Properties of Criteria

- Program-based
- To recognize a good adequacy criteria
- And to discard poor choices
- Objective, well-defined properties

1. Applicability Property

- For every program, there exists an adequate test set
- Every program must be adequately testable

Criteria

- Statement coverage
- Branch coverage
- Path coverage
- Def-use coverage

- One cannot algorithmically determine whether more testing must be performed

Exhaustive test set

- If all representable points of the specification’s domain have been tested
  - Set of all inputs for which the program should produce the desired output
- Exhaustive test set is surely adequate
  - No matter what criterion is used
- There can be no additional testing possible
- Practical if domain is small
- A criterion that always requires an exhaustive test set is unacceptable
2. Non-exhaustive Applicability

• There is a program P and (not exhaustive) test set T such that P is adequately tested by T

3. Monotonicity

• Once a program has been adequately tested, running some additional test cases cannot cause the program to be deemed inadequately tested
• If T is adequate for P, and T ⊆ T’ then T’ is adequate for P
• “Stop when we find less than 50 errors per 1000 hours of testing”
• Note
  • An exhaustive test set is always adequate

4. Inadequate empty set

• If no testing has been performed, then the program cannot be considered adequately tested
• The empty set is not an adequate test set for any program

Program Equivalence

• P ≡ Q
  – P is equivalent to Q
• For x (input vector) in the specification’s domain
• P(x) = Q(x)
  – Results of P and Q on every x are same
5. Antiextentionality

- There are programs P and Q, such that $P \equiv Q$, and a test set T is adequate for P but T is not adequate for Q.
- Remember
  - Program-based
-Semantic equivalence of two programs does not necessarily imply that they be tested the same way.
- Program-based testing should consider the implementation, not the functions computed.

6. General Multiple Change

- There are programs P and Q, which are the same shape, and a test set T is adequate for P but T is not adequate for Q.
- Syntactic closeness of programs does not imply that they should be tested the same way.

Syntactic Closeness

- Two programs have the same shape
  - If one can be transformed into another by applying the following transformations, any number of times:
    - Replace relational operator $r_1$ in a predicate with relational operator $r_2$.
    - Replace constant $c_1$ in a predicate or assignment statement with constant $c_2$.
    - Replace arithmetic operator $a_1$ in an assignment statement with arithmetic operator $a_2$.

Program Decomposition

- A component Q of a program P is any contiguous sequence of statements of P.
7. Antidecomposition

- There exists a program $P$, and
- component $Q$,
- such that test set $T$ is adequate for $P$,
- $T'$ is the set of vectors of values that variables can assume on entrance to $Q$ for some $t$ in $T$, and
- $T'$ is not adequate for $Q$.

Explanation

- Although a program has been adequately tested, it does not necessarily imply that each of its component pieces has been properly tested.
- A routine that has been adequately tested in some environment or context has not necessarily been tested for other environments.
- Even though $P$ appears to be more complicated than $Q$ (p syntactically contains $Q$), semantically, $Q$ may be more complex than $P$.
Program Composition

- Assume a structured programming language
  - Programs are single-entry/single-exit
  - All input statements appear at the start of the program
  - All output statements appear at the end of the program
- Programs P and Q
  - Using the same set of identifiers
  - Remove all output statements of P
  - Remove all input statements of Q
- P;Q is the composed program

8. Anticomposition

- There exist programs P and Q, and test set T,
- such that T is adequate for P, and
- the set of vectors of values that variables can assume on entrance to Q for inputs in T is adequate for Q, but
- T is not adequate for P;Q
Criteria

- Statement coverage
- Branch coverage
- Anticomposition property eliminates criteria that do not have provision for testing the interaction of program pieces

Gödel Numbering

- Definition
  - A unique numerical value for each program, such that the program can be algorithmically retrieved from this value
- For a program P with Gödel number p
  - A test set T is Gödel adequate for P if $p \in T$
- Any test set T that contains a program P's Gödel number is adequate for P

Examining Gödel Adequacy

- Gödel adequacy has nothing to do with a program's semantics, syntax or specifications
- Every program will always have an adequate test set of size one
- Does this criterion satisfy all the properties that we have discussed?
- Do you think that this criterion is useful?

Program Renaming

- P is a renaming of Q if
  - P is identical to Q, except
  - All instances of an identifier $x_i$ of Q have been replaced by an identifier $x_j$ where $x_j$ does not appear in Q, or
  - If there exists a sequence $Q = P_1, P_2, P_3, ..., P_n = P$, where
    - $P_{i+1}$ is a renaming of $P_i$ for $i = 1, ..., n-1$
9. Renaming Property

- Let P be a renaming of Q.
- Test set T is adequate for P iff T is adequate for Q.
- Intuitively, an “inessential” change in a program, such as changing variable names, should not change the test data required to adequately test the program.
- Gödel adequacy does not satisfy this property!!

Canonical Representation

- Given a Program P with k variables
  - Obtain its canonical representation by
  - Renaming variables using the set \{x_1, x_2, ..., x_k\} where \(x_i\) replaces the first variable used in the program and \(x_k\) replaces the k\textsuperscript{th} variable used; \(x_i\) replaces the i\textsuperscript{th} variable used.

Gödel-class Numbering

- Definition
  - A unique numerical value for each program’s canonical form, such that it can be algorithmically retrieved from this value.
- For a program P with Gödel-class number p
  - A test set T is Gödel-class adequate for P if p \(\in\) T.
- Any test set T that contains a program P’s Gödel-class number is adequate for P.
- Does it satisfy Renaming Property?
- And all other 8 properties?

10. Statement Coverage

- If T is adequate for P, then T causes every executable statement of P to be executed.