ONE-WAY FUNCTIONS AND CIRCUIT COMPLEXITY

R. Boppana

Massachusetts Institute of Technology
Cambridge, Massachusetts

J. C. Lagarias

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

A finite function f is a mapping of $\{1,2,\ldots,m\}$ into $\{1,2,\ldots,m\} \cup \{\#\}$ where $\#$ is a symbol to be thought of as ''undefined.'' This paper defines a measure $M(f)$ of the difficulty of inverting a finite function f, which is given by $M(f) = \text{MIN}\left\{\dfrac{\log_2 C(g)}{\log_2 C(f)} : g \text{ an inverse of } f\right\}$ where $C(f)$ is a circuit complexity measure of the difficulty of computing f. We say that one-way functions exist (in a circuit complexity sense) if and only if $M(f)$ is unbounded. We prove that one-way functions exist if and only if the satisfiability problem SAT has polynomial sized circuits.

This paper also defines an analogous measure $M_d(f)$ in which only circuits of depth $\leq d$ are allowed. We show that one-way functions exist in this bounded-depth circuit complexity model, by showing for the permutations σ_n on $\{1,2,\ldots,2^n\}$ defined by $\sigma_n(k) \equiv 3k \pmod{2^n}$ that for $d \geq 4$ there is a positive constant c_d such that $M_d(\sigma_n) > c_d \log n$ as $n \to \infty$.

ONE-WAY FUNCTIONS AND CIRCUIT COMPLEXITY

R. Boppana

Massachusetts Institute of Technology
Cambridge, Massachusetts

J. C. Lagarias
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. Introduction

A one-way function is a function that is easy to compute, whose inverse is hard to compute. Such functions play an important role in cryptography and in generating pseudo-random numbers. Several distinct concepts of one-way functions have been proposed ([B], [L], [Sel]). However the existence of one-way functions has never been proved for any of these concepts. There are particular classes of functions, such as the discrete logarithm [O] which *appear* to possess some one-wayness properties. In this paper we study a non-uniform notion of one-wayness based on circuit complexity.

We associate to a finite function f combinatorial measures $M(f)$ and $M_d(f)$ of ''degree of one-wayness'' based on circuit complexity. We consider functions f: $I_m \to I_m \cup \{\#\}$ having the finite domain $I_m = \{1, 2, \ldots, m\}$, and whose range may include the object #, where # is a symbol to be thought of as ''undefined.'' We assign a circuit complexity to a finite function f by associating to f a set of 0-1 valued Boolean functions $\{B_i(f)\}$ (''*bit functions*'') that describe f, and then assigning a circuit complexity to this set of Boolean functions. We define a notion of an inverse g to a finite function f and assign a similar measure of circuit complexity of computing an inverse function g. The measures $M(f)$ and $M_d(f)$ are constructed from these.

We first define for a Boolean function $B : \{0, 1\}^n \to \{0, 1\}$ the circuit complexity measures $C(B)$ and $C_k(B)$ we use. We define a *stratified circuit* to be one having $2n$ inputs $\{x_1, \ldots, x_n\}$ and $\{\neg x_1, \ldots, \neg x_n\}$, and which are stratified into levels having only OR gates at even levels $2i$ and AND gates at odd levels $2i + 1$, with arbitrary fan-in and fan-out allowed at each level. The inputs to gates at level n must all be outputs of gates at level $n-1$. We define the *circuit complexity* $C(B)$ of a Boolean function B to be the number of inputs n plus the minimum number of gates needed in any stratified circuit that computes B. The

depth of a stratified circuit is the number of levels in a circuit. Every Boolean function can be computed by some stratified circuit of depth 2 in this model. We define the *depth-d circuit complexity* $C_d(B)$ of a Boolean function B to be the number of input n plus the minimum number of gates needed in any stratified circuit of depth at most d that computes B.

The circuit complexity measure $C(B)$ is essentially the same up to a polynomial factor as a general circuit complexity measure. An arbitrary circuit having G gates (AND, OR, and NOT) with fan-in and fan-out at most two and an acyclic graph of connections can be computed by a stratified circuit having at most $2G^2$ gates. (See Appendix A.) However the depth of circuit measure $C_d(B)$ does depend in a critical way on the allowability of circuits with unbounded fan-in and fan-out. For example, not all Boolean functions can be represented by depth 2 circuits with bounded fan-in.

Now we define a circuit complexity measure $C(f)$ for a finite function $f : I_m \to I_m \cup \{\#\}$. We first extend the domain of f to be the next higher power of 2 by using #: If $2^{n-1} < m \le 2^n$ we define $f^* : I_{2^n} \to I_{2^n} \cup \{\#\}$ by

$$f^*(k) = \begin{cases} f(k) & \text{for } 1 \le k \le m\,, \\ \# & \text{for } m+1 \le k \le 2^n\,. \end{cases} \tag{1.1}$$

