Characteristic Polynomial Method for Verification and Test of Combinational Circuits

Vishwani D. Agrawal
David Lee

AT&T Bell Laboratories
Murray Hill, NJ 07974

Abstract — *This paper gives a new and efficient method of determining the equivalence of two given Boolean functions. We define a characteristic polynomial directly from the sum-of-product form of the logic function. The polynomial contains a real variable corresponding to each Boolean variable. Logical operations on the Boolean function correspond to arithmetic operations on the polynomial. We show that if the characteristic polynomials of two Boolean functions, when evaluated at the same randomly sampled values of their variables, produce identical result then the two corresponding Boolean functions are identical with probability* 1. *In a typical application, one characteristic function may be derived from the truth table specification while the other is obtained from a logic implementation. The proposed method is very efficient as it allows to prove correctness by just one evaluation of the two polynomials. We further show that when the real variables in the polynomial are restricted to the range* $[0,1]$, *the value of the polynomial is the same as the probability of the Boolean function producing a true output. This result is applied to testing of combinational circuits. We derive the length of a random test sequence that will detect any fault in the circuit with any given arbitrarily high probability.*

1. INTRODUCTION

The subject of research reported here is closely related to a recent paper by Jain *et al* [3]. In that paper, the authors probabilistically establish the equivalence between two given Boolean functions. They assign randomly selected integers to the input variables and compute integer-valued transform functions. If the evaluations give the same value, the Boolean functions are shown to be identical with a low probability of error. The error probability is reduced as the domain from which the integers are obtained is enlarged. Also, for a fixed domain, the probability of error can be reduced by taking multiple samples for inputs.

There are two main differences in the present work. First, we assign randomly selected *real* numbers to the input variables. We show that when the characteristic polynomials of two Boolean functions, computed for a random input, give the same value then the functions are identical with probability 1. This result is valid for any finite real domain from which inputs are randomly assigned. Second, when the inputs are sampled from the real domain [0,1], and are interpreted as probabilities of logic 1, then the corresponding value of the characteristic polynomial gives the probability of output logic 1 of the Boolean function. This observation provides a new look at random or syndrome testing. Furthermore, it leads to a stopping criterion for random testing with a confidence measure.

2. CHARACTERISTIC POLYNOMIALS AND THEIR EVALUATION

2.1. Notation

We use small letters for Boolean variables with values in $\{0, 1\}$: x_i, $i = 1, \cdots, n$, and Boolean vectors $\vec{x} = (x_1, \cdots, x_n)$. Capital letters are used for real variables with values in R: X_i, $i = 1, \cdots, n$, and real vectors $\vec{X} = (X_1, \cdots, X_n)$. Similarly, small letters are for Boolean functions $f : \{0, 1\}^n \to \{0, 1\}$ and capital letters for real functions $F : R^n \to R$.

2.2. Characteristic Polynomials

We consider Boolean functions (expressions) of n variables $\vec{x} = (x_1, \cdots, x_n)$. Two Boolean functions are identical $f \equiv g$ if and only if for all $\vec{x} \in \{0, 1\}^n$ $f(\vec{x}) = g(\vec{x})$.

A Boolean function f can be uniquely represented by a truth table. The standard *sum of products (SOP)* form of f can be obtained from the given truth table by taking a minterm for each combination of variables which produces a 1 in the function, and then taking the OR of all those minterms:

$$f = E_1 \vee E_2 \vee \cdots \vee E_k \tag{2.1}$$

where each minterm is

$$E_j = \alpha_{j,1} \wedge \alpha_{j,2} \wedge \cdots \wedge \alpha_{j,n}, \; j = 1, \cdots, k,$$

and

$$\alpha_{j,i} = x_i \;\; or \;\; x_i{}'$$

Note that x' and $\overline{x}$ are interchangeably used in this paper to denote the complement of x. If we replace each Boolean variable x_i by a real variable X_i, $x_i{}'$ by $1 - X_i$, AND operation by product, and OR by summation, then we obtain a real valued polynomial of n variables. Specifically, we construct a real valued polynomial by substitutions:

$$\begin{array}{c} x_i \rightarrow X_i \\ x_i{}' \rightarrow 1 - X_i \\ \vee \rightarrow + \\ \wedge \rightarrow \cdot \end{array} \tag{2.2}$$

We obtain a real valued polynomial of n variables: $F(\vec{X}) = F(X_1 , \cdots , X_n)$.

