INFORMATION AND CONTROL 71, 87–94 (1986)

An Exponential Lower Bound for Real-Time Branching Programs

STANISLAV ŽÁK

Institute for Computation Techniques, Technical University, Horská 3, 128 00 Praha 2, Czechoslovakia

Received September 1985; accepted March 3, 1986

Branching programs are a general model of sequential computation. One of their computational features is their possibility to question (repeatedly) the information from each input bit. Real-time branching programs make at most n questions when computing on an input of length n. The restriction "real-time" allows to find a simple language which requires the lower bound $2^{\sqrt{2n}/8}$ on memory (= the state space). © 1986 Academic Press, Inc.

INTRODUCTION

By a branching program we mean an oriented acyclic finite graph with the following properties:

(a) There is exactly one source.

(b) Every node has outdegree at most 2.

(c) Every node with outdegree 2 is labelled by a number i, $1 \leqslant i \leqslant n$, one of the out-edges is labelled by 0 and the other one by 1.

(d) Every sink is labelled by 0 or 1.

Such a branching program P computes a Boolean function f_P, $f_P: \{0,1\}^n \to \{0,1\}$, as follows: Given an input $a = a_1 \cdots a_n \in \{0,1\}^n$.

The computation starts at the source. If the computation has reached a node v and only one edge leaves v, then the computation proceeds via this edge. If 2 edges leave v and v is labelled by i, then the computation proceeds via the edge whose label equals to a_i. Once the computation reaches a sink, the computation ends and $f_P(a)$ is defined to be the label of that sink. Branching programs are known for a long time. Independently they were introduced by the present author in Pudlák and Žák (1982) and Žák (1983); as a generalization of sublinear space bounded computations on Turing machines, with the aim to prove lower space bounds on *Tm*'s. Let us briefly describe this idea.

0019-9958/86 $3.00

Assume, we have a Turing machine with a unique input tape. The input head is two-way and read-only. Further the machine may have worktapes, oracle tapes, pushdowns, stacks and so on. The corresponding branching program for input words of length n is constructed as follows. By a configuration we shall mean the total state of the machine except of the content of the input tape. The initial configuration means the state when the input head is at the leftmost cell of the input tape, the finite control is in the initial state and the work-storages are empty. The configurations which are reachable from the initial configuration each during a computation on a word of length n will be the nodes of the constructed branching program. Edges are given by the transition function of the machine. Out-degree is at most 2, since in a configuration the next action depends on the content -0 or $1-$ of the cell scanned by the input head. The initial configuration is the source. Each branching node is labelled by the position of the input head. Moreover we suppose that the machine is space bounded—therefore the resulting graph is finite—and that it has a time counter—therefore the graph is acyclic. The sinks are labelled according to the state of the finite control—if the state is rejecting then the sink is labelled by 0, and by 1 otherwise.

Now, it is clear that each $S(n)$-space bounded Turing machine with Q states, one one-headed worktape with m work symbols, can be simulated by a sequence of branching programs $\{P_n\}$ such that P_n is responsible for computation on words of length n and P_n has at most $C(n) = n \cdot Q \cdot m^{S(n)} \cdot S(n)$ nodes. For $S(n) \geqslant \log n$, $C(n) \leqslant 2^{c \cdot S(n)}$. From this follows: the language which cannot be computed on branching programs within a bound $C(n) \geqslant n$, cannot be computed on *Tm*'s within the space bound $\log C(n)$. In other words, the lower bound $C(n)$ on branching programs implies the lower space bound $\log C(n)$ on *Tm*'s. The present author thinks that this fact is useful for proving space lower bounds for *Tm*'s.

On the other hand, Pudlák (1982) proved the simulation in the other direction. Any sequence $\{P_n\}$ of branching programs each with at most $C(n) \geqslant n$ nodes can be simulated on a $\log C(n)$ space bounded Turing machine with an oracle where the oracle queries are binary words whose lengths depend on the lengths of inputs in question and are equal at most $\log C(n)$.

THEOREM. *Branching programs of complexity* $C(n)$, $C(n) \geqslant n$, *and* $\log C(n)$*—space and oracle—bounded Turing machines are equivalent.*

R. Aleliunas *et al.* (1979) use sequences of finite automata which are a computing device similar to our branching programs. They proved that the reachability problem for undirected graphs is decidable within the

polynomial complexity (= the number of states). Using the above theorem we may reformulate: the problem is decidable on log—space and oracle—bounded Turing machines (it is not known whether on log space bounded). This allows Pudlák (1982) to prove that the complexity on branching programs and on contact schemes are polynomially related.

