数理解析研究所講究録
第522巻 1984年 62-76

On Depth Complete Problems

安浦 寛人
Hiroto Yasuura

京都大学 工学部
Faculty of Engineering, Kyoto University

1.Introduction

It is one of the most fundamental questions in the theory of parallel computation how much the computation speed is accelerated by parallel computation scheme. In this paper, we define the log-depth complete problems which might not exponentially accerelate by parallel computation.

For the model of parallel computation, we use combinational logic circuits constructed of fan-in restricted logic elements. The computation time is measured by the depth of the circuits. We mainly discuss the log-depth completeness for the class of problems computable by polynomial size circuits, SIZE(poly(n)). If a problem P is depth complete for SIZE(poly(n)), all problems in SIZE(poly(n)) can be transformed to P by circuits with depth O(log n). Therefore if P can be computed by circuits with depth O(log n), we can conclude that all problems in SIZE(poly(n)) can also be computed in time O(log n). The circuit value problem introduced by Ladner [1] is clearly log-depth complete for SIZE(poly(n)). We show that several problems which are known as

log-space complete for DTIME(ploy(n)) [2] are also log-depth complete for SIZE(poly(n)).

2.Computation Models and Complexity Classes

We assume that the reader is familiar to the time/space complexity theory of Turing machines. Our Turing machine model is an "off-line" and "deterministic (or nondeterministic)" machine with a two-way read only input tape and an arbitrary but finite number of work tapes. We sometimes refer DTIME(T(n)), DSPACE(S(n)) and NSPACE(S(n)) as a set of decision problems on {Ø,1} computed by T(n)-time bounded deterministic Turing machines (DTM's), by S(n)-space bounded DTM's and by S(n)-space bounded nondeterministic Turing machines (NTM's), respectively.

In this paper, we adopt combinational logic circuits as a model of parallel computation. Combinational logic circuits are the most fundamental components of digital systems and many researches have been carried out on the complexity theory of logic functions realized by combinational circuits[3]. There are two measures to evaluate the complexity of circuits, size and depth, which correspond to the number of resources and the computation time spent in the computation respectively.

A basis is a finite set of logic functions. A combinational logic circuit C over a basis B is a labeled acyclic directed graph. Vertices with indegree Ø are input vertices each of which

is labeled with an element in $\{x_1, x_2, \ldots, x_n, \emptyset, 1\}$. Output vertices each of which is labeled with an output variable have indegree 1. Other vertices are computation ones each of which is labeled with a logic function in B. Indegree of a vertex labeled with an i-variable function f is just i. These edges into a computation vertex are linearly ordered from 1 to i.

Each vertex represents a logic function of variables and constants in $\{x_1, x_2, \ldots, x_n, \emptyset, 1\}$. An input vertex with label x represents a logic function x. A computation vertex v labeled with i-variable function f represents a function $f(p_1, p_2, \ldots, p_i)$, where p_j $(j=1,2,\ldots,i)$ is a function represented by a vertex u such that the edge $e=(u,v)$ is the j-th edge into v. The function represented by an output vertex v is the same function of vertex u such that the edge (u,v) is in C (i.e., v is adjacent to u).

[Definition 1] For a function $f:\{\emptyset,1\}^n \rightarrow \{\emptyset,1\}^m$, we say a combinational logic circuit C computes f, if there exists a labeling of input vertices and a set of output vertices $V=\{v_1, v_2, \ldots, v_m\}$ in C such that f equals to the set of logic functions represented by vertices in V.

Levels of computation vertices in a circuit C are defined in the following way: the level of each input vertex is $\emptyset$; the level of a computation vertex v is one greater than the maximum of the levels of the vertices to which v is adjacent.

The size of a combinational circuit C, denoted size(C), is

the number of computation vertices in C. The depth of C, denoted depth(C), is the maximum level of the computation vertices in C. When we assume that the delay of computation only depends on the delay in each computation vertex (gate) and they have same value, depth(C) is linearly proportional to the delay of computation on C.

A basis B is said to be complete if for any logic function f there exists a circuit on basis B which computes f. The combinational (or circuit) complexity of a function f relative to a basis B, denoted $C_B(f)$, is the size of the smallest size circuit over B that computes f. The delay complexity of f relative to B, denoted $D_B(f)$, is the depth of the smallest depth circuit over B that computes f. For any complete bases B and B', there are positive constants c and d such that

$$C_B(f) \leq cC_{B'}(f)$$

and

$$D_B(f) \leq dD_{B'}(f)$$

for every function f. It is well known that for all n-variable functions

$$C_B(f) = O(2^n/n)$$

and

$$D_B(f) = O(n)$$

for any complete basis B. Assuming that B is complete, we sometimes omit the suffix B from C_B and D_B notations.

