2. Circuits and Alternating Turing Machines

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.

We shall assume that every node v has a path from v to some output. That is, we assume there are no syntactically superfluous nodes.

Let $c(\alpha)$ (the complexity of α) be the number of *gates* (i.e. nodes other than inputs) in α, and let $d(\alpha)$ (depth of α) be the length of the longest path in α. If $f \in B_n$, then $c(f) = \min\{c(\alpha) \mid \alpha \text{ computes } f\}$ and $d(f) = \min\{d(\alpha) \mid \alpha \text{ computes } f\}$.

If $A \subseteq \{0, 1\}^*$, then $A^n = A \cap \{0, 1\}^n$. We can regard A^n as a member of B_n by the convention $A^n(x_1, ..., x_n) = 1$ iff $(x_1 \ldots x_n) \in A^n$. A family $\{\alpha_n\}$ of circuits *computes* A iff α_n computes A^n for all n, and each α_n has a unique output node.

Notation. Let $S, T : N^+ \to R$. Then

$$\text{SIZE}(T) = \{A \mid \exists \{\alpha_n\} : \{\alpha_n\} \text{ computes A and } c(\alpha_n) = O(T(n))\}$$
$$\text{DEPTH}(S) = \{A \mid \exists \{\alpha_n\} : \{\alpha_n\} \text{ computes A and } d(\alpha_n) = O(S(n))\}$$

We shall always assume $T(n) \geqslant n$ and $S(n) \geqslant \log n$.

These complexity classes are strange in that they include nonrecursive sets A. In fact, by Lupanov's result (see [S3]) $\text{SIZE}(2^n/n) = 2^{\{0,1\}^*}$, and by disjunctive normal form $\text{DEPTH}(n) = 2^{\{0,1\}^*}$. Nevertheless they are mathematically interesting, and have intuitive significance especially for lower bound results. In particular, a proof that $A \notin \text{DEPTH}(S)$ means that no parallel computer with fixed circuitry could compute A in time O(S). This is because the parallel computation could be unwound to form a circuit with constant delay at each gate. Our assumption that circuits have bounded fan-in (in fact fan-in two) is justified by engineering experience that any general design for a gate with n inputs has a delay at least proportional to $\log n$. On the other hand, Hoover [H1] gives results that show that an assumption of fan-*out* two would not materially alter either the depth or the size complexity of a set A.

Although the circuit depth to compute A is a reasonable lower bound on the parallel time required, it is not a reasonable upper bound in general (unless we want parallel machines to compute nonrecursive sets). Borodin [B1] proposed making it reasonable by requiring that the family $\{\alpha_n\}$ computing A be uniform in some sense. The trouble is there is no clearly correct choice for the definition of *uniform*. (See Ruzzo [R1] for a discussion of various possibilities.) Here we shall adopt the following definition, which has gained some acceptance (see [C1], [R1], and [P1]):

Definition. A family $\{\alpha_n\}$ of circuits is *uniform* provided some deterministic Turing machine can compute the transformation $1^n \to \bar{\alpha}_n$ in space

$O(\log c(\alpha_n))$. (Here $\bar{\alpha}_n$ is a binary string coding the circuit α_n in some reasonable fashion.)

We can now define the uniform complexity classes

$$\begin{aligned}\text{USIZE (T)} &= \{A \mid \exists \text{ uniform } \{\alpha_n\} : \{\alpha_n\} \text{ computes A and } c(\alpha_n) \\ &= O(T(n))\} \\ \text{UDEPTH (S)} &= \{A \mid \exists \text{ uniform } \{\alpha_n\} : \{\alpha_n\} \text{ computes A and } d(\alpha_n) \\ &= O(S(n))\}\end{aligned}$$

Notice that the size $c(\alpha_n)$ is not mentioned in the definition of UDEPTH so it can be taken to be as large as possible consistent with $d(\alpha_n)$. In fact, every circuit of depth d with a unique output can be expanded into an equivalent tree circuit of size $2^d - 1$. Also, our assumption of no superfluous nodes implies that no unique-output circuit of depth d can have more than $2^d - 1$ nodes. This leads to the following

PROPOSITION 2.1. *The class* UDEPTH (S) *remains unchanged if the definition of uniform family* $\{\alpha_n\}$ *is changed to require that the transformation* $1^n \to \bar{\alpha}_n$ *be computable in deterministic space* $O(d(\alpha_n))$ *instead of* $O(\log(c(\alpha_n)))$.

