

# IA169 System Verification and Assurance

## Fundamentals of Testing

<http://www.testingeducation.org/BBST/>

## Course Organization

## Topics to be covered ...

- Introduction to Formal Verification and Testing
- Symbolic Execution
- Deductive Verification
- LTL (Linear Temporal Logic) Model Checking
- CTL (Computation Tree Logic) Model Checking
- Bounded Model Checking
- CEGAR and Abstract Interpretation
- Verification of Real-Time and Hybrid Systems
- Verification of Probabilistic Systems
- Assurance, Threat Models, Relevant Security Standards

## Prerequisites

- Formally none, but we expect ...
- ... capability of basic math reasoning and abstractions.
- ... some experience with coding.
- ... you can handle Unix as a user.

## Mutual Exclusion with

- IV113 and IV101

## Possible Follow-Up

- IA159 Formal Verification Methods

## Structure

- 2/2/2 credits
- Lecture, Seminar, Homework

## Marking

- Final exam 70%
- Assignments 30%
- 50% for E or Colloquy or Credit
- 60% for D
- 70% for C
- 80% for B
- 90% for A

## Fundamentals of Testing

**Testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test.**

## **Empirical Technical**

- Conduct experimental measurements.
- Logic and math.
- Modelling.
- Employs SW tools.

## **Investigation**

- Organised and thorough.
- Self-reflecting.
- Challenging.

## **Product or Service**

- Software.
- Hardware.
- Data.
- Documentation and specification.
- ... other parts that are delivered.

## **Information**

- Not know before.
- Has some value.

## **Stakeholders**

- Who has interest in the success of testing effort.
- Who has interest in the success of the product.



# Fundamental Questions of Testing

## Mission

- Why do we test? What we want to achieve?

## Strategy

- How to proceed to fulfil the mission efficiently?

## Oracle

- How to recognise success of the test.

## Incompleteness

- Do we realise that testing cannot prove absence of error?

## Measure

- How much of our testing plan has been completed?
- How far we are to complete the mission?

## Most Typical Mission

- Bug hunting.
- Identification of factors that reduce quality.

## Other Missions

- Collect data to support manager decisions, such as: Is the product good enough to be released?
- How much different is the product from product available on market?
- Is the product complete with respect to specification?
- Are individual components logically and ergonomically connected.
- ...

## Other Missions – continued

- Support manager decision with empirical results.
- Evaluate the cost of support after release.
- To check compatibility with other products.
- Confirm accordance with the specification.
- Find safe scenarios of product usage.
- To acquire certification.
- Minimise consequences of low quality.
- Evaluate the product for third party.
- ...

## Strategy

**Strategy is a plan, how to fulfil the mission in the given context.**

**Example:** Consider spreadsheet computation in four different contexts.

- a) Computer game.
- b) Early stage of development of database product.
- c) Late stage of development of database product.
- d) Driver for medical X-ray scanning device.

**Question:**

- Will you proceed with the same strategy?

## What factors influence strategy selection

- What is the mission?
- How aggressively we need to detect bugs.
- What bugs are less important than others?
- How thoroughly testing will be documented?

## Discussion

- Assume that a program has an enter field that is expecting numerical values. Is it meaningful to test the product for situation when we enter non-numeric value? (Not mentioned in specification at all.)

Oracle

**Oracle (in the context of testing) is a detection mechanism or principle to learn that the product passed or failed a test.**