We can consider complexity classes of problems on the combinational logic circuits. For simplicity, we only consider decision problems on alphabet {0,1}. Let P be a decision problem

on $\{\emptyset,1\}$. P_i denotes a subproblem of P with length n, namely, $P_n=P\cap\{\emptyset,1\}^n$.

Thus P_n can be considered as an n-variable logic function. The parallel computational complexity of P is defined by asymptotic behavior of the complexities $C_B(P_n)$ and $D_B(P_n)$ for a complete base B.

We define complexity classes related with size and depth of logic circuits. We use the following notations:

$SIZE(C(n))=\{P|$ For each n, $C(P_n)=O(C(n))\}$

$DEPTH(D(n))=\{P|$ For each n, $D(P_n)=O(D(n))\}$

We use the notation of polynomial complexity such as

$$SIZE(poly(n))=\bigcup_{i=1,2,\ldots} SIZE(n^i).$$

We can define completeness on the delay complexity, i.e., depth completeness, as well as log-space completeness in the sequential computation.

[Definition 2] For a positive integer k, a problem P is said to be $\log^k$-depth complete for a class of problems A iff

(1) P is in A,

and

(2) for any Q in A, there is a set of functions $\{f_n|\ f_n: \{\emptyset,1\}^n \to \{\emptyset,1\}^{p(n)}\times\{\emptyset,1,\ \ldots,\ p(n)\}\}$ and a polynomial p(n) such that f_n transforms Q_n to one of $\{P_1,\ P_2,\ \ldots,\ P_{p(n)}\}$ and $D(f_n)=O(\log^k n)$. Namely, for a string w in $\{\emptyset,1\}^n$, if $f_n(w)=(w',\ z)$, then w is in Q iff substr(w', 1, z) is in P, where $substr(w,\ i,\ j)=a_ia_{i+1}\ldots a_j$ for $w=a_1a_2\ldots a_n$ and $\emptyset\leqq i\leqq j\leqq n$. We sometimes say that P is 'uniformly' $\log^k$-depth complete for A, if a circuit C_n in a set

$\{C_n \mid C_n$ computes f_n and depth$(C_n)=O(\log^k n)\}$ is generated from each n by a (log n)-space bounded Turing machine[4].

We sometimes use 'log-depth complete (or depth complete)' rather than '$\log^1$-depth complete'. Of cause, if P is $\log^k$-depth complete for A for any k, then P is also $\log^{k'}$-depth complete for A for every k' greater than k.

Ladner introduced the circuit value problem (CVP) which is clearly depth complete for SIZE(poly(n)).

(Ø) CVP (the Circuit Value Problem)
Given: A combinational circuit C and an input x to C.
To determine: Whether C outputs 1 for x.

[Theorem 1] CVP is log-depth complete for SIZE(poly(n)).
(Proof) Let P be a problem in SIZE(poly(n)). Then there are a polynomial p(n) and a set of circuits $\{C_n\}$ such that C_n computes P_n and size$(C_n)<p(n)$ for each n. Since C_n can be coded on {Ø,1} with length O(p(n) log n), it is sufficient for reduction from P_n to CVP to generate a code representing C_n. The code generation requires no computation vertex, because C_n is fixed for each n. Then CVP is log-depth complete for SIZE(poly(n)). Q.E.D.

Borodin showed that if a problem P is log-space complete for A then P is also uniformly $\log^2$-depth complete for A. Thus he claimed that a problem P which is log-space complete for DTIME(poly(n)) is also uniformly $\log^2$-depth complete for

DTIME(poly(n))[4]. In this paper, we will show that several problems which are known as log-space complete for DTIME(poly(n)) are also log-depth complete for SIZE(poly(n)).

3.The Problems

In this section, we list the problems to be shown depth complete for SIZE(poly(n)).

(1) DHGAP (the Directed HyperGraph Accessibility Problem)

Given: A directed hypergraph $H=(V,E)$, a set of vertices S in V and a vertex v in V.

To determine: Whether v is reachable from S.

[Definition: V is a set of vertices and E is a set of directed hyperedges. A hyperedge e is an ordered pair of a set of vertices V_e in V and a vertex v_e in V, denoted (V_e,v_e). In H, a vertex v is reachable from a set of vertices S iff v is included in S or there exists a hyperedge $e=(V_e,v)$ such that all vertices in V_e are reachable from S.]

(2) UNIFICATION[5]

Given: A term graph $G=(V,E)$ and two vertices v_1 and v_2 in V.

To determine: Whether terms t_1 and t_2, represented by v_1 and v_2 respectively, are unifiable.

