An Experimental Comparison of the Effectiveness of the All-uses and All-edges Adequacy Criteria

Phyllis G. Frankl * Stewart N. Weiss†

Abstract

An experimental comparison of the effectiveness of the all-uses and all-edges test data adequacy criteria was performed. A large number of test sets was randomly generated for each of nine subject programs with subtle errors. For each test set, the percentages of (executable) edges and definition-use associations covered were measured and it was determined whether the test set exposed an error. Hypothesis testing was used to investigate whether all-uses adequate test sets are more likely to expose errors than are all-edges adequate test sets. All-uses was shown to be significantly more effective than all-edges for five of the subjects; moreover, for four of these, all-uses appeared to guarantee detection of the error. Further analysis showed that in four subjects, all-uses adequate test sets appeared to be more effective than all-edges adequate test sets of the same size. Logistic regression showed that in some, but not all of the subjects there was a strong positive correlation between the percentage of definition-use associations covered by a test set and its error-exposing ability.

1 Introduction

Considerable effort in software testing research has focussed on the development of software test data adequacy criteria, that is, criteria which are used to determine when software has been tested "enough", and can be released. Numerous test data adequacy criteria have been proposed, including those based on control flow analysis [22, 23], data flow analysis [26, 28, 31], and program mutation [8]. Tools based on several of these criteria have been built [7, 14], and many theoretical studies of their formal properties and of certain aspects of their relation to one another have been done [6, 13, 15, 29, 31]. But surprisingly, relatively little work has focussed on the crucial question of how good adequate test sets (as determined by various adequacy criteria) are at exposing errors in programs.

In this paper, we describe an experiment which addresses this question. In Section 1.1, below, we define a notion of the effectiveness of an adequacy criterion, which, for a given erroneous program and specification, measures the likelihood that an adequate set will expose an error. The higher the effectiveness of criterion C, the more confidence we can have that a program which has been tested on a C-adequate test set without exposure of an error is indeed correct. The primary goal of this study was to measure and compare the effectiveness of several adequacy criteria for a variety of subject programs.

We limited our attention to three adequacy criteria, the all-edges criterion (branch testing), which is fairly widely used in practice, the all-uses criterion, which is considered promising by many testing researchers, and the "null criterion" (which deems any test set to be adequate). In addition to measuring the effectiveness of the criteria, our experiment design allowed us to address two related questions:

- If all-uses adequate test sets are more effective than all-edges adequate test sets, is this simply because they are bigger, or are they more effective even for test sets of a fixed size?
- What is the relationship between the percentage of definition-use associations covered by a test set and the test set's effectiveness?

*Author's address: Computer Science Dept., Polytechnic University, 333 Jay St., Brooklyn, N.Y. 11201. Supported by NSF Grant CCR-8810287 and by the New York State Science and Technology Foundation Center for Advanced Technology program.

†Author's address: Computer Science Dept., Hunter College, 695 Park Ave., New York, NY 10021. Supported by NSF Grant CCR-9008394 and by PSC-CUNY Award 661341.

We believe that the results reported here should be of interest both to testing researchers and to testing practitioners. While practitioners may be primarily interested in the experiment's results and their implications for choosing an adequacy criterion, researchers may also find the novel design of the experiment interesting.

In the remainder of this introduction, we define effectiveness, review the definitions of the relevant adequacy criteria, and survey related work. Section 2 describes the design of the experiment, Section 3 describes the statistical analysis techniques used, and Section 4 describes the subject programs on which the experiment was performed. The experiment results are presented in Section 5 and discussed further in Section 6.

1.1 Effectiveness of an adequacy criterion

Consider the following model of the testing process:

- A test set is generated using some test data generation technique.
- The program is executed on the test set, the outputs are checked, and the adequacy of the test set is checked.
- If at least one test case exposes an error, the program is debugged and regression tested; if no errors are exposed but the test set is inadequate, additional test cases are generated.
- This process continues until the program has been executed on an adequate test set which fails to expose an error.

At this point, the program is released. Although the program is not guaranteed to be correct, the "better" the adequacy criterion, the more confidence one can have that it is correct.

We now define a measure of the "goodness" of an adequacy criterion which captures this intuition. Let P be an incorrect program whose specification is S, and let C be an adequacy criterion. Consider all the test sets T which satisfy C for P and S. Some but not necessarily all of these test sets will expose an error. If a large percentage of the C-adequate test sets expose an error, then C is an effective criterion (for this program.)

More formally, we define the *effectiveness* of adequacy criterion C for a program P which is incorrect with respect to specification S, denoted $p_C(P,S)$, to be the probability that a C-adequate test set T will expose at least one error in P. Similarly, we can relativize this notion to a particular test generation strategy and define the *effectiveness of criterion C for P and S relative to test generation strategy G* to be the probability that a C-adequate test set generated by G will expose an error in P. Weiss has defined a more general notion of the effectiveness of an adequacy criterion [35], of which our definition is a special case.