For a Boolean function f, the corresponding polynomial F is unique, and we call it the *characteristic polynomial* of the given Boolean function. If $g \equiv f$ then the two Boolean functions f and g have the same truth table, standard form, and hence the same characteristic polynomial. We denote this transformation from a Boolean function f to its characteristic polynomial F by

$$F = \tau(f) \tag{2.3}$$

Proposition 2.1 below is a special case of the following general result. For any finite field, there is a unique embedding of Boolean functions into a polynomial ring over the field such that they have the same value when all the variables take values 0 or 1.

Proposition 2.1. Two Boolean functions f and g are identical $f \equiv g$ if and only if their characteristic polynomials are identical: $\tau(f) \equiv \tau(g)$.

Sketch of Proof. Only if. If $f \equiv g$ then $\tau(f) \equiv \tau(g)$.

If. For a constant Boolean vector $\vec{x^*}$, let $\vec{X^*}$ be a constant integer valued vector from a substitution in (2.2). For a Boolean function f, $f(\vec{x^*}) = 0$ if and only if $F(\vec{X^*}) = \tau(f)(\vec{X^*}) = 0$. Therefore, if $f \neq g$ then $\tau(f) \neq \tau(g)$. □

2.3. Evaluation of Characteristic Polynomials — A Greedy Method

We want to evaluate the characteristic polynomial of a Boolean function f: $F = \tau(f)$; given a constant vector $\vec{X^*} \in R^n$, we want to compute $F(\vec{X^*})$. A naive approach is to derive the standard form of F explicitly from f (using the truth table, for instance) and then evaluate $F(\vec{X^*})$. However, the number of terms in the standard form can be exponential in n in the worst case.

In general, evaluation of F is hard. More specifically, the following problem is NP-complete [6]: Given a Boolean function f of n variables and a constant real n-vector $\vec{X^*}$, whether $F(X^*) = 0$? We can reduce the satisfiability problem of Boolean expressions to this evaluation problem by taking $X_i^* = 0.5$, $i = 1, \cdots , n$. A given Boolean expression $f(x)$ is satisfiable if and only if $F(X^*) \neq 0$.

We now propose a greedy method for an evaluation of the characteristic polynomials without explicitly constructing them. Given a Boolean function f, its *Shannon expansion* is:

$$f(\vec{x}) = (x_1 \wedge f_{x_1=1}(\vec{x})) \vee (x_1{}' \wedge f_{x_1=0}(\vec{x})) \tag{2.4}$$

where $f_{x_1=1}(\vec{x})$ is obtained from $f(\vec{x})$ by assigning $x_1 = 1$ and $f_{x_1=0}(\vec{x})$ by assigning $x_1 = 0$. On the other hand, the characteristic polynomial

$$\begin{aligned} F(\vec{X}) &= \tau(f)(\vec{X}) \\ &= X_1 \cdot F_1(X_2,\cdots,X_n) + (1-X_1) \cdot F_0(X_2,\cdots,X_n) \end{aligned} \tag{2.5}$$

where F_1 is the characteristic polynomial of $f_{x_1=1}(\vec{x})$ and F_0 is the characteristic polynomial of $f_{x_1=0}(\vec{x})$. For a constant real vector $\vec{X^*}$,

$$F(\vec{X^*}) = X_1^* \cdot F_1(X_2^*,\cdots,X_n^*) + (1-X_1^*) \cdot F_0(X_2^*,\cdots,X_n^*) \tag{2.6}$$

From (2.4) to (2.6), the evaluation of the characteristic polynomial F of f is reduced to the evaluations of the characteristic polynomials of $f_{x_1=1}$ and $f_{x_1=0}$. We then evaluate the two polynomials of $n - 1$ variables, and continue recursively until we have one variable left. Based on this observation, we propose the following method.

For a Boolean expression, we define its *size* as the number of its *literals*. We recursively ($m = n$, $n-1$, $\cdots$ 1) use the Shannon expansion to reduce the evaluation of a characteristic polynomial of m variables to evaluations of two characteristic polynomials of $m - 1$ variables. At each step we determine which variable to expand on. For each variable x_i, we compute the sum of the sizes of the following two Boolean expressions: $f_{x_i=1}$ and $f_{x_i=0}$. We choose for the expansion in (2.4)-(2.6) the variable x_i with a minimal sum of sizes of the two corresponding characteristic polynomials of $m - 1$ variables. The rationale is: the expansion of that variable results in the computation of two Boolean expressions of a minimal total size.

