# STRUCTURED vs GENERAL MODELS IN COMPUTATIONAL COMPLEXITY *

by A. BORODIN

## I. INTRODUCTION AND CONCLUSION

The goal of this expository paper is to make explicit a certain viewpoint of computational complexity, indeed a viewpoint of computation itself. Since I am calling attention to distinctions which are (at least, intuitively) recognized by most logicians and computer scientists, I feel obliged to immediately draw some conclusions about the usefulness of these distinctions. I will devote the remaining sections to some evidence regarding the conclusions.

I use the term *structured* model of computation in almost the same sense as Pippenger and Valiant [76] (who say "conservative" rather than "structured") to mean that a computation in such a model can only proceed within a fixed, well-defined, mathematical structure; that is, it uses only the relations and operations within that structure for the computation. Hence all intermediate results as well as the inputs and the outputs are from the underlying domain. Two well studied examples are:

S1. Comparison trees (also called pure comparison trees, computation trees) where the structure is $[D; \leqslant]$ with $D$ a partially ordered set, $\{\leqslant, >\}$ or $\{<, =, >\}$ predicates, and no operations. Linear comparison trees extend this model by allowing linear functions of the inputs.

S2. Arithmetic circuits (also called arithmetic straight line programs) where the usual domains are $F[x_1, ..., x_n]$, $F[[x_1, ..., x_n]]$ or $F(x_1, ..., x_n)$, there are no predicates, and the operations are $+, -, \times, \div$.

In contrast, a *general* model of computation can be viewed as a string (over a finite alphabet) processing machine. While the input and output strings for a given problem may arise as a "natural encoding" of a set of

* This article has already been published in *Logic and Algorithmic*, an international Symposium in honour of Ernst Specker, Zürich, February 1980. Monographie de L'Enseignement Mathématique N° 30, Genève 1982.

elements from the domain of a particular mathematical structure, there is no obligation to process these objects in any prescribed way relating to the intended mathematical structure. The main criterion for a general model is that we can "get at" the encoding; that is, we can access and arbitrarily manipulate the individual characters (or bits, since we usually consider binary encoding) of all the inputs and intermediate results. As common examples of such models we have:

G1. Boolean circuits.

G2. Turing machines (TM) in all styles, sizes and colours.

Surely, this distinction between structured and general is quite intuitive. Moreover, similar distinctions have already been made explicit in computer science (and, of course, mathematical logic) in the context of programming language semantics; in particular, the theory of program schemes compares language features by having uninterpreted predicate and function symbols. (Indeed, perhaps the first structured TIME-SPACE tradeoff result for simulating linear recursion schemes by flowchart schemes originates from Paterson and Hewitt's [70] seminal paper—see Chandra [73] and Savage [79].) And we should also note some analogy with the distinction between the first order theory of the reals (or complex numbers) under $+$, $\times$ in contrast to the first order theory of integers (or rationals). In the latter, we can (via the Gödel function) get at the encoding of a domain element, which is precisely why we get undecidability. In the former, we get decidability (see Specker and Strassen [76] for a discussion of the complexity of such decision problems and how we can get at the expressibility and thus encoding of "small" integers).

My purpose here is to argue for the importance of this distinction in the study of computational complexity. I am particularly interested in lower bounds. As a point of reference, let me review the somewhat embarrassing state of affairs in computational complexity with regard to the "general" or string processing viewpoint. If we ignore "diagonalization based results", the following barriers are well recognized:

GB1. The inability to establish a nonlinear lower bound on sequential Time or circuit SIZE.

GB2. The inability to establish a non-logarithmic (i.e. $\omega(\log n)$) lower bound on Space.

GB3. The inability to establish a non-logarithmic lower bound on Parallel Time or circuit Depth.

Although our concepts of "Parallel Time" are still evolving, we should at least note that GB2 and 3 are quite related by the Parallel Time-Space simulations (see Cook's [80] paper in this conference).

For the general viewpoint, one measures complexity as a function of the length of the encoding of the inputs and outputs. In contrast, one measures complexity in the structured viewpoint as a function of the number of inputs and outputs. In either case we can refer to the *size* of the problem. In the structured setting, the barriers are a little less precise yet the analogies do persist.

SB1. We do have some important $\Omega(n \log n)$ lower bounds for algebraic complexity based on degree (Strassen [73]) and for pure and linear comparison trees based on information theoretic arguments (e.g. sorting and related problems—see Reingold [72]). There are even some $\Omega(n^2)$ lower bounds for linear comparison trees (also based on information theoretic arguments—Dobkin and Lipton [78]).

SB2. I do not know of any non-logarithmic Space bounds except for the result of Cook and Rackoff [80]—see section IV.

SB3. Again, I do not know of any non-logarithmic Depth bounds except for some trivial arguments in algebraic complexity based on degree.

We should note that a general model may be considered from a structured point of view in the context of a specific problem and complexity measure. This is the case for the recent result on sorting (see Paul's [80] paper in this conference).

This also seems like an appropriate place to comment on two other concepts, *uniformity* and *restricted models* which relate to our theme. Circuits (arithmetic or Boolean) and Comparison trees or Branching Programs (see Tompa [78]) are non-uniform models in that for each size $n$, we have a different solution. The derivation of these solutions may be completely unrelated (for different $n$) and arbitrarily complex. In contrast, Turing machines are uniform. It turns out that known lower bounds (except for "on-line" computations—see Tarjan [77]) in the structured setting are achieved with respect to non-uniform models. Non-uniformity makes the lower bounds results stronger, but it also reflects the fact that we don't know how to take advantage of uniformity. But the point here is that uniformity considerations are appropriate in either of our settings. We call a model, structured or general, *restrictive* (for a particular class of

problems or for all computable problems) if there is some "reasonable" or "natural" model which can (seemingly) solve the intended problems more efficiently. For example, a one tape TM is provably restrictive and there are senses (on-line computation—see Hennie [66]) in which multitape and multidimensional Turing machines are also restrictive. Valiant [79*c*] demonstrates that monotone $(+, \times)$ arithmetic circuits are restrictive. Comparison trees which use only $\{=, \neq\}$ tests are also provably restrictive in the structured setting (Reingold [72]). I would resist the temptation to think of structured models as being a-priori restricted general devices because the relevant computational domain used need not be finitely encodeable.