Suppose that it has been established that for a wide variety of programs, criterion C1 is more effective than criterion C2. Let P be a program and let T be a test set for P. If P is incorrect, then the probability that T exposes an error is greater if T is C1-adequate than if T is only C2-adequate. Consequently, if T does not expose an error, the likelihood that P is incorrect is smaller if T is C1-adequate than if T is C2-adequate. Thus, the more effective a criterion is, the more confidence its adequate test sets can inspire in the correctness of the program under test.

Most previous comparisons of adequacy criteria have been based on investigating whether one criterion *subsumes* another. Criterion C1 subsumes criterion C2 if, for every program P and specification S, every test set which satisfies C1 also satisfies C2. It might seem, at first glance, that if C1 subsumes C2, then C1 is guaranteed to be more effective than C2 for every program. This is not the case. It may happen that for some program P, specification S, and test generation strategy G, test sets which only satisfy C2 may be better at exposing errors than those which satisfy C1. Hamlet has discussed related issues [20]. The relationship between subsumption and error-detecting ability is examined further in [16].

1.2 Definitions of the adequacy criteria

This study compares the effectiveness of three adequacy criteria: the all-edges criterion, the all-uses criterion, and the null criterion. Two of these criteria, all-edges and all-uses, are members of a family of criteria, sometimes called structured testing criteria, which are based on requiring the test data to cause the execution of representatives of certain sets of paths through the flow graph of the program under test. The null criterion considers any test set to be adequate; thus application of the null criterion is the same as not using any adequacy criterion at all. We have included the null criterion in this study in order to allow comparison of all-edges and all-uses adequate sets to arbitrary sets.

The all-edges criterion, also known as branch testing, demands that every edge in the program's flow graph be executed by at least one test case. All-edges and its variants (such as 80% branch coverage) are the most widely used formal test data adequacy criteria (other than the null criterion). All-edges is known to be a relatively weak criterion, in the sense that it is often easy to devise a test set which covers all the edges in a buggy program without exposing the bug.

Data flow testing criteria [26, 28, 31] require the test data to exercise paths from points at which variables are defined to points at which their values are subsequently used. Occurrences of variables in the program under test are classified as being either definitions, in which values are stored, or uses, in which values are fetched. The all-uses criterion [31] demands that the test data cover every *definition-use association* (dua) in the program, where a dua is a triple (d,u,v) such that d is a node in the program's flow graph in which variable v is defined, u is a node or edge in which v is used, and there is a definition-clear path with respect to v from d to u. A test case t *covers* dua (d,u,v) if t causes the execution a path which goes from d to u without passing through any intermediate node in which v is redefined. [1]

One problem with the all-edges and all-uses criteria, as originally defined, is that for some programs no adequate test set exists. This problem arises from infeasible paths through the program, i.e. paths which can never be executed. The problem is particularly serious for the all-uses criterion because for many commonplace programs (e.g. for any program having a **for** loop in which the lower and upper bounds are non-equal constants) no all-uses adequate test set exists. Frankl and Weyuker defined a new family of criteria, the *feasible data flow testing criteria*, which circumvents this problem by eliminating unexecutable edges or definition-use associations from consideration [13, 15], and showed that under reasonable restrictions on the program under test, all-uses* subsumes all-edges* (where the * denotes the feasible versions of the criteria). As pointed out above, this does not imply that all-uses* is more effective than all-edges* for a particular test generation strategy.

It is important to note that the original (unstarred) criteria are not really used in practice: they do not apply to those programs which have infeasible edges or duas, and they are the same as the starred versions for other programs. Ideally, testers should examine the program to eliminate infeasible edges and duas from consideration. In reality they often stop testing when some arbitrary percentage of the edges or duas has been covered, without worrying about whether the remaining edges/duas are infeasible, or whether they indicate deficiencies in the test set. For this reason, we felt that it was also important to examine the relationship between the percentage of the edges/duas covered by a test set and its likelihood of exposing an error. In the remainder of this paper, we will, by abuse of notation, refer to the all-edges* and all-uses* criteria as all-edges and all-uses, respectively.

[1] The formalism we use here is that given in [15].

1.3 Related Work

There have been several studies of the fault-detection ability of various test data *selection strategies*, but few studies of the fault-detection ability of *adequacy criteria*. Many previous investigations have been analyses [10, 33, 24] or simulations [9, 21, 32] using hypothetical programs, fault distributions, and testing criteria, but only a few have been experiments evaluating real testing strategies on real programs with naturally occurring faults [9, 17, 34].

As Hamlet pointed out [20], many previous experiments may give misleading results because they rely on a small number of test sets to represent each testing technique. While there have been several studies of related software quality issues which did use rigorous statistical techniques, e.g. [2, 25, 34], none of these were aimed at evaluating the effectiveness of adequacy criteria, and so their designs could not be used for our experiment.

2 Experiment Design

The goal of this experiment was to measure and compare the effectiveness of various adequacy criteria for several different subject programs. To measure the effectiveness of criterion C, we can

- generate a large number of C-adequate test sets,
- execute the subject program on each test set,
- check the outputs and consider a test set *exposing* if and only if the program gives the wrong output on at least one element of the test set, and
- calculate the proportion of C-adequate test sets which expose an error.

