A PROBABILISTIC REMARK ON ALGEBRAIC PROGRAM TESTING

Richard A. DEMILLO
School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA

Richard J. LIPTON
Computer Science Department, Yale University, New Haven, CT 06520, USA

Received 8 August 1977; revised version received 27 March 1978

Software reliability, program testing

Until very recently, research in software reliability has divided quite neatly into two – usually warring – camps: methodologies with a mathematical basis and methodologies without such a basis. In the former view, "reliability" is identified with "correctness" and the principle tool has been formal and informal verification [1]. In the latter view, "reliability" is taken to mean the ability to meet overall functional goals to within some predefined limits [2,3]. We have argued in [4] that the latter view holds a great deal of promise for further development at both the practical and analytical levels. Howden [5] proposes a first step in this direction by describing a method for "testing" a certain restricted class of programs whose behavior can – in a sense Howden makes precise – be *algebraicized*. In this way, "testing" a program is reduced to an equivalence test, the major components of which become

(i) a combinatorial identification of "equivalent" structures;

(ii) an algebraic test

$$f_1 \equiv f_2 ,$$

where f_i, $i = 1, 2$ is a multivariable polynomial (multinomial) of degree specified by the program being considered.

In arriving at a method for exact solution of (ii), Howden derives an algorithm, that requires evaluation of multinomials $f(x_1, ..., x_m)$ of maximal degree d at $O[(d + 1)^m]$ points. For large values of m (a typical case for realistic examples), this method becomes prohibitively expensive.

Since, however, a test for reliability rather than a certification of correctness is desired, a natural question is whether or not Howden's method can be improved by settling for less than an exact solution to (ii).

We are inspired by Rabin [6] and, less directly, by the many successes of Erdös and Spencer [7] to attempt a *probabilistic* solution to (ii). Using these methods, we show that (ii) can be tested with probability of error ϵ * with only $O(g(\epsilon))$ evaluations of multinomials, where g is a slowly growing function of only ϵ. In particular, 30 or so evaluations should give sufficiently small probability of error for most practical situations. The remainder of this note is devoted to proving this result.

Let us denote by $P_{\neq 0}(m, d)$ the class of multinomials

$$f(x_1, ..., x_m) \not\equiv 0$$

(over some arbitrary but fixed integral domain) whose degree does not exceed $d > 0$. We define

$$P(m, d, r) = \min \operatorname{Prob}\{1 \leqslant x_1 \leqslant r, f(x_1, ..., x_m) \neq 0\}$$

$$f \in P_{\neq 0}(m, d) .$$

* See Rabin's account of algorithms that may err with fixed probability [6].

We think of $P(m, d, r)$ as the minimal relative frequency with which witnesses to the non-nullity of a multinomial of the appropriate kind can occur in the chosen interval. We will derive a lower bound p for $P(m, d, r)$. Then $(1 - p)$ is an upper bound on the error in selecting a random point from the m-cube. We then iterate the procedure by t independent random selections to obtain a small probability of error $(1 - p)^t$. Notice, in particular, that since a polynomial of degree d has at most d roots (ignoring multiplicity), the largest probability of finding a root must be at least the probability of finding a root by randomly sampling in the interval $1 \leqslant x_1 \leqslant r$; thus

$$P(1, d, r) \geqslant 1 - d/r .$$

Now, consider some

$$f(x_1, ..., x_m, y) \not\equiv 0$$

of degree at most d. But there are then multinomials $\{g_i\}_{i<d}$, not all $\not\equiv 0$, such that

$$f(x_1, ..., x_m, y) = \sum_{i=0}^{d} g_i(x_1, ..., x_m)\, y^i .$$

Let us suppose that $g_k \in P_{\neq 0}(m, d)$. Thus

$$\mathrm{Prob}\{1 \leqslant x_i \leqslant r, f(x_1, ..., x_m, y) \neq 0\}$$
$$\geqslant \mathrm{Prob}\{\, g_k(x_1, ..., x_m) \neq 0 \text{ and } y \text{ is not a root}\}$$
$$\geqslant P(m, d, r)(1 - d/r) .$$

Continuing inductively, we obtain a lower bound in $P(m, d, r)$ as follows:

$$P(m, d, r) \geqslant (1 - d/r)^m . \tag{1}$$

But

$$\lim_{m\to\infty} (1 - d/r)^m = \lim_{m\to\infty} \left[1 + \frac{1}{m}\left(\frac{-dm}{r}\right)\right]^m$$
$$= \exp(-dm/r) \tag{2}$$

Combining (1) and (2), we have for large m, $r = dm$,

$$P(m, d, dm) \geqslant \mathrm{e}^{-1} .$$

Thus, with t evaluations of f for independent choices of points from the m-cube with sides $r = dm$, the probability of missing a witness to the non-nullity of $f(x_1, ..., x_m)$ is at most

$$(1 - \mathrm{e}^{-1})^t .$$

Table 1 shows the probable error in testing $f \equiv 0$ by t evaluations of f at randomly chosen points for some typical values of d, m, r, t. Notice that for $dm = r$, $t = 30$, this is already $<10^{-5}$.

References

[1] Z. Manna, Mathematical Theory of Computation (McGraw-Hill, New York, 1974).

Table 1
Probable error in testing $f(x_1, ..., x_m) \equiv 0$ (degree $\leqslant d$) by t random evaluations in $\{1, ..., r\}$

$[1 - P(m,d,r)]^t$

dm	r	$t + 10$	$t = 20$	$t = 30$	$t = 50$	$t = 100$
10	10	10×10^{-3}	106×10^{-6}	1×10^{-6}	109×10^{-12}	12×10^{-21}
20	10	233×10^{-3}	54×10^{-3}	13×10^{-3}	695×10^{-6}	483×10^{-9}
50	10	935×10^{-3}	873×10^{-3}	816×10^{-3}	713×10^{-3}	509×10^{-3}
10^2	10	~1	~1	~1	~1	~1
10	10^2	61×10^{-12}	$<10^{-20}$	$<10^{-20}$	$<10^{-20}$	~0
20	10^2	38×10^{-9}	1×10^{-15}	$<10^{-20}$	$<10^{-20}$	~0
50	10^2	88×10^{-6}	8×10^{-9}	704×10^{-15}	$<10^{-20}$	$<10^{-20}$
10^3	10^2	~1	~1	~1	~1	~1
10	10^3	$<10^{-20}$	$<10^{-20}$	$<10^{-20}$	~0	~0
20	10^3	9×10^{-18}	$<10^{-20}$	$<10^{-20}$	~0	~0
50	10^3	76×10^{-15}	$<10^{-20}$	$<10^{-20}$	~0	~0

[2] J.R. Brown, M. Lipow, Testing for software reliability, Intern. Conf. in Reliable Software, SIGPLAN Notices, 10, b, (June 1975) 518–527.
[3] A.I. Llewelyn, R.F. Wilkins, The testing of computer software, 1969 Conf. on Software Engineering, 189–199.
[4] R.A. DeMillo, R.J. Lipton, A.J. Perlis, Social processes and proofs of theorems and programs, Fourth ACM Symposium in Principles of Programming Languages (to appear in CACM).
[5] W.E. Howden, Algebraic program testing, Computer Science Technical Report No. 14 (November 1976) UC-San Diego, La Jolla, CA.
[6] M.O. Rabin, Probabilistic algorithms, in: J. Traub, ed., Algorithms and Complexity (Academic Press, New York, 1976) 21–40.
[7] P. Erdös, J. Spencer, Probabilistic Methods in Combinatorics (Academic Press, New York, 1974).