The alternative definition of uniform is in fact the one given by Borodin [B1].

Borodin expresses the general thesis in [B1] that circuit size corresponds to Turing machine time and circuit depth corresponds to Turing machine space. (If we identify uniform circuit depth with parallel time, then the second assertion is an instance of the parallel computation thesis stated in section 1.) One precise statement of Borodin's thesis is the following:

THEOREM 2.1. *If* [log T] *is fully space constructable, then*

$$\text{USIZE}(T^{O(1)}) = \text{DTIME}(T^{O(1)}).$$

If S *is fully space constructable, then*

$$\text{UDEPTH}(S^{O(1)}) = \text{DSPACE}(S^{O(1)}).$$

(See [HU] for the definitions of *constructable*, *DTIME* and *DSPACE*. We have altered the definitions of the latter so they contain only subsets of $\{0, 1\}^*$.)

The first equation is easy in this crude form, and in fact can be made considerably more precise (see [P1]).

The second equation is a consequence of the following result of Borodin [B1], which states that the inclusions (1.1) hold for uniform circuit depth.

THEOREM 2.2. *If* S *is fully space constructable, then*

$$\text{UDEPTH (S)} \subseteq \text{DSPACE (S)}, \quad \textit{and}$$
$$\text{NSPACE} \quad \subseteq \text{UDEPTH } (\text{S}^2).$$

COROLLARY (Savitch's Theorem). *If* S *is fully space constructable, then* NSPACE (S) $\subseteq$ DSPACE (S^2).

Let us sketch the proof of theorem 2.2. The *circuit value problem* (see [HU]) is the set of all binary strings encoding systems $\langle x_1, \ldots, x_n; \alpha \rangle$ where each $x_i \in \{0, 1\}$ and α is a circuit whose unique output is 1 when its n inputs take on the values $x_1, \ldots, x_n$.

LEMMA 2.1. *There is a deterministic Turing machine* M *which recognizes the circuit value problem, and on an input encoding* $\langle x_1, \ldots, x_n; \alpha \rangle$, M *uses space* $\text{O}(d(\alpha))$.

The idea is to perform a depth first search of α from the output node taking left descendants first. M stores the number of the node v currently examined, together with one symbol for each node on the path followed from the root to v. This symbol is either a marker L, if the search is proceeding on through the left input of the node; or the value of the left input if this value has been determined and the search is proceeding on to the right.

The first inclusion of Theorem 2.2 follows from Lemma 2.1 and Proposition 2.1.

To prove the second inclusion, recall that the graph reachability problem (GRP) (see [HU]) is the set of all binary strings encoding the adjacency matrix of a digraph G on nodes $\{1, 2, \ldots, \text{N}\}$ such that G has a path from node 1 to node N.

LEMMA 2.2. GRP $\in$ UDEPTH ($\log^2 n$).

The proof involves constructing a circuit which computes the transitive closure of a Boolean matrix by repeated squaring. The circuit has O ($\log n$) stages, and each stage has depth O ($\log n$) and computes the Boolean square of the matrix resulting from the previous stage. The circuit can be constructed by a deterministic Turing machine in space O ($\log n$).

Given a nondeterministic S space bounded Turing machine M and the input length n, a circuit α_n is constructed which does the following on an

input string w of length n. α_n first computes the adjacency matrix A of the graph whose nodes are the possible configurations of M with an input of length n, and whose edges represent possible steps in a computation with input w. We can assume M has an initial configuration labelled 1 and a unique accepting configuration labelled N. α_n now solves the graph reachability problem for A according to Lemma 2.2. The solution to the problem is positive iff M accepts w. Using Lemma 2.2 it is not hard to see that α_n has depth $O(S^2)$ and can be constructed in deterministic space $O(S^2)$ (in fact, space $O(S)$).