If the proportion of C1-adequate test sets which expose an error is significantly higher than the proportion of C2-adequate test sets which expose an error, we can conclude that criterion C1 is more effective than C2 for the given program and test generation strategy.

In the present experiment, we generated test sets randomly and compared the all-edges, all-uses, and null criteria. For each program, we identified the unexecutable edges and duas and eliminated them from consideration, then recorded the number of executable edges (duas) covered by each test set. This allowed us to measure the effectiveness of all-edges and all-uses and to investigate the correlation between percentage of edges (duas) covered and error exposing ability.

For each subject program, we first generated a large set of test cases called the universe. We then executed each test case in the universe, checked its output, recorded whether it was correct in a table, and saved a trace of the path executed by that test case (generated by ASSET [12, 14]). Finally, we selected and checked test sets of size S by randomly selecting S elements of the universe, looking them up in the table to see whether at least one of them exposed an error, and using ASSET to check how many executable edges or duas were not covered by any of the paths corresponding to the test set.

Some care was necessary in choosing appropriate test set sizes. If the generated test sets were too small, then relatively few of them would cover all edges (all duas), so the results would not be statistically significant. On the other hand, if the test sets were too large, then almost all of them would expose errors, making it difficult to distinguish the effectiveness of the two criteria. To overcome this problem, we generated our test sets in a sequence of "batches" where each batch contained sets of a fixed size. After observing which sizes were too large or too small, we generated additional batches in the appropriate size range, if necessary. This "stratification" of test sets by size also allowed us to investigate whether all-uses is more effective than all-edges for test sets of a given size.

The design of the experiment imposed several constraints on the subject programs.

- The input domain of the program had to have a structure which allowed for some reasonable way to generate the universe of test cases. For example, while there are several reasonable ways to randomly generate matrices, it is less clear how to randomly generate inputs to an operating system.
- Because of the large number of test cases, it was necessary to have some means of automatically checking the correctness of the outputs.
- The failure rate of the program had to be low; i.e. errors should be exposed by relatively few inputs in the universe. Otherwise, almost any test set which is big enough to satisfy all-edges will be very likely to expose an error. We were surprised to discover that many of the programs we considered as candidates for this experiment (including many which had been used in previous software quality studies) had to be rejected because of their high failure rates.

The available tool support (ASSET) imposed an additional constraint, namely, that the subject programs had to be either written in Pascal or short enough to translate manually. When translation was necessary, program structure was changed as little as possible.

Note that our experiment considers the effectiveness of all-edges adequate sets in general, *not* the effectiveness of those all-edges adequate sets which fail to satisfy all-uses. This models the situation in which the tester releases the program when it has passed an all-edges adequate test set without caring whether or not the test set also satisfies all-uses. In an alternative model, the tester would classify a test set as all-edges adequate only if it satisfied all-edges and did *not* satisfy all-uses.

Also note that our design introduces a bias in favor of all-edges. We used test set sizes which were big enough to insure the selection of a statistically significant number of all-uses adequate sets, *not* just a significant number of all-edges adequate sets. This resulted in the selection of many all-edges adequate sets which were bigger, thus more likely to expose an error, than the all-edges adequate sets which would be selected by a practitioner using our model of the testing process.

3 Data Analysis Techniques

Recall that we are interested in comparing the effectiveness of all-uses to all-edges and to the null criterion for *each* of a variety of subject programs. We treat each subject program's data as that of a separate experiment. Throughout this section, when we refer to the effectiveness of a criterion we mean its effectiveness for a particular program. For clarity, we describe the techniques used to compare all-uses with all-edges. The techniques for comparing all-uses to the null criterion are identical.

If we have randomly chosen N C-adequate test sets, and X is the number of these that exposed at least one error, then $\hat{p_C} = X/N$, the sample proportion, is a good estimator of p_C, the effectiveness of C. In fact, if the probability that a C-adequate test set exposes an error is governed by a binomial distribution, then $\hat{p_C}$ is a minimum variance unbiased estimator of the effectiveness of C [3].

3.1 Overall comparison of criteria

The first question posed was whether or not all-uses adequate test sets are significantly more effective than all-edges adequate test sets. Let $\hat{p_u}$ be the proportion of all-uses adequate sets which exposed an error and let $\hat{p_e}$ be the proportion of all-edges adequate sets which exposed an error. If $\hat{p_u}$ is significantly higher than $\hat{p_e}$ then there is strong statistical evidence that all-uses is more effective than all-edges. If not, the data do not support this hypothesis.

This observation suggests that hypothesis testing techniques are suitable for answering this question. In hypothesis testing, a research, or alternative, hypothesis is pitted against a null hypothesis, and the data are used to determine whether one hypothesis is more likely to be true than the other. Our research hypothesis, that all-uses is more effective than all-edges, is expressed by the assertion $p_e < p_u$. The null hypothesis, that the two criteria are equally effective, is expressed by $p_e = p_u$. It is important to realize that the goal in hypothesis testing is quite conservative; we uphold the null hypothesis as true unless the data is strong testimony against it, in which case we reject the null hypothesis in favor of the alternative.

