that self-reduces less efficiently than it does to a given oracle, then $\mathcal{P} \neq \mathcal{LOGSPACE}$.
(3) If a problem in $\mathcal{NLOGSPACE}_c$ is found that self-reduces less efficiently than it does to a given oracle, then $\mathcal{NLOGSPACE} \neq \mathcal{LOGSPACE}$. $\square$

Note that problems obeying the hypotheses cannot be complete (by our previous corollary) nor can they belong to a lower complexity class; hence they are incomplete problems (in the terminology of [GJ]).

Theorem 4 and Corollary 1 can be used as circumstantial evidence that a problem is not complete. For example, since group isomorphism has polylogarithmic oracle complexity, Theorem 4 suggests that it is highly unlikely that group isomorphism with natural evidence is $\mathcal{NP}_c$-complete. Luks' oracle algorithm for graph isomorphism, together with Corollary 1, provides evidence (further to that of [GMW]) that graph isomorphism is not $\mathcal{NP}$-complete, since no one knows of a self-reduction using a sublinear number of queries.

Oracle complexity can be a useful tool in the design of algorithms. Consider the problem of determining whether a given graph has a vertex cover of size at most k, for any fixed k. The results of Robertson and Seymour immediately imply that there exists a quadratic-time algorithm for this problem. Using this (unknown) decision algorithm as an oracle, we can develop a cubic time search algorithm for the problem, which we then turn into a simple linear-time algorithm by eliminating the unknown decision oracle. Select any edge (u, v); delete u and, using the oracle, ask whether the remaining graph has a vertex cover of size at most $k - 1$. If so, put u in the vertex cover and recur; otherwise, put v in the vertex cover and recur. In k queries, a vertex cover has been generated when any exists. Now one can eliminate the oracle by trying all 2^k possible sequences of responses; since 2^k is a constant, the resulting algorithm runs in linear time for each value of k. Better yet, we *know* the algorithm!

4 Blind Oracles and Reductions

Many natural self-reduction algorithms actually make no essential use of the input [FL3]. Formalizing this observation provides another perspective on oracle algorithms and a way to measure their efficiency.

Definition 7: A blind oracle algorithm is an oracle algorithm which has only access to the length of the input, not the input itself; on a query to the oracle, the input is automatically prefixed to the query. $\square$

Thus a blind oracle algorithm attempting to produce evidence for string x only has access to the value $|x|$; however, on query string y, the oracle actually decides membership for the string xy.

Definition 8: The blind oracle complexity of a problem $\mathcal{R}$, call it $\mathrm{boc}(\mathcal{R})$, is the minimum number of oracle calls made to some fixed, but unrestricted language by an oracle algorithm (running within appropriate resource bounds) which uncovers acceptable evidence for the problem. $\square$

Information-theoretic arguments immediately give lower bounds on blind oracle complexity. For example, if $\mathcal{R}$ is the equality relation, we clearly have $\mathrm{boc}(\mathcal{R}) = n$. More interesting are bounds for some standard $\mathcal{NP}$-complete problems.

Theorem 6: Let *VC/Nat* be the vertex cover problem with natural evidence (the cover) and *HC/Nat* the Hamiltonian circuit problem with natural evidence (the circuit).

$$\left\lceil \log \binom{n}{\lfloor n/2 \rfloor - 1} \right\rceil \leq \mathrm{boc}(\mathit{VC/Nat}) \leq \left\lceil \log \binom{n}{\lfloor n/2 \rfloor} + \log n \right\rceil$$

$$\left\lceil \log(1 + \frac{n!}{2n}) \right\rceil \leq \mathrm{boc}(HC/Nat) \leq (n-1)\lceil \log(n-1) \rceil$$

Hence $\mathrm{boc}(VC/Nat) \in \Theta(n)$ and $\mathrm{boc}(HC/Nat) \in \Theta(n \log n)$ □

Proof: The upper bounds are derived from simple oracle algorithms for each problem (that for *VC* actually finds a minimal cover for the problem). The lower bounds come from simple counts of the number of distinct possible arrangements that may have to be checked to identify a solution.

Q.E.D.

Is blind oracle complexity preserved in some sense through reductions? The main problem here is that our reductions are not themselves blind and so defeat the blindness of the oracle algorithms by giving them a description of the input as a side-effect. We need a type of reduction which preserves blindness (such is theory!). Such a reduction must perforce use a very different mechanism from that of constructive many-one reductions. Let us use an anthropomorphic analogy. In the latter style of reduction, the scientist with the new problem calls upon a scientist with a known complete problem, who then serves as a one-shot oracle: the first communicates to the second the transformed instance and the second returns to the first suitable evidence for the transformed instance. The second scientist acts in mysterious ways (i.e., unknown to the first) and thus has attributes of deity. But in a blind reduction, neither scientist is allowed to see the input and yet the instance given the first scientist must be transformed into the instance given the second; both scientists then sit at the same level, as humble supplicants to some all-powerful deity. The reduction goes as follows: the first scientist asks the oracle to carry out the transformation $x \rightarrow f(x)$ implicitly (neither x nor $f(x)$ will be made known), then uses the oracle algorithm provided by the second scientist to establish a certificate y' for the unknown $f(x)$, and finally applies g to recover evidence y for instance x from the known values of y' and $|x|$. Since the oracle algorithm of the second scientist assumes knowledge of the instance size, in this case $|f(x)|$, it is imperative that $|f(x)|$ be computable from $|x|$; hence a blind reduction must be uniform with respect to instance sizes. Note that what gets communicated in the blind reduction is the oracle protocol, whereas what gets communicated in the normal many-one reduction is the (transformed) instance.

