Technical Report CS76003- R

Finiteness and Bounds of Complete Test Point Sets for Program Verification

by

Johannes J. Martin

Virginia Polytechnic Institute
and
State University

Abstract

It is proven that there exists a finite set of test points that suffices to establish the equivalence of two programs s and t if some finite set S of programs can be identified that contains both s and t. It is also proven that the expected number of those test points is bounded by the logarithm of the cardinality of S.

Key Words and Phrases: program testing, correctness proofs, existence of test points, number of test points.

CS categories: 4.42, 5.24

Introduction

We define a program s to be correct with respect to some specification t if, for some given domain, s and t are equivalent, that is, describe the same function. By this definition, program testing, the conventional debugging method, is believed unfit to prove program correctness unless it is done exhaustively (for examples in the more recent literature see references [1,2]). Exhaustive testing is, of course, almost always practically impossible.

However, the requirement that testing must be exhaustive is difficult to understand if we interpret testing as a process that gathers information about the program tested. Clearly, the program as a finite string of symbols represents a finite amount of information. Further, with every well-chosen test we should learn something new about the program. Hence, with some finite number of tests we should be able to obtain all information represented.

The purpose of this paper is to prove that this is true; in fact, we will be able to show that the sufficient number of test points is linearly related to the length of the program and its specification.

We will obtain this result by separating the information theoretical question concerning the sufficient number of test points from the question that haunts all methods of program verification: the decidability of the equivalence of two given descriptions [3].

As a consequence, our test points may have two severe shortcomings:

a) if the program s is not equivalent to the specification t it may happen that s never halts for certain test values;

b) although we know that there exists a complete test point set for any program (i.e. a test point set sufficient to prove correctness) we may have no effective method for finding all of the test points.

Nevertheless, we feel that our results build a basis for further work in the area of program testing and we trust that nontrivial, practically interesting classes of programs (and specifications) can be found for which (i) complete sets of test points can be computed and (ii) bounds n can be determined such that, for any test point, the program s will terminate in not more than n steps if it is equivalent to its specification t.

The sufficient number of tests

We regard a program as a string of symbols over some alphabet T. We assume that such a program specifies in some well defined way a sequence of computational steps. Applied to values x in some domain X a program computes, if it halts, values y of some range Y.

The function f: $X \to Y$ computed by a program s will be denoted by A(s), the value of A(s) at some point x will be denoted by s(x). We will call two programs s and t equivalent, denoted $s \equiv t$, if both are applicable to the same domain X and A(s) = A(t), or if both s and t are meaningless.

Theorem 1

Let S be a finite set of programs and $s \in S$ then there exists a finite set $V \subseteq X$ such that $\forall\, t \in S\ (\forall\, x \in V\ s(x)=t(x)) \Rightarrow s \equiv t$.

Proof

The equivalence relation '$\equiv$' partitions S into equivalence classes f. The set of all such equivalence classes F_0 has the cardinality $N_0 = |F_0| \leq |S|$. In order to prove or disprove that s and t are equivalent we wish to derive a sequence of sets

$F_0 \supset F_1 \supset \ldots \supset F_k$ such that

$F_k = \{A(s)\}$ if $s \equiv t$ or $F_k = \phi$ otherwise.

We obtain F_{i+1} from F_i by the condition

$F_{i+1} = \{f | f \in F_i \wedge f(x_{i+1}) = s(x_{i+1}) = t(x_{i+1})\}$ with $x_{i+1} \in V$.
We ensure that F_{i+1} is a proper subset of F_i by choosing x_{i+1} such that $\exists(f,g \in F_i)\ f(x_{i+1}) \neq g(x_{i+1})$. Obviously such x must exist unless F_i contains only one function and, thus, we know that $s \equiv t$.
Clearly, $k \leq |F_0|$. □

Although theorem 1 assures us that there exist k test points that suffice to prove equivalence of two programs it does not offer a favorable upper bound for k. In fact, if $S = \{s | s \in T^* \wedge |s| \leq n\}$ with $n = MAX(|s|,|t|)$ (and, hence, $|S| \simeq |T|^n$) then the proof of the theorem only shows that $k \leq |F_0| \leq |T|^n$. This bound is quite useless since even for moderate values of $|T|$ and n (say 50 and 100) a set of $|T|^n$ elements is practically infinite for it can not be exhaustively processed.

Therefore, theorem 1 merely asserts that the sufficient number of test points does not depend on the cardinality of the program's domain or range but on the size of the program itself.

Theorem 2

The expected value of the cardinality of V is $\overline{k} \leq \lg_2 |F_0|$.

Proof

Suppose the set F_{i+1} obtained from F_i by the i+1st test contains $\alpha \cdot |F_i|$ elements with $\alpha < 1$. We want to show that with the proper choice of x_{i+1} we obtain $\alpha \leq 1/2$ for all but possibly one function in F_i.

