INFORMATION AND COMPUTATION 74, 226–240 (1987)

# One-Way Functions and Circuit Complexity*

R. B. BOPPANA

*Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139*

AND

J. C. LAGARIAS

*AT&T Bell Laboratories, Murray Hill, New Jersey 07974*

A finite function $f$ is a mapping of $\{0, 1\}^n$ into $\{0, 1\}^m \cup \{\#\}$, where "#" is a symbol to be thought of as "undefined." A family of finite functions is said to be *one-way* (in a circuit complexity sense) if it can be computed with polynomial-size circuits, but every family of inverses of these functions cannot. In this paper we show that, provided functions that are not one-to-one are allowed, one-way functions exist if and only if the satisfiability problem SAT does not have polynomial-size circuits. A family of functions $f_i(x)$ can be *checked* if some family of polynomial-size circuits with inputs $x$ and $y$ can determine if $f_i(x) = y$. A family of functions $f_i(x)$ can be *evaluated* if some family of polynomial-size circuits with input $x$ can compute $f_i(x)$. Can all families of total functions that can be checked also be evaluated? We show that this is true if and only if the nonuniform versions of the complexity classes $P$ and $UP \cap \text{co-}UP$ are equal. A family of functions $f_i$ is *one-way for constant depth circuits* if $f_i$ can be computed with unbounded fanin circuits of polynomial size and constant depth, but every family of inverses $f_i^{-1}$ cannot. We give two provably one-way functions (in fact permutations) for constant-depth circuits. The second example has the stronger property that no bit of its inverse can be computed in polynomial size and constant depth. © 1987 Academic Press, Inc.

## 1. INTRODUCTION

A one-way function is a function that is easy to compute, whose inverse is hard to compute. Such functions are important for cryptography. Without one-way functions, it is impossible to design secure public-key cryptosystems. One-way functions are also crucial for generating pseudo-random numbers. Blum and Micali (1984), Yao (1982), and Levin (1987)

* A preliminary version of this paper appeared in "Proceedings, ACM Structure in Complexity Theory Conference, Berkeley," Lecture Notes in Computer Science Vol. 223, Springer-Verlag, Berlin, 1986, pp. 51–65. The first author was supported in part by an NSF Graduate Fellowship and by an NSF Mathematical Sciences Postdoctoral Fellowship.

0890-5401/87 $3.00

each show that the existence of certain types of one-way functions implies the existence of pseudo-random number generators.

Several distinct definitions of one-way functions have been proposed (see Brassard (1983), Selman (1984), and Levin (1987)). These definitions are all based on Turing machine complexity, i.e., they are uniform concepts. Despite the importance of one-way functions, the existence of such functions has never been proved. Levin (1987) constructs a universal function that is one-way in his terminology if some one-way function exists. There are particular functions, such as the inverse of discrete logarithm (Odlyzko (1984)), that *appear* to be one-way.

In this paper we study a nonuniform notion of one-wayness based on circuit complexity. A *Boolean circuit* is a directed acyclic graph with input nodes $\{x_1, x_2, ..., x_n\}$ and output nodes $\{y_1, y_2, ..., y_m\}$, together with AND gates, OR gates, and NOT gates. The *size* of a circuit is the number of gates in the circuit. In the natural way, each circuit computes a finite function. We consider finite functions $f$ mapping $\{0, 1\}^n$ into $\{0, 1\}^m \cup \{\#\}$, where "#" is a symbol to be thought of as "undefined." An *inverse* of $f$ is a function that, on input $y$, will output some $x$ satisfying $f(x) = y$ if such an $x$ exists, and outputs # otherwise. We say that $f$ is *one-to-one* if $x_1 \neq x_2$ implies that either $f(x_1) \neq f(x_2)$ or $f(x_1) = f(x_2) = \#$. A function $f$ has a unique inverse function if and only if it is one-to-one in this sense. We say that $f$ is *onto* if every $y$ in $\{0, 1\}^m$ has some $x$ in $\{0, 1\}^n$ such that $f(x) = y$. Note that with these definitions, a one-to-one and onto function does not have to be a permutation. We say that $f$ is *total* if $f$ does not take the value #.

We say that a family of functions $f_i$ is *one-way* if the functions $f_i$ can be computed with polynomial-size circuits, but every family of inverses of $f_i$ cannot. In Section 2 of this paper we show that, provided functions that are not one-to-one are allowed, one-way functions exist if and only if the satisfiability problem SAT does not have polynomial-size circuits. Previous results by Grollman and Selman (1984) imply complementary results. They show that a family of one-way one-to-one functions exists if and only if the nonuniform versions of the complexity classes $P$ and $UP$ are not equal, and that a family of one-way one-to-one and onto functions exists if and only if the non-uniform versions of $P$ and $UP \cap \text{co-}UP$ are not equal. Karp and Lipton (1982, Theorem 6.1) show that if SAT has polynomial-size circuits, then the polynomial-time hierarchy collapses at the second level, i.e., the hierarchy $\bigcup_{i=1}^{\infty} \Sigma_i^p$ equals $\Sigma_2^p$. This may be taken as evidence that one-way functions exist, provided functions that are not one-to-one are allowed.

In Section 3, we study similar questions about the relative complexity of checking a function versus evaluating a function, first studied by Valiant (1976). A function $f$ can be *checked* if some polynomial-size circuit with inputs $x$ and $y$ can determine if $f(x) = y$. A function can be *evaluated* if some polynomial-size circuit with input $x$ can compute $f(x)$. Can all families of total functions that can be checked also be evaluated? We show that this is true if and only if the nonuniform versions of the complexity classes $P$ and $UP \cap$ co-$UP$ are equal, solving an open problem of Valiant (1976). We show there is a parallel between the results on checking versus evaluating (Theorems 3.1–3.3) and the results on one-way functions (Theorems 2.1–2.3). We also ask: can all families of total relations that can be checked also be evaluated? We show that this is true if and only if polynomial-size many–one reducibility is equivalent to beta reducibility—see Section 3 for definitions of these concepts.