Theorem 2.1 represents one way to make precise Borodin's thesis that size corresponds to time and depth to space. Alternatively, instead of making the circuit family $\{\alpha_n\}$ uniform one can make Turing machines nonuniform (see [S1]). We borrow from Pippenger's terminology [P1]. Suppose $g : \{0, 1\}^* \to \{0, 1\}^*$. We say that a (deterministic or nondeterministic) multitape Turing machine *accepts* A *modulo* g provided that M accepts A under the condition that in addition to the normal input $x \in \{0, 1\}^*$ on a read only input tape M is also provided with $g(x)$ on a separate read only tape called the *reference* tape. The space used by M is the work tape space plus $\lceil \log |g(x)| \rceil$, where $|w|$ is the length of w. (The term $\lceil \log |g(x)| \rceil$ was not counted in [P1], but it should be, since it represents the amount of information stored by the position of the head on the reference tape.) The function g is *length determined* if $g(x)$ depends only on $|x|$, and not otherwise on x. A *nonuniform* machine is a machine M together with a length determined function g. It accepts A provided it accepts A modulo g. We add (NONUNIFORM) after a complexity class to indicate the machines are allowed to be nonuniform.

There is an alternative and more elegant definition of nonuniform space. We say that A is in DSPACE (S) (NONUNIFORM) provided there is a family $\{F_n\}$ of finite automata, each with a two-way read only input tape, such that F_n recognizes A^n, and $\log |F_n| = O(S(n))$, where $|F_n|$ is the number of states of F_n. It is not hard to verify that this definition is equivalent to the one in the previous paragraph (recall our convention that $S(n) \geqslant \log n$).

The above definition does not work for time. However, Les Valiant pointed out that we could change the definition of nonuniform Turing machine to be a family $\{M_n\}$ of Turing machines instead of a single Turing machine with a reference tape. The time complexity $T(n)$ of such a family would be the maximum of $|M_n|$ and the worst case running time of M_n on inputs of length n. The space complexity $S(n)$ would be $\log |M_n|$ plus

the worst case space used by M_n on inputs of length n. This gives the same definition of nonuniform space as before, but the nonuniform time is only the same up to application of a polynomial.

In any case, theorems 2.1 and 2.2 have the following analogs for nonuniform machines:

THEOREM 2.3.

$$\text{SIZE}(T^{O(1)}) = \text{DTIME}(T^{O(1)}) \text{ (NONUNIFORM), } \textit{and}$$
$$\text{DEPTH}(S^{O(1)}) = \text{DSPACE}(S^{O(1)}) \text{ (NONUNIFORM).}$$

THEOREM 2.4.

$$\text{DEPTH}(S) \subseteq \text{DSPACE}(S) \text{ (NONUNIFORM), } \textit{and}$$
$$\text{NSPACE}(S) \text{ (NONUNIFORM)} \subseteq \text{DEPTH}(S^2).$$

To prove these results, the nonuniform machines simulate the circuits by letting $g(x)$ provide a description of the circuit for inputs of length $|x|$. Conversely, a circuit family $\{\alpha_n\}$ can simulate a nonuniform machine by building into α_n the value of $g(x)$ for $|x| = n$.

Note that the following nonuniform version of Savitch's theorem is a consequence of Theorem 2.4:

COROLLARY.

$$\text{NSPACE}(S) \text{ (NONUNIFORM)} \subseteq \text{DSPACE}(S^2) \text{ (NONUNIFORM).}$$

In other words, a 2^s-state 2NFA can be simulated by a $2^{O(s^2)}$-state 2DFA for inputs of length $n \leqslant 2^s$.

A second interesting model of parallel computation, which falls in the fixed structure category, is the *alternating Turing machine* (ATM) ([CS], [K1], [CKS]). An ATM is a generalization of a nondeterministic multitape Turing machine. A nondeterministic machine has *existential* states, for which there are several possible next states, and at least one of the alternatives must lead eventually to an accepting state. In addition to existential states, an ATM also has *universal* states, for which *all* of the possible next states must lead to an accepting state. We define the accepting state to be a universal state with no successors. Every state is either universal or existential. Thus an *accepting computation* of an ATM M *with input* w is a finite tree whose nodes are labelled with configurations of M, such that i) every universal node (i.e. node whose configuration has a universal state) must have all possible next configurations as children, ii) every existential node

must have at least one possible next configuration as a child, and iii) the root is the initial configuration. In order for M to operate in sublinear time we assume it has "random access" to the bits of w instead of a read only input tape. That is, M has a special index tape, and when M writes an index i on the index tape and assumes one of a distinguished set of index states, the i-th symbol of the input w is placed on the index tape. We say M *accepts* w in time s and space l if there is an accepting computation of M with input w whose longest path from root to leaf is s or less, and such that no configuration in the computation has tapes of length exceeding l. The complexity classes for time and space for ATM's are designated ATIME (S) and ASPACE (L), respectively, and we always assume $S(n), L(n) \geqslant \log n$.