Given the barriers GB1, 2, 3, we can see why the following *conjectures* remain fundamental challenges (for notation, see Hopcroft and Ullman [79]).

GC1. $P \neq NP$. I use this to represent all the associated issues, like completeness, $\# P$ problems (Valiant [79*a*]), etc.

GC2. DSPACE(S) $\neq$ NSPACE(S) for reasonable (= constructible) bounds $S$.

GC3. $P \notin \text{DSPACE}(\log^k)$ for any $k$ (and, in particular, for $k=1$). Indeed $P \notin \bigcup_k \text{DSPACE}(\log^k)$. (This is implied by GC3.)

GC4. $P \notin \text{DEPTH}(\log^k)$ for any $k$ (uniform or non-uniform circuit depth). And again, $k = 1$ is of special interest.

GC5. $P \cap \bigcup_k \text{DSPACE}(\log^k)$ (or $P \cap \text{DSPACE}(\log^2)$) is not equal to the class of problems which can be computed *simultaneously* in small Time (polynomial) and Space $(\bigcup_k \log^k)$. The latter class lacks a good notation (Cook [79] calls it PLOPS) so I'll add nothing to the confusion and either call it Polytimelogspace or "SC" (for reasons explained below).

GC6. $P \cap \bigcup_k \text{DEPTH}(\log^k)$ is not equal to Polysizelogdepth. (The last term is a test to see if anyone skipped GC5—Cook refers to this class as "NC", referring to (Nick) Pippenger [79]. If NC takes hold, we would have to use SC for Polytimelogspace.) Referring back to our barriers, we could add insult GB4: "The inability to prove a SIZE · DEPTH lower bound of $\omega(n \log n)$" to the injury of GB1 and GB3.

I would argue that these barriers and many of these same conjectures are of fundamental importance in the structured setting although here one has to look at each specific model to formulate appropriate questions. But this is precisely my main, albeit obvious, conclusion—namely, that it is important and productive to formulate and study the analogous barriers and conjectures in reasonably natural structured settings. Of course, I should admit that my perspective may distort the fact that specific instances of these questions were studied in structured settings *before* the issues were formulated generally. But in the general framework these issues have come into clearer focus. There are several reasons why these issues should also be pursued in structured settings.

1. The issues are usually of significant independent interest in the different settings, especially when the model represents the "natural" model.
2. Some structured models have the property, often by design, that they are sufficiently "general with respect to a specific complexity issue" that results in such a setting will yield a direct corollary for the general theory.
3. A structured model may not be sufficiently general to yield direct corollaries but nevertheless the proof techniques which are developed may become paradigms for the general model.

In the following sections I would like to substantiate these points by primarily considering the two structured models, S1 (comparison based models) and S2 (arithmetic models) mentioned initially. We will then discuss a few other examples outside of these models to further emphasize the utility of this viewpoint.

## II. Comparison Based Models

I want to concentrate on a few examples of models which hopefully will exemplify the utility of the structured viewpoint. The first model, or rather class of models, is the comparison tree (see Knuth [73]). In a pure comparison tree, we label the nodes of a tree by questions of the form "$x_i \leqslant x_j$?". The model then can only solve problems dealing with "searching and sorting". It is also sufficient to consider the input domain to be $\{1, 2, \ldots, n\}$ for a problem of size $n$. On a given input, the computation follows an appropriate path to a leaf, where the output takes place. Since

every problem under consideration is completely determined by the permutation of the input, and since we can sort in $n \log n + o(n)$ comparisons, this simple model cannot address itself to many of the "larger issues" (e.g. P vs NP). Yet, we do get an $\Omega(n \log n)$ lower bound not only for sorting but for set recognition problems like "$X$ distinct ?", "$X = Y$?". The sorting argument simply observes that we need at least one leaf for each of the possible $n!$ permutations and hence the depth of the tree (= number of comparisons = Time in this model) must be at least $\lceil \log n! \rceil = n \log n + o(n)$. As simple as this argument is, it provides a paradigm for asking and answering the same complexity question in a more interesting setting, namely for Random Access Machines (see Paul [80]) with $+$, $-$, $\times$ as unit cost operations. The same questions apparently remain open if integer division is also allowed as a unit cost operation.

One way to establish the set recognition lower bounds can be obtained by considering an extension of the model, namely linear comparison trees where nodes are labelled "$\sum_{i=1}^{n} c_i x_i \geqslant c_{n+1}$?", with $\{c_i\}$ in some underlying field (say $\mathbf{Q}$ for definiteness). For linear comparison trees, we can consider the input domain to be $\mathbf{Q}^n$. It is easily seen that the set of inputs leading to any leaf is a convex subset of $\mathbf{Q}^n$. Dobkin and Lipton [78] then observe that if a subset $A$ (of $\mathbf{Q}^n$) which is being recognized is the disjoint union of $k$ open sets, we must have at least one leaf (in the linear comparison tree) for each such open set (by convexity). Again, it follows that $\lceil \log k \rceil$ time is required. For example, if $A = \{ < x_1, ..., x_n > \mid x_i \neq x_j \}$ it follows that $k = \Omega(n!)$ and hence the $\Omega(n \log n)$ lower bound. By the same proof technique, Dobkin and Lipton show that the knapsack problem ($\{ < x_1, ..., x_n > \mid \exists i_1, ..., i_r : \sum x_{i_j} = 1 \}$) requires $\Omega(n^2)$ comparisons.