Next we express f^* as $n+1$ Boolean functions (''bit functions'') $B_i = B_i(f^*)$ for $0 \le i \le n$ by writing the input k and output $f(k) =$ ➳ in binary as

$$k = x_n 2^{n-1} + \ldots + x_1\,; \quad \text{all} \quad x_i \in \{0, 1\}$$

and then for $1 \le i \le n-1$ defining

$$B_0(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } f(k) = \#, \\ 0 & \text{otherwise}, \end{cases} \tag{1.2}$$

and

$$B_i(x_1, \ldots, x_n) = \begin{cases} y_i & \text{if } f(k) = \sum_{i=1}^{n} y_i \, 2^{i-1} \text{ with } y_i \in \{0, 1\}, \\ 0 & \text{if } f(k) = \#, \end{cases} \tag{1.3}$$

for $1 \leq i \leq n$. Now we define the circuit complexity $C(f)$ of f by

$$C(f) = C(f^*) = n + \sum_{i=0}^{n} C(B_i(f^*)) \ . \tag{1.4}$$

We have a similar depth-d measure $C_d(f)$ circuit complexity given by

$$C_d(f) = C_d(f^*) = n + \sum_{i=0}^{n} C_d(B_i(f^*)) \ . \tag{1.5}$$

Next we define the circuit complexity of the problem of inverting a function $f : I_m \to I_m \cup \{\#\}$. We say a function $g : I_m \to I_m \cup \{\#\}$ is *an inverse* of f provided that

$$g(x) = \begin{cases} y & \text{if } x \in \text{Range } (f) \text{ and } f(y) = x, \\ \# & \text{if } x \notin \text{Range } (f) \ . \end{cases}$$

A function f may have more than one inverse.

We define the *canonical inverse function* $f^{\#}$ by:

$$f^{\#}(x) = \begin{cases} \text{MIN}\{y : f(y) = x\} & \text{if } x \in \text{Range } (f) \ , \\ \# & \text{if } x \notin \text{Range } (f). \end{cases} \tag{1.6}$$

We define the *circuit complexity* $C^{-1}(f^*)$ *of the function inversion problem* to be

$$C^{-1}(f^*) = \text{MIN}\{C(G) : G \text{ is an inverse of } f^*\} \tag{1.7}$$

Analogously we define the *depth d circuit complexity* $C_d^{\#}(f)$ *of the function inversion problem* by

$$C_d^{-1}(f^*) = \text{MIN}\{C_d(g) : g \text{ is an inverse of } f^*\} \tag{1.6}$$

Finally we define the *measure of one-wayness* $M(f)$ of a finite function f by

$$M(f) = \frac{\log_2 C^{-1}(f^*)}{\log_2 C(f^*)} \tag{1.8}$$

We have an analogous *depth d measure of one-wayness* $M_d(f)$ defined by

$$M_d(f) = \frac{\log_2 C_d^{-1}(f^*)}{\log_2 C_d(f^*)} \tag{1.9}$$

The point of these definition of $M(f)$ is that $M(f) \geq k$ if and only if $C^{-1}(f^*) \geq C(f^*)^k$, so that $M(f)$ is unbounded if and only if a superpolynomial increase in the number of gates is needed in any circuit computing an inverse function G of f^* compared to the minimum number of gates needed to compute f^*. In this context the purpose of including the number of inputs n in the circuit complexity $C(B)$ is to avoid $M(f)$ possibly being large because $C(f^*)$ is small. In terms of this circuit complexity model we say that *one-way finite functions exist* if and only if there exists an infinite sequence of functions $\{f_i : 1 \leq i < \infty\}$ defined on larger and larger domains such that $\{M(f_i)\}$ is unbounded.

Our main result for the general circuit complexity model is as follows.

Theorem 1. The following are equivalent:

(1) There is a bound c_0 such that $M(f) \leq c_0$ for all finite functions f.

(2) The satisfiability problem SAT has polynomial sized circuits.

Karp and Lipton [KL] have shown that if SAT has polynomial size circuits then the polynomial hierarchy collapses at the second level, i.e. $\sum_2^p = \prod_2^p$. This may be taken as evidence that one-way finite functions exist.

For the bounded-depth circuit complexity model we are able to show that one-way functions exist.

Theorem 2. Let $\sigma_n : I_{2^n} \to I_{2^n}$ be the permutation

$$\sigma_n(k) \equiv 3k \pmod{2^n} .$$

Then for any fixed $d \geq 4$, there is a positive constant c_d such that

$$M_d(\sigma_n) \geq c_d \log n .$$

This result is proved by showing that σ_n has polynomial sized circuits of depth 4 and second, that a circuit to invert σ_n can be used to compute $k \equiv 0 \pmod 3$, a problem known to require superpolynomial size constant depth circuits by a result of Furst, Saxe and Sipser [FSS]. The quantitative bound $M_d(\sigma_n) \geq c_d \log n$ follows from a sharper bound of Ajtai [A] for the same problem.

Theorem 2 may be taken more as a indication of the restrictiveness of the bounded-depth circuit complexity model than as a indication of the existence of one-way functions.

2. Proof of Theorem A

Before proving the result, we make precise the statement ''SAT has polynomial-sized circuits.'' Define the *length* of a Boolean formula B to be the number of symbols in it, where the variable x_n is counted as $\log_2 n$ symbols. Let $BF_{n,m}$ denote all Boolean formulae of length at most m, involving only the variables $x_1, \ldots, x_n$. There is an encoding $\rho : BF \to N$ which has the properties

(1) ρ is one-to-one and is computable in polynomial time.