## Facts

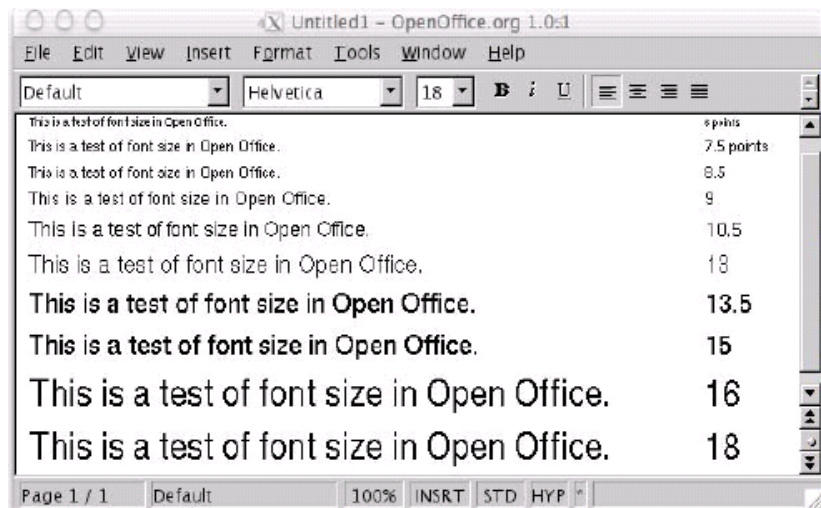
- If tester claims that the program passed a test, it does not mean the program is correct with respect to the tested property. It depends on the oracle used.
- Basically, any test may fail or pass with a suitable oracle.

## Example

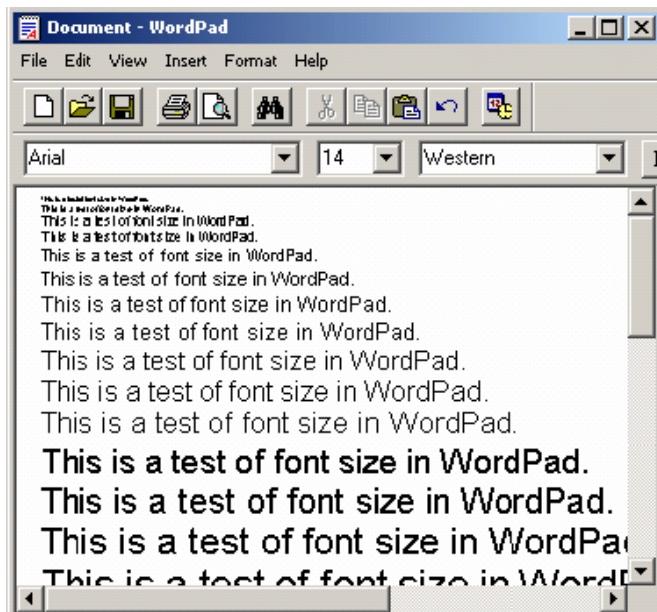
- Does font sizes work properly in OpenOffice, WordPad, and MS Word text editors?