The linear tree model allows us to pose more challenging questions; that is, beyond what can be done with a pure comparison tree. Although we can view linear trees as an extension of the pure model, I would claim that, relative to its scope of intended problems, this model is in a sense more structured. This will become clearer if we enlarge our discussion to Space considerations (where it is possible to force larger Time bounds). Each node of a comparison tree can be thought of as representing a state (or I.D.) of the computation. In order to introduce Space complexity, we should coalesce identical states (that is, those with identical subtrees) and let outputs take place at any step of the computation. We are then led to Pippenger's comparison branching programs (see Tompa [78]), which are directed acyclic graphs (rather than trees) whose nodes are labelled as in

comparison trees and whose edges are labelled to denote possible outputs. The Space used by a branching program is defined as log ( # nodes or states), which is precisely Cobham's [66] notion of *capacity* which was defined for general models. (We should also note that a general version of branching programs was also studied by Masek [76] prior to their introduction into a structured setting).

We can construct branching programs for any set of predicates, in particular we can have $\{=, \neq\}$, $\{\leqslant, >\}$, or linear comparisons. And now we can try to clarify why linear branching programs appear to be more structured. Cook, and Tompa [78] observe that $\{\leqslant, >\}$ (or $\{=, \neq\}$) branching programs are "general with respect to Space and Time (within a factor of $n$) complexity" in the following sense:
*Suppose* we have a problem for which we can establish that any branching program which works correctly for problem size $n$ must have Space $= \Omega(S(n))$ (or for which we can establish a Time-Space tradeoff of the form $f$ (Time, Space) $= \Omega(P(n))$—e.g. Space $= O(\log^k n) \Rightarrow$ Time $= \Omega(n^{\log n})$). Then a corresponding result will hold for a general model because the structured model can simulate the general model on a "representative set of inputs". Suppose the $\{\leqslant, >\}$ (respectively, $\{=, \neq\}$) comparison problem is to compute $f_1(x_1, ..., x_n), ..., f_r(x_1, ..., x_n)$; the analogous general problem is to output $\overline{f_1(x, ..., x_n)}, ..., \overline{f_r(x_1, ..., x_n)}$ given $\bar{x}_1, ..., \bar{x}_n$ where $\bar{y}$ denotes a binary encoding of the integer $y$. (Note, the model insures that each $f_j(x_1, ..., x_n) = x_{i_j}$ for some index $i_j$). We only assume that the general model has a read-only input (not necessarily a tape), with a fixed number of reading heads. In fact, since we are permitting random access on the input we can assume that there is only one input head. Now suppose the general model solves this analogous general problem in Space (= Capacity) $\tilde{S}(n)$, and Time (= # of steps or just read instructions) $\tilde{T}(n)$. In particular, the general machine must work on a special class of inputs, those that satisfy the property that $x_i$ is the rank of $x_i$ in $\{x_1, ..., x_n\}$ (resp. $x_i$ is the smallest index $j(i) \leqslant i$ such that $x_i = x_{j(i)}$). But now the structured comparison program can simulate the behaviour of the general machine on this class of inputs in the following way: we need to simulate a move $\delta$: (present state, input bit being read) $\mapsto$ (new state, new head position). But we can use the $\{\leqslant, >\}$ (resp. $\{=, \neq\}$) tests to determine any $x_i$ (and hence any bit of $x_i$) using at most $n-1$ (resp. $i-1$) comparisons so that the branching program has Time complexity

$T \leqslant n\tilde{T}$, and increasing the number of states by a factor of $n^2$ (resp. $n$) so that the Space $S$ of the branching program is $O(\tilde{S})$ since Space $\geqslant \log n$ (whenever the output depends on all $n$ inputs). The bounds $S$ and $T$ then apply as upper bounds within the structured model for the original problem.

This is the sense in which we mean that such a model is "general relative to a particular issue". The model is not general (in that it cannot always get at the encoding), but it can simulate the general machine on a "representative set of inputs". Savitch [73], and Cook and Rackoff [80] have made such observations before with respect to conjecture GC2 (see section IV).

The question then arises as to whether or not linear tree or linear branching programs may be sufficiently general in this same sense. But now it is not clear whether or not there exists an appropriate representative set of inputs for the type of problems we would like to consider. I want to mention one such problem, the shortest path problem, relating to conjecture GC5. The problem can be formulated as a set recognition problem or as a function; given $A = (a_{ij})$, where $a_{ij}$ is the distance (or cost) associated with edge $< i, j >$, compute $D = (d_{ij})$, where $d_{ij}$ is the distance (or the path itself) of the shortest (i.e. of least cumulative cost) path from node $i$ to node $j$. This problem has received considerable attention from the point of view of Time complexity. The most structured model for the problem is a straight-line program or circuit (i.e. no predicates) with operations "min" and $+$. Kerr [70] showed that such "oblivious (the sequence of operations is independent of the inputs) programs require $\Omega(n^3)$, where $n$ is the number of nodes and hence $n^2$ is the size of the problem. A more challenging setting is provided by linear tree programs. In this setting Fredman [76] demonstrates an $O(n^{2.5})$ method (which can be used as the basis for a $O(n^3 (\log\log n)^{1/3}/(\log n)^{1/3}) = o(n^3)$ uniform algorithm). With regard to lower bounds, we only have a negative result by Rivest and Yao [78] that a particularly appealing approach cannot yield a sought after $\Omega(n^2 \log n)$ lower bound. I am interested in Time-Space bounds for this problem in the context of linear branching programs. It seems necessary to extend the model to allow assignments $y_i :=$ linear combination of previously defined $\{y_j\}$ and inputs $\{x_k\}$. Then Space is defined as Capacity plus the number of extra variables $y_i$. The shortest path problem is an excellent example of a problem which is in $P \cap \bigcup_k \text{DSPACE}(\log^k)$ but not, apparently, in Polytimelogspace. (Here I use these terms to have the obvious meaning for a structured setting like linear branching programs as well as their more standard meaning in the general setting). It is con-