(2) The unique inverse ρ^{-1} is computable in polynomial time.

(3) $\rho(BF_{n,m}) \subseteq [1, 2^{3m}]$.

(See Appendix B). Note that

$$\rho^{-1}(k) = \begin{cases} F & \text{if } k \in \text{Range } (\rho) \text{ and } \rho(F) = k \\ \# & \text{if } k \notin \text{Range } (\rho) \end{cases}$$

Now set $\blacksquare = 2^{3m}$, and define the finite function $SAT^*_{n,m} : I_{\blacksquare} \to I_{\blacksquare} \cup \{\#\}$ by

$$SAT^*_{n,m}(k) = \begin{cases} 1 & \text{if } k = \rho(F) \text{ for } F \in BF_{n,m} \text{ and } F \text{ is satisfiable.} \\ 0 & \text{if } k = \rho(F) \text{ for } F \in BF_{n,m} \text{ and } F \text{ is unsatisfiable.} \\ \# & \text{if } k \notin \rho(BF_{n,m}). \end{cases}$$

Then ''SAT has polynomial size circuits'' means there exists an integer j such that

$$C(SAT^*_{n,m}) \leq (n + m)^j + j .$$

for all $m \geq 1$.

Proof of Theorem A. (1) => (2). Suppose that the function $M(f)$ is bounded by the integer c_0 for all finite functions *f*. We will construct polynomial size circuits for SAT.

The TEST be the set

$$\text{TEST} = \left\{(B, \mathbf{x}) : B \text{ a Boolean formula with } n \text{ variables, } \mathbf{x} \in \{0, 1\}^n, n \geq 1 .\right\}$$

Certainly TEST is recognizable in polynomial time, and it is easy to construct (by the same method as used in Appendix B) a one-to-one polynomial-time computable encoding $\Psi : \text{TEST} \rightarrow \mathbf{N}$ such that:

(1) It is *honest* in the sense that

$$4 \text{ length } (B, \mathbf{x}) \geq \text{ length } \Psi((B, \mathbf{x})) \geq \text{ length}(B, \mathbf{x}) \tag{2.1}$$

for all inputs $(B, \mathbf{x})$.

(2) The range $\Psi(\text{TEST})$ is recognizable in deterministic polynomial time.

Define

$$\text{TEST}_{n,m} = \left\{(B, \mathbf{x}) : B \in BF_{n,m} \quad \text{and} \quad x \in \{0, 1\}^n\right\}.$$

and observe that

$$\Psi(\text{TEST}_{n,m}) \subseteq [1, 2^{4(n+m)}] . \tag{2.2}$$

We use the following well-known fact:

Lemma 2.1. Let T be a Turing machine used as a transducer that halts on all inputs of length ≤ n in time $f(n)$ and which has s states. There is a stratified Boolean circuit Ω_n of depth ≤ $6sf(n)$ with ≤ 20 $s^2[f(n)]^2$ gates which simulates T on all inputs of length ≤ n. More precisely, Ω_n has $2n$ input gates encoding the n inputs to T as 00 = 0, 11 = 1 and 01 = #, and 2 $f(n)$ outputs encoding the output of T similarly.

We sketch a proof of Lemma 2.1 in Appendix C.

We first infer from Lemma 2.1 that since the collection of functions $\{\rho_{n,m}^{-1}\}$ given by

$$\rho_{n,m}^{-1}(k) = \begin{cases} B \text{ if } k = \rho(B) \text{ and } B \in BF_{n,m} \\ \\ \# \text{ otherwise} \end{cases}$$

for integers k in the interval $[1, 2^{3m}]$ is computable by a polynomial time Turing machine, each such functions $\rho_{n,m}^{-1}$ is computable by a polynomial size circuit, having at most $p_2(n,m)$ gates.

Next we consider the finite function $f_{n,m}$ on domain $[1, 2^{4(m+n)}]$ define by

$$f_{n,m}(k) = \begin{cases} \Psi((B, B(\mathbf{x})^n)) \text{ if } k = \Psi(B, \mathbf{x}) \text{ is in } \Psi(\text{TEST}_{n,m}), \\ \\ \# \qquad\qquad\qquad \text{if } k \text{ isn't in } \Psi(\text{TEST}_{n,m}). \end{cases}$$

Here B denotes a Boolean expression with n variables, and $B(\mathbf{x})$ is its value on input $\mathbf{x}$. Since the complete set of functions $\{f_{n,m}\}$ is computable by a polynomial time Turing machine used as a transducer, by Lemma 2.1 the collection $\{f_{n,m}\}$ is computable by polynomial size circuits, having at most $p_3(n, m)$ gates x_1. By assumption $M(f_{n,m}) \leq c_0$ so that there is an inverse function $g_{n,m}(\cdot)$ for $f_{n,m}(\cdot)$ that uses at most $p_4(n, m)$ gates, where $p_4(x_1, x_2) = [p_3(x_1, x_2)]^{c_0}$. In particular, this inverse $g_{n,m}$ has the property

$$F \in BF_{n,m} \text{ and } F \notin \text{SAT} <=> F \in BF_{n,m} \text{ and } g_{n,m}(\Psi(F, 1^n)) = \#.$$

Consequently $g_{n,m}$ can be used to recognize UNSAT and hence SAT. A polynomial sized circuit for $\text{SAT}_{n,m}$ is easily constructed from that for $g_{n,m}(\cdot)$ using the flowchart in Figure 1, and the resulting circuit has at most $p_5(n,m) = p_3((n+m)c_0, (n+m)^{c_0}) + p_3(n,m) + p_2(n,m) + 4(n+m)$ gates.