In Section 4, we discuss one-way functions for the constant depth circuit model. A family of functions $\{f_i\}$ is *one-way for constant depth circuits* if $\{f_i\}$ can be computed in polynomial size and constant depth, but every inverse of $\{f_i\}$ cannot. We provide two examples of provably one-way functions (in fact permutations) for constant depth circuits. The second example has the stronger property that no bit of the inverse can be computed in polynomial size. These examples, whose inverses are trivially computable using general circuits of polynomial size, may be taken as an indication of the restrictiveness of the bounded depth circuit model rather than as evidence that one-way functions exist. Ajtai and Wigderson (1987) have constructed a pseudo-random bit generator for constant depth circuits, thus showing that probabilistic constant depth circuits can be simulated in deterministic subexponential time.

Finally we remark that the existence of one-to-one one-way functions in the nonuniform case does not seem to imply the existence of one-to-one one-way functions in the uniform case, or vice versa.

## 2. One-Way Functions for General Circuits

In this section we will show that, provided functions that are not one-to-one are allowed, one-way functions exist if and only if the satisfiability problem SAT does not have polynomial-size circuits. First we need some definitions.

Given a function $f$ mapping a set $A$ into a set $B \cup \{\#\}$, define the *domain* of $f$ to be the set of $x$ in $A$ such that $f(x) \neq \#$. Let the *range* of $f$ be the set of $y$ in $B$ such that some $x$ in $A$ satisfies $f(x) = y$. Given a function $g$ mapping $B$ to $A \cup \{\#\}$, we say that $g$ is an *inverse* of $f$ if $f(g(y)) = y$ for all $y$ in the range of $f$, and $g(y) = \#$ otherwise.

A *circuit* $C$ is a directed acyclic graph with nodes of five different types:

(i) input nodes having indegree 0 and outdegree 1, with a label from the set $\{x_1, x_2, ..., x_n\}$;

(ii) output nodes having indegree 1 and outdegree 0, with a label from the set $\{y_1, y_2, ..., y_m\}$;

(iii) AND gates having indegree 2 and arbitrary outdegree, with a label "$\wedge$";

(iv) OR gates having indegree 2 and arbitrary outdegree, with a label "$\vee$";

(v) NOT gates having indegree 1 and arbitrary outdegree, with a label "$\neg$".

In the natural way, a circuit $C$ computes a function $C$ from $\{0, 1\}^n$ to $\{0, 1\}^m$. The *size* of a circuit is the number of nodes in the circuit. Note that size($C$) is at least the number of inputs plus the number of outputs of the circuit $C$.

A *family of functions* is a sequence $f_1, f_2, ...,$ such that the function $f_i$ maps $\{0, 1\}^{n_i}$ into $\{0, 1\}^{m_i}$. The family $\{f_i\}$ is *computable in polynomial size* if there is a sequence of circuits $C_1, C_2, ...,$ such that the circuit $C_i$ has $n_i$ inputs and $1 + m_i$ outputs, has size polynomial in $n_i$, and satisfies $C_i(x) = (1, f_i(x))$ for all $x$ in the domain of $f_i$, and $C_i(x) = (0, 0^{m_i})$ otherwise. The family $\{f_i\}$ is *one-way* if $n_i$ and $m_i$ are within polynomial factors of each other, the family $\{f_i\}$ is computable in polynomial size, and no inverse of $\{f_i\}$ is computable in polynomial size.

The main result of this section is the following theorem.

THEOREM 2.1. *The following are equivalent:*

(1) *There exists a one-way family of functions.*

(2) *The satisfiability problem SAT does not have polynomial size circuits.*

We need to make precise the statement "SAT has polynomial-size circuits." Let BF denote the set of Boolean formulas. Let $\rho$ be any reasonable one-to-one encoding from BF into $\{0, 1\}^*$, i.e., such that $\rho$ and its unique inverse $\rho^{-1}$ are computable by polynomial-time Turing machines, and $\rho$ preserves lengths of formulas to within a constant factor (where a variable $x_i$ is considered to have length $\lfloor \log_2 i \rfloor + 1$). We define functions $\mathrm{SAT}_{m,n}$ from $\{0, 1\}^m$ to $\{0, 1, \#\}$ as follows:

(i) $\mathrm{SAT}_{m,n}(y) = 1$ if $\rho^{-1}(y)$ is a formula of $n$ variables that has a satisfying assignment;

(ii) $\mathrm{SAT}_{m,n}(y) = 0$ if $\rho^{-1}(y)$ is a formula of $n$ variables that has no satisfying assignments;

(iii) $\mathrm{SAT}_{m,n}(y) = \#$ otherwise.

By "SAT has polynomial-size circuits," we mean that $\{\mathrm{SAT}_{m,n}\}$ has circuits of size bounded by a polynomial in $m$ and $n$.

*Proof of Theorem* 2.1. (1) *implies* (2) Suppose that SAT has polynomial-size circuits. Let CIRCUIT-SAT be the language that consists of Boolean circuits having a satisfying assignment. By a nonuniform version of Cook's theorem (Cook, 1971), the language CIRCUIT-SAT would also have polynomial-size circuits. In fact, in polynomial size we could find the lexically first satisfying assignment of a circuit (simply make $n$ successive calls to CIRCUIT-SAT). Let $\{f_i\}$ be a family of functions computable in polynomial size using circuits $\{C_i\}$. We show how to compute the lexically first inverse of $\{f_i\}$ in polynomial size. Note that $y$ is in the domain of an inverse of $f_i$ iff $y$ is in the range of $f_i$, in other words iff some $x$ satisfies $C_i(x)=(1, y)$. But we can test this in polynomial size by calling CIRCUIT-SAT. In fact, by the remarks above, we can actually find such an $x$ in polynomial size. Therefore, the lexically first inverse of $\{f_i\}$ is computable in polynomial size.