It is often necessary to deal with logic networks described as interconnection of Boolean gates. The Shannon expansion method, discussed here, can be applied to such networks also. A good heuristic is to expand with respect to the variables that fanout and then reconverge. Also, in large circuits, partitioning may be necessary. Partitioning into *supergates*, as applied to signal probability calculation, is applicable to the calculation of the characteristic polynomial [5]. Other methods, as discussed by Jain *et al* [3] and those based on binary decision diagrams [2], can also be used.

2.4. Signal Probability Formulation

The following approach gives a different perception and hence a different evaluation method. Assume that $0 \le X_i^* \le 1$, $i = 1, \cdots, n$. Suppose that X_i^* is the probability that $x_i = 1$. Then for each minterm in (2.1):

$$E_j = \alpha_{j,1} \wedge \alpha_{j,2} \wedge \cdots \wedge \alpha_{j,n} \ , \ j = 1, \cdots, k,$$

$\tau(E_j)(\vec{X}^*)$ is the probability that minterm takes Boolean value 1. Taking the summation, $F(\vec{X}^*) = \tau(f)(\vec{X}^*)$ is the probability that the Boolean function f takes value 1:

Proposition 2.2. Let $0 \le X_i^* \le 1$ be the probability that the Boolean variable x_i takes value 1, $i = 1, \cdots, n$. Then $\tau(f)(\vec{X}^*)$ is the probability that the Boolean function f takes value 1. □

Therefore, an evaluation of the characteristic polynomial is identical to computing the probability for f to take value 1. When all inputs are made equiprobable, i.e., all input variables are assigned the value 0.5, the output probability is simply the fraction of 1's in the truth table. The number of 1's in the truth table is also known as the *syndrome* and has been used in testing [4]. The methods for calculating the output 1 probability for any given input probabilities have been discussed in the literature [5].

3. FUNCTIONAL TESTING (VERIFICATION)

Given two Boolean function $f(\vec{x})$ and $g(\vec{x})$, we want to verify if $f \equiv g$. From Proposition 2.1, this is the case if and only if their characteristic polynomials are identical: $\tau(f) \equiv \tau(g)$. For a constant real vector $\vec{X}^*$, if $\tau(f)(\vec{X}^*) \ne \tau(g)(\vec{X}^*)$ then definitely the two polynomials are different and consequently $f \ne g$. However, if $\tau(f)(\vec{X}^*) = \tau(g)(\vec{X}^*)$ then the two polynomials (and hence the two given Boolean functions) may or may not be identical. We now show that it is ''very likely'' that they are identical if we sample uniformly at random.

Example 3.1. Consider two Boolean functions:

$$f = a \wedge b$$

$$g = \overline{a \vee b}$$

where a and b are Boolean variables. The functions are AND and NOR, respectively. The corresponding characteristic polynomials are:

$$\tau(f) = F(A,B) = AB$$
$$\tau(g) = G(A,B) = (1-A)(1-B)$$

where A and B are real numbers. Suppose we evaluate the two polynomials at a randomly sampled point (A^*,B^*) and find that the two polynomials give different values, then we correctly conclude: *f and g are different*. However, the two polynomials will give the same value when:

$$A^* B^* = (1-A^*)(1-B^*) \quad or \quad A^* + B^* = 1$$

Given any continuous domain of real numbers, the probability of picking a sample that lies on the straight line $A^* + B^* = 1$ is almost 0. The situation is illustrated in Figure 1.

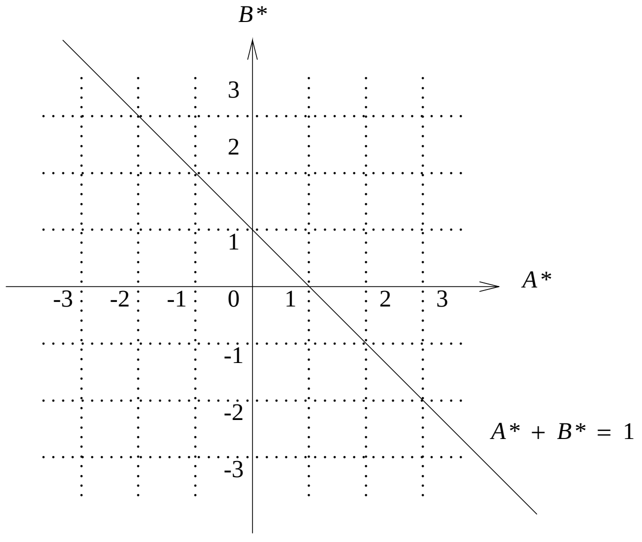

Fig. 1. Sample space for Example 3.1.