Definition 9: A blind (constructive) many-one reduction is a many-one constructive reduction, (f, g), where the first map, f, is length-uniform (i.e., $|f(x)|$ depends only on $|x|$) and the second map, g, only has access to the size of the original input. □

Now we can check that blind reductions indeed preserve blind oracle complexity; we state this only for the case of most interest to us.

Lemma 1: If $\mathcal{R}_1 \in \mathcal{NP}_c$ blindly reduces to $\mathcal{R}_2 \in \mathcal{NP}_c$ and $\mathcal{R}_2$ has polylogarithmic oracle complexity, so does $\mathcal{R}_1$. □

The proof is obvious from our anthropomorphic discussion above.

With a blind transformation, we can define blind completeness in the obvious way. Perhaps surprisingly, blindness does not appear to affect the power of reductions very much, as the following claim shows.

Claim 1: All known $\mathcal{NP}$-complete sets are the domains of blindly $\mathcal{NP}_c$-complete relations. $\square$

Obviously, we cannot offer a proof of this statement; we simply remark that all known reductions between $\mathcal{NP}$-complete problems can easily be made length-uniform and that, in all of these reductions, the evidence assumes such characteristic form that, armed only with the length of the original instance and evidence for the transformed instance, we can easily reconstruct evidence for the original instance. (This obviously is strongly reminiscent of the observation that all known reductions among $\mathcal{NP}$-complete problems can be made weakly parsimonious, so that the number of different certificates for one problem can easily be recovered from the number of different certificates for the transformed instance.)

A fundamental question in classical complexity theory concerns the density of sets in various classes (recall that the density of a set is the rate of growth of its membership as a function of the length of the elements). A set S is *sparse* if $|\{x | x \in S,\ |x| = n\}| \leq p(|x|)$ for some polynomial p; it has *subexponential* density (or is *semi-sparse*) if $|\{x | x \in S,\ |x| = n\}| \in O(2^{log^k n})$, for some positive integer k. It is widely suspected that $\mathcal{NP}$-complete sets must have exponential density; however, all that is known at this time is that $\mathcal{NP}$-complete sets cannot be sparse unless $\mathcal{P} = \mathcal{NP}$ (even their complexity cores cannot at present be shown to have more than subexponential density [OS]). Blind oracles allow us to prove in our context a much stronger result about density.

Theorem 7: There is no blindly $\mathcal{NP}_c$-complete relation with domain of subexponential density. $\square$

Proof: We have shown that *VC/Nat* has linear blind oracle complexity, which is not a polylogarithmic function. Hence the domain of *VC/Nat* has density greater than subexponential. Since polylogarithmic oracle complexity is preserved through blind reductions, it follows than no other $\mathcal{NP}_c$-complete problem can have a domain of subexponential density.
Q.E.D.

Combining this result with a proof for our claim would allow us to transfer our conclusion to $\mathcal{NP}$-complete sets; it might also help in characterizing the relationship between the sets $\mathcal{NP}$ and $\mathcal{POLYLOGSPAC}\ calE$.

5 Conclusions

We have presented a proposal for a constructive theory of complexity. By examining reducibility and oracle algorithms, we have been able to establish a number of simple results which show that our theory has a sound basis and holds much promise. Indeed, through the use of blind oracle methods, we have been able to prove within our framework a much stronger result than has been shown to date in the classical theory.

Much work obviously remains to be done. Problems of particular interest to us at this time include a further study of the relationship between decision, checking, and search. For instance, Schnorr [Sn] conjectures that there exist $\mathcal{P}$-decidable predicates that are not $\mathcal{P}$-searchable if we require particularly concise evidence; this is the type of question that may be advantageously addressed within our framework. Another problem of special interest is the characterization of blindly complete relations in a variety of classes and the connections between blind reductions and communication complexity. Of potential interest is a study of the higher classes of complexity ($\mathcal{PSPACE}$, $\mathcal{EXP}$); although these classes can hardly be deemed constructive from a practical standpoint (any relation that is not $\mathcal{P}$-checkable is only "solvable" in some abstract sense), the greater resources which they make available may enable us to derive some interesting results with respect to reducibility.