Figure 1. Flowchart for circuit to compute $\mathrm{SAT}_{n,m}$.

(2) => (1). Assume that SAT has polynomial sized circuits. Now let f be an arbitrary finite function $f : I_{2^n} \rightarrow I_{2^n} \cup \{\#\}$ and suppose f has circuit complexity $C(f)$. For $0 \leq i \leq n+1$ let $\sum_f^{(i)}$ be stratified circuits computing the ''bit functions'' $B_0(f)$, . . . , $B_{n+1}(f)$ using the minimal number of gates G_i, and note that by definition of the circuit complexity measure $C(f)$ we have

$$G_i \leq C(f) \quad \text{for} \quad 0 \leq i \leq n+1 \ . \tag{2.3}$$

We shall combine these circuits with appropriately sized SAT circuits to create a circuit $\sum_{f^\#}$ computing the canonical inverse function $f^\#$ which has at most $(C(f))^{c_0} + c_0$ gates, where c_0 is a constant independent of f. Assuming this is accomplished, we may conclude that

$$\begin{aligned} M(f) &\leq \frac{\log_2 C(f^\#)}{\log_2 C(f)} \\ &\leq \frac{c_0 \log_2 C(f) + c_0}{\log C(f)} \\ &\leq 2c_0 \ , \end{aligned}$$

and the desired implication follows.

The circuit $\sum_{f^\#}$ will use suitably encoded versions of the circuits $\sum_f^{(i)}$ as inputs to SAT circuit in order to ''guess'' the lexicographically smallest value for the inverse function $f^\#$ when it exists. To encode the circuits $\sum_f^{(i)}$ as Boolean formulae we use the following result.

Lemma 2.2. Let $g : \{0,1\}^n \to \{0,1\}$ be any Boolean function of n variables, and suppose g is computable by a stratified circuit having G gates. Then there is a Boolean formula $\overline{B}_g(x_1, \ldots, x_n, y_1, \ldots, y_G)$ in $n+G$ variables such that:

(1) For all $(x_1, \ldots, x_n) \in \{0,1\}^n$ we have

$$g(x_1, \ldots, x_n) = 1 \Leftrightarrow \exists y_1 \exists y_2 \cdots \exists y_H \; \overline{B}_g(x_1, \ldots, x_n, y_1, \ldots, y_G) = 1 .$$

(2) The Boolean formula $\overline{B}_g(x_1, \ldots, x_n, y_1, \ldots, y_G)$ contains at most $2G^2 + 2nG + 2G$ variable symbols and has length at most $12G(n+G) \log_2(n+G)$.

Proof. (Sketch) We add dummy variables $y_1, \ldots, y_G$ corresponding to the gates of the stratified circuit and add appropriate equality conditions forcing the values y_i to simulate the gates of the circuit. See Appendix E for a detailed proof. ■

We apply Lemma 2.2 to the ''bit functions'' $B_i(f)$ to obtain Boolean formulae $\overline{B}_i(x_1, \ldots, x_n, y_1, \ldots, y_{G_i})$ such that for $0 \le i \le n$,

$$B_i(f)(x_1, \ldots, x_n) = 1 \Leftrightarrow \exists y_1 \, \exists y_2 \cdots \exists y_{G_i} \; \overline{B}_i(x_1, \ldots, x_n, y_1, \ldots, y_{G_i}) = 1 . \quad (2.4)$$

We may bound the length L_i of the Boolean formula $\overline{B}_i$ using (2.3) and (2) of Lemma 2 to obtain

$$L_i \le 24C(f)(n+C(f)) \log_2 C(f) \le 48C(f)^3 \quad (2.5)$$

The overall structure of the circuit $\sum_{f^\#}$ to compute $f^\#$ is pictured in Figures 2a and 2b. The main ingredient in the circuit $\sum_{f^\#}$ is the circuits used to compute Round i for $1 \le i \le n$. The detailed structure of a Round i circuit (excluding Round 1) is pictured in Figure 3. It has two main ingredients, a circuit simulating a Turing machine computation and a SAT circuit. The purpose of the Turing machine computation is to preprocess the question asked in Round i: 'Do there exist values $x_{i+1}, \ldots, x_n$ such that $f(\tilde{x}_1, \ldots, \tilde{x}_i, x_{i+1}, \ldots, x_n) = k$?' into a form suitable for input to the SAT circuit, where

$k = \sum_{i=1}^{n} z_i \, 2^{i-1}$.

