I. Introduction and Conclusion

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