In order to do so we consider two verification problems:

i) we assume that two programs, s and t, are equivalent and compute the function $f_0 \in F_{i+1}$;

ii) we assume that two other equivalent programs, p and q, compute a function $f_1 \in F_i - F_{i+1}$.

Clearly the same point x_{i+1} could be used in both cases. In the first case it would isolate the set F_{i+1} and thus reduce the number of candidates by the factor α. In the second case it would isolate a set of $\overline{F}_{i+1} \subset F_i - F_{i+1}$ and thus reduce the number of candidates by at least the factor $1-\alpha$. Moreover, if x_{i+1} has been optimally chosen for f_0 i.e. such that α becomes minimal then any point $\overline{x} | f_0(\overline{x}) \neq f_m(\overline{x})$ with an arbitrary function f_m will split F_i in such a way that the subset which contains f_m does not have more than $(1-\alpha) \cdot |F_i|$ elements for otherwise $\overline{x}$ would be a better test point for f_0 than x_{i+1} which is impossible if x_{i+1} is optimal.

Now suppose $\alpha \geq 1/2$ and, hence, $1-\alpha \leq 1/2$ and observe that for every function $f_m \in F_i$ except f_0 there exists a test point (such as $\overline{x}$ in the above argument) which eliminates at least half of the candidates in F_i.

Now we will inductively establish the upper bound for the expected number of test points.

We conjecture that the expected number $\overline{k}$ of tests that remain after F_i is obtained, is bounded by $\overline{k} \leq \lg_2 |F_i|$ which, if true, proves the theorem.

The base case for $|F_i| = 1$ is trivially correct. Now consider $|F_i| = N_i > 1$. From the previous argument we know the next point that aims to verify f_m will reduce F_i to a set F_{i+1}^m with $|F_{i+1}^m| = \alpha_m \cdot N_i$ where $\alpha_m \leq 1/2$ for all but possibly one function $f_m \in F_i$. With the inductive hypothesis that the expected value for the number of tests remaining for F_{i+1}^m is bounded by $\overline{k}_m \leq \lg_2 \alpha_m + \lg_2 N_i$ we compute the expected value $\overline{k}$ for F_i to

$$\overline{k} = 1/N_i \cdot \sum_{m=0}^{N_i-1} (\overline{k}_m + 1) \leq 1/N_i \cdot \sum_m (\lg_2 \alpha_m + \lg_2 N_i + 1) = A + \lg_2 N_i + 1 \quad \text{with}$$

$$A = 1/N_i \cdot \sum_m \lg_2 \alpha_m .$$

Two cases emerge:

1) $\alpha_m \leq 1/2$ for all m. Here $A \leq -1$ and hence $\overline{k} \leq \lg_2 N_i$.

2) $\alpha_0 = 1/2+\varepsilon$, $\alpha_m \leq 1/2-\varepsilon \mid m \neq 0$ for some nonnegative $\varepsilon \leq 1/2$. Now

$$A \leq 1/N_i (\lg_2 1/2(1+2\varepsilon) + \sum_{m=1}^{N_i-1} \lg_2 1/2(1-2\varepsilon))$$

$$= 1/N_i (-1 + \lg_2 (1+2\varepsilon) - (N_i - 1) + \lg_2 (1-2\varepsilon)^{N_i-1})$$

$$= -1 + 1/N_i \cdot \lg_2 (1-4\varepsilon^2)(1-2\varepsilon)^{N_i-2} \leq -1. \quad \square$$

Remarks

As stated above, an upper bound for N_0 is $|S|$ which was assumed to be approximately $|T|^n$. Thus theorem 2 asserts that $\overline{k} \leq n \cdot \lg_2 |T|$. This bound is very conservative for two reasons.

1) Most strings s over T are meaningless (a random sequence of symbols is rarely an executable program). The remaining set is further partitioned into groups of programs applicable to the same domain and, finally, many programs in every such group are equivalent.

2) A test partitions the set considered into what could be termed classes of locally equivalent programs. In the proof of theorem 2 we have assumed that a test splits the set into two such classes. However, if the range R of a program is a set of $|R|$ values the number of classes that arise could possibly be $|R|$. As a result, the selection process would be greatly accelerated.

The next steps

In order to make the results presented practically useful the following steps must be taken.

1) A nontrivial class of programs must be identified for which complete sets of test points can be determined.

2) Methods for determining complete sets of test points for programs of this class must be found.

3) If such methods can not be found at least criteria must be found that decide if a given set of test points is complete.

4) A method of testing complex programs should be developed that proceeds by testing program components in a bottom-up fashion.

References

1. Boyer, R.S. SELECT-- A formal system for testing and debugging programs by symbolic execution. SIGPLAN Notices 10, 6 (June 1975), 234-245.

2. Huang, J.C. An approach to program testing. ACM Computing Surveys 7, 3 (Sept. 1975), 113-128.

3. Ritchie, R.W. Classes of predictably computable functions. Transactions of the American Mathematical Society 106 (1963), 139-173.