The required Turing machine has the following properties. It takes as inputs n, G, i, and $n+1$ Boolean formulae $\overline{B}_i(x_1, \ldots, x_n, y_1, \ldots, y_G)$ for $0 \le i \le n$, plus the i values $(\tilde{x}_1, \ldots, \tilde{x}_i)$ and the n values $(z_1, \ldots, z_n)$ where $k = \sum_{i=1}^{n} z_i \, 2^{i-1}$. It produces as output the encoding $\rho(F_i)$ of the Boolean formula

$$F_i(x_{i+1}, \ldots, x_n, y_1, \ldots, y_G) = \bigwedge_{j=1}^{n} \text{`}\overline{B}_j(\tilde{x}_1, \ldots, \tilde{x}_i, x_{i+1}, \ldots, x_n, y_1, \ldots, y_G) = z_i\text{'} \quad (2.6)$$

where $\overline{B}_j(\tilde{x}_1, \ldots, \tilde{x}_i, x_{i+1}, \ldots, x_n, y_1, \ldots, y_G)$ is the Boolean formula obtained from $\overline{B}_j$ of (2.4) obtained by substituting the specific values $(\tilde{x}_1, \ldots, \tilde{x}_i)$ for the variables $(x_1, \ldots, x_i)$, and where '$\overline{B}_j = z$' is an abberivation for $(\overline{B}_j \wedge z) \vee (\neg\overline{B}_j \wedge \neg z)$. It is clear that there is a polynomial time Turing machine to do this computation, hence by Lemma 2.1 this can be simulated by a circuit of size polynomial in $C(f)$.

Now we bound the size of the SAT circuit required to test $\rho(F_i)$. Now the formula F_i involves at most $n + C(f)$ variables and is of length at most $1536C(f)^4$, using the bounds (2.5) and $n \le C(f)$. Consequently $F_i \in BF_{2C(f),\,1536C(f)^4}$. Thus we may test $\rho(F_i)$ using a $SAT^*_{2C(f),\,1536C(f)^4}$-circuit. Observe that

$$\begin{aligned} SAT^*(\rho(F_i)) = 1 &\Leftrightarrow \exists x_{i+1} \cdots \exists x_n \exists y_1 \cdots \exists y_G \, \tilde{B}_i(\tilde{x}_1, \ldots, \tilde{x}_i, x_{i+1}, \ldots, x_n, y_1, \ldots, y_G) = 1, \\ &\Leftrightarrow \exists x_{i+1} \cdots \exists x_n \, f(\tilde{x}_1, \ldots, \tilde{x}_i, x_{i+1}, \ldots, x_n) = k, \end{aligned}$$

as required.

By hypothesis SAT has polynomial sized circuits, so the resulting circuit $\sum_{f^\#}$ is of size bounded by a polynomial in $C(f)$, so is $\le (C(f))^{c_0} + c_0$ for a sufficiently large fixed constant c_0. ■

Figure 2. Flowchart for computing $f^{\#}$.

3. Proof of Theorem B

Let $\sigma_n : I_{2^n} \to I_{2^n}$ be the permutation $\sigma_n(k) \equiv 3k \pmod{2^n}$. We first show that $C_d(\sigma_n)$ is bounded by a polynomial in n, for $n \geq 4$. We first exhibit a depth 6 stratified circuit Σ_n that computes σ_n using $O(n^2)$ gates. Write the input k in binary as

$$k = \sum_{i=0}^{n-1} x_i 2^i$$

and define the binary bits z_i by

$$3k = \sum_{i=0}^{n+1} z_i \, 2^i \, .$$

It suffices to find a circuit that when given $\{x_i : 0 < i \leq n-1\}$ computes the bits $\{z_i : 0 \leq i \leq n+1\}$ and then discards the overflow bits z_n and z_{n+1}. We view $3k = k + 2k$ and note that the i^{th} ''carry bit'' can be computed with a depth 4 circuit with $3i + 4$ gates. Indeed consider the i^{th} carry bit w_i in adding $\twoheadrightarrow_1 = \sum_{i=0}^{n} x_i \, 2^i$ to $\twoheadrightarrow_2 = \sum_{i=0}^{h} y_i \, 2^i$ Then

$$\begin{aligned} w_i = 1 \iff & \text{ either } x_i \wedge y_i = 1 \\ & \text{or } x_{i-1} \vee y_{i-m} = 1 \quad \text{for} \quad 0 \leq m \leq j \quad \text{and} \\ & x_{i-j} \wedge y_{i-j} = 1 \quad \text{for some} \quad j \quad \text{with} \quad 1 \leq j \leq i \, . \end{aligned}$$

