1. Introduction

TOWARDS A COMPLEXITY THEORY OF SYNCHRONOUS PARALLEL COMPUTATION

by Stephen A. Cook [1])

Abstract. This is largely an expository paper on the general theory of synchronous parallel computation. The models of parallel computers discussed include uniform circuit families, alternating Turing machines, conglomerates, vector machines, and parallel random access machines. A classification of these models indicates the need for still more; so "aggregates" and "hardware modification machines" are introduced. The resources sequential time, space, parallel time, circuit size and depth, hardware size etc., are discussed and interrelated. Work in progress at Toronto is mentioned and basic open questions are listed.

1. Introduction

There is now a well developed computational complexity theory of sequential computation. The precisely "right" computer model is not completely clear, but the main contenders for this model do not differ markedly from each other in their computing efficiency. These contenders are multitape Turing machines, possibly with storage structures more general than linear tapes, and various versions of random access machines. Of these models, the storage modification machine (SMM) made popular by Schönhage [S2] carries the most conviction as a stable and general model of a sequential computer; where we take sequential to mean the number of active elements is bounded in time.

To be sure, there is a feeling that one step of an SMM may be a little too powerful. It is hard to imagine a mechanism for reconnecting a given edge out of a node v_1 in the storage structure to a node v_2 in one step, when the candidates for v_2 from the perspective of the whole computation are unlimited. But the fact remains that if we restrict ourselves to fixed storage

[1]) Presented at the *Symposium über Logik und Algorithmik* in honour of Ernst Specker, Zürich, February 1980.

structures, there is no single structure or class of structures which seems to be just right. (Certainly multitape Turing machines are too restrictive.) On the other hand, for random access machine models one is never quite sure which set of operations should be primitive, and whether to charge more than one time unit for an operation capable of manipulating arbitrarily large integers.

Whatever the sequential model, it is clear that the main resources of interest are time and space. Let me repeat that the differences among the leading models in the time and space needed to execute algorithms are minor. And the theory of sequential time and space complexity is a rich and interesting one.

In the past few years it has become increasingly clear that the most powerful computers of the future will not be sequential but parallel. An entire processor can now be placed on a VLSI chip that is so small and cheap that it is not hard to imagine a machine of the future consisting of millions of such processors connected together and operating synchronously. The questions then become: How should the machine be organized and what can be done with the result? Hence the need for a theory of parallel computation. (A second motivation, of course, is that the human brain appears to be a parallel computer.)

I should point out here that the theory I have in mind deals only with synchronous computers. There is indeed a great and interesting literature on asynchronous processes, and the theory has applications when the processes in question cannot easily be synchronized (such as distributed computer systems or operating systems). The theory discussed here assumes one parallel computer whose elements have been designed from scratch to operate synchronously.

The first problem in this theory is to find the right mathematical model of a parallel computer. The parallel models in the literature fall roughly into two classes: those with fixed structure and those with modifiable structure. The fixed structure parallel models correspond to sequential machines with fixed storage structure, namely Turing machines with "tapes" which may be more general than linear arrays, but cannot be modified. The parallel analogs of these include Borodin's uniform circuit families [B1], Goldschlager's conglomerates [G1], [G2], and Hoover's uniform infinite circuits [H1].

The modifiable sequential machines include SMM's and random access machines (RAM's). (Indirect addressing in a RAM gives the effect of a modifiable storage structure, and in fact RAM's which can only add and

subtract one are equivalent to SMM's [S2].) The modifiable parallel machines include various parallel RAM's, such as SIMDAG's [G1] and P-RAM's [SS] and [FW], as well as vector machines as defined in [PS]. As yet no parallel analog of SMM's has appeared, but a tentative candidate is introduced in section 5.