[Definition: A term graph $G=(V,E)$ is a directed acyclic graph. Each vertex has a label in a set of i-adic function symbols A_i or a set of variables X. The outdegree of a vertex with a label in X is Ø and no two vertices have a same label in X. A vertex v with a label f in A_i ($i \geq 0$) has i outgoing edges. v represents a term $f(t_1, t_2, \ldots, t_i)$ where t_j is a term represented by a vertex u such that (v,u) is the j-th outgoing edge of v. Terms t_1 and t_2 are said to be unifiable iff there is a substitution s such that $t_{1s}=t_{2s}$.]

(3) GEN

Given: A set X, a binary operation * on X, a subset S of X, and an element x in X.

To determine: Whether x is contained in the smallest subset of X which contains S and is closed under *.

(4) PATH[6]

Given: A path system (X,R,S,T) where S and T are subsets of X, and R is in $X \times X \times X$.

To determine: Whether there is an admissible node in S.

[Definition: x is admissible iff x is in T or there exist y and z in X such that (x, y, z) is in R and both y and z are admissible.]

4. Depth Completeness of DHGAP

In this section, we prove depth completeness of DHGAP defined above.

[Lemma 1] DHGAP is in SIZE(poly(n)).

(Proof) We first show a polynomial time algorithm on a DTM for DHGAP (H, S, v). For a given hypergraph H=(V,E), we first mark all vertices in S. We make a new subset S_1 of V from S. Initially, let S_1 be S. Scanning all hyperedges, we add a vertex u into S_1 if there is a hyperedge (V_e,u) in E and V_e is included in S. Next we make S_2 from S_1 by the same manner. Repeating the operation, we have S_m, where m is the number of hyperedges. If S_m contains v, then v is reachable from S. This algorithm clearly requires time at most $O(m^6)$ steps on a DTM. Since SIZE(poly(n)) contains DTIME(poly(n)), DHGAP is in SIZE(poly(n)). Q.E.D.

Now we show that DHGAP is depth complete for SIZE(poly(n)). Namely, DHGAP is the hardest problem in SIZE(poly(n)) as well as CVP.

[Theorem 2] DHGAP is log-depth complete for SIZE(poly(n)).

Before proving Theorem 2, we prepare several terms and a lemma. For a hyperedge $e=(V_e, v_e)$, we define the rank of e by the number of elements in V_e. The rank of hypergraph H=(V,E) is also defined by the maximum rank of all hyperedges in E. A directed hypergraph

H is called AND-OR graph, if any hyperedges $e=(V_e, v_e)$ with rank more than 1 does not share the vertex v_e as the second element with other hyperedges. Namely, if the indegree of a vertex v is more than 1, all hyperedges pointing v are simple edges (i.e., the rank is 1). Such vertices are called an OR-nodes and others are called an AND-nodes.

[Lemma 2] For any combinational logic circuit C over a basis {2-AND, NOT} with a single output vertex v and for any input vector $(x_1, x_2, \ldots, x_n)$ for C, there is an acyclic hypergraph $H=(V,E)$ with rank 2, a subset of vertices S and a vertex v_1 in V such that v_1 is reachable from S iff C outputs 1 for the input vector.

(Proof) We construct a hypergraph H as follows:

(1) First, for each vertex u in C, place two vertices $u_\emptyset$ and u_1 in H.

(2) If u is a computation vertex with label 2-AND and there are edges (u',u) and (u",u) in C, make hyperedges $(\{u_\emptyset', u_\emptyset"\}, u_\emptyset)$, $(\{u_\emptyset', u_1"\}, u_\emptyset$, $(\{u_1', u_\emptyset"\}, u_\emptyset)$ and $(\{u_1', u_1"\}, u_1)$ in H.

(3) If u is a computation vertex with label NOT and there are edge (u',u) in C, make edges $(u_\emptyset', u_1)$ and $(u_1', u_\emptyset)$.

(4) For the output vertex v in C, if u is adjacent to v then make edges $(u_\emptyset, v_\emptyset)$ and (u_1, v_1). The terminal vertex in DHGAP is v_1.

(5) Suppose that w is an input vertex with label x_i. When $x_i=1$ in the given input vector, let w_1 be in the initial set S. When $x_i=\emptyset$, let $w_\emptyset$ be in S.

It is clear that v_1 is reachable from S if and only if C outputs

1 for a given input vector. Note that the number of vertices and hyperedges in H is not greater than 2(size(C)+n+n'+1) and 2(2size(C)+1), respectively. Here n and n' are the numbers of input vertices with variable labels and of ones with constant labels, respectively. Q.E.D.

Now we will return to the proof of Theorem 2.