ceivable to me that ideas from linear geometry may enable someone (apparently, not me) to establish an $\omega(\log n)$ lower bound on Space, and even more generally establish the structured analogue of conjecture GC5. But I don't see how this would directly yield a corollary for the general theory. It seems fair to augment the Space measure by the precision of the coefficients used in the program to reflect the fact that such coefficients would have to be represented. Now given bounds on Space and Time, we can put bounds on the precision of the inputs needed for a potentially representative input set. Unfortunately, the bound, which clearly exists given the decision procedure for the first order theory of $\mathbf{Q}$ under $+$ (see Specker and Strassen [76]), would be exponential in $t$, the Time bound for the branching program; hence we do not readily obtain a representative set as for the pure branching models since it appears to take time $t$ to decode each input.

But still the problem is of enough independent interest that it is worth pursuing. And, moreover, this gives me an opportunity to argue that even if a direct corollary does not follow, the proof method may generalize. The case in point is the Time·Space $= \Omega(n^2)$ lower bound established by Borodin, *et al.* [79] for sorting on comparison branching programs. This result is "too low-level" to employ the previously discussed simulation for inferring a meaningful lower bound in the general setting. Yet, in this case Borodin and Cook [80] were able to show that the proof method does generalize and a bound of $\Omega(n^2/\log n)$ was established to sort $n$ integers, each of length $O(\log n)$. Hence in terms of input string length $m = O(n \log n)$, we have the time-space bound $\Omega(m^2/\log^3 m)$. The idea in producing this general bound was to take a fairly structured view of the input without giving up any generality. I should also mention that Yao extended the original $\Omega(n^2)$ result to linear branching programs (i.e. a more powerful structured model). Unfortunately, there are no comparable lower bounds for a set recognition problem, in either setting.

Before leaving this section, we should mention another important structured comparison based model, the Batcher comparator network (see Knuth [73]). The model consists of a network (or circuit) with one type of gate, a comparator, which takes $< x, y >$ on input and outputs $< \max(x, y), \min(x, y) >$. Pippenger and Valiant [76] study an extension of this model, called ordering networks, where comparators are replaced by gates

$$\text{computing i)} \quad f(x,y) = \begin{cases} 1 & x \geqslant y \\ 0 & x < y \end{cases}$$

$$\text{and ii)} \quad g(b,x,y) = \begin{cases} x & b = 1 \\ y & b = 0, \end{cases}$$

and then augmented by the usual Boolean operations. Both models can be studied with respect to Depth or Size (= number of gates) of the network. The merging problem is relatively well understood on both models, with $\log n$ Depth and $n \log n$ Size being asymptotically necessary and sufficient. The sorting problem is relatively open. In particular, we know, $\Omega(n \log n) = \text{Size} = O(n \log^2 n)$ and simultaneously $O(\log^2 n)$ Depth on both models. For Depth alone the model is more critical. The conjecture is that sorting requires $\Omega(\log^2 n)$ Depth on comparator networks, whereas the results of Muller and Preparata [75] show that $O(\log n)$ Depth is sufficient (with Size $= O(n^2)$) for ordering networks. However, this raises the question as to whether or not we can simultaneously achieve $O(\log n)$ Depth and quasilinear Size (i.e. $O(n \log^k n)$). (For a much more powerful non oblivious parallel model, namely a comparison tree with $n$ comparisons permitted in parallel, Valiant [75] can derive such simultaneous bounds—see also Preparata [78]). Even for comparator networks, there is no proof that Size $\cdot$ Depth $= \omega(n \log^2 n)$. This same issue concerning sorting on ordering networks can be viewed in the general setting of Boolean circuits which are to sort $n$ numbers, each of binary length $O(\log n)$.

The Size-Depth problem for sorting (in contrast to Time-Space) seems to have a very interesting complexity behaviour, also observed for a variety of other problems involving simultaneous resource bounds. This behaviour is as follows: An optimal bound for measure 1 (say Depth $= O(\log n)$) can be achieved when the bound for measure 2 is essentially pessimal (say Size $= O(n^2)$), whereas by relaxing the measure 1 somewhat (to $O(\log^2 n)$) we can get good (i.e. quasilinear) measure 2 bounds. At the other extreme, an optimal measure 2 bound, say if $O(n \log n)$ Size were possible, seems to be achievable only with an essentially pessimal measure 1 bound. As other examples consider Space (as measure 1) and Time (measure 2) for the problems of the median (see Munro and Paterson [78] for the upper bound) and the string pattern matching problem (see Galil and Seiferas [77] for the upper bound). This latter problem exhibits a more quantitative statement of the behaviour, namely that with approximately $k$ registers (which in our terms would be $k \log n$ Space since each register holds a pointer) one can solve the string pattern matching problem in Time

$O(n^{1+1/k})$. (Pippenger [personal communication] has recently shown that a similar upper bound can also be achieved for the sorting problem in the context of ordering networks.) This same quantative behaviour has a corresponding lower bound for the (structured) simulation of linear recursion schemes by flow-chart schemes that was referred to in Section I.