The present author shows in Pudlák and Žák (1982) and Žák (1983) that the computation on any branching program has also geometrical aspect. The set of inputs $\{0, 1\}^n$ is understood as the set of vertices of the n-dimensional cube. Each node of a branching program is represented by the set of inputs the computations on them reach it. Therefore each node is represented by a subset of the cube. Further, we see that the computation on the branching program can be understood as an iterative application of two operations

(a) Cutting of a subset of vertices according to a hyperplane perpendicular to an axis,

(b) set-union of the generated subsets.

(One application of the operation (a) corresponds to one branching node.) The aim of such a computation is to separate two sets of vertices—one are the vertices to be accepted and the other the vertices to be rejected.

Pudlák (1982) proved that the complexity of computation on the cube with operations (a), (b), and on the cube with operations a', b, where a' allows not only perpendicular but arbitrary hyperplanes, are polynomially related.

Now, let us turn our attention to lower bounds.

The main goal is to prove a superpolynomial bound on a language which can be accepted within nondeterministic log-space or within polynomial time (i.e., to solve the problems LOG = ?NLOG, LOG = ?P). In general case the largest known bound is $n^2/\log^2 n$-due to Netchiporuk (1966)—and superlinear—due to Pudlák (1984).

Other authors attempt to prove lower bounds for restricted models of branching programs. Borodin *et al.* (1983) introduced width-two branching programs and proved for them an $\Omega(n^2/\log n)$ bound. Yao (1983) announced an exponential lower bound for width-two branching programs computing majority function.

The present author Žák (1983, 1984) introduced another restricted type which is now called one-time-only branching programs due to Wegener (1984). (On such a branching program for each i, $1 \leqslant i \leqslant n$, any computation goes through a branching node labelled by i at most one-time.) The present author proved an exponential lower bound for one-time-only branching program computing a simple language of half-cliques which can be computed within polynomial time. Wegener (1984) proved a slightly

weaker bound for the *NP*-complete language of graphs containing a clique. Moreover Wegener (in press) found a language which is exponentially hard on one-time-only branching programs and only polynomially hard on width-two branching programs—it is a counterpart of the result of Yao mentioned above (the majority function is only polynomially hard on one-time-only branching programs).

A motivation for one-time-only branching programs is as follows. In any general branching program, if a computation goes through a branching node the information about a bit is remembered. On the other hand, the joining of computations can be viewed as (in sense) forgetting of an information. It seems that there are two kinds of information—the atomic one—it is the knowledge about the content of bits of the input, and the general one—it is the knowledge of the kind for example "that part of the graph forms a tree." Then (maybe) the computation on an input is characterized by a continuous growth of the general information and by the remembering and forgetting of the atomic information. In this context, one-time-only branching programs represent the idea that during the growth of the global information each bit of the input is investigated at most one-time. The exponential bound shows that the naïve hope that one-time-only branching programs are optimal is false since there is a simple log-space Turing machine which computes the language of half-cliques investitating each input cell at most two-times. So, according to the above theorem, this language is of only polynomial complexity on two-times-only branching programs. (Two-time-only are better.) Wegener (1984) constructed a two-times-only branching programs of quadratic complexity for half-cliques. There are many questions about hierarchies according the number of allowed investigations.

Another motivation (due to Wegener (1986)) is to consider k-times-only branching programs as a model of time bounded computations. This motivation is followed by Ftáčník and Hromkovič (in press). They introduced real-time bounded branching programs (during a computation on an input of length n only n investigations of bits are allowed) and proved quadratic lower bound. In this paper we prove an exponential lower bound. The witness language is only slightly more complicated then for the case of one-time-only branching programs. The method is similar to those from Žák (1984) , Wegener (1984), and Ftáčník and Hromkovič (in press).

2. Real-Time Branching Programs and Exact Languages

A branching program is called "real-time" if each computation on any input (of length n) goes through at most n branching nodes of the program.

The labels of the branching nodes need not be different, i.e., a bit of the input can be investigated (asked) repeatedly. In this case another bit is not investigated at all.

A language L, $L \subseteq \{0, 1\}^n$, is said to be "exact" if all words from L have the same number of 0's (and 1's).

For any real-time branching program computing an exact language the following three lemma's hold.

LEMMA 1. *Each accepting computation asks each bit of the input exactly one-time.*

Proof. The computation has to ask each bit of the input since two inputs which differ in only one bit have to be separated (the language is exact).

