ELSEVIER

Information Processing Letters 57 (1996) 31-34

Information Processing Letters

Complexity of Boolean functions over bases with unbounded fan-in gates

Vlado Dančík [a,b,1,2]

[a] *Department of Mathematics, University of Southern California, Los Angeles, CA 90089-1113, USA*
[b] *Mathematical Institute, Slovak Academy of Sciences, Grešákova 6, 040 01 Košice, Slovak Republic*

Received 19 June 1995; revised 12 October 1995
Communicated by L.A. Hemaspaandra

Abstract

Let Ω be the basis consisting of a negation and logical "and" and "or" operations over any number of inputs. Every Boolean function of n variables can be realised by a Boolean circuit over Ω using at most $2.122 \cdot 2^{n/2} + n + 1$ gates ($2 \cdot 2^{n/2} + n + 1$ for even n). We also show that almost all Boolean functions have circuit complexity at least $1.914 \cdot 2^{n/2} - 4n$.

Keywords: Boolean functions; Computational complexity

1. Introduction and preliminaries

Shannon [5] and Lupanov [2] have shown that almost all Boolean functions over the basis of all binary gates have circuit complexity $2^n/n$. This classical results can be naturally extended to the case of more general bases [3]. Let k_Ω be the maximal fan-in of gates in a (fixed) basis Ω, then almost all Boolean functions of n variables have circuit complexity $2^n/(k_\Omega - 1)n$ over basis Ω. Similar results for other bases and complexity measures can be found in the literature [1,4,6]. In this note we shall consider circuit complexity of Boolean functions over the basis with unbounded fan-in $\vee$- and $\wedge$-gates.

A Boolean function f of n variables is any function $f : \{0,1\}^n \to \{0,1\}$. Boolean functions can be computed by Boolean circuits. For our purposes a Boolean circuit is defined as a directed graph satisfying the following conditions. Every vertex (gate) is labeled either with one of variable names $x_1, \ldots, x_n$ or with one of the gate operations $\neg, \vee, \wedge$. The in-degree of gates marked with variable names is 0, the in-degree of $\neg$-gates is 1, and the in-degree of $\vee$- and $\wedge$-gates is arbitrary, but at least 2. There is a special output gate with out-degree 0, all other gates have arbitrary out-degree. Without loss of generality we can suppose that all gates are ordered so that every arc points according to the order of gates and the output gate is the last one in the order.

The size $\mathbf{C}(C)$ of a Boolean circuit C is the number of $\neg, \vee, \wedge$ gates in the circuit C. The (circuit size) complexity $\mathbf{C}(f)$ of Boolean function f is the minimal size of a circuit computing the function f.

[1] The author is partially supported by the NIH grant 2R01GM36230-10, the EPSRC grant GR/J 17844 and ALCOM II.

[2] Email: dancik@hto.usc.edu, vlado@linux1.saske.sk.

2. Upper bounds

Let $f = p_1 \vee \cdots \vee p_k$ be a disjunctive normal form of a Boolean function of n variables $x_1, \ldots, x_n$. We can separate variables into two halves, $x_1, \ldots, x_{\lfloor n/2 \rfloor}$ and $x_{\lfloor n/2 \rfloor + 1}, \ldots, x_n$. Then every p_i can be written as $p_i = q \wedge r$, where $q = x_1^* \wedge \cdots \wedge x_{\lfloor n/2 \rfloor}^*$ and $r = x_{\lfloor n/2 \rfloor + 1}^* \wedge \cdots \wedge x_n^*$. Here x^* denotes a literal that can be either x or $\neg x$. This allows us to rewrite f in the following form,

$$f = (q_1 \wedge (r_{1,1} \vee \cdots \vee r_{1,k_1})) \vee \cdots \vee (q_m \wedge (r_{m,1} \vee \cdots \vee r_{m,k_m})),$$

where $q_i \neq q_j$ for $i \neq j$. We denote the function $r_{i,1} \vee \cdots \vee r_{i,k_i}$ by f_i and let $f_i(x_{\lfloor n/2 \rfloor + 1}, \ldots, x_n) = s_{i,1} \wedge \cdots \wedge s_{i,j_i}$ be a conjunctive normal form of f_i. Thus

$$f = (q_1 \wedge (s_{1,1} \wedge \cdots \wedge s_{1,j_1})) \vee \cdots \vee (q_m \wedge (s_{m,1} \wedge \cdots \wedge s_{m,j_m})).$$