(2) *implies* (1) Suppose that SAT does not have polynomial-size circuits. Consider the function $\mathrm{PROJ}_{m,n}$ from $\{0,1\}^m \times \{0,1\}^n$ to $\{0,1\}^m \cup \{\#\}$ defined as follows: if $\rho^{-1}(y)$ is a formula of $n$ variables that outputs 1 on input $x$, then set $\mathrm{PROJ}_{m,n}(y,x)=y$; otherwise, set $\mathrm{PROJ}_{m,n}(y,x)=\#$. Clearly $\mathrm{PROJ}_{m,n}$ is computable in polynomial size. On the other hand, the range of $\mathrm{PROJ}_{m,n}$ is precisely the set of $y$ such that $\mathrm{SAT}_{m,n}(y)=1$. Thus even the domain of each inverse of $\mathrm{PROJ}_{m,n}$ requires more than polynomial size to compute, so $\mathrm{PROJ}_{m,n}$ is a one-way function. ∎

The proof method of Theorem 2.1 can be used to establish a stronger result, applying to finite functions having arbitrary circuit size, which we now describe. For a function $f$ from $\{0,1\}^n$ to $\{0,1\}^m \cup \{\#\}$, define the circuit complexity of $f$ by $C(f)=\min\{\mathrm{size}(C): C \text{ computes } f\}$, and the circuit complexity of inverting such a function by $C^{-1}(f)=\min\{C(g): g \text{ is an inverse of } f\}$. Now define the *measure of one-wayness* $M(f)$ of a finite function by $M(f)=\log(C^{-1}(f))/\log(C(f))$.

The point of this measure is that $M(f) \geqslant k$ if and only if $C^{-1}(f) \geqslant C(f)^k$. Thus $M(f)$ is unbounded if and only if a superpolynomial increase in the number of gates is needed in computing each inverse function of $f_i$ for a suitable family of functions $\{f_i\}$, with *no restriction* on the number of gates needed for the $f_i$. We have the following result.

THEOREM 2.1a. *The following are equivalent:*

(1) *The measure of one-wayness $M(f)$ is unbounded.*

(2) *The satisfiability problem SAT does not have polynomial-size circuits.*

*Proof.* (1) *implies* (2) Suppose that $\mathrm{SAT}_{m,n}$ has circuits of size $p(m, n)$ bounded by a polynomial in both $m$ and $n$; we show that $M(f)$ is bounded by a constant. Let $f$ be a function from $\{0, 1\}^n$ to $\{0, 1\}^m \cup \{\#\}$, and let $g$ be the lexically first inverse of $f$. Define predicates $Q_i$ by

$$Q_i(w_1, ..., w_i, y) \equiv \exists x_{i+1} \cdots \exists x_n [f(w_1, ..., w_i, x_{i+1}, ..., x_n) = y],$$

where $w_1, ..., w_i \in \{0, 1\}$ and $y \in \{0, 1\}^m$. We will show that the predicates $Q_i$ can be used to construct small circuits for $g$. Consider the following algorithm for computing $g(y)$:

```
if $Q_0(y)$ then
  for $i \leftarrow 1$ to $n$ do
  if $Q_i(w_1, ..., w_{i-1}, 0, y)$ then $w_i \leftarrow 0$ else $w_i \leftarrow 1$
  endfor
  $g(y) \leftarrow (w_1, ..., w_n)$
else $g(y) \leftarrow \#$
```

It is easy to check that the above algorithm computes the lexically first inverse of $f$. Set $s = C(f)$. Provided that the predicates $Q_i$ have circuits of size polynomial in $s$, the above algorithm can be converted into a circuit computing $g$ of size polynomial in $s$; hence $M(f)$ would be bounded by a constant. We now construct small circuits for the predicates $Q_i$.

The predicate $Q_i$ can be easily computed with a nondeterministic circuit of size $s + O(m) = O(s)$ having $n - i \leqslant n$ nondeterministic variables. By adding dummy variables corresponding to the gates of this nondeterministic circuit and using Boolean conditions to force a correct simulation of the circuit, we can construct a nondeterministic Boolean formula computing $Q_i$ of size $O(s)$ and length $O(s \log s)$, having $O(s)$ nondeterministic variables. Using a $\mathrm{SAT}_{m',n'}$ circuit, where $m' = O(s \log s)$ and $n' = O(s)$, we can compute $Q_i$ with a deterministic circuit of size $p(O(s \log s), O(s))$. But this size is polynomial in $s$, so we are done.

(2) *implies* (1) Same as for Theorem 2.1. ▌

These results are complemented by previous work on the existence of one-way one-to-one functions. To state these results, we need some definitions. We call a function $f$ mapping $\{0, 1\}^n$ to $\{0, 1\}$ a Boolean function. Define PSIZE ("nonuniform P," sometimes denoted by P/poly) to be the set of families of Boolean functions that are computable in polynomial size. Define NPSIZE ("nonuniform NP") to be the set of families $\{f_i\}$ of Boolean functions for which there is a sequence of circuits $\{C_i\}$ such that $C_i$ has $n_i + p_i$ inputs (for some $p_i$ polynomial in $n_i$) and 1 output, has size polynomial in $n_i$, and, for all $x$ in $\{0, 1\}^{n_i}$, we have $f_i(x) = 1$ iff some $y$ in $\{0, 1\}^{p_i}$ satisfies $C_i(x, y) = 1$. Define UPSIZE ("nonuniform UP") to be the set of families $\{f_i\}$ of Boolean functions for which there is a sequence of circuits $\{C_i\}$ that satisfies the conditions for NPSIZE and furthermore for all $x$ in $\{0, 1\}^{n_i}$ there is at most one $y$ such that $C_i(x, y) = 1$. Let co-$A$ denote the set of families of Boolean functions whose negations are in the class $A$.