## III. Arithmetic Models — Algebraic Complexity

I would now like to turn attention to the complexity of arithmetic problems, and to the straight line or circuit model with operations $+$, $-$, $\times$ (and perhaps $\div$). Fortunately, I need not pursue this topic in too much detail since Valiant [80] in this conference will be addressing just this topic. Indeed, Valiant [79*a*] and [79*b*] has always provided compelling evidence for the importance of the interrelation between a structured problem setting (algebraic complexity) and the general theory. The correspondence between algebraically structured arithmetic circuits computing (say) formal polynomials in $F[x_1, ..., x_n]$ and general Boolean circuits computing Boolean functions of $n$ variables is readily apparent. In the former, gates represent the ring operations $(\times, +, -)$ and the inputs are the (indeterminates) $\{x_i\} \cup F$, while in the latter, the gates represent some basis set of Boolean operations (say $\wedge$, $\vee$, $\neg$) and the inputs are the (Boolean variables) $\{x_i\} \cup \{0 \text{ or false}, 1 \text{ or true}\}$. Since $\vee$, $\wedge$, $\neg$ can be easily simulated by $+$, $-$, $\times$ when restricted to $\{0, 1\{$ (e.g. $x \vee y$ by $x + y - x \times y$), positive results for the arithmetic case often carry over immediately to the Boolean setting. The usual measures of complexity are SIZE (= number of gates = sequential Time complexity), DEPTH (= length of longest path in the circuit = parallel time complexity) and FORMULA SIZE (= number of gates in a circuit having fan-out one; i.e. a formula). One of the first (pair of) results that demonstrated to me the importance of keeping this correspondence in mind, is the relating of the FORMULA SIZE and DEPTH measures. Independently, Spira [71] (for the Boolean case over any basis) and Brent [74] showed how to convert any formula of size $m$ to an equivalent formula (and hence circuit) of depth $O(\log m)$; the converse that any circuit of depth $d$ can be converted to a formula of size $2^d$ is immediate. It is interesting to note that the Spira result seems to depend intrinsically on the Boolean domain, whereas the Brent result is proven in a more abstract setting using only that $\times$ (resp. $\wedge$) distributes over $+$ (resp. $\vee$). Then, here again, is a situation where a result need not

yield a direct corollary (the simulation of $x \vee y$ by $x + y - x \times y$ requires fan-out two) yet the proof technique can be applied. In this regard, it is interesting to note that whereas Pippenger [74] shows every *symmetric* Boolean function has polynomial Formula Size (i.e. can be computed in $O(\log n)$ Depth), the corresponding result is not known for the arithmetic elementary symmetric functions ($O(\log^2 n)$ Depth is easy to establish via polynomial multiplication).

While the relationship between FORMULA SIZE and DEPTH is relatively well understood, the relation between Size and Depth is a more fundamental issue (see conjectures GC4 and GC6). The relating of TIME to SIZE and SPACE to DEPTH in the general setting (for example, see Borodin [77]) also shows the relationship between conjectures GC3 and GC5 but the classes Polytimelogspace and Polytimelogdepth (in conjectures GC4 and GC6) may be quite different (see Cook [80]). In the algebraic setting, these general relationships take on added interest when combined with some important results concerning Depth. First, Csanky proved that a number of central problems (including $A^{-1}, A^n$) have the same depth complexity as computing det $(A)$, and, more important, these problems can be computed in $O(\log^2 n)$ depth and simultaneously in $O(n^4)$ size. Hyafil [79] took Csanky's result further in showing that any set of multivariate polynomials of degree $\leqslant d$ and computable in Size $\leqslant t$, can be computed in Depth $= O(\log d \cdot \log t)$. Clearly $\Omega(\log d)$ is a lower bound on depth, so that Hyafil's result is a major challenge to conjecture GC4 in the algebraic (i.e. structured) setting. It is the concept of degree which gives us an opportunity to relate Size and Depth in the algebraic setting. The concept of degree (in the sense of algebraic geometry) also gave Strassen [73] the means to establish a non trivial $\Omega(n \log n)$ lower bound on Size. At present, we do not have a meaningful analogy to *degree* in the Boolean setting, and hence all the barriers and conjectures remain intact.

Hyafil's [79] result leaves open the analogue of conjecture GC6. The construction which converts from deg $d$, Size $t$ to Depth $O(\log d \cdot \log t)$ results in a Size of $t^{O(\log d)}$; that is, it does not preserve polynomial size. This contrasts with Csanky's [76] result that our concrete examples (det, $A^n$, $A^{-1}$) are in Polytimelogdepth. It also remains open as to whether or not these concrete problems (or perhaps all small degree polynomials computable in polynomial Size) can be computed in smaller Depth (e.g. $O(\log n)$). In this regard, one can note the similarity between the arithmetic $A^n$, $A^{-1}$ problems and the Boolean Depth requirements for integer powering and division (see Cook [80]). We also note the similarity between the arith-

metic $A^n$ and the Boolean $A^*$ (transitive closure). The latter problem is complete for the issue of NSPACE(S) vs DSPACE(S). $A$ (uniform) positive result for $A^n$ (say $O(\log^\alpha n)$ Depth with $\alpha < 2$) would directly improve Savitch's [70] deterministic simulation of nondeterministic space bounded computations. It appears to me then that a first attempt to break barriers GB2, 3 would be to try to establish a nonlogarithmic lower bound for the depth of $A^n$ (equivalently, the det). The only known lower bound, Shamir and Snir [77], is that $A^n$ does require depth $\Omega(\log^2 n)$ for monotone $(+, \times)$ circuits. However, Valiant [79*c*] has shown that monotone circuits can be *exponentially* inefficient.

To me, the most compelling evidence of the importance of the algebraic viewpoint for the general theory is Valiant's [79*a*] result that the *permanent* (say when restricted to integer matrices) is complete for the class $(\#P)$ of problems associated with counting the number of solutions of problems computable in nondeterministic polynomial time (NP). The difference between the determinant and permanent is thus made quite explicit in complexity terms (even though we can't yet translate this into provable lower bounds). Recently, Valiant [79*b*] uncovers the central role that the determinant and permanent problems play within algebraic complexity itself (with the result of further insights into the general theory). Valiant is directly motivated (see his introductory paragraph) by the kind of completeness results one obtains in the general theory. But rather than use complexity based reductions (as is usually done) he is able to base his reductions on purely algebraic properties. Again, following Schnorr's [76] original beliefs in this regard, I would also argue that the algebraic setting is a reasonable framework for attacking what many feel to be the fundamental issue of complexity (P vs NP), even though (or maybe because) lower bounds in the sense of algebraic complexity do not seem to have any direct corollaries to the general theory.