Now we can describe a circuit of depth three (actually there is the fourth level of negations of input variables) for f induced by this decomposition. In the bottom level of the circuit we have $2^{\lceil n/2 \rceil}$ $\vee$-gates computing all $2^{\lceil n/2 \rceil}$ maxterms over variables $x_{\lfloor n/2 \rfloor + 1}, \ldots, x_n$. In the middle level we have at most $2^{\lfloor n/2 \rfloor}$ $\wedge$-gates, the ith of them computing the function $q_i \wedge f_i = x_1^* \wedge \cdots \wedge x_{\lfloor n/2 \rfloor}^* \wedge s_{i,1} \wedge \cdots \wedge s_{i,j_i}$. In the top level there is an $\vee$-gate computing f.

Thus for every Boolean function f there is a circuit over Ω computing f and having $2^{\lfloor n/2 \rfloor} + 2^{\lceil n/2 \rceil} + n + 1$ gates. For even n we then have $\mathbf{C}(f) \leqslant 2 \cdot 2^{n/2} + n + 1$, for odd n we have $\mathbf{C}(f) \leqslant 2^{(n-1)/2} + 2^{(n+1)/2} + n + 1 = (\sqrt{2} + 1/\sqrt{2}) 2^{n/2} + n + 1 < 2.122 \cdot 2^{n/2} + n + 1$.

3. Lower bounds

We try to answer the question how good with respect to the number of gates is the construction from the previous section. A trivial lower bound can be obtained using simple counting argument. Let $N(m)$ be the number of different Boolean circuits over Ω with not more than m gates. For every gate we have three possibilities of choosing the type of a gate and 2^n possibilities of selecting variables contributing to the gate. There are $m(m-1)/2$ pairs of gates which gives $2^{m(m-1)/2}$ possibilities of selecting wires. Therefore,

$$N(M) \leqslant \sum_{m=1}^{M} 3^m (2^n)^m 2^{m(m-1)/2} g.$$

For $M < \sqrt{2}.2^{n/2} - 2n$ we have

$$\begin{aligned} \log N(M) &\leqslant \log(M 3^M (2^n)^M 2^{M(M-1)/2}) \\ &\leqslant \log M + M \log 3 + nM + \frac{M(M-1)}{2} \\ &\leqslant \frac{M^2}{2} + 2nM \leqslant \frac{M^2}{2} + 2nM + 2n^2 - n \\ &\leqslant \frac{M^2}{2} + 2nM + 2n^2 - n \\ &= \frac{(M+2n)^2}{2} - n < 2^n - n. \end{aligned}$$

Since there are 2^{2^n} different Boolean functions of n variables, then almost all of them (at least $2^{2^n}(1 - 1/2^n)$) must have a circuit size complexity at least $\sqrt{2} \cdot 2^{n/2} - 2n$.

There can be many different circuits computing the same Boolean function. To improve lower bound we shall try to count only as few of such circuits as possible. To do so we define *simple* circuits, these are circuits satisfying the following four conditions:

(1) every edge connects gates of different type,
(2) for every gate there is at most one gate negating it, moreover this $\neg$-gate is next one according to the order of gates,
(3) no input of any gate is the negation of another input of that gate,
(4) first (at most n) gates are negations of variables.

Lemma 1. *For every circuit C computing a Boolean function f there is a simple circuit C' computing f such that* $\mathbf{C}(C') \leqslant \mathbf{C}(C)$.

Proof. Let C be any circuit computing function f. We show how to transform it into a simple circuit without increasing the size.

First we check $\neg$-gates. If output of some gate is the input for more than one $\neg$-gate, all redundant $\neg$-gates are deleted. Also order can be changed so that $\neg$-gate follows immediately the negated gate. Also we can move all negations of input variables to the gates with the smallest numbers. Any $\neg$-gate negating another $\neg$-gate is redundant and can be deleted. Similarly, if the output of a $\wedge$-gate ($\vee$-gate) is the input of