As different as ATM's may seem from uniform circuit families, there is a remarkably close correspondence between alternating time and circuit depth, and between alternating space and circuit size. Unfortunately, our definition of *uniform* for circuits is too weak to express the correspondence precisely. Ruzzo gives a number of alternative definitions, of which the strongest is the following: $\{\alpha_n\}$ is U_E *uniform* iff the connection language L_{EC} can be recognized by a deterministic Turing machine in time $O(\log c(\alpha_n))$. Here L_{EC} consists of those quadruples (n, g, p, x) such that if g' is the gate reached by following the path $p \in \{L, R\}^*$ (where $|p| \leqslant \log c(\alpha_n)$) in circuit α_n back from gate g (L, R refer to left and right input, respectively) then g' has label x if $x \in B_2$, and $g' = x$ otherwise. (Assume n, g and x are expressed in binary notation.)

If we use the notation, for example, U_EDEPTH (S) to indicate this notion of uniformity, then we have

$$\text{ATIME (S)} = U_E\text{DEPTH (S)}, \quad \text{and}$$
$$\text{ASPACE (L)} = U_E\text{SIZE } (2^{O(L)}),$$

assuming S (n) can be computed in deterministic time S (n), and L (n) can be computed in deterministic time L (n) given n in binary notation. In fact, Ruzzo [R1] proves the stronger result that the equivalences hold simultaneously. Let us use the notation ATIME-SPACE (S, L) for the class of sets accepted simultaneously in time S and space L on an ATM, (note that this may be a proper subset of the intersection of ATIME (S) and ASPACE (L)), and analogous notation for other simultaneous classes. Then

THEOREM 2.5. ATIME-SPACE (S, L) = U_EDEPTH-SIZE (S, $2^{O(L)}$), *provided* S *and* L *are computable in deterministic time* O (S).

Ruzzo shows the above result still holds when U_E is replaced by U, provided $S \geqslant L^2$.

From their definition, ATM's appear to model a restricted form of parallel computation, because the "processors" in the model are restricted to be Turing machines, and they must be organized in the form of an and-or-tree. This makes Theorem 2.5 all the more interesting. On the other hand, ATM's are more pleasing in one way than circuit families, because there is no question of how to define *uniform*. Each ATM is automatically uniform. In fact, ATM's may be the best candidate proposed so far for defining parallel time, at least in the fixed structure category. But this remains to be seen. The one clear drawback of ATM's is that they do not seem to have any resource that corresponds to hardware size (see section 4).

3. Log Depth vs Log Space

As far as we know, the second inclusions in Theorems 2.2 and 2.4 cannot be improved, even when NSPACE is replaced by DSPACE. (Of course an improvement for NSPACE would improve Savitch's theorem.) Taking $S(n) = \log n$ as the most basic case, it is interesting to look for examples of sets in DSPACE $(\log n)$ which do not appear to be in DEPTH $(\log n)$. Addition of n n-digit binary numbers, and multiplication of two n-digit binary numbers both can be done in $O(\log n)$ circuit depth (see [S3]), as can sorting n n-digit binary numbers (see [MP]). On the other hand, the "cycle free problem" is in DSPACE $(\log n)$ but does not appear to be in DEPTH $(\log n)$.

Definition. The cycle free problem (CFP) is the set of all binary codes for symmetric Boolean $N \times N$ adjacency matrices A of undirected cycle-free graphs.

One can define functions $f: \{0,1\}^* \to \{0,1\}^*$ computable in depth S (or uniform depth S) using circuits with several outputs. We say A_1 is *log depth reducible* to A_2 (respectively *uniformly log depth reducible*) iff there is some function f computable in depth $O(\log n)$ (respectively uniform depth $O(\log n)$) such that $w \in A_1$ iff $f(w) \in A_2$, for all w. We say A is *log depth complete* for the class $\mathcal{S}$ iff $A \in \mathcal{S}$, and every $A' \in \mathcal{S}$ is log depth reducible to A. The uniform case is defined similarly. The main ideas in the proof of the following result appear in Hong [H2].