I want to conclude this section by considering Space complexity, and, more precisely, Time-Space tradeoffs for the algebraic setting. Space is not usually considered within algebraic complexity but I think it is quite relevant to our thesis. The Space measure here evolves naturally from the attempt to execute a circuit as a straight-line program; namely, it is the number of intermediate locations needed to store partial results. This measure has been well studied in the guise of a certain *pebble game* introduced in the schematology paper of Paterson and Hewitt [70] and used in other structured settings (e.g. Cook [73] and Cook and Sethi [76] where conjecture GC3 is formulated in a structured setting). Indeed, the pebble

game has been studied extensively and is the basis for a number of results that have "TIME-SPACE tradeoffs" as part of a title. If one wishes to conserve on the number of intermediate locations (the number of pebbles) then it may be necessary to often recompute results (i.e. repebble the same node of the circuit). Tompa [78] uses the connectivity properties of the FFT problem to demonstrate a Space (number of pebbles) · Time (number of pebble moves) = $\Omega(n^2)$ lower bound. I find it interesting that, independently, Grigoryev [76] produces a similar Time-Space = $\Omega(n^2)$ tradeoff for multiplication in $Z_2[x]$ (which extends to integer multiplication) with respect to Boolean circuits (i.e. general setting) by using arguments about the range of subfunctions.

My interest stems from the fact that the same duality (between connectivity and subfunctions) again provides the basis for two results concerning VLSI design; namely, using similar models, Thompson [79] shows the product of Area (= length of wire) and Parallel Time$^2$ = $\Omega(n^2)$ for the FFT (structured setting) while Brent and Kung [79] show Area · Parallel Time$^2$ = $\Omega(n^2)$ for integer multiplication (general setting). Recently, Brent and Goldschlager have established an analogous Area · Parallel Time tradeoff result for a set recognition problem.

## IV. Other Structured Models

I should use this last section to briefly indicate that many other structured models can be found in a variety of problem areas. Yet, these models are often more appropriate to particular problems rather than for a large class of problems. Hence, the real purpose of this section is to indicate a need for structured models natural to important problem areas.

Perhaps I should constrain this concluding discussion to an obvious candidate, a "model for graph-theoretic problems". But given the scope of graph theory, this seems far too ambitious. What has been done thus far? We have already discussed the use of linear comparison trees and branching programs for the study of shortest path problems. This model does seem to abstract the underlying tests and operations needed for such problems while suppressing any data structures needed for both searching and representation. The comparison tree becomes a rather uninteresting general model if we study unlabelled graph problems; since any such problem can be "solved" by looking at each entry of the input adjacency matrix. The

solutions (Kirkpatrick [74], Rivest and Vuillemin [76]) to the Rosenberg-Aanderaa conjecture show that most graph problems require every entry of the adjacency matrix be probed. (Obviously, there are other ways to represent a graph. But unlabelled graph problems do again become non-trivial if we consider time-space considerations with regard to the generalized version of branching programs).

There is a class of structured models that have been developed by Savitch [73], and Cook and Rackoff [80] for studying the space complexity of the directed and undirected versions of the graph reachability (alias transitive closure) problem. The models reflect the kinds of path traversal strategies one often uses in graph theoretic algorithms. Similar models have been employed for the problem of searching mazes (see Blum and Kozen [78]). Essentially, the JAG model of Cook and Rackoff consists of a finite control of say $q$ states plus a set of $p$ labelled markers. The Space charge is $\log q + p \log n$, the latter term reflecting the fact that a marker is used to remember a node in the graph. The graph is oriented (i.e. edges leaving a node are numbered) and the model traverses a graph by moving markers along edges or to be coincident with other markers. Moves are determined by the state and by the presence of markers. If the model does not allow backward traversal of edges, then Cook and Rackoff can demonstrate an $\Omega(\log^2 n/\log\log n)$ Space lower bound in the directed case. For the undirected case, the result of Aleliunas *et al.* [79] shows that the reachability problem can be solved in $O(\log n)$ Space by a Monte Carlo algorithm, and hence in non-uniform $O(\log n)$ Space. Indeed, a JAG with only 2 markers and a polynomial number of states can solve the undirected reachability problem.

If the JAG model does allow backward edge traversal, then the model becomes "general for the issue of NSPACE ($\log n$) $\neq$ DSPACE ($\log n$)". The Aleliunas *et al.* result is based on a universal covering sequence (for all $n$ node cubic graphs) of polynomial ($n$) length; it is this covering sequence which leads to a "representative set of inputs" on which a JAG with backward edge capability can simulate a general algorithm for the directed reachability problem (which is log space complete for NSPACE ($\log n$)).

Recently there has been a set of interesting results (see Bland and Las Vergnas [78], and Lovász [79]) concerning matroid properties where the model is essentially an oracle for determining the *independence* of a set of elements. Since matroid properties (and algorithms, like the Greedy algorithms) are often viewed as generalizations of graph theoretic properties, one might view the independence oracle as a structured model for graph

theory. As such, this is a different kind of structured model than I have been trying to sell in this paper. The results here proceed by an adversary argument which constructs two similar looking matroids to force an exponential number of oracle calls in order to determine a certain property. But the matroids being constructed need not be, and are not, constructed from the same "domain" (e.g. graphs, with cycle free paths as independent sets). It is rather like relativized complexity theory (see Baker, *et al.* [75]) or the construction of non standard models in logic, which gives insight into what kind of arguments will not work. However, I am trying to emphasize structured models where the domain is "standard" and the structure issues hinge on the accessing and processing of such domain elements.