We compute the clauses $x_j \wedge y_j$ at depth 1, the clauses $x_j \vee y_j$ at depth 2, combine them to compute the $i + 1$ clauses $A_{i,0} = x_i \wedge y_i$ and

$$A_{i,j} = \bigwedge_{i=m}^{j-1} (x_{i-m} \vee y_{i-m})) \wedge (x_{i-j} \wedge y_{i-j}) \qquad 1 \leq j \leq i$$

at depth 3 and finally compute

$$w_i = A_0 \vee A_1 \vee \ldots \vee A_i \tag{3.1}$$

at depth 4. Using all the carry bits for $0 \leq i \leq n$, we can now compute all the bits of $\twoheadrightarrow_1 + \twoheadrightarrow_2$ in a depth 6 circuit using $O(n^2)$ gates. To do this we create other depth 4 circuits computing $\overline{w}_i = \neg w_i$ and then compute the j^{th} bit of $\twoheadrightarrow_1 + \twoheadrightarrow_2$ to be

$$x_j \oplus y_j \oplus w_{j-1} = (x_j \wedge y_j \wedge w_{j-1}) \vee (x_j \wedge \overline{y}_j \wedge \overline{w}_{j-1})$$
$$\vee(\overline{x}_j \wedge y_j \wedge \overline{w}_{j-1}) \vee (\overline{x}_j \wedge \overline{y}_j \wedge w_{j-1}) \quad \text{for} \quad 0 \le j \le n+1 ,$$

where by convention $w_0 = x_{n+1} = y_{n+1} = 0$. We apply this construction to obtain the desired depth 6 stratified circuit Σ_n computing σ_n using $O(n^2)$ gates.

Now we obtain a depth 4 circuit Σ_n^* computing σ_n using a polynomial number of gates. We use the distributive laws to exchange two adjacent levels of AND and OR gates with at most a polynomial blow-up in the number of gates, which is possible if the *higher* level has bounded fan-in. Then we may collapse one level. If both levels exchanged have bounded fan-in, the same is true for the exchanged levels. We use this procedure to eliminate levels 5 and 6 from Σ_n, obtaining the desired stratified circuit Σ_n^*. (See Appendix D.)

Next we bound the complexity of computing the inverse $C_d^{\#}(\sigma_n)$ from below. Since σ_n is a permutation, it has a unique inverse σ_n^{-1} given by

$$\sigma_n^{-1}(k) \equiv \frac{1}{3}k \pmod{2^n} .$$

and by (1.6) we have

$$C_d^{-1}(\sigma_n) = C_d(\sigma_n^{-1}) .$$

Let Π_n be a depth d circuit competing σ_n^{-1} with G_n gates. We will use Π_n to construct a circuit Δ_n of depth $d + \neg$ which has $G_n + O(n^2)$ gates and which computes $\equiv 0 \pmod 3$ on $[\frac{n+1}{2}]$ variables, i.e. it computes the Boolean function

$$P_3(y_1, \ldots, y_{[\frac{1}{2}n]}) \equiv \begin{cases} 1 & \text{if } y_1 + \ldots + y_{[\frac{1}{2}n]} \equiv 0 \pmod 3, \\ 0 & \text{otherwise .} \end{cases}$$

We use the fact that

$$k \equiv 0 \pmod 3 \quad \text{and} \quad 0 \le k < 2^n \iff k = 3\,\sigma_n^{-1}(k) \quad \text{over} \quad \mathbf{N} . \tag{3.2}$$

Note here that $k \equiv 3\,\sigma_n^{-1}(k) \pmod{2^n}$, so the assertion on the right side of (3.2) is that the overflow bits z_n

and z_{n+1} in $3\ \sigma_n^{-1}(k)$ are both zero. We next observe that if $0 \le k < 2^n$ and k has the binary expansion $\sum_{i=0}^{n-1} x_i\ 2^i$ then

$$k \equiv 0 \pmod 3 \iff x_0 + 2x_1 + x_2 + 2x_3 + \ldots + \left\lfloor \frac{3 + (-1)^n}{2} \right\rfloor x_{n-1} \equiv 0 \pmod 3 . \quad (3.3)$$

Hence (3.2) and (3.3) imply that given $\{y_i : 1 \le i \le [\frac{n+1}{2}]\}$ we have

$$y_1 + \ldots + y_{[\frac{n+1}{2}]} \equiv 0 \pmod 3 \iff k = \sum_{i=0}^{[\frac{n}{2}]} y_i\ 2^{2i-1} \text{ has } k = 3\sigma_n^{-1}(k) . \quad (3.4)$$

We obtain the desired depth d + $\neg$ circuit Δ_n with $[\frac{n+1}{2}]$ inputs using the flowchart in Figure 4. The circuit Δ_n has $G_n + O(n^2)$ gates.

Figure 4: Circuit Computing $\equiv 0 \pmod 3$

By a result of Furst, Saxe, and Sipser ([FSS], Corollary 3.6) any circuit Δ_n to compute $\equiv 0 \pmod 3$ on I_{2^n} requires a number of gates that grows at rate superpolynomial in n as $n \to \infty$. Ajtai's [A] methods in fact

imply that there is a positive constant c_0 such that any such circuit has at least $\Omega(n^{c_0 \log n})$ gates as $n \to \infty$.