Since we are using the sample proportions as estimators of the effectiveness of the criteria, our decision to accept or reject the null hypothesis reduces to a decision as to whether or not the difference between the sample proportions is significantly large. In particular, we should reject $p_e = p_u$ if and only if $\hat{p}_u - \hat{p}_e$ is greater than some prespecified *critical value.*

We use a standard statistical technique [5] for establishing the critical values for each experiment. Call a sample ***sufficiently large*** if there are at least five exposing and five unexposing test sets in both the all-edges and all-uses samples. For sufficiently large samples, the difference $\hat{p}_u - \hat{p}_e$ is approximately normally distributed, enabling us to calculate critical values, significance probabilities and confidence intervals for $p_u - p_e$. The significance probabilities indicate the strength of the evidence for rejection of hypotheses, and the confidence intervals give an indication of how much better one criterion is than the other, if at all. The lower the significance probability, the stronger the evidence that all-edges is less effective than all-uses. To be conservative in our interpretation of the data, we chose a significance level of $\alpha = 0.01$, meaning that if the null hypothesis is rejected, the probability that all-edges is *actually* as effective as all-uses is at most 1/100.

In several of our subjects, every all-uses adequate test set exposed an error, so that the normal approximation could not be used. In these cases, we calculated confidence intervals separately for p_u and p_e. Inspection of the data showed that all-uses was clearly more effective than all-edges for these subjects, making further analysis unnecessary.

3.2 Comparison of criteria for fixed size test sets

The second question we asked dealt with the effect of test set size on the previous results. The all-uses adequacy criterion in general requires larger test sets than does the all-edges criterion. Since the probability that a test set exposes an error increases as its size increases, for some subjects all-uses may be more effective than all-edges simply because it demands larger test sets. On the other hand, the increased effectiveness of all-uses may result from the way the criterion subdivides the input domain [24].

To determine whether differences in the effectiveness of the criteria were primarily due to differences in the sizes of adequate test sets, we analyzed the data on a "by-size" basis. In Table 5, we display the sample data for each of the subject programs by size, arranging close sizes into groups. The intent of this table is to give descriptive evidence of the relationship between all-uses and all-edges for fixed size test sets. Where there was enough data and $\hat{p}_u \neq 1$, we also did hypothesis testing on the individual size groups and reported the results in the right hand columns. Similar "by-size" analyses of all-edges and all-uses versus the null criterion are presented in [36].

3.3 Relationship between coverage and effectiveness

The last question to be answered is whether there is a relationship between the extent to which a test set satisfies the all-uses criterion and the probability that the test set will expose an error. This is the most difficult of the questions, and the technique we employed to answer it is logistic regression.

A regression model gives the mean of a response variable in a particular group as a function of the numerical characteristics of the group. If Y is the response variable and $X_1, X_2, \ldots X_n$ are the predictors, we denote the mean of Y, given fixed values $\bar{x} = x_1, x_2, \ldots x_n$, by $\mu_{Y|\bar{x}}$. Ordinary linear (or higher order) regression models are not suitable for data in which the response variable takes on yes-no type values such as "exposing" or "not exposing", in part because regression equations such as

$$\mu_{Y|\bar{x}} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_n x_n \qquad (1)$$

put no constraints on the value of $\mu_{Y|\bar{x}}$. The right hand side can take on any real value whereas the left hand side must lie between 0 and 1. The right hand side is also assumed to follow a normal distribution whereas the left hand side generally does not. There are other serious problems that make linear regression a poor choice for modeling proportions [1].

Logistic regression overcomes these problems and provides many important advantages as well. In logistic regression the left hand side of equation 1 is replaced by the *logit* of the response variable,

$$\log\left(\frac{\mu_{Y|\bar{x}}}{1-\mu_{Y|\bar{x}}}\right)$$

and the right hand side can be any real-valued function of the predictors.

In our analysis, we treated test set size and fraction of coverage of all-uses as the predictor variables, and used logistic regression to determine the extent, if any, to which the probability of exposing an error was dependent upon these variables. We used maximum likelihood estimation as the method of inference, and various chi square tests as measures of goodness of fit.

subject	edges	duas	exec edges	exec duas	failure rate
detm	78	298	74	103	0.001
find1	34	114	34	93	0.066
find2	34	114	34	93	0.018
matinv1	78	298	74	106	0.001
matinv2	30	81	30	62	0.001
strmtch1	13	49	13	49	0.032
strmtch2	14	56	14	54	0.062
textfmt	21	50	21	42	0.052
transpose	44	97	42	88	0.023

Table 1: Subject Programs

4 Subject Programs

Our nine subjects were obtained from seven programs, all of which had naturally occurring errors. Three of our programs were drawn from the Duran and Ntafos study [9]; high failure rates made the rest of the Duran-Ntafos subjects unsuitable for our experiment. We obtained more than one subject from two of the programs by using different input distributions and by instrumenting different procedures.

The subjects are described briefly below, and more thoroughly in [36]. Table 1 gives the numbers of edges, duas, executable edges, executable duas, and proportions of failure causing test cases in each universe.