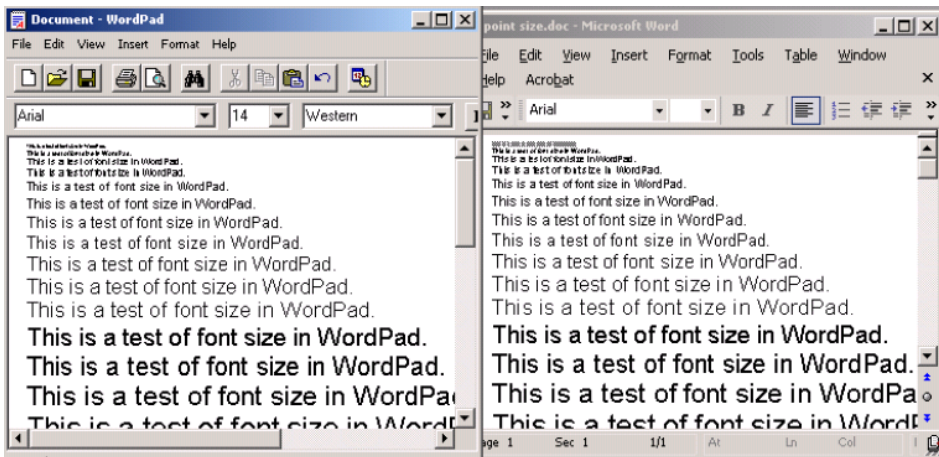
# Example – OpenOffice 1.0



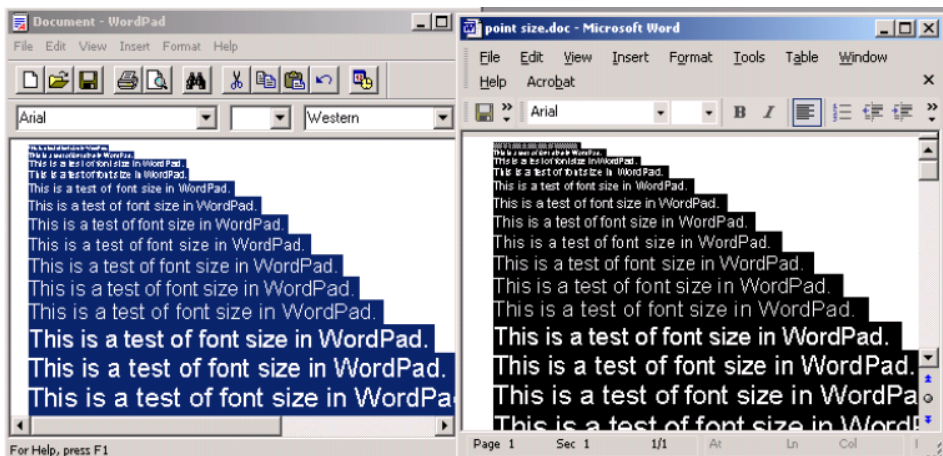
# Example – WordPad



# Example – WordPad versus MS Word



# Example – WordPad versus MS Word (highlighted)



The image displays two side-by-side windows illustrating text highlighting in WordPad and Microsoft Word. Both windows contain the text: "This is a test of font size in WordPad." repeated ten times. In the WordPad window (left), the text is highlighted in a solid blue color. In the Microsoft Word window (right), the text is highlighted in black with a white background. The WordPad window has a menu bar with File, Edit, View, Insert, Format, and Help. The Microsoft Word window has a menu bar with File, Edit, View, Insert, Format, Tools, Table, and Window. The status bar at the bottom of the WordPad window reads "For Help, press F1". The status bar at the bottom of the Microsoft Word window reads "Page 1 Sec 1 1/1 At Ln Col".

Document - WordPad

File Edit View Insert Format Help

Arial Western

This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.

For Help, press F1

point size.doc - Microsoft Word

File Edit View Insert Format Tools Table Window

Help Acrobat

Arial B I

This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.  
This is a test of font size in WordPad.

Page 1 Sec 1 1/1 At Ln Col

## Questions

- Is the observed difference in font sizes a bug in WordPad?
- Is the observed difference in font sizes a bug in MS Word?
- Is the observed difference in font sizes a bug at all?

## Possible Conclusions

- We do not know if sizes are correct, but we have tendency to believe MS Word rather than to WordPad.
- For WordPad it is not necessary to stick precisely to typographic standards.
- For WordPad it is possibly a bug, but definitely it is not a problem.

## Possible (Pragmatic) Position

- It is/isn't a bug?  $\implies$  It is/isn't a problem?
- It is necessary to know the context, to guess the metrics that the final consumer will use to judge the issue.
- With some risk we can achieve simplification of the decision.

## Simplification in Testing Process

- Avoid tests that obviously does not reveal any problems.
- Avoid tests that obviously reveal only uninteresting problems.

## How much do we actually know about typography?

- Point definition is unclear.  
(<http://www.oberonplace.com/dtp/fonts/point.htm>)
- Absolute sizes are difficult to measure.  
(<http://www.oberonplace.com/dtp/fonts/fontsize.htm>)

## From Uncertainty to Heuristics

- How precisely must the sizes agree in order to declare that the sizes are correct?
- Obtaining complete information and evaluation of all the facts is too complicated and costly.
- **Heuristics** are used instead.

## Decision Heuristics

- Allows for simplification of decision problem.
- Advice, recommendation, or procedure to be used within the given context.
- Should not build on any hidden knowledge.
- Does not guarantee a good decision.
- Various heuristics may lead to contradictory decisions.

## Disadvantages

- Heuristics might be subjective