LEMMA 2. *Let two computations meet at a node. Suppose they have asked the same bits and they can be prolonged in accepting computations. Then each content of the remaining bits causes the input is accepted–rejected in both cases simultaneously.*

Proof. Let A be the set of bits asked before the meeting. Let us fix a content of the remaining bits ($\bar{A}$). We have inputs a, b; a (resp. b) corresponds to the first (resp. second) computation on A and to the fixed content on $\bar{A}$. If one from computations on a and b is accepting then after the meeting it can ask only bits from $\bar{A}$ (Lemma 1). However, on $\bar{A}$ is cannot branch with the second computation which is therefore accepting, too.

For the case of meeting of two computations we define the following sets:

$A = \{i/\text{the } i\text{th bit has been asked by the both computations}\}$,

$B = \{i/\text{the } i\text{th bit has been asked by only the first computation}\}$,

$C = \{i/\text{the } i\text{the bith has been asked by only the second computation}\}$,

$D = \{i/\text{the } i\text{the bit has not been asked}\}$.

Lemma 3. *Let two computations meet at a node. Suppose that*

(a) *There are contents of bits non-asked by the first* $(C \cup D)$ *and by the second* $(B \cup D)$ *computations which equal on* D, *and which cause the inputs are accepted or*

(b) *There is a content of bits non-asked by the first computation* $(= C \cup D)$ *which causes the first input is accepted, and which is equal to the second input on* C.

Then before the meeting the computations have asked the same bits ($B = C = \emptyset$).

Proof of the case (a). We have two inputs a, b; a corresponds to the first computation on $A \cup B$, and to the first supposed content on $C \cup D$.

b corresponds to the second computation on $A \cup C$ and to the second supposed content on $B \cup D$. After the meeting the computations on inputs a, b are not allowed to branch on $A \cup B \cup C$ since they are accepting (Lemma 1), and they cannot branch on D since on D a, b are equal. We see that the computation on a (resp. b) is accepting and does not ask on C (resp. B). According to Lemma 1, $B = C = \emptyset$.

Now we prove the case (b). Suppose $B \neq \emptyset$. We have two inputs a, b; a corresponds to the first computation on $A \cup B$, and to the supposed content on $C \cup D$.

b corresponds to the second computation on $A \cup C$, on B b is defined arbitrarily and on D b equals a. The computation on b is also accepting since after the meeting the computations on a, b are not allowed to branch on $A \cup B$ (Lemma 1) and on $C \cup D$ a, b are equal. We see that the computation on b does not ask the bits from B. According to Lemma 1, $B = \emptyset$ — a contradiction.

Now, suppose $B = \emptyset$, $C \neq \emptyset$. Let a, b be inputs as above. The computation on a is accepting and therefore after the meeting it does not ask in A. On C, D, a equals b, hence the computation on a, b do not branch. Therefore the computation on b is accepting, too. We see that after the meeting (a) the computation on a, b must not ask in C, since the (accepting) computation on b has asked in C before the meeting, and (b) the computation in a must ask in C (Lemma 1). A contradiction.

3. An Exponential Lower Bound for Half-Cliques with a Corona

Let us first define the language of half-cliques with a corona. Let $G = (V, E)$; $E \subseteq V \times V$, be finite nonempty undirected graph. Let us have a numbering of its vertices $V = (v_i)_{i=1}^m$, $m = \text{card } V$. By the corresponding matrix we mean the matrix $(a_{ij})_{i,j=1}^m$ where $a_{ij} =_{\text{df}} 1$ if $(v_i, v_j) \in E$, and $a_{ij} =_{\text{df}} 0$ otherwise. This matrix is symmetric since the graph is undirected. By a code of the graph $C = (V, E)$ we mean the word a, $a = a_{1,2} \cdots a_{1,m} a_{2,3} \cdots a_{2,m}$, ..., $a_{m-1,m}$. Let n be the length of a. We see that $n = m \cdot (m-1)/2$ and therefore $m \geqslant \sqrt{2n}$.

By a half-clique with a corona we mean an undirected graph $G = (V, E)$ where $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, $\text{card } V_1 = \text{card } V_2$, $G_1 =_{\text{df}} (V_1, E \restriction V_1 \times V_1)$ is the full graph (the clique). By the size of the corona we mean the number of edges in G which are not in G_1. The degree of each vertice of the

half-clique is at least $m/2-1$. Hence any graph which is a half-clique with a corona of size v, $v<m/2-1$, can be divided to the clique and the corona by a unique way. The degree of vertices from the corona is less than $m/2-1$.