Subjects **find1** and **find2** both have Buggyfind [4] as the subject program. Buggyfind takes as input an array of integers and an array index. In **find1**, the test universe consisted of 1000 randomly generated arrays, with sizes in range [0..10], elements in range [1..100], and index in range [1..array-size]. In **find2** the universe contained one test case for each array with elements selected from $\{0,1,2,3,4\}$ and range $[0\ldots n]$, for each n from 0 to 5.

textfmt is the text formating program analyzed in [18], with all the problems except N5 and N6 corrected. The universe consisted of one thousand 15-character long pieces of text, each generated by repeated uniform random selection of a character from the set consisting of letters, blank and newline characters.

transpose is the transpose routine of a sparse-matrix package [19, 27]. The universe had 1,000 randomly generated R by C matrices ($0 \leq C \leq R \leq 50$) with densities between 0 and 66%.

strmtch1 and **strmtch2** are string matching routines, each with an error which we had previously inadvertently introduced. **strmtch1** failed whenever the null pattern was input, and **strmtch2** had the maximum pattern length shorter than specified. The universe consisted of all (text, pattern) pairs on a two letter alphabet with text length and pattern length ranging from zero to four. [2]

matinv1, **matinv2**, and **determinant** were matrix manipulation programs based on LU decomposition [30], written by graduate students. The programs failed on some, but not all, singular matrices [3]. In **matinv1** and **determinant** the lower-upper decomposition procedure was instrumented, while in **matinv2** the back-solving procedure was instrumented. In each case, the universe consisted of 1,000 square matrices with sizes uniformly selected between 0 and 5 and with integer entries uniformly selected between 0 and 24.

5 Results

Tables 2, 3, and 4 summarize the results of the comparisons of effectiveness of all-uses to all-edges, all-uses to null, and all-edges to null. The columns labeled N_e, N_u, and N_n give the total numbers of adequate test sets for criteria all-edges, all-uses, and null, respectively, and the columns labeled $\hat{p_e}$, $\hat{p_n}$, and $\hat{p_u}$ give the proportions of these which expose errors. The sixth column of each table gives the significance probability to three significant digits, where applicable; an entry of **** indicates that hypothesis testing could not be applied. A "yes" in the column labeled "$p_e < p_u$?" indicates that all-uses is significantly more effective than all-edges. The columns labeled "$p_n < p_u$?" and "$p_n < p_e$?" answer analogous questions. Where the answer to this question is "yes", confidence intervals are shown in the last column. In those cases where the normality assumption held, a confidence interval for the *difference* in effec-

[2] We had previously discovered that all-uses was guaranteed to discover the error in **strmtch1**, but did not know how effective the other criteria were for this program.

[3] Interestingly, the determinant and matrix inversion programs did not fail on the same set of inputs.

Subj.	N_e	$\hat{p_e}$	N_u	$\hat{p_u}$	Sig.	$p_e < p_u$?	Confidence
detm	169	0.041	7	1.000	****	yes	[0.00,0.08] vs. [0.52,1.00]
find1	1678	0.557	775	0.667	0.000	yes	[0.06,0.16]
find2	3182	0.252	43	0.256	0.476	no	
matinv1	3410	0.023	76	1.000	****	yes	[0.02,0.03] vs. [0.94,1.00]
matinv2	4789	0.001	4406	0.001	0.500	no	
strmtch1	1584	0.361	238	1.000	****	yes	[0.33,0.39] vs. [0.98,1.00]
strmtch2	1669	0.535	169	0.615	0.015	no	
textfmt	1125	0.520	12	1.000	****	yes	[0.48,0.56] vs. [0.68,1.00]
transpose	1294	0.447	13	0.462	0.456	no	

Table 2: All-edges vs. All-uses

Subj.	N_n	$\hat{p_n}$	N_u	$\hat{p_u}$	Sig.	$p_n < p_u$?	Confidence
detm	6400	0.032	7	1.000	****	yes	[0.03,0.04] vs. [0.52,1.00]
find1	2000	0.484	775	0.667	0.000	yes	[0.13,0.24]
find2	3500	0.234	43	0.256	0.366	no	
matinv1	4000	0.020	76	1.000	****	yes	[0.01,0.03] vs. [0.94,1.00]
matinv2	5000	0.001	4406	0.001	0.500	no	
strmtch1	2000	0.288	238	1.000	****	yes	[0.26,0.31] vs. [0.98,1.00]
strmtch2	2000	0.456	169	0.615	0.000	yes	[0.06,0.26]
textfmt	2000	0.391	12	1.000	****	yes	[0.36,0.42] vs. [0.68,1.00]
transpose	3000	0.407	13	0.462	0.336	no	