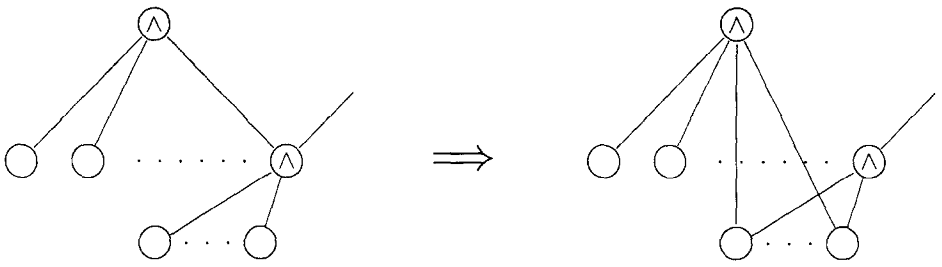

Fig. 1. Elimination of $\wedge$-$\wedge$ connections.

another $\wedge$-gate ($\vee$-gate respectively), the connection between the gates can be removed and the links from the sources of the first gate can lead straight to the inputs of the second gate (see Fig. 1). If a gate has as an input the negation of its other input, is computes a constant function and is redundant. □

Let $S(i,j,k,l)$ be the number of circuits having i $\neg$-gates negating $\wedge$-gates, j $\neg$-gates negating $\vee$-gates, k negation-free $\wedge$-gates, and l negation-free $\vee$-gates. We can treat negation gate together with the gate it negates as double "gate". There are

$$\binom{i+j+k+l}{i}\binom{j+k+l}{j}\binom{k+l}{l} < 4^{i+j+k+l}$$

ways to choose the type of a gate. Each gate (except a $\neg$-gate) can be connected to variables $x_1^*,\dots,x_n^*$, this gives at most 3^{nm} possibilities. Now we specify the maximal number of possible connections among "gates" $\neg\wedge,\neg\vee,\wedge,\vee$ in simple circuits. For "gates" without the negation there are two possibilities of connection if the gates are different and only one possibility when the gates are the same. For "gates" with the negation there are at most three possibilities of connection if the gates are different and at most two possibilities when the gates are the same. The following table shows the number of possible connections among "gates" in simple circuits.

$\neg\wedge$	$\neg\vee$	$\wedge$	$\vee$	
2	3	2	3	$\neg\wedge$
	2	3	2	$\neg\vee$
		1	2	$\wedge$
			1	$\vee$

Thus

$$S(i,j,k,l) \leqslant 4^{i+j+k+l}3^{n(i+j+k+l)} \times 2^{i(i-1)/2+j(j-1)/2+ik+jl+kl}3^{ij+il+jk}.$$

Let $m = 2i+2j+k+l$ be the total number of gates, then $i^2/2+j^2/2+ik+jl+kl+(ij+il+jk)\log 3$ has maximal value μm^2 when $i=j=\alpha m$ and $k=l=\beta m$, where

$$\alpha = \frac{-1+\log 3}{-2+6\log 3},\quad \beta = \frac{1+\log 3}{-2+6\log 3},$$

and $\mu = \dfrac{\log 3(1+\log 3)}{-4+12\log 3}$.

Thus

$$S(i,j,k,l) \leqslant 2^{4nm+\mu m^2}$$

Let $S(M)$ be the number of different simple circuits over Ω with not more than M gates, then

$$S(M) \leqslant \sum_{m=1}^{M}(m+1)^4 2^{3nm+\mu m^2}.$$

If $M < \mu^{-1/2}2^{n/2} - 4n$, then

$$\log S(M) \leqslant \mu M^2 + 3nM + 5\log M \leqslant (\mu M + 4n)^2 - n < 2^n - n$$

and therefore almost all Boolean function of n variables have complexity at least $\mu^{-1/2}2^{n/2} - 4n = 1.914\cdot 2^{n/2} - 4n$.

Acknowledgement

I would like to thank Professor M. Paterson for his help, many valuable comments and suggestions. I also thank J. Tarui and P. Dunne for many useful discussions.

References

[1] P.E. Dunne, *The Complexity of Boolean Networks* (Academic Press, London, 1988).

[2] O.B. Lupanov, A method of circuit synthesis, *Izv. Vussh. Uchebn. Zaved. Radiofiz.* **1** (1958) 120–140 (in Russian).

[3] M.S. Paterson, private communication.

[4] J.E. Savage, *The Complexity of Computing* (Wiley, New York, 1976).

[5] C.E. Shannon, The synthesis of two-terminal switching circuits, *Bell System Tech. J.* **28** (1949) 59-98.

[6] I. Wegener, *The Complexity of Boolean Functions* (Wiley-Teubner, 1987).