The following theorem is the nonuniform version of a result by Grollman and Selman (1984), and can be proved by adapting their argument to the nonuniform case.

THEOREM 2.2. *The following are equivalent:*

(1) *There exists a one-way family of one-to-one functions.*

(2) PSIZE $\neq$ UPSIZE.

The next theorem is also the nonuniform version of a result by Grollman and Selman, which again is proved following their argument.

THEOREM 2.3. *The following are equivalent:*

(1) *There exists a one-way family of one-to-one and onto functions.*

(2) PSIZE $\neq$ UPSIZE $\cap$ co-UPSIZE.

Recall that, with our conventions on the meaning of one-to-one and onto, a one-to-one onto function does not have to be a permutation. It is an open problem to give a necessary and sufficient condition for the existence of a one-way family of permutations in terms of separation of complexity classes.

## 3. CHECKING VERSUS EVALUATING

In this section we will consider questions concerning the relative complexity of checking versus evaluating. They were first raised by Valiant (1976). To formulate the questions we require a few definitions.

Given two sets $A$ and $B$, a *relation* $R$ between $A$ and $B$ is just a subset of the Cartesian product $A \times B$. The *domain* of $R$ is the set of $x$ in $A$ such that some $y$ in $B$ satisfies $(x, y) \in R$. The *range* of $R$ is the set of $y$ in $B$ such that some $x$ in $A$ satisfies $(x, y) \in R$.

A *family of relations* is a sequence of relations $\{R_i\}$ such that $R_i$ is a relation between $\{0, 1\}^{n_i}$ and $\{0, 1\}^{m_i}$, where $m_i$ and $n_i$ are within polynomial factors of each other. This family can be *checked* if there is a sequence of circuits $\{C_i\}$ such that the circuit $C_i$ has $n_i + m_i$ inputs and 1 output, has size polynomial in $n_i$, and for all $x$ in $\{0, 1\}^{n_i}$ and $y$ in $\{0, 1\}^{m_i}$, the pair $(x, y)$ is in $R_i$ if and only if $C_i(x, y) = 1$, i.e., the circuit $C_i$ tests membership in $R_i$. The family of relations $\{R_i\}$ can be *evaluated* if there is a sequence of circuits $\{C_i\}$ such that the circuit $C_i$ has $n_i$ inputs and $m_i$ outputs, has size polynomial in $n_i$, and for all $x$ in the domain of $R_i$ the ordered pair $(x, C_i(x))$ is in $R_i$.

Valiant (1976) asked questions of the form: can all relations (in a particular class) that can be checked also be evaluated? There is a relationship between one-way functions and checking versus evaluating questions. Define the *inverse* of a relation $R$ between $A$ and $B$ to be a new relation $R^{-1}$ between $B$ and $A$, defined by $(y,x) \in R^{-1}$ iff $(x,y) \in R$. Note that $R$ can be checked iff $R^{-1}$ can be checked. If $f$ is a one-way function, then $f^{-1}$ is a relation that can be checked but not evaluated. In fact, it is easy to see that $f$ is a one-way function if and only if $f$ can be checked and evaluated but $f^{-1}$ cannot be evaluated.

There are three complexity results for the checking versus evaluating problem parallel to the three complexity results (Theorems 2.1–2.3) for one-way functions given in Section 2. The first of these results is the nonuniform version of a result of Valiant (1976), which can be proved by his methods.

THEOREM 3.1. *The following are equivalent:*

(1) *Every family of relations that can be checked can also be evaluated.*

(2) *The satisfiability problem SAT has polynomial-size circuits.*

A relation $R$ is a *function* if for all $x$ in $A$ at most one $y$ in $B$ satisfies $(x, y) \in R$. The next theorem is also the nonuniform version of a result by Valiant, which his methods also establish.

THEOREM 3.2. *The following are equivalent:*

(1) *Every family of functions that can be checked can also be evaluated.*

(2) PSIZE = UPSIZE.

A relation $R$ between $A$ and $B$ is *total* if its domain is all of $A$, i.e., for all $x$ in $A$ at least one $y$ in $B$ satisfies $(x, y) \in R$. Valiant left open the question: can all total functions that can be checked also be evaluated? We give a necessary and sufficient condition for this to be true in the nonuniform case (the proof can be adapted to establish the uniform case as well).

THEOREM 3.3. *The following are equivalent:*

(1) *Every family of total functions that can be checked can also be evaluated.*

(2) PSIZE = UPSIZE $\cap$ co-UPSIZE.

*Proof.* (1) *implies* (2) Suppose every total function that can be checked can also be evaluated. Let $\{f_i\}$ be a family of functions in UPSIZE $\cap$ co-UPSIZE, where $f_i$ maps $\{0, 1\}^{n_i}$ into $\{0, 1\}$. Let $\{C_i\}$ be UPSIZE circuits for $\{f_i\}$ (with $p_i$ nondeterministic bits) and $\{C_i'\}$ be UPSIZE circuits for $\{\neg f_i\}$ (with $q_i$ nondeterministic bits). Define the function $g_i$ from $\{0, 1\}^{n_i}$ to $\{0, 1\} \times \{0, 1\}^{p_i} \times \{0, 1\}^{q_i}$ by: if $f_i(x) = 0$, then $g_i(x) = (0, 0^{p_i}, z)$, where $z$ is the unique solution to $C_i'(x, z) = 1$; otherwise $g_i(x) = (1, y, 0^{q_i})$, where $y$ is the unique value satisfying $C_i(x, y) = 1$.