Following the approach of [3], if we restrict the random sampling to integers (grid points in Figure 1), we make an error whenever a point on the line $A^* + B^* = 1$ is sampled. Thus, the error probability is determined by the number of such points in the sample space. This is given in Table 1.

Table 1 - Error probability for integer inputs in Example 3.1.

Sample space for A^*, B^*	Error Probability
0,1	$\frac{2}{4} = 0.50$
-1,0,1	$\frac{2}{9} = 0.22$
-2,-1,0,1,2	$\frac{4}{25} = 0.16$
-3,-2,-1,0,1,2,3	$\frac{6}{49} = 0.12$

This example illustrates the influence of the random sampling procedure on the error probability. Sampling from a continuous real domain gives a vanishingly small error probability. On the other hand, if we sample in the integer domain, the error probability reduces as the domain is enlarged. As indicated in [3], the error probability can also be reduced without increasing the domain but by taking multiple samples.

Proposition 3.1. Suppose that D is a compact set in R^n with a Lebesgue measure μ and $\mu(D) = 1$. Given a polynomial $F(\vec{X})$ of n variables that are not identically zero, the algebraic variety of F is $V = \{\vec{X} : F(\vec{X}) = 0\}$. Then $\mu(V) = 0$. □

Corollary 3.1. Suppose that D is a compact set in R^n with a Lebesgue measure μ and $\mu(D) = 1$. Given two polynomial $F(\vec{X})$ and $G(\vec{X})$ of n variables, if $F \neq G$ then they have the same value on a subset of D of measure zero. □

Corollary 3.1 has an interesting implication. We want to determine whether two Boolean functions f and g are identical. Suppose that we can sample uniformly at random (according to the Lebesgue measure) in D and obtain $\vec{X}^* \in D$. We compute $\tau(f)(\vec{X}^*)$ and $\tau(g)(\vec{X}^*)$. If they are different then definitely $f \neq g$. Otherwise, we claim that they are identical. The only case our claim is incorrect is that $f \neq g$ and the sample $\vec{X}^*$ is in the algebraic variety of $\tau(f) - \tau(g)$, which is of measure zero. Therefore, the probability that we make an incorrect claim is zero. We summarize:

Algorithm 3.1. (Verification)

input: two Boolean functions f and g of n variables;
output: whether they are identical.

```
begin
    find a compact subset $D \subseteq R^n$;
    sample uniformly at random from D and obtain $\vec{X}^* \in D$;
    if ($eval(f, \vec{X}^*) \neq eval(g, \vec{X}^*)$)
        then return "$f \neq g$";
    else return "$f \equiv g$";
end
```

Proposition 3.2. The probability that Algorithm 3.1 returns an incorrect answer is zero. □

For an easy implementation, we can take an n-cylinder for D: $-\infty < a_i < b_i < +\infty$, $i = 1, \cdots, n$, or a unit n-cube where $a_i = 0$ and $b_i = 1$. To sample uniformly at random from an n-cylinder, we can just sample uniformly at random from each interval $[a_i, b_i]$ independently for X_i^*, $i = 1, \cdots, n$, and obtain a sample $\vec{X}^*$.

Example 3.2. Consider the functions f and g shown in Figure 2. The function f is a logic AND function of variables a, b and c. Its characteristic polynomial is $F(A,B,C) = ABC$. The function g is a multiplexer function with characteristic polynomial, $G(A,B,C) = AB + (1-B)C$. In the latter case, the polynomial is obtained by Shannon expansion as explained in Section 2.3. It is advantageous to expand about the fanout variables (input variable b in these example circuits.)