Fortunately, all these models are roughly equivalent from the point of view of computation time, in the sense that each can simulate another while at most cubing the computation time. In fact, the sets or functions computed by each in time $S^{O(1)}$ (i.e. time polynomial in S: this notation appears in [P1]) are the same as those computed by a Turing machine in space $S^{O(1)}$ for any well behaved time bound S. (This phenomenon was observed, for example, in [CS], and called the "parallel computation thesis" in [G1].)

This thesis can be made more specific as follows: For the fixed structure parallel machines; namely, uniform circuit families, conglomerates, "aggregates" (see section 4) and uniform infinite circuits (see [H1]),

(1.1) parallel time (S) $\subseteq$ DSPACE (S) $\subseteq$ NSPACE (S) $\subseteq$ parallel time (S^2)

(See [HU] for the meaning of DSPACE and NSPACE.)
On the other hand, the modifiable parallel machines tend to be more powerful, and the inclusions become (at least for SIMDAG's and the P-RAM's of [FW]):

(1.2) parallel time (S) $\subseteq$ DSPACE (S^2) $\subseteq$ parallel time (S^2).

(For SIMDAG's, the stronger statement NSPACE (S) $\subseteq$ parallel time (S) also holds [G1]).

The modifiable parallel models that have been proposed so far all share the same problem as the sequential RAM models: The choice of primitive operations seems arbitrary, and most of these operations (such as shifts in vector machines and random access to global storage in P-RAM's) seem too powerful to be primitive. Hence I propose a new modifiable parallel model: "Hardware Modification Machines" (HMM's), to be the parallel analog of SMM's. These are discussed in section 5.

Time and space are the fundamental resources in sequential complexity theory. What are their analogs in the parallel theory? Obviously, parallel time plays a fundamental role. The second important parallel resource, I think, should be hardware size; that is, the number of elements of a machine which are active during a computation. For conglomerates, hard-

ware size is the number of active finite state machines, and for vector machines it is the sum of the lengths of the vectors. For SIMDAG's and P-RAM's it corresponds roughly to the number of processors, although it should take into account the total memory used. For circuits, the circuit size is an upper bound on hardware size, but the traditional restriction that circuits are acyclic disallows elements to be reused during a computation and hence may give an unrealistically large value for size. Hence "aggregates" are introduced in section 4. These can be thought of either as circuits with cycles, or as finite conglomerates.

Section 2 discusses two fundamental fixed structure parallel models; namely, uniform circuit families and alternating Turing machines. These turn out to be nearly equivalent. Section 3 gives examples which are log depth complete for deterministic log space, and hence may distinguish between two similar classes: deterministic log space and uniform log circuit depth. Section 4 discusses two fixed structure models useful for considering hardware size as well as parallel time; namely, conglomerates and aggregates. Section 5 introduces hardware modification machines, and section 6 surveys other modifiable parallel models, such as vector machines and parallel RAM's. Section 7 discusses characterizations and interrelationships between two complexity classes defined by simultaneous resource bounds; namely, NC and SC Finally, section 8 lists some open problems.

2. Circuits and Alternating Turing Machines

Perhaps the simplest model for measuring the parallel time to compute a function is the combinational circuit (or simply a circuit). (See [S3] and [P2] for general discussions of circuits.)

Notation. $B_n = \{f \mid \{0, 1\}^n \to \{0, 1\}\}$ = the set of all Boolean functions of rank n.

Definition. A *circuit* α *with* n *inputs* is a finite directed acyclic graph such that each node has a label from $\{x_1, ..., x_n\} \cup B_0 \cup B_1 \cup B_2$. A node labelled x_i must have indegree zero, and is called an *input* node. A node v with label $g \in B_i$ must have indegree i, and one edge into v is associated with each argument of g. Certain nodes are designated *output* nodes. When the variables x_i are assigned values from $\{0, 1\}$ every node v assumes a unique value in $\{0, 1\}$, so that v computes some function f_v of $x_1, ..., x_n$. We say the circuit α *computes* f if $f = f_v$ for some output node v.