Observe that $g_i(x) = (b, y, z)$ iff either $(b = 0 \wedge y = 0^{p_i} \wedge C_i'(x, z) = 1)$ or $(b = 1 \wedge z = 0^{q_i} \wedge C_i(x, y) = 1)$. This shows that the family $\{g_i\}$ can be checked. The $g_i$'s are total functions so, by our assumption, the family $\{g_i\}$ can be evaluated. But the first bit of $g_i$ equals $f_i$, so we can compute the family $\{f_i\}$ in polynomial size. Thus our assumption implies that PSIZE = UPSIZE $\cap$ co-UPSIZE.

(2) *implies* (1) Suppose that PSIZE = UPSIZE $\cap$ co-UPSIZE. Let $\{f_i\}$ be a family of total functions, where $f_i$ maps $\{0, 1\}^{n_i}$ into $\{0, 1\}^{m_i}$, that are checked by the polynomial-size circuits $\{C_i\}$. Let $y_j$ be the $j$th bit of $f_i$. Note that $y_j = 1$ iff some $z$ satisfies $C_i(x, z) = 1$ and $z_i = 1$, and that $y_j = 0$ iff some $z$ satisfies $C_i(x, z) = 1$ and $z_i = 0$. Since $f_i$ is single-valued, there exists at most one $z$ in either case. The first case shows that each bit of $f_i$ is in UPSIZE, whereas the second case shows that each bit of $f_i$ is in co-UPSIZE. But we assumed that PSIZE = UPSIZE $\cap$ co-UPSIZE, so each bit of $f_i$ can be computed in polynomial size. Thus the family $\{f_i\}$ can be evaluated. ▌

Let $\{R_i\}$ be a family of relations such that $R_i$ is a relation between $\{0, 1\}^{n_i}$ and $\{0, 1\}^{m_i}$, where $n_i$ and $m_i$ are within polynomial factors of each other. The family $\{R_i\}$ can be *nondeterministically checked* if there is a sequence of circuits $\{C_i\}$ such that the circuit $C_i$ has $n_i + m_i + p_i$ inputs (for some $p_i$ polynomial in $n_i$) and 1 output, has size polynomial in $n_i$, and for all $x$ in $\{0, 1\}^{n_i}$ and $y$ in $\{0, 1\}^{m_i}$, the pair $(x, y)$ is in $R_i$ iff some $z$ satisfies $C_i(x, y, z) = 1$.

Long (1981) studied the uniform version of the statement "all total *relations* that can be *nondeterministically* checked can be evaluated." Long showed that this statement is true if and only if gamma reducibility (see Adleman and Manders (1977)) is equivalent to polynomial-time many–one reducibility. We define the nonuniform versions of these reducibilities below.

A *family of languages* is a sequence $\{A_i\}$ such that $A_i$ is a subset of $\{0, 1\}^{n_i}$. Let $\{A_i\}$ and $\{B_i\}$ be two families of languages such that $A_i \subseteq \{0, 1\}^{n_i}$ and $B_i \subseteq \{0, 1\}^{m_i}$, where $n_i$ and $m_i$ are within polynomial factors of each other. We say that $\{A_i\}$ is *polynomial-size many–one reducible* to $\{B_i\}$ (written as $\{A_i\} \leqslant_m^p \{B_i\}$) if there is a family of functions $\{f_i\}$ such that

(i) $f_i$ maps $\{0, 1\}^{n_i}$ into $\{0, 1\}^{m_i}$,

(ii) $\{f_i\}$ is computable in size polynomial in $n_i$, and

(iii) for all $i$ and all $x$ in $\{0, 1\}^{n_i}$, we have $x \in A_i$ iff $f_i(x) \in B_i$.

We say that $\{A_i\}$ is *gamma reducible* to $\{B_i\}$ (written as $\{A_i\} \leqslant_\gamma \{B_i\}$) if there is a family of relations $\{R_i\}$ such that

(i) $R_i$ is a relation between $\{0, 1\}^{m_i}$ and $\{0, 1\}^{n_i}$,

(ii) $\{R_i\}$ can be nondeterministically checked,

(iii) $R_i$ is total, and

(iv) for all $i$ and all pairs $(x, y) \in R_i$, we have $x \in A_i$ iff $y \in B_i$.

The following theorem is the nonuniform version of a result of Long (1981), and can be proved by adapting the proof of Theorem 3.5 below.

THEOREM 3.4. *The following two statements are equivalent:*

(1) *All families of total relations that can be nondeterministically checked can also be evaluated.*

(2) *Polynomial-size many-one reducibility is equivalent to gamma reducibility.*

Valiant asked the following question: can all total *relations* that can be checked also be evaluated? Valiant showed that a sufficient condition for this to be true is that SAT have polynomial-size circuits, and a necessary condition is that PSIZE = NPSIZE ∩ co-NPSIZE. We give below a necessary and sufficient condition.

We say that $\{A_i\}$ is *beta reducible* to $\{B_i\}$ (written as $\{A_i\} \leqslant_\beta \{B_i\}$) if there is a family of relations $\{R_i\}$ such that

(i) $R_i$ is a relation between $\{0, 1\}^{n_i}$ and $\{0, 1\}^{m_i}$,

(ii) $\{R_i\}$ can be checked,

(iii) $R_i$ is total, and

(iv) for all $i$ and all pairs $(x, y) \in R_i$, we have $x \in A_i$ iff $y \in B_i$.