Given the significant progress in the field of graph theoretic algorithms, it is relatively disappointing how few structured models have been proposed for this area. In particular, we seem to have adopted the model of algebraic complexity to study graph theoretic variants of P vs NP, and to study lower bounds for graph theoretic parallel computation (see Reghbati and Corneil [78] for some upper bounds in this context). To be fair, we sometimes adopt graph theory as a means to proving lower bounds in algebraic complexity (see Valiant [77]).

Clearly, I have not nearly exhausted the variety of computational problems whose complexity has been studied from both the general and structured viewpoints. But I hope I have begun to defend my earlier conclusions that the general theory provides a standard for assessing complexity results about structured models and conversely that the structured setting gives insight for the general theory.

*Note added at the end of Symposium:* M. Rabin observed to me that Khachian's polynomial time solution for the Linear Programming problem provides a dramatic example of the structured vs general distinction. The present polynomial bound is based on the precision of the inputs and not just the number of inputs. Similar remarks can be made about the Transportation Problem and the Edmonds-Karp solution.

*Acknowledgement.* S. Cook made a number of valuable insights and suggestions to me about this topic. His ideas permeate this paper. I also want to thank, but not hold responsible, R. Aleliunas, P. Dymond, N. Pippenger, C. Rackoff, A. Rosenberg and J. von zur Gathen for their suggestions and corrections. Finally, I want to thank E. Engeler, E. Specker and V. Strassen for many enjoyable conversations about this and other topics during my stay at the E.T.H. (1976).

## REFERENCES


ALELIUNAS, R., R. M. KARP, R. J. LIPTON, L. LOVÁSZ and C. RACKOFF [79]. Random Walks, Universal Traversal Sequences, and Complexity of Maze Problems. *Proc. of 20th Annual IEEE FOCS*, Oct. 1979, pp. 218-223.

BAKER, J., J. GILL and R. SOLOVAY [75]. Relativizations of the $P = ?\ NP$ Question. *JCSS, Vol. 4, No. 4* (1975), pp. 431-442.

BLAND, R. and M. LAS VERGNAS [78]. Orientability of Matroids. *J. of Combinatorial Theory, Series B, 24* (1978), pp. 94-123.

BLUM, M. and D. KOZEN [78]. On the Power of the Compass (or, Why Mazes are Easier to Search than Graphs. *Proc. of 19th Annual IEEE FOCS*, Oct. 1978, pp. 132-142.

BORODIN, A. [77]. On Relating Time and Space to Size and Depth. *SICOMP, Vol. 6* (1977), pp. 733-744.

BORODIN, A., M. FISCHER, D. KIRKPATRICK, N. LYNCH and M. TOMPA [79]. A Time-Space Tradeoff for Sorting and Related Non-Oblivious Computations. *Proc. of 20th Annual IEEE FOCS*, Oct. 1979, pp. 319-327.

BORODIN, A. and S. COOK [80]. A Time-Space Tradeoff for Sorting on a General Sequential Model of Computing. *To appear in Proc. of 12th Annual ACM STOC*, May 1980.

BRENT, R. [74]. The Parallel Evaluation of Several Arithmetic Expressions. *JACM 21* (1974), pp. 201-206.

BRENT, R. and H. J. KUNG [79]. *The Area Time Complexity of Binary Multiplication.* Dept. of Computer Science Tech. Rep., Carnegie-Mellon Univ., 1979.

CHANDRA, A. K. [73]. Efficient Compilation of Linear Recursive Programs. *Proc. of IEEE 14th Annual SWAT Conf.*, Oct. 1973.

COBHAM, A. [66]. *The Recognition Problem for the Set of Perfect Squares.* Research Paper RC-1704, IBM Watson Research Center, Yorktown Heights, New York, April 1966.

COOK, S. [73]. An Observation on Time-Storage Tradeoff. *Proc. of 5th Annual STOC*, May 1973, pp. 29-33.

COOK, S. and R. SETHI [76]. Storage Requirements for Deterministic Polynomial Time Recognizable Languages. *JCSS, Vol. 13, No. 1* (1976), pp. 25-37.

COOK, S. [79]. Deterministic CFL's are Accepted Simultaneously in Polynomial Time and Log Squared Space. *Proc. of ACM 11th Annual STOC*, May 1979.

—— [80]. The Theory of Unbounded Parallel Computation. *This conference.*

COOK, S. and C. RACKOFF [80]. Space Lower Bounds for Maze Threadability on Restricted Machines. *To appear in SICOMP.*

CSANKY, L. [76]. Fast Parallel Inversion Algorithms. *SICOMP, Vol. 5, No. 4* (1976), pp. 618-623.

DOBKIN, D. and R. LIPTON [78]. A Lower Bound of $\frac{1}{2} n^2$ on Linear Search Programs for the Knapsack Problem. *JCSS, Vol. 16, No. 3* (June 1978), pp. 413-417.

FREDMAN, M. [76]. New Bounds on the Complexity of the Shortest Path Problem. *SICOMP, Vol. 5, No. 1* (March 1976), pp. 83-89.

GALIL, Z. and J. SEIFERAS [77]. Saving Space in Fast String-Matching. *18th Annual Symposium on Foundations of Computer Science*, Providence, Rhode Island, Oct.-Nov. 1977, pp. 179-188.

GRIGORYEV, D. Yu. [76]. An Application of Separability and Independence Notions for Proving Lower Bounds of Circuit Complexity (in Russian). *Notes of Scientific Seminars, Vol. 60, Steklov Mathematical Institute*, Leningrad Department, 1976, pp. 38-48.

HENNIE, F. [66]. On-line Turing Machine Computations. *IEEE Trans. Elec. Comp. EC-15* (1966), pp. 35-44.

HOPCROFT, J. and J. ULLMAN [79]. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

HYAFIL, L. [79]. On the Parallel Eval of Multivariate Polynomials. *SICOMP, Vol. 8, No. 2* (May 1979), pp. 120-123.

KERR, L. [70]. *The Effect of Algebraic Structure on the Computational Complexity of Matrix Multiplication.* Ph.D. Thesis, Cornell University.

