ONEWAY PERMUTATIONS IN NC^0

Johan Hastad*
Laboratory of Computer Science, MIT

Abstract: We prove that there is a family of permutations which are computable by a family of NC^0 circuits while their inverses are P-hard to compute. Thus these permutations are oneway from the point of view of parallel computation.

Keywords: Oneway permutation, P-completeness, NC

1. Permutations that are hard to invert.

A family of circuits $(C_n)_{n=1}^{\infty}$ is said to belong to NC^0, if for every n, C_n has n inputs and the depth of C_n is bounded by a constant K which is independent of n and the fanin of any gate in the circuit is 2. We will extend the notion slightly by having a family $(C_{i,n})_{n=1}^{\infty}{}_{i=1}^{n}$ where $C_{i,n}$ has n inputs and the depth of any $C_{i,n}$ is bounded by K. Observe that the restriction implies that the size of any $C_{i,n}$ is bounded by 2^K. We will in this paper investigate permutations from Σ^* to Σ^* which for each n takes Σ^n to Σ^n and such that the ith output bit is computed by $C_{i,n}$ for NC^0 family of circuits, and we call such a permutation a NC^0 permutation. We say that such a family of circuits is LOGSPACE uniform if there is a Turing machine M which on input 1^n operates in $O(\log n)$ workspace and outputs the n NC^0 circuits which defines the permutation from Σ^n to Σ^n.

Previous work has been done to determine the complexity of inverting permutations of the above type. Boppana and Lagarias [2] proved that there are permutations which were computable in NC^0 but whose inverses were as hard to compute as parity. These permutations can be said to be oneway since parity is known not to be computable even by unbounded fanin polynomial-size circuits [5]. Barrington [1] gave another example of a oneway function which was computable in AC^0, but computing its inverse was LOGSPACE-complete. We prove the following stronger result.

Theorem: *There is a LOGSPACE-uniform family of NC^0 permutations which are P-complete to invert.*

Proof: We will reduce the problem of evaluating a straight line program to the problem of inverting an NC^0 permutation. Since the former problem is well known to be P-complete [6] the latter will be P-hard. We will use the term P-complete to mean P-complete under LOGSPACE-reductions. Thus if a P-complete problem is in LOGSPACE every problem in P is in LOGSPACE. In a similar manner a problem is defined to be P-hard if it has the above property but is not known to be in P.

* Supported by an IBM Postdoctoral fellowship

Let us set up some notation for the straight line program. The program contains a sequence of variables which are either defined as a part of the input or in terms of two previously defined variables. To make this formal let i_k and j_k be two indices which are less than k and let f_k be arbitrary functions of two Boolean inputs to one Boolean output.

Using this notation we define an instance of straight line program.

INPUT: Boolean variables $x_1, x_2, \ldots x_n$
PROGRAM: $x_k = f_k(x_{i_k}, x_{j_k}), \quad k = n+1, n+2, \ldots m$
OUTPUT: Value of x_m.

We will reduce this problem to the question of inverting an NC^0 permutation. Let us denote the permutation by g. It will be defined from $\{0,1\}^m$ to $\{0,1\}^m$ where m is the number of variables occurring in the straight line program. Let $z_1, z_2, \ldots z_m$ denote the input bits and $g_1, g_2 \ldots g_m$ the output bits. Let $\oplus$ be exclusive or.

Then,

$$g_k(z) = z_k \quad k = 1, 2, \ldots, n$$

$$g_k(z) = z_k \oplus f_k(z_{i_k}, z_{j_k}) \quad k = n+1, n+2, \ldots m$$

where i_k, j_k and the functions f_k are the same as in the straight line program.

Let us establish that the reduction is correct.

Fact 1 g is a permutation.

We need only show that g is onto. Given $y \in \{0,1\}^m$ find $z \in \{0,1\}^m$ such that $g(z) = y$ by solving for $z_1, z_2, \ldots z_m$ in increasing order. This can be done since the equations can be written $z_k = y_k \oplus f(z_{i_k}, z_{j_k})$.

Fact 2 The m'th bit of $g^{-1}(x_1, x_2 \ldots x_n, 0, \ldots, 0)$ is the output of the straight line program.