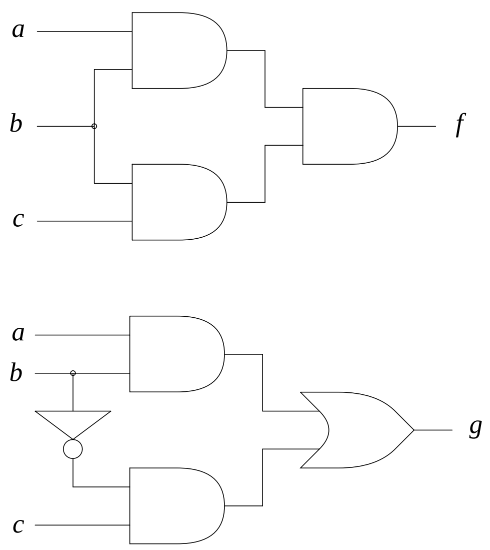

Fig. 2. Circuits of Example 3.2.

Consider a random sample of real numbers from $[-1,1]$: $A^*=0.75$, $B^*=0.30$ and $C^*=-0.65$. We get, $F(A^*,B^*,C^*) = -0.4625$ and $G(A^*,B^*,C^*) = -0.23$. Although we have not checked all cases, it is unlikely that any two or more of the 2^8 Boolean functions of three variables will have the same value for the characteristic polynomial at this point. For inputs from the integer set [0,1], the functions f and g will appear identical with probability $5/8 = 0.625$. For the increased range $[-1,0,1]$, the error probability becomes $10/27 = 0.370$.

4. TESTING

We have a *specification* Boolean function $f(\vec{x})$ of n variables and we know f explicitly. We have an *implementation* Boolean function $g(\vec{x})$ which is a ‘‘black box’’; for an input Boolean n-vector $\vec{x}^*$, it outputs a Boolean value $g(\vec{x}^*)$ but we do not know its internal structure. The *testing* problem is to determine whether $g \equiv f$.

A naive approach is to test on the implementation g all possible 2^n input vectors. If for every input vector $\vec{x}$ $g(\vec{x}) = f(\vec{x})$ then $g \equiv f$. Otherwise, there is a discrepancy on at least one of them, and $g \neq f$. However, for large n, which is typical in practice, this is infeasible. Due to the large number of inputs, there is a variety of techniques in circuit testing relying on fault models that reduce the complexity of testing [1].

4.1. Basic Idea

We sample uniformly at random from the unit n-cube and obtain $\vec{X}^*$ where $0 \leq X_i^* \leq 1$, $i = 1, \cdots, n$. For the specification function f, we evaluate its characteristic polynomial and obtain $\tau(f)(\vec{X}^*)$. Suppose that we could also compute $\tau(g)(\vec{X}^*)$ then, from Corollary 3.1 and Proposition 3.2, we conclude the implementation g is correct if and only if the two values are the same. Unfortunately, it is impossible to evaluate $\tau(g)(\vec{X}^*)$; we do not even know $g(\vec{x})$. However, we can estimate $\tau(g)(\vec{X}^*)$ and then determine whether g is faulty.

4.2. Estimation of $\tau(g)(\vec{X}^*)$

We take X_i^* as the probability that variable x_i takes value 1, $i = 1, \cdots, n$. Then $\tau(g)(\vec{X}^*)$ is the probability that $g(\vec{x})$ takes value 1. This observation leads to an estimation of $\tau(g)(\vec{X}^*)$. We run each test as follows: Take 1 for Boolean variable x_i with probability X_i^*, $i = 1, \cdots, n$. This can be easily done by sampling uniformly at random from $[0,\ 1]$ to obtain q. If $q \geq X_i^*$ then $x_i = 1$ and 0, otherwise. For the chosen Boolean vector $\vec{x}$, we compute $g(\vec{x})$ using logic simulation of the circuit. We repeat N tests and suppose that for Q times g takes value 1. Then Q/N is an estimate of the probability that g takes value 1, which is $\tau(g)(\vec{X}^*)$.

4.3. Confidence

Obviously, we do not know $\tau(g)(\vec{X}^*)$ exactly; we only have an estimation. Our confidence increases as N becomes larger. Suppose that for $N \rightarrow \infty \quad Q/N \rightarrow p$ which is our estimation of $\tau(g)(\vec{X}^*)$. We then compare it with $\tau(f)(\vec{X}^*)$ and compute

$$\Delta = |p - \tau(f)(\vec{X}^*)| \tag{4.1}$$

If Δ is large then it is very likely that g is faulty. On the other hand, if Δ is small then it is very likely that g is correct. We need a threshold value for Δ; it should depend on the number of variables n and the number of tests N.