The exponential lower bound we prove for (the language of codes of) half-cliques with coronas of size $v=m/2-2$. It is clear that on $\{0,1\}^n$ this language is exact for each $n \in N$.

For real-time branching programs computing this language the two following lemma's hold.

LEMMA 4. *Let two computations meet at a node. If they have asked the same set A of bits then* card $A \geqslant m/4$.

Proof. Suppose card $A<m/4$. There is a bit in A on which the computation have branched. Assume that the content of this bit is 1 for the case of the first computation (and 0 for the second computation). Let us choose an input a such that on A it corresponds to the first computation, it is a code of a half-clique with a corona of size $m/2-2$, and the bit mentioned above is an edge of the half-clique. It is possible since there are $m/2$ vertices with no edge in A. According to Lemma 2 the input such that on A it corresponds to the second computation and on $\bar{A}$ it equals a is accepted, too. However, it is not a half-clique with a corona of size $m/2-2$. A contradiction.

LEMMA 5. *Let two computations meet at a node. Then* card $(A \cup B) \geqslant m/8$ *or* card$(A \cup C) \geqslant m/8$.

Proof. (by a contradiction). Let a be the following input. On A, B a corresponds to the first computation, on C to the second computation, and on D a is such that a is a half-clique with a corona of size $m/2-2$. Such a content of bits in D exists, since in $A \cup B \cup C$ there is less than $m/4$ of possible edges. According to Lemma 3(b), $B=C=\varnothing$. This contradicts to Lemma 4.

THEOREM 6. *Real-time branching programs computing the language of half-cliques with coronas of size $m/2-2$ have at least $2^{m/8} \geqslant 2^{\sqrt{2n}/8}$ nodes.*

Proof. According to Lemma 5, the initial part of any such program is a tree of depth at least $m/8$.

4. OPEN PROBLEMS

(1) To prove an exponential lower bound for weaker restriction than "real-time."

(2) To investigate the trade-off between the number of nodes (the complexity) and the number of asking.

(3) To prove a hierarchy according the number of asking for a smaller class of programs; e.g., for programs of polynomial complexity (Wegener, 1986; Ftáčnik and Hromkovič, in press).

REFERENCES

ALELIUNAS, R., KARP, R. M., LIPTON, R. J., LOVÁCZ, L., AND RACKOFF, C., (1979), "Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems," Proc. 20th IEEE Symp. on Foundations of Computer Science, pp. 218–223.

BORODIN, A., DOLEV, D., FICH, F. E., AND PAUL, W. (1983), "Bounds for Width Two Branching Programs," 15th Annual ACM Symposium on Theory of Computing, pp. 87–93.

CHANDRA, A. K., FURST, M. L., AND LIPTON, R. J. (1983), "Miltiparty Protocols," 15th Annual ACM Symposium on Theory of Computing, pp. 94–99.

MASEK, W. (1976), "A Fast Algorithm for the String Editing Problem and Decision Graph Complexity," M. Sci. thesis, MIT, Cambridge, Mass., May 1976.

WEGENER, I. (1984), Optimal decision trees and one-time-only branching programs for symmetric boolean functions, *Inform. and Control* **62**, 129–143.

WEGENER, I. (1984), "On the Complexity of Branching Programs and Decision Trees for Clique Functions," Interner Bericht 5/84.

WEGENER, I. (1986), Time-space trade-offs for branching programs, *J. Comput. System Sci.* **32**, 91–96.

PUDLÁK, P., AND ŽÁK, S. (1982), "Space complexity of computations, a manuscript, 1982.

PUDLÁK, P. (1984), "A Lower Bound on Complexity of Branching Programs," Proc. 11th Symp. MFCS, pp. 480–489.

YAO, A (1983), "Lower bounds by probabilistic arguments," Twenty-first Symp. on Foundations of Computer Science, pp. 420–428.

ŽÁK, S. (1983), Information in Computation Structures (preliminary version), *Acta Polytecn.* **20**, (IV. 4), 47–54.

ŽÁK, S. (1984), "An Exponential Lower Bound for One-Time-Only Branching Programs," Proc. 11th Symp. MFCS, pp. 480–489.

FTÁČNIK, M., AND HROMOKOVIČ, J. (in press), "Nonlinear Lower Bound for Real-Time Branching Programs."

NETCHIPORUK, E. J. (1966), A Boolean function, *Dokl. Acad. Nauk SSSR* **169**, no. (4), 765–766. [Russian], English translation in *Sov. Math. Dokl.* **7** (4), 999–1000.