Table 3: Null Criterion vs. All-uses

Subj.	N_n	$\hat{p_n}$	N_e	$\hat{p_e}$	Sig.	$p_n < p_e$?	Confidence
detm	6400	0.032	169	0.041	0.255	no	
find1	2000	0.484	1678	0.557	0.000	yes	[0.03,0.12]
find2	3500	0.234	3182	0.252	0.040	no	
matinv1	4000	0.020	3410	0.023	0.179	no	
matinv2	5000	0.001	4789	0.001	****	no	
strmtch1	2000	0.288	1584	0.361	0.000	yes	[0.03,0.11]
strmtch2	2000	0.456	1669	0.535	0.000	yes	[0.04,0.12]
textfmt	2000	0.391	1125	0.520	0.000	yes	[0.08,0.18]
transpose	3000	0.407	1294	0.447	0.006	yes	[0.00,0.08]

Table 4: Null Criterion vs. All-edges

Subj.	size	N_e	$\hat{p_e}$	N_u	$\hat{p_u}$	Sig.	$p_e < p_u$?
detm	1-6	5	0.000	0	-	-	
	7-12	5	0.000	0	-	-	
	13-18	10	0.000	0	-	-	
	19-24	25	0.040	1	1.000	****	
	25-30	73	0.000	0	-	-	
	>30	51	0.118	6	1.000	****	yes
find1	1-5	211	0.299	1	0.000	****	
	6-10	470	0.440	97	0.474	0.270	no
	11-15	497	0.616	285	0.653	0.151	no
	16-20	500	0.718	392	0.727	0.383	no
find2	1-5	205	0.088	0	-	-	
	6-10	482	0.168	0	-	-	
	11-15	496	0.228	4	0.000	****	
	16-20	1999	0.296	39	0.282	0.575	no
matinv1	1-6	205	0.000	0	-	-	
	7-12	484	0.015	6	1.000	****	yes
	13-18	550	0.013	7	1.000	****	yes
	19-24	579	0.021	12	1.000	****	yes
	25-30	595	0.024	14	1.000	****	yes
	31-35	498	0.032	16	1.000	****	yes
	36-40	499	0.042	21	1.000	****	yes
matinv2	1-5	4090	0.001	3714	0.001	0.500	no
	6-10	699	0.001	692	0.001	****	
strmtch1	1-5	194	0.155	0	-	-	
	6-10	416	0.298	77	1.000	****	yes
	11-15	484	0.388	161	1.000	****	yes
	16-20	490	0.469	0	-	-	
strmtch2	1-5	238	0.366	1	1.000	****	
	6-10	447	0.438	16	0.438	0.500	no
	11-15	486	0.591	49	0.592	0.495	no
	16-20	498	0.649	103	0.650	0.492	no
textfmt	1-5	99	0.354	0	-	-	
	6-10	258	0.399	0	-	-	
	11-15	348	0.511	2	1.000	****	
	16-20	420	0.640	10	1.000	****	yes
transpose	10-20	359	0.306	1	0.000	****	
	22-32	935	0.501	12	0.500	0.503	no

Table 5: All-edges vs. All-uses By Size

tiveness between the two criteria (e.g. $p_u - p_e$) is given, while in the other cases, confidence intervals around the effectiveness of *each* criterion are shown. For example, the first row of Table 2 indicates that for **determinant** we are 99% confident that the effectiveness of all-edges lies between 0 and 0.08, whereas that of all-uses lies between 0.52 and 1.0. The second row indicates that for **find1** we are 99% confident that the effectiveness of all-uses is between 0.06 and 0.16 *greater* than that of all-edges.

Examination of these tables shows that for five of the nine subjects, all-uses was more effective than all-edges at 99% confidence; for six subjects, all-uses was more effective than the null criterion; and for five subjects all-edges was more effective than null. Note that for **strmtch2** all-uses would be considered more effective than all-edges at 98% confidence. Further interpretation of these results is given in Section 6.

In Table 5 the test sets are grouped according to their sizes. In four of the nine subjects, all-uses adequate test sets are more effective than all-edges adequate sets of the same (or similar) size. Thus it appears that in four of the five subjects for which all-uses was more effective than all-edges, the improved effectiveness can be attributed to the inherent properties of all-uses, not just to the fact that all-uses adequate test sets are larger than all-edges adequate test sets.

Subject	$f(c,s)$
detm	****
find1	$-7.234 + 6.866c - 0.657s + 0.732cs$
find2	$-3.826 - 0.009s^2 - 0.707c^2s + 1.028cs$
matinv1	$17.940 - 584.6c + 572.0c^2 + 0.005cs^2 - 0.265cs$
matinv2	$-29.513 + 24.146cs - 7.225cs\ln s - 24.182c^2\ln s$
strmtch1	$-1.579 - 19.093c^2 + 139.3c^{16} - 282.1c^{26} + 201.5c^{39}$ $+16.816\ln s - 17.346c^4\ln s + 16.598\ln c\ln^2 s$
strmtch2	****
textfmt	$2600.2 + 1125.0\ln c - 5203.1c + 3668.2c^2 - 1043.3c^3 - 0.116c^3\ln s$
transpose	$-6.830 - 52.359\ln c - 213.4\ln^2 c + 1.932c^6\ln s$

Table 6: Logistic Regression Results

The results of logistic regression are shown in Table 6. Each regression equation is of the form

$$Prob(exposing) = \frac{\exp f(s,c)}{1 + \exp f(s,c)}$$

where $f(s,c)$ is a function of the predictor variables, s, the test set size, and c, the fraction of duas covered by the test set. The table gives the functions $f(s,c)$ for each subject program for which we were able to find a good-fitting model. The asterisks in the table entries for **determinant** and **strmtch2**, indicate that the data are so scattered that any function that gives a good fit is too complex to offer much insight into the relationship, if any exists, between coverage and effectiveness.