Beta reducibility differs from gamma reducibility only in that the witness relations $\{R_i\}$ are required to be checked instead of just nondeterministically checked.

THEOREM 3.5. *The following two statements are equivalent:*

(1) *All families of total relations that can be checked can also be evaluated.*

(2) *Polynomial-size many-one reducibility is equivalent to beta reducibility.*

*Proof.* (1) *implies* (2) Suppose that (1) is true. Let $\{A_i\}$ and $\{B_i\}$ be two families of languages. We will show that $\{A_i\} \leqslant_m^p \{B_i\}$ iff $\{A_i\} \leqslant_\beta \{B_i\}$. The reduction $\{A_i\} \leqslant_m^p \{B_i\}$ always implies the reduction $\{A_i\} \leqslant_\beta \{B_i\}$, so we just need to show the reverse implication. Suppose that $\{A_i\} \leqslant_\beta \{B_i\}$; let $\{R_i\}$ be the family of witnessing relations promised by the definition of beta reducibility. Since $\{R_i\}$ is total and can be checked, the assumption that (1) is true implies that $\{R_i\}$ can be evaluated, i.e., there is a sequence of polynomial-size circuits $\{C_i\}$ satisfying $(x, C_i(x)) \in R_i$ for all $i$ and $x$. The definition of beta reducibility implies that for all $i$ and $x$, we have $x \in A_i$ iff $C_i(x) \in B_i$, which in turn implies that $\{A_i\}$ is polynomial-size many-one reducible to $\{B_i\}$. This completes the first half of the theorem.

(2) *implies* (1) Suppose that (2) is true. Let $\{R_i\}$ be a family of total relations that can be checked. We will show that $\{R_i\}$ can be evaluated.

We reduce to the case that $R_i$ is one-to-one, i.e., for all $i$ and $y$ there is at most one $x$ such that $(x, y) \in R_i$. If $R_i$ is not one-to-one, then set $R_i'$ equal to $\{(x, (x, y)): (x, y) \in R_i\}$. The relation $R_i'$ is one-to-one, and it is obvious that if $R_i'$ can be evaluated, then $R_i$ can be evaluated as well. Thus we assume in what follows, without loss of generality, that $R_i$ is one-to-one.

Given a set $A_i \subseteq \{0, 1\}^{n_i}$, let $B(R_i, A_i)$ denote the set of $y$ in $\{0, 1\}^{m_i}$ such that some $x$ in $A_i$ satisfies $(x, y) \in R_i$. Since $R_i$ is one-to-one, it is easy to see that every family $\{A_i\}$ is beta reducible to $\{B(R_i, A_i)\}$, with $\{R_i\}$ witnessing the reduction. Hence by hypothesis the family $\{A_i\}$ is polynomial-size many-one reducible to $\{B(R_i, A_i)\}$.

Let $g_i$ be the unique function from $\{0, 1\}^{m_i}$ to $\{0, 1\}^{n_i} \cup \{\#\}$ such that $(g_i(y), y) \in R_i$ for $y$ in the range of $R_i$, and $g_i(y) = \#$ otherwise. For a subset $A_i$ of $\{0, 1\}^{n_i}$, we see that $y \in B(R_i, A_i)$ iff $g_i(y) \in A_i$ for all $y$ in $\{0, 1\}^{m_i}$. Note that $\{g_i\}$ need not be computable in polynomial size. We will need the following result.

Claim 3.6. *Let $R_i$ be a one-to-one, total relation. Then there is an $A_i \subseteq \{0, 1\}^{n_i}$ such that for all circuits $C$, the condition*

$$\forall x [x \in A_i \Leftrightarrow C(x) \in B(R_i, A_i)] \qquad (*)$$

*implies that*

$$|\{x \in \{0, 1\}^{n_i}: (x, C(x)) \notin R_i\}| < 10 \cdot \text{size}(C) \cdot \log_2(\text{size}(C) + 1). \qquad (**)$$

*Proof.* The proof is by a probabilistic argument. Choose $A_i$ uniformly at random from $\{0, 1\}^{n_i}$. We will show that $A_i$ satisfies the claim with positive probability. The probability that there is a circuit $C$ of size $s$ that satisfies $(*)$ but not $(**)$ is bounded above by (the number of circuits of size $s$) times (the probability that a particular circuit satisfies $(*)$ but not

(**)). The number of circuits of size $s$ is bounded above by $s^{3s}$ (see Savage (1976)).

Let $C$ be a circuit of size $s$ that does not satisfy (**). By definition of $B(R_i, A_i)$, the condition (*) can be rewritten as

$$\forall x[x \in A_i \Leftrightarrow g_i(C(x)) \in A_i].$$

Define an undirected graph $G$ with vertex set $V = \{0, 1\}^{n_i} \cup \{\#\}$, having edges between vertices $x$ and $g_i(C(x))$ for all $x$ in $\{0, 1\}^{n_i}$. The circuit $C$ satisfies (*) iff every two vertices connected by an edge in $G$ are either both in $A_i$ or both not in $A_i$ (where # is considered to not be in $A_i$). The probability that $C$ satisfies (*) is thus exactly $2^{a-|V|}$, where $a$ is the number of connected components of $G$. It is easy to see that $a \leqslant |V| - b/2$, where $b$ is the number of edges in $G$ that are not self-loops. If $(x, C(x)) \notin R_i$, then $x$ has an edge that is not a self-loop associated with it. Since $C$ does not satisfy (**), we must have $b \geqslant 10s\log_2(s+1)$. Thus the probability that $C$ satisfies (*) is at most $2^{-5s\log_2(s+1)} = (s+1)^{-5s}$.