(Proof of Theorem 2) Since DHGAP is in SIZE(poly(n)), we only show that any problem Q in SIZE(poly(n)) can transform into P by circuits with depth O(log n). For any positive integer n, there exists a circuit with size not greater than p(n) which computes Q_n. Thus from Lemma 2 we can make a hypergraph with size O(p(n)) for any Q_n. For a given input w for Q_n, we should only generate the initial set S of DHGAP. Clearly S can be computed from w by a circuit with constant depth. Q.E.D.

Observing the proof of Lemma 2, we only used DHGAP on hypergraphs with rank 2. Moreover, it is easy to convert the hypergraph in the proof to an AND-OR graph with rank 2 [5]. Then the depth completeness is remained even if the hypergraph is an AND-OR graph with rank 2.

[Corollary 1] DHGAP on AND-OR graphs with rank 2 is still log-depth complete for SIZE(poly(n)).

DHGAP is a generalized problem of GAP (the Graph

Accessibility Problem) on usual directed graphs, because a directed graph is just a directed hypergraph with rank 1. Reducing the simulation of (log n)-space bounded NTM's and DTM's to GAP and DGAP (GAP on a directed graph in which the outdegree of each vertex is not more than 1), respectively, we can show that GAP and DGAP are uniformly log-depth complete for NSPACE(log n) and DSPACE(log n), respectively [3][4]. As a generalization of this technique, we can show that DHGAP is uniformly log-depth complete for DTIME(poly(n)) by reducing the simulation of (log-n)-space bounded alternating Turing machines [7].

5. Depth Completeness of Other Problems

Using Corollary 1, we can directly proved that UNIFICATION, GEN, and PATH are also log-depth complete for SIZE(poly(n)). Since these problems are in DTIME(poly(n)), it is clear that they are in SIZE(poly(n)).

[Theorem 3] UNIFICATION is log-depth complete for SIZE(poly(n)). (Proof is included in [5])

[Theorem 4] GEN is log-depth complete for SIZE(poly(n)).
(Proof) Let a hypergraph H=(V,E) with rank 2, a set S in V and a vertex v in V be given. Define a binary operation * on V as follows: for x, y and z in V, x*y=z iff there is a hyperedge ({x,

y}, z). Clearly, v is reachable from S iff v is contained in the smallest subset of V contains S and is closed under *. Q.E.D.

[Theorem 5] PATH is log-depth complete for SIZE(poly(n)).
(Proof) Consider the same DHGAP in the proof of Theorem 4. Construct a path system (V, E', {v}, S) where E' includes (x, y, z) iff E includes ({z, y}, x). It is trivial that v is reachable from S iff v is admissible. Q.E.D.

By the similar argument in [2], we can easily show that the following problems are also log-depth complete for SIZE(poly(n)) by reduction GEN.

(5) CFØ
Given: A context-free grammar G=(N,T,P,S).
To determine: Whether L(G)=ø.

(6) CF
Given: A context-free grammar G=(N,T,P,S).
To determine: Whether L(G) is infinite.

(7) CFMEMBER
Given: A context-free grammar G=(N,T,P,S) and x in T^*.
To determine: Whether x is in L(G).

Since DHGAP on AND-OR graph is also log-depth complete for SIZE(poly(n)) from Corollary 1, it is easy to show directly from

this fact that GAME [2] is log-depth complete for SIZE(poly(n)).

Acknowledgment

The author expresses sincere appreciation to Professor Shuzo YAJIMA of Kyoto University for his suggestions and critisisms. He also thanks Associate Professor Yahiko KAMBAYASHI, Dr. Hiromi HIRAISHI and Mr. Naofumi TAKAGI in Kyoto University to their comments and discussions on directed hypergraphs. Thanks are also due to Dr.Kenichi HAGIHARA in Oosaka University. This work is partially supported by grant in aid for scientific research no.58460135.

References

[1] Ladner, R. E., "The circuit value problem is log-space complete for P", SIGACT news, vol.7, no.1, 1975.

[2] Jones, N.D. and Laaser, W.T., "Complete problems for deterministic polynomial time", Theoretical Computer Science, vol.3, no.1, 1976.

[3] Yasuura, H., "Studies on complexity theory of logic circuits and hardware algorithms for VLSI systems", Doctorial Dissertation of Kyoto Univ. 1982.

[4] Borodin,A., "On relating time and space to size and depth", SIAM J.Computing, vol.6, no.4, 1977.

[5] Yasuura, H., "On the parallel computational complexity of unification" Tech. Rep. of IECEJ, vol.83, no.163, AL83-30, 1983. also available as ICOT Technical Report TR-027.

[6] Cook, S.A., "Path systems and language recognition", Proc. of 2nd Annual ACM Symp. on Theory of Computing, 1970.

[7] Yasuura, H., "The Reachability Problem on Directed Hypergraphs and Computational Complexity", Tech. Rep. of IECEJ, vol.83, no.184, AL83-42, 1983.