Inspection of the graphs of these functions, given in [36], shows that for four of the subjects (**find1**, **textfmt**, **strmtch1**, and **transpose**), there is strong positive correlation between coverage and effectiveness, but that for the remaining three there is no correlation at all.

6 Conclusion

We have described an experimental comparison of the effectiveness of randomly generated test sets which are adequate according to the all-edges, all-uses, and null test data adequacy criteria. Our experiment was designed to allow for comparison of *adequacy criteria*, not just of *test generation techniques*. The data was analyzed rigorously, using well established statistical techniques. For five of the nine subjects, all-uses was significantly more effective than all-edges, and for six of the nine, all-uses was significantly more effective than the null criterion. All-uses adequate test sets appeared to be more effective than all-edges adequate sets of the same size in four of the nine subjects. Logistic regression showed that in some, but not all of the programs, there was a strong correlation between the percentage of duas that a test set covered and the likelihood that it exposed an error.

Close examination of the tables led us to several other interesting observations. While all-uses was not always more effective than all-edges and the null criterion, in most of those cases where it was more effective, it was *much* more effective. In contrast, in those cases in which all-edges was more effective than the null criterion, it was usually only a little bit more effective.

For **buggyfind**, all-uses performed significantly better than all-edges when the **find1** universe was used, but not when the **find2** universe was used; also, the effectiveness of each criterion is dramatically better for **find1** than for **find2**. This shows that even relatively minor changes in the test generation strategy can profoundly influence the effectiveness of an adequacy criterion and that blanket statements about "random testing" without reference to the particular input distribution used can be misleading.

In four of the subjects, **determinant**, **matinv1**, **textfmt**, and **strmtch1**, coverage of all duas appears to guarantee detection of the error. This was already known (prior to the experiment) for **strmtch1**, but was a surprise for the other three programs. The raw data [36] show that in each of these, there is a number x, such that below x% coverage, effectiveness decreases suddenly. It appears that in each of these cases there are one or more duas whose coverage guarantees detection of the error.

This phenomenon has profound consequences for testing practitioners, who might deal with the unexecutable feature problem by testing until some arbitrary predetermined percentage of the duas have been covered. If it so happens that the test set fails to cover any of the "crucial" duas, the test set may be much less likely to detect the error than if it had covered 100% of the executable duas.

For example, in matinv1, there are a total of 298 def-use associations, only 206 (69%) of which are executable. Suppose it has been decided that test sets which cover 200 of the duas will be deemed adequate. Examination of the raw data [36] shows that such test sets have probability close to zero of exposing an error, whereas those which cover all of the executable duas appear certain to expose an error.

Consequently, we recommend that practitioners using data flow testing put in the effort required to weed out unexecutable def-use associations, and only accept test sets which achieve 100% coverage of the executable duas. A heuristic for doing this is presented in [11] and the issue of how this effects the cost of using a criterion is discussed in [35].

In summary, the all-uses criterion receives mixed reviews according to our experiment. While it sometimes performs much better than all-edges and the null criterion, this is not always the case. On the other hand, our results show that all-uses can be extremely effective, appearing to guarantee error detection in several of the subjects. Further analysis of the data can be found in [36].

There are many directions for future research, foremost among them being the performance of similar experiments on a large variety of subjects, including large programs, and on other adequacy criteria. In addition, experiments comparing the effectiveness of various adequacy criteria when non-random test generation strategies are used would be useful. We hope that other researchers will join us in performing such experiments in the future.

Acknowledgments: The authors would like to thank Prof. Al Baranchik of the Hunter College Mathematics Department for advice on statistical methods, Mohammed Ghriga and Roong-Ko Doong for helping prepare the subject programs, Zhang Ming for helping with the data analysis, and the Hunter College Geology and Geography Department for letting us run SAS on their system.

References

[1] A. Agresti. *Analysis of Ordinal Categorical Data.* John Wiley & Sons, Inc., New York, 1984.

[2] V. R. Basili and R. W. Selby. Comparing the effectiveness of software testing strategies. *IEEE Transactions on Software Engineering*, SE-13(12):1278–1296, Dec. 1987.

[3] G. K. Bhattacharyya and R. A. Johnson. *Statistical Concepts and Methods.* John Wiley & Sons, Inc., New York, 1977.

[4] R. Boyer, B. Elspas, and K. Levitt. SELECT – a formal system for testing and debugging programs by symbolic execution. *SIGPLAN Notices*, 10(6):234–245, June 1975. International Conference on Reliable Software.

[5] D. R. Byrkit. *Elements of Statistics.* D. van Nostrand Company, New York, 1980.