The probability that there is a circuit of size $s$ that satisfies (*) but not (**) is thus at most $s^{3s} \cdot (s+1)^{-5s} \leqslant (s+1)^{-2s}$. The sum of $(s+1)^{-2s}$ over all $s \geqslant 1$ is still less than 1. Hence there exists an $A_i$ that satisfies the claim. ▌

For our family of relations $\{R_i\}$, let $\{A_i\}$ be the family of languages promised by the above claim. We know that $\{A_i\} \leqslant_m^p \{B(R_i, A_i)\}$, so some sequence of circuits $\{C_i\}$ of size polynomial in $n_i$ witnesses the reduction. Set $p_i$ equal to the size of $C_i$. By the above claim, the circuit $C_i$ also satisfies (**), i.e., we have

$$|\{x: (x, C_i(x)) \in R_i\}| > 2^{n_i} - 10p_i \log_2(p_i + 1).$$

The circuit $C_i$ evaluates $R_i$ almost everywhere—it fails on less than $10p_i\log_2(p_i+1)$ inputs. But by constructing a lookup table for these inputs, we can construct a circuit $D_i$ of size $O(p_i \log p_i)$ that evaluates $R_i$ exactly. This completes the proof of the theorem. ▌

## 4. One-Way Permutations for Constant-Depth Circuits

In this section we will give two examples of provably one-way functions for constant-depth circuits. Throughout this section, a *circuit* will have alternating levels of AND gates and OR gates of unbounded fanin and fanout. The inputs are Boolean variables and their negations. The *depth* of a circuit is the number of alternating levels in it.

A *family of permutations* is a sequence $\{f_i\}$ such that $f_n$ is a permutation of $\{0, 1\}^n$. The family $\{f_i\}$ is *one-way for constant-depth circuits* if it can be computed in polynomial size and constant depth, but its inverse $\{f_i^{-1}\}$ cannot.

Let $B_n$ be the mapping from $\{0, 1\}^n$ to $\{0, 1, ..., 2^n - 1\}$ obtained by interpreting binary strings as natural numbers, i.e., set $B_n(x_1, x_2, ..., x_n)$ equal to $\sum_{i=1}^{n} x_i 2^{n-i}$. Let $\sigma_n: \{0,1\}^n \to \{0,1\}^n$ be defined by $\sigma_n(x) = B_n^{-1}(3B_n(x) \bmod 2^n)$.

THEOREM 4.1. *The sequence $\{\sigma_n\}$ is a one-way family of permutations for constant-depth circuits.*

*Proof.* The function $\sigma_n$ is a permutation because $\gcd(3, 2^n) = 1$. We first show that $\{\sigma_n\}$ has polynomial-size constant-depth circuits. Chandra *et al.* (1984) showed that adding two $n$-bit numbers can be done in constant depth and polynomial size. Since $\sigma_n(x) = B_n^{-1}(B_n(x) + B_n(x) + B_n(x) \bmod 2^n)$, we need only two additions to compute $\sigma_n(x)$. Thus $\{\sigma_n\}$ can be computed in polynomial size and constant depth.

We next show that $\{\sigma_n^{-1}\}$ requires more than polynomial size to compute in constant depth. Let $\pi_n: \{0, 1\}^n \to \{0, 1\}$ be defined by setting $\pi_n(x_1, x_2, ..., x_n)$ equal to 1 if the number of $x_i$'s that are 1 is divisible by 3, and equal to 0 otherwise. We show that $\{\pi_n\}$ can be reduced in constant depth to $\{\sigma_n^{-1}\}$. Since the methods of Ajtai (1983) and Furst *et al.* (1984) (see also Yao, 1985 and Hastad, 1987) show that $\{\pi_n\}$ cannot be computed in constant depth and polynomial size, this will complete the proof.

Observe that $B_n(\sigma_n^{-1}(y)) \leqslant \lfloor 2^n/3 \rfloor$ if and only if $B_n(y)$ is divisible by 3. But $B_n(y)$ is divisible by 3 if and only if $y_1 - y_2 + \cdots + (-1)^n y_n$ is divisible by 3, where $y = (y_1, y_2, ..., y_n)$. Thus $\pi_n(x_1, x_2, ..., x_n) = 1$ if and only if $B_n(\sigma_n^{-1}(x_1, 0, x_2, 0, x_3, ..., 0, x_n)) \leqslant \lfloor 2^n/3 \rfloor$. Comparison of two $n$-bit numbers can be computed in polynomial size and constant depth (see Chandra *et al.*, 1984); thus we have a constant depth reduction from $\{\pi_n\}$ to $\{\sigma_n^{-1}\}$. ∎

A family of permutations $\{f_n\}$ is *bit-wise one-way for constant-depth circuits* if it can be computed in constant depth and polynomial size, but no bit of its inverse $\{f_n^{-1}\}$ can.

Let $\tau_n: \{0, 1\}^n \to \{0, 1\}^n$ map the string $(x_1, x_2, ..., x_n)$ to the string $(y_1, y_2, ..., y_n)$, where $y_i = x_i \oplus x_{i+1}$ for $1 \leqslant i \leqslant n-1$, and $y_n = x_1 \oplus x_{\lfloor n/2 \rfloor} \oplus x_n$, and with $\oplus$ denoting addition modulo 2.