Hence

$$G_n + O(n^2) >> \left[\frac{n}{2}\right]^{c_0 \log \frac{n}{2}}$$

so that

$$G_n >> n^{c_1 \log n} \tag{3.5}$$

with $c_1 = \frac{c_0}{2}$, for large n. Consequently for $d \geq 4$,

$$\begin{aligned} M_d(\sigma_n) &\geq \frac{\log C_d^{-1}(\sigma_n)}{\log C_d(\sigma_n)} \\ &\geq \frac{\log G_n}{\log n} \\ &>> \log n \end{aligned}$$

using (3.5). Theorem B is proved. ■

Appendix A. Relations Between Circuit Complexity Models

A *general circuit* is a circuit having inputs the variables $x_1, \ldots, x_n$ and using AND, OR and NOT gates and having an acyclic graph of connections with fan-in and fan-out at most two at each gate. A *stratified circuit* is a circuit with inputs the *literals* $x_1, \ldots, x_n$ and $\neg x_1, \ldots, \neg x_n$, which uses only AND and OR gates which are stratified into levels such that gates at level i receive inputs only from gates at level $i - 1$, and output only to level $i + 1$, and such that all gates at even levels $2i$ are AND gates, all gates at odd levels $2i + 1$ are OR gates, and fan-in and fan-out are unrestricted. Stratified circuits also may have ''no-operation'' gates at each level using e.g. $B \wedge B$ at even levels, $B \vee B$ at odd levels.

Lemma A.1

(1) For any Boolean function f in n variables computable by a general circuit with G gates, there is a stratified circuit computing f using at most $2G^2$ *gates.*

(2) For any Boolean function f in n variables computable by a stratified circuit with G gates, there is an equivalent general circuit computing f using at most $2G^2 + n$ *gates.*

Proof.

(1) Define the *depth* of a gate to be the longest directed path in the input-output graph which inputs to the gate. We first eliminate the NOT-gates by proceeding from the largest depth upwards by induction, using the distributive laws, as in Figure A-1.

Figure A.1 Eliminating NOT-gates

At stage i we eliminate NOT nodes at level i, possibly inserting new NOT nodes at levels $i - 1$ or lower. At the end of this process we obtain a circuit whose inputs are the literals $x_1, \ldots, x_n, \overline{x}_1 (= \neg x_1), \ldots, \overline{x}_n$ and only AND and OR gates, and this circuit has at most G gates since the total number of AND and OR gates remains constant during this process.

Next add dummy (''no-operation'') gates at each level so that all of each gate's inputs come from the immediately preceding level, and all its outputs go to the next higher level. Since the maximum depth is at most G, we add at most $G^2 - G$ dummy gates. Finally split each depth level into two levels, with all AND gates at the top level and all OR gates at the bottom level, adding extra dummy gates at each level as necessary. The resulting circuit is stratified and has at most $2G^2$ gates.

(2) The main problem is to eliminate fan-in. The stratified circuit has at most depth G and fan-in and fan-out at most G at each level. Split each gate with fan-in k and fan-out $\rightarrowtail$ into $(k - 2)$ gates with fan-in 2 and fan-out 1 and a final gate in this process with fan-in 2 and fan-out $\rightarrowtail$. Now split this last gate into $\rightarrowtail$ gates having fan-in 1 and fan-out 2. (See Figure A-2).

Figure A.2

After doing this process with all original gates, the resulting circuit has at most $2G^2$ gates. Finally add n NOT gates at level one to compute $\overline{x}_1$, ... $\overline{x}_n$. ■

Appendix B. Encoding of Boolean Formulae as Natural Numbers

Let *BF* denote the collection of all well-formed Boolean formulae constructed using the symbols (,), $\wedge$, ,$\neg$ and *variables* x_j. We define the *length* of a Boolean formula to be the number of symbols in the formula, using the convention that the variable x_n is counted as being $\log_2 n$ symbols. Let $BF_{n,m}$ denote the subset of *BF* consisting of all formulae using only the symbols $x_1, \ldots, x_n$ and of length at most *m*. The map $\rho : BF \to \mathbf{N}$ encodes a formula *F* from left to right using the correspondence

$$
\begin{array}{cl}
 & \vert\!\rightarrow\ 100 \\
) & \vert\!\rightarrow\ 101 \\
\wedge & \vert\!\rightarrow\ 110 \\
\vee & \vert\!\rightarrow\ 111 \\
\neg & \vert\!\rightarrow\ 011
\end{array}
$$

and where the variable x_n is encoded as */n/*, where *n* is given in binary and then encoded further using

$$
\begin{array}{ll}
0 & \rightarrow\ 000 \\
1 & \rightarrow\ 001 \\
/ & \rightarrow\ 010
\end{array}
$$

.

Note that ρ is *honest* in the sense that for any Boolean formula *B* we have

$$3 \text{ length } (B) \geq \text{length } (\rho(B)) \geq \text{length } (B) \ .$$

Lemma B-1. The mapping $\rho : BF \to \mathbf{N}$ *has the following properties:*

(1) It is one-to-one (but not onto) and computable in polynomial time.

(2) The (unique) inverse ρ^{-1} *is computable in polynomial time.*

(4) $\rho(BF_{n,m}) \subseteq [1, 2^{3m}]$.