Acknowledgements: We would like to thank Len Adleman, Eric Allender, Juris Hartmanis, Richard Karp, Gene Luks, Steve Mahaney, and Doug Stinson for helpful conversations and encouragement.

6 References

[Ba] J.L. Balcazar, "Self-reducibility," *Proc. Symp. on Theoretical Aspects of Comp. Sci.* STACS-87 (1987), 136–147.

[BC] G. Brassard and C. Crepeau, "Nontransitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond," *Proc. 27th Symp. on Foundations of Comp. Sci.* FOCS-86 (1986), 188–195.

[Be] M. Beeson. *Foundations of Constructive Mathematics.* Springer-Verlag, 1980.

[FL1] M. R. Fellows and M. A. Langston, "Nonconstructive Advances in Polynomial-Time Complexity," *Info. Proc. Letters* 26 (1987), 157–162.

[FL2] M. R. Fellows and M. A. Langston, "Nonconstructive Tools for Proving Polynomial-Time Decidability," *J. of the ACM* 35 (1988), 727–739.

[FL3] M. R. Fellows and M. A. Langston, "Fast Self-Reduction Algorithms for Combinatorial Problems of VLSI design," *Proc. 3rd Aegean Workshop on Computing* (1988), 278–287.

[FL4] M. R. Fellows and M. A. Langston, "On Search, Decision, and the Efficiency of Polynomial-Time Algorithms," *Proc. 21st Ann. ACM Symp. on Theory of Computing* STOC-89 (1989), 501–512.

[FRS] H. Friedman, N. Robertson and P. D. Seymour, "The Metamathematics of the Graph Minor Theorem," in *Applications of Logic to Combinatorics*, American Math. Soc., Providence, RI, to appear.

[GJ] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, 1979.

[GMW] O. Goldreich, S. Micali and A. Wigderson, "Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design,," *Proc. 27th Symp. on Foundations of Comp. Sci.* FOCS-86 (1986), 174–187.

[IT] R. Impagliazzo and E. Tardos, private communication reported by E. Allender.

[Jo] D. S. Johnson, "The Many Faces of Polynomial Time," in "The NP-Completeness Column: An Ongoing Guide," *J. of Algorithms* 8 (1987), 285–303.

[KUW] R. M. Karp, E. Upfal and A. Wigderson, "Are Search and Decision Problems Computationally Equivalent," *Proc. 17th ACM Symp. on Theory of Computing* STOC-85 (1985), 464–475.

Ko] Ker-I Ko, "On Helping by Robust Oracle Machines," *Theoret. Comp. Sci.* 52 (1987), 15–36.

[Kr] M. Krentel, "The Complexity of Optimization Problems," Ph.D. Dissertation, Dept. of Comp. Sci., Cornell U., 1987.

[La] R. E. Ladner, "The Circuit Value Problem is Log-Space Complete for $\mathcal{P}$," *SIGACT News* 7 (1975), 18–20.

[Le] L.A. Levin, "Universal Sequential Search Problems," *Problemy Peredachi Informatsii* 9 (1973), 115–116 (in Russian).

[Lu] E. M. Luks, private communication.

[Mi] G. L. Miller, "On the $n^{\log n}$ Isomorphism Technique," *Proc. 10th ACM Symp. on Theory of Computing* STOC-78 (1978), 51–58.

[NT] J. Nesetril and R. Thomas, "Well Quasi Orderings, Long Games and a Combinatorial Study of Undecidability," *Contemporary Mathematics* 65 (1987), 281–293.

[OS] P. Orponen and U. Schöning, "The Density and Complexity of Polynomial Cores for Intractable Sets," *Inf. and Control* 70 (1986), 54–68.

[Ra] M. O. Rabin, "Probabilistic Algorithms for Testing Primality," *J. Number Theory* 12 (1980), 128–138.

[RS1] N. Robertson and P. Seymour, "Disjoint paths—a survey," *SIAM J. Alg. Disc. Meth.* 6 (1985), 300–305.

[RS2] N. Robertson and P. Seymour, "Graph minors—a survey," in *Surveys in Combinatorics* (I. Anderson, ed.), Cambridge Univ. Press (1985), 153-171.

[RSh] R. L. Rivest and A. Shamir, "Efficient Factoring Based on Partial Information," Eurocrypt 85, *Lecture Notes in Computer Science* 219, 31–34.

[Sa] W.J. Savitch, "Nondeterministic $\log n$ Space," *Proc. 8th Ann. Princeton Conf. on Information Sciences and Systems* (1974), 21–23.

[Sn] C. Schnorr, "On Self-Transformable Combinatorial Problems," *Math. Progr. Study* 14 (1981), 225–243.

[So] U. Schöning, "Robust Algorithms: A Different Approach to Oracles," *Theoret. Comp. Sci.* 40 (1985), 57–66.

[Va] L. Valiant, "Relative Complexity of Checking and Evaluating," U. of Leeds Tech. Report, 1974.