4.4. Cross Checking with Specification

For a test, we generate a Boolean vector $\vec{x}$ for testing on g and obtain $g(\vec{x})$. Naturally, we can compute $f(\vec{x})$ and compare that with $g(\vec{x})$. If there is a discrepancy, definitely g is faulty. On the other hand, if $f = g$ for all N tests, we need more information to conclude that g is not faulty.

Suppose that f takes value 1 Q' times. Then $p' = Q'/N$ is an estimate of $\tau(f)(\vec{X}^*)$ which we can compute. Therefore, when $\Delta' = |p' - \tau(f)(\vec{X}^*)|$is small, we are confident that we have run enough tests so that the estimate p of $\tau(g)(\vec{X}^*)$ is also accurate. This also serve as a stopping criterion for N in Section 4.3.

4.5. Cross Checking of Samples

Since we can only have an estimation of $\tau(g)(\vec{X}^*)$ the following cross checking provides more information of g. We sample uniformly at random from the unit n-cube a few times and obtain samples $\vec{X}^{*(1)}$, $\vec{X}^{*(2)}$, $\cdots$. We then compute the corresponding values of $\tau(f)$ and have p_1, p_2, $\cdots$. Among them we take two, the smallest p_l and the largest p_u, for testing as in Section 4.2. We have the corresponding estimates in (4.1): Δ_l and Δ_u. If either one is large, then we claim that g is faulty.

The reason we choose the two extreme probabilities is that the two test sets are less ‘‘correlated’’. Suppose that the probability that we make an incorrect claim is $e_i << 1$ for the ith test set. Then the probability we have an incorrect conclusion for the two test sets is $e_1 e_2$. Of course, more test sets could further increase our confidence.

4.6. Syndrome Testing

In syndrome testing [4], we count the occurrence of logic 1 state at the output of combinational logic, while all possible logic states are applied to inputs. For many types of faults in the circuit, the syndrome (1-count) will change. However, there are faults that can be masked and produce the same syndrome as the fault free circuit.

Consider the Boolean functions in Example 3.1. Both have the same number of 1’s in the truth table, and hence have the same syndrome. Indeed, if we assign 0.5 to the inputs, the two characteristic polynomials will give the same value. It shows that randomly selected input probabilities will reduce fault masking in the output probability.

5. CONCLUSION

The main idea in this paper is the definition of a real-valued function that provides a unique representation of a polynomial for a Boolean function. The probabilistic interpretation of the evaluation of the polynomial further points to new applications of signal probabilities in verification and test. We find that the use of real numbers in this context has a definite advantage over the use of integers that has been proposed previously. We have outlined several applications, namely, logic verification, random testing, and syndrome testing. However, much work is required in actually implementing these applications. Further, the issue of efficient evaluation of the characteristic polynomials for large combinational circuits remains to be further explored.

Logic verification and testing are difficult problems. We believe the new ideas given in this paper will have an impact on the future solutions of these problems.

REFERENCES

[1] V.D. Agrawal and S.C. Seth, *Test Generation for VLSI Chips,* IEEE Computer Society Press, Los Alamitos, CA, 1988.

[2] R.E. Bryant, ‘‘Graph-Based Algorithms for Boolean Function Manipulation,’’ *IEEE Trans. Comput.*, Vol. C-35, pp. 677-691, August 1986.

[3] J. Jain, J. Bitner, D.S. Fussell, and J.A. Abraham, ‘‘Probabilistic Verification of Boolean Functions,’’ *Formal Methods in System Design*, Vol. 1, pp. 63-117, 1992.

[4] J. Savir, ‘‘Syndrome-Testable Design of Combinational Circuits,’’ *IEEE Trans. Comput.*, Vol. C-29, pp. 442-451, June 1980, (Also see, pp. 1012-1013, November 1980).

[5] S.C. Seth and V.D. Agrawal, ‘‘A New Model for Computation of Probabilistic Testability in Combinational Circuits,’’ *Integration, the VLSI Journal*, Vol. 7, pp. 49-75, 1989.

[6] M. Yannakakis, *Private Communication.*