Solving for this input as described above performs the computation of the straight line program.

Fact 3 The reduction from straight line programs to permutations is effective and g is computable by NC^0 circuits.

The reduction is trivial computationally since it just replaces equality signs by $\oplus$. The second part of Fact 3 follows from the fact that any function that only depends on a constant number of inputs can be computed in NC^0.

To prove the theorem we need to establish that there is a uniform family of straight line programs which are P-complete to evaluate. To see this let us recall the reduction from any problem to a straight line program.

Let M be a Turing machine which solves a P-complete problem. The standard reduction to a straight line program uses the *computation tableaux* of M on input x which is a matrix $(m^x_{i,j})$ where $m^x_{i,j}$ contains information about the the ith square of M's tape at time step j. In particular $m^x_{i,j}$ contains the following information: the content of the square, whether the head is there and in this case which state the machine is in. Each $m^x_{i,j}$ will correspond to a constant number of variables of the straight line program using a suitable coding scheme. Now since $m^x_{i,j}$ only depends on $m^x_{i-1,j-1}$, $m^x_{i,j-1}$, and $m^x_{i+1,j-1}$ it is easy to make a short piece of straight line code which computes the variables corresponding to

$m^x_{i,j}$ from the variables corresponding to $m^x_{i-1,j-1}$, $m^x_{i,j-1}$, and $m^x_{i+1,j-1}$. The code for this computation is identical for all i and j and hence the only thing the machine constructing the straight line program has to remember is the index of the variables and this can be done in space $O(\log n)$. The input x enters as bits specifying $m^x_{i,0}$. This consludes the proof of the teorem. ∎

We know ([3], [4], [7], [9]) that in the sequential setting the existence of oneway functions implies the existence of good cryptosystems.

There are two obvious obstacles to using the present results to construct parallel cryptosystems.

The first problem is that the function needs to be hard to invert on a random input. This is not quite achieved since even if we start with a straight line program which is hard to compute for a random input it is not necessarily true that the corresponding NC^0 permutation is hard to invert for random outputs. This is so as our reduction only maps to values of the permutation whose last $m - n$ bits are 0.

The second problem is that the reductions from oneway functions to cryptographic generators is sequential i.e. even if the oneway function is easy to compute in parallel the resulting cryptographic generator will require large parallel time. For a discussion of "parallel cryptography" we refer to Reif and Tygar [8].

We have proved that there is a sequence of uniform NC^0 circuits which are P-complete to invert. An interesting open question is whether inverting every uniform family of NC^0 permutations is in P.

Acknowledgment: I would like to thank Mike Sipser and David Barrington for fruitful discussions.

2. References

[1] Barrington D. personal communication 1985

[2] Boppana R. and Lagarias J. "One way functions and circuit complexity" in Structure in Complexity Theory *Lecture Notes in Computer Science 223* eds G. Goos and J. Hartmanis, pp 51-66.

[3] Blum M. and Micali S. "How to generate Cryptographically Strong Sequences of Pseudo Random Bits", *Proceedings of 23'rd IEEE symposium on Foundations of Computer Science* 1982 pp 112-117. Also *SIAM J. on Computing* 13 (1984) pp 850-864.

[4] Goldwasser S. and Micali S., "Probabilistic Encryption" *Journal of Computer and System Sciences* Vol. 28, No.2 pp 270-299.

[5] Furst M., Saxe J. and Sipser M., "Parity, Circuits, and the Polynomial Time Hierarchy", *Proceedings of 22nd Annual IEEE Symposium on Foundations of Computer Science*, 1981,pp 260-270.

[6] Ladner, R.E., "The Circuit Value Problem is Log Space Complete for P", SIGACT News, vol. 7, No 1, Jan 1975 (18-20).

[7] Levin, L.A., "One-Way Functions and Pseudorandom Generators", *Proceedings of the*

17'th Annual ACM Symposium on Theory of Computing 1985 pp 363-365.

[8] Reif J.H. and Tygar J.D., "Efficient Parallel Pseudo-Random Number Generation", presented at Crypto 85.

[9] Yao A. "Theory and Applications of Trapdoor Functions", *Proceedings of 23'rd IEEE Symposium on Foundations of Computer Science* (1982) pp 80-91.