[6] L. Clarke, A. Podgurski, D. Richardson, and S. Zeil. A formal evaluation of data flow path selection criteria. *IEEE Transactions on Software Engineering*, SE-15(11):244–251, Nov. 1989.

[7] R. A. DeMillo, D. S. Guindi, W. McCracken, A. Offutt, and K. King. An extended overview of the Mothra software testing environment. In *Proceedings Second Workshop on Software Testing Verification and Analysis*, pages 142–151. IEEE Computer Society Press, July 1988.

[8] R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41, Apr. 1978.

[9] J. W. Duran and S. C. Ntafos. An evaluation of random testing. *IEEE Transactions on Software Engineering*, SE-10(7):438–444, July 1984.

[10] J. W. Duran and J. Wiorkowski. Quantifying software validity by sampling. *IEEE Transactions on Reliability*, R-29:141–144, 1980.

[11] P. G. Frankl. Partial symbolic evaluation of path expressions. submitted.

[12] P. G. Frankl. ASSET user's manual. Technical Report 318, Computer Science Department, Courant Institute of Mathematical Sciences, New York University, Sept. 1987.

[13] P. G. Frankl. *The Use of Data Flow Information for the Selection and Evaluation of Software Test Data.* PhD thesis, Courant Institute of Mathematical Sciences, New York University, Oct. 1987.

[14] P. G. Frankl, S. N. Weiss, and E. J. Weyuker. ASSET – a system to select and evaluate tests. In *Proceedings of the IEEE Conference on Software Tools*, pages 72–79, Apr. 1985.

[15] P. G. Frankl and E. J. Weyuker. An applicable family of data flow testing criteria. *IEEE Transactions on Software Engineering*, SE-14(10):1483–1498, Oct. 1988.

[16] P. G. Frankl and E. J. Weyuker. Assessing the fault-detecting ability of testing methods. In *ACM SIGSOFT '91 Conference on Software for Critical Systems.* ACM Press, Dec. 1991.

[17] M. Girgis and M. Woodward. An experimental comparison of the error exposing ability of program testing criteria. In *Proceedings IEEE Workshop on Software Testing*, pages 64–73. IEEE Computer Society Press, July 1986.

[18] J. Goodenough and S. Gerhart. Toward a theory of test data selection. *IEEE Transactions on Software Engineering*, SE-1(2):156–173, June 1975.

[19] F. Gustavson. Remark on algorithm 408. *ACM Transactions on Mathematical Software*, 4:295, 1978.

[20] D. Hamlet. Theoretical comparison of testing methods. In *Proceedings ACM SIGSOFT Third Symposium on Software Testing, Analysis, and Verification*, pages 28–37. ACM Press, Dec. 1989.

[21] D. Hamlet and R. Taylor. Partition testing does not inspire confidence. *IEEE Transactions on Software Engineering*, 16(12):206–215, Dec. 1990.

[22] W. E. Howden. A survey of dynamic analysis methods. In *Tutorial: Software Testing and Validation Techniques*, pages 209–231. IEEE Computer Society Press, 1978.

[23] J. Huang. An approach to program testing. *ACM Computing Surveys*, 7(3):113–128, Sept. 1975.

[24] B. Jeng and E. J. Weyuker. Some observations on partition testing. In *Proceedings ACM SIGSOFT Third Symposium on Software Testing, Analysis, and Verification*, pages 38–47. ACM Press, Dec. 1989.

[25] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Transactions on Software Engineering*, SE-12(1), Jan. 1986.

[26] J. W. Laski and B. Korel. A data flow oriented program testing strategy. *IEEE Transactions on Software Engineering*, SE-9(3):347–354, May 1983.

[27] J. M. McNamee. Algorithm 408: A sparse matrix package (part i) [f4]. *Commun. ACM*, 14(4), Apr. 1971.

[28] S. Ntafos. On required element testing. *IEEE Transactions on Software Engineering*, SE-10(6):795–803, Nov. 1984.

[29] S. Ntafos. A comparison of some structural testing strategies. *IEEE Transactions on Software Engineering*, 14(6):868–874, June 1988.

[30] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 1986.

[31] S. Rapps and E. J. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, SE-14(4):367–375, Apr. 1985.

[32] J. H. Rowland and Y. Zuyuan. Experimental comparison of three system test strategies: preliminary report. In *Proceedings ACM SIGSOFT Third Symposium on Software Testing, Analysis, and Verification*. ACM Press, Dec. 1989.

[33] K. C. Tai. Program testing complexity and test criteria. *IEEE Transactions on Software Engineering*, SE-6(6):531–538, Nov. 1980.

[34] M. Vouk, D. McAllister, and K. Tai. An experimental evaluation of the effectiveness of random testing of fault-tolerant software. In *Proceedings IEEE Workshop on Software Testing*. IEEE Computer Society Press, July 1986.

[35] S. N. Weiss. Methods of comparing test data adequacy criteria. In *COMPSAC 90*, pages 1–6, Oct. 1990.

[36] S. N. Weiss and P. G. Frankl. Comparison of all-uses and all-edges: Design, data, and analysis. Technical Report CS-91-03, Hunter College Computer Science Dept., Mar. 1991.