Proof. (1), (2) and (3) are all clear from the encoding procedure. ■

Appendix C. Simulating Turing Machine Computations with Circuits: Proof of Lemma 2.1

We are given a Turing machine T having s internal states which halts on all inputs of length $\leq n$ in time $f(n)$. We construct a stratified circuit Ω_n simulating T on all inputs of length $\leq n$ as follows. The idea is the Ω_n has $f(n)$ special levels called *tape levels* in which the ith of these levels record the state of the Turing machine and work tapes and output tapes of the Turing machine at time i. A tape level has $2n + 4f(n) + s$ gates, where $2f(n)$ of these gates are used to record the work tape data and indicate the tape's read head location, and $3f(n)$ gates are used to record the output tape data, encoded in the same way as the input data (0 encoded as 00, 1 as 11, # as 01, / as 10), and $2n$ gates are used to record the input data, and s gates are used to indicate the Turing machine's current internal state. There are a bounded level of *intermediate levels* between each tape level. There are used to carry out one step of the Turing machine computation. Each set of intermediate levels has at most $6\ s^2 f(n)$ gates and at most depth $2s + 4$. The lemma follows. ■

Appendix D. Collapsing Levels of Circuits

Lemma D-1. Let Ω *be a stratified Boolean circuit of depth* $d \geq 3$ *with* G *gates, and suppose there are* n_j *gates on level* j *with fan-in bounded by* f_j. *Then there exists a stratified Boolean circuit* Ω^* *computing the same Boolean function with depth* $d - 1$ *and at most* $G + n_d (f_{d-1})^{f_d}$ *gates.*

Proof. Suppose the d^{th} level consists of AND gates. Use the distributive laws in the form

$$\bigwedge_{i=1}^{f_d} (C_1^{(i)} \vee \ldots \vee C_{f_{d-1}}^{(i)}) = \bigvee_{\substack{m_1, \ldots, m_{f_d} \\ 1 \leq m_i \leq f_{d-1}}} (\bigwedge_{j=1}^{f_d} C_{m_i}^{(j)}) ,$$

We think of the $C_j^{(k)}$ as Boolean functions computed by the gates at level $d - 2$, and dummy variables $C_j^{(a)} = 0$ are added to fill out the identity to fan-in f_{d-1} at each gate. This identity can be used to interchange levels d and $d - 1$, putting one OR gate on level d with fan-in at most $(f_{d-1})^{f_d}$ and $(f_{d-1})^{f_d}$ new AND gates on level $d - 1$ each with fan-in at most f_d. Now since level $d - 2$ and the new level both consist of AND gates $d - 1$ they may be coalesced to obtain the stratified circuit Ω^*.

In the case where the d^{th} level in OR gates, use the corresponding distributive law and dummy variables $C_j^{(k)} = 1$.

Note that the new circuit has fan-in $f_i^* = f_i$ for $1 \leq i \leq d - 3$, $f_{d-2}^* \leq f_{d-2} + f_d$ and $f_{d-1}^* \leq (f_{d-1})^{f_d}$. ■

The construction of Lemma D-1 can also be used to exchange two interior levels i and $i + 1$. This collapses the depth by 2 and the number of new gates is at most $n_{i+1} (f_i)^{f_{i+1}}$.

Appendix E. Relations between circuits and existentially quantified Boolean formulae: Proof of Lemma 2.2

Lemma 2.2 asserts that function computable by a Boolean circuit may be calculated by an existentially quantified Boolean formula of approximately the same size.

Proof. To obtain the formula B we replace each gate g with a variable y_2. If g is an AND-gate with gates $g_1, g_2, ..., g_m$ as inputs, add the clause

$$C_g = ``y_g = (((y_{g_1} \wedge y_{g_2}) \wedge ...) \wedge y_{g_m})''$$

using

$$``B_1 = B_2'' \quad \text{to mean} \quad ((B_1) \vee (\neg B_2)) \wedge (\neg B_1) \vee B_2))$$

Proceed similarly for OR-gates. The formula is

$$B(x_1, ..., x_n, y_1, ..., y_g) = \bigwedge_{\substack{\text{all} \\ g}} C_g$$

Input variables $x_1, ..., x_n, \overline{x}_1, ..., \overline{x}_n$ are treated as literals. The number of literals in B is at most $2E + 2G$, where E is the number of edges in the input-output interconnect graph of the circuit Ω. Since $E \leq G^2 + nG$ the bound on the number of literals in B follows. ■

Lemma 2.2 holds for a general circuit also, using the same proof together with the distributive law to eliminate NOT gates.

References

[A] M. Ajtai, Σ-formulae on finite structures, Ann. Applied and Pure Logic *24* (1983), 1-48.

[B] G. Brassard, Relatived Cryptography, Proc. 20th IEEE Symposium on Foundations of Computer Science, 1979, pp. 383-392 (Section 7)

[F] M. Furst, J. Saxe, and M. Sipser, Parity, Circuits and the Polynomial Time Hierarchy, Math. Systems Theory *17* (1984) 13-27.

[KL] R. M. Karp and R. J. Lipton, Some connections between non-uniform and uniform complexity classes, Proc. 12th Annual ACM Symp. on Theory of Computing, 1980, pp. 302-309.

[L] L. Levin, One-way functions and pscudo random generators, preprint.

[O] A. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, preprint.

[S] A. Selman, Remarks about Natural Self-Reducible Sets in NP and Complexity Measures for Public Key Cryptosystems, preprint.