KIRKPATRICK, D. [74]. Determining Graph Properties from Matrix Representations. *Proc. 6th Annual SIGACT STOC,* May 1974, pp. 84-90.

KLEIN, P. [79]. *A Lower Bound for the Sorting Complexity on a RAM.* Universität Bielefeld (preprint).

KNUTH, D. E. [73]. *The Art of Computer Programming, Vol. 3.* Sorting and Searching, Addison-Wesley Pub. Co., Reading, Massachusetts, 1973.

LOVÁSZ, L. [79]. *The Matroid Parity Problem.* Working Paper, Dept. of Combinatorics and Optimization, Univ. of Waterloo.

MASEK, W. J. [76]. *A Fast Algorithm for the String Editing Problem and Decision Graph Complexity.* M.Sc. thesis, Massachusetts Institute of Technology, Cambridge, Mass., May 1976.

MULLER, D. E. and F. P. PREPARATA [75]. Bounds to Complexities of Networks for Sorting and for Switching. *JACM, Vol. 22* (April 1975), pp. 195-201.

MUNRO, I. and M. PATERSON [78]. Selection and Sorting with Limited Storage. *Proc. of IEEE 19th Annual FOCS,* Oct. 1978.

PATERSON, M. S. and C. E. HEWITT [70]. Comparative Schematology. *Project MAC Conf. on Concurrent Systems and Parallel Computation,* Woods Hole, Massachusetts, June 1970, pp. 119-127.

PAUL, W. [80]. Decision Trees and Random Access Machines. *This conference.*

PIPPENGER, N. [74]. *Short Formulas for Symmetric Functions.* IBM Res. Report RC-5143, Yorktown Heights, 1974.

PIPPENGER, N. and L. G. VALIANT [76]. Shifting Graphs and Their Applications. *Journal of the Association for Computing Machinery, Vol. 23* (July 1976), pp. 423-432.

PIPPENGER, N. [79]. On Simultaneous Resource Bounds, *Proc. of IEEE 20th Annual FOCS,* Oct. 1979.

PREPARATA, F. [78]. New Parallel-Sorting Schemes. *IEEE Trans. on Computers, Vol. c-27, No. 7* (July 1978), pp. 669-673.

REGHBATI, E. and D. CORNEIL [78]. Parallel Computations in Graph Theory. *SICOMP, Vol. 7, No. 2* (May 1978), pp. 230-237.

REINGOLD, E. M. [72]. On the Optimality of Some Set Algorithms. *Journal of the Association for Computing Machinery, Vol. 19* (Oct. 1972), pp. 649-659.

RIVEST, R. and J. VUILLEMIN [76]. On Recognizing Graph Properties from Adjacency Matrices. *Theoretical Computer Science, Vol. 3,* (1976), pp. 371-384.

RIVEST, R. and A. YAO [78]. On the Polyhedral Decision Problem. *To appear in SICOMP.*

SAVAGE, J. and S. SWAMI [79]. *A Time-Space Tradeoff for Linear Recursion.* Brown Univ. Tech. Rep., 1979.

SAVITCH, W. J. [70]. Relationships between Nondeterministic and Deterministic Tape Complexities. *JCSS, Vol. 4, No. 2* (1970), pp. 177-192.

SAVITCH, W. [73]. Maze Recognizing Automata and Nondeterministic Tape Complexities. *JCSS, Vol. 7* (1973), pp. 389-403.

SCHNORR, C. [76]. A Lower Bound on the Number of Additions in Monotone Computations. *Theoretical Computer Science, Vol. 2* (1976), pp. 305-315.

SHAMIR, E. and M. SNIR [77]. *Lower Bounds on the Number of Multiplications and the Number of Additions in Monotone Computations.* IBM Report RC 6757, 1977.

SPECKER, E. und V. STRASSEN [76]. *Komplexität von Entscheidungsproblemen.* Springer-Verlag, Lecture Notes in Computer Science, 1976.

SPIRA, P. M. [71]. On Time Hardware Complexity Tradeoffs for Boolean Functions. *Proc. 4th Hawaii Symp. on Systems Science*, 1971, pp. 525-527.

STRASSEN, V. [73]. Die Berechnungskomplexität von elementary-symmetrischen Functionen und von Interpolationskoeffizienten. *Numerische Mathematik, Vol. 20, Nr. 3* (1973), pp. 238-251.

TARJAN, R. [77]. Reference Machines Require Nonlinear Time to Maintain Disjoint Sets. *Proc. of ACM 9th Annual STOC*, May 1977.

THOMPSON, C. D. [79]. Area-Time Complexity for VLSI. *Proc. 11th Annual ACM STOC*, May 1979, pp. 81-88.

TOMPA, M. [78]. *Time-Space Tradeoffs for Straight-Line and Branching Programs.* Univ. of Toronto, Comp. Sci. Dept. Tech. Rep. #122, July 1978.

VALIANT, L. [75]. Parallelism in Comparison Problems. *SICOMP, Vol. 4* (Sept. 1975), pp. 348-355.

—— [77]. Graph-Theoretic Arguments in Low-Level Complexity. *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, Vol. 53, Springer-Verlag, Berlin, 1977, pp. 162-176.

—— [79*a*]. The Complexity of Computing the Permanent. *Theoretical Computer Science, Vol. 8, No. 2* (1979), pp. 189-202.

—— [79*b*]. Completeness Classes in Algebra. *Proc. of IEEE 20th Annual FOCS*, Oct. 1979, pp. 249-261.

—— [79*c*]. Negation can be Exponentially Powerful. *Proc. of ACM 16th Annual STOC*, 1979, pp. 189-196.

—— [80]. Reducibility by Algebraic Projections. *This conference.*

*(Reçu le 16 juillet 1980)*

A. Borodin

Department of Computer Science
University of Toronto