THEOREM 4.2. *The sequence $\{\tau_n\}$ is a bit-wise one-way family of permutations for constant depth circuits.*

*Proof.* The function $\tau_n$ is a permutation. In fact, it is straightforward to check that its inverse is given by $\tau_n^{-1}$ $(y_1, y_2, ..., y_n) = (x_1, x_2, ..., x_n)$, where $x_i = (y_1 \oplus \cdots \oplus y_{i-1}) \oplus (y_{\lfloor n/2 \rfloor} \oplus \cdots \oplus y_n)$ for $1 \leqslant i \leqslant \lfloor n/2 \rfloor$, and $x_i = (y_1 \oplus \cdots \oplus y_{\lfloor n/2 \rfloor - 1}) \oplus (y_i \oplus \cdots \oplus y_n)$ for $\lfloor n/2 \rfloor \leqslant i \leqslant n$. The function $\tau_n$ can be computed with linear-size depth 2 circuits, since each bit of $\tau_n$ depends on at most three variables. On the other hand, note that each bit of $\tau_n^{-1}$ is the modulo-2 sum of at least $\lfloor n/2 \rfloor$ of the $y_i$. Thus the parity function is constant-depth reducible to each bit of $\tau_n^{-1}$. Since the parity function cannot be computed in polynomial size and constant depth (Ajtai, 1983 and Furst *et al.*, 1984), we are done. ∎

Two additional examples of one-way permutations for constant-depth circuits have since been discovered. Barrington (1985) has constructed a family of permutations that can be computed with constant-depth polynomial-size circuits, whose inverse is complete for LOGSPACE under first-order reductions (see Immerman, 1983, for the definition of first-order reductions). Even stronger, Hastad (1986) has constructed a family of permutations each of whose output bits depends on at most three input bits (like the permutation $\tau_n$ above), but whose inverse is complete for *P* under first-order reductions.

RECEIVED November 1985; ACCEPTED February 17, 1987

## REFERENCES

ADLEMAN, L., AND MANDERS, K. (1977), Reducibility, randomness, and intractibility, *in* "Proceedings, 9th Annual ACM Symposium on Theory of Computing, Boulder," pp. 151–163.

AJTAI, M. (1983), $\sum_1^1$-formulae on finite structures, *Ann. Pure Appl. Logic* **24**, 1–48.

AJTAI, M., AND WIGDERSON, A. (1987), Deterministic simulation of probabilistic constant depth circuits, *in* "Randomness and Computation" (S. Micali, Ed.), Advances in Computer Research Vol. 5, JAI Press, Greenwich; preliminary version, *in* "Proceedings, 26th Annual IEEE Symposium on Foundations of Computer Science, Portland, 1985," pp. 11–19.

BARRINGTON, D. (1985), personal communication.

BLUM, M., AND MICALI, S. (1984), How to generate cryptographically strong sequences of pseudo random bits, *SIAM J. Comput.* **13**, 850–864.

BRASSARD, G. (1983), Relativized cryptography, *IEEE Trans. Inform. Theory* **29**, 877–894.

CHANDRA, A. K., STOCKMEYER, L., AND VISHKIN, U. (1984), Constant depth reducibility, *SIAM J. Comput.* **13**, 423–439.

COOK, S. A. (1971), The complexity of theorem proving procedures, *in* "Proceedings, 3rd Annual ACM Symposium on Theory of Computing, Shaker Heights," pp. 151–158.

FURST, M., SAXE, J. B., AND SIPSER, M. (1984), Parity, circuits, and the polynomial time hierarchy, *Math. Systems Theory* **17**, 13–28.

GROLLMAN, J., AND SELMAN, A. L. (1984), Complexity measures for public-key cryptosystems, *in* "Proceedings, 25th Annual IEEE Symposium on Foundations of Computer Science, Singer Island," pp. 495–503.

HASTAD, J. (1986), One-way permutations in $NC^0$, submitted to *Inform. Process. Lett.*

HASTAD, J. (1987), Improved lower bounds for small depth circuits, *in* "Randomness and

Computation" (S. Micali, Ed.), Advances in Computer Research Vol. 5, JAI Press, Greenwich; preliminary version, *in* "Proceedings, 18th Annual ACM Symposium on Theory of Computing, Berkeley, 1986," pp. 6–20.

IMMERMAN, N. (1983), Languages which capture complexity classes, *in* "Proceedings, 15th Annual ACM Symposium on Theory of Computing, Boston," pp. 347–354.

KARP, R. M., AND LIPTON, R. J. (1982), Turing machines that take advice, *Enseign. Math.* **28**, 191–209; preliminary version, *in* Some connections between non-uniform and uniform complexity classes, "Proceedings, 12th Annual ACM Symposium on Theory of Computing, Los Angeles, 1980," pp. 302–309.

LEVIN, L. A. (1987), One-way functions and pseudorandom generators, *Combinatorica*, in press; preliminary version, *in* "Proceedings, 17th Annual ACM Symposium on Theory of Computing, Providence, 1985," pp. 363–365.

LONG, T. J. (1981), On $\gamma$-reducibility versus polynomial time many-one reducibility, *Theoret. Comput. Sci.* **14**, 91–101.

ODLYZKO, A. (1985), Discrete logarithms in finite fields and their cryptographic significance, *in* "Advances in Cryptology—Proceedings of Eurocrypt 84" (T. Beth, N. Cot, and I. Ingemarsson, Eds.), Lecture Notes in Computer Science Vol. 209, pp. 224–314, Springer-Verlag, Berlin.

SAVAGE, J. E. (1976), "The Complexity of Computing," Wiley, New York.

SELMAN, A. L. (1984), Remarks about natural self-reducible sets in NP and complexity measures for public key cryptosystems, preprint.

VALIANT, L. (1976), Relative complexity of checking and evaluating, *Inform. Process. Lett.* **5**, 20–23.

YAO, A. C. (1982), Theory and applications of trapdoor functions, *in* "Proceedings, 23rd Annual IEEE Symposium on Foundations of Computer Science, Chicago," pp. 80–91.

YAO, A. C. (1985), Separating the polynomial-time hierarchy by oracles, *in* "Proceedings, 26th Annual IEEE Symposium on Foundations of Computer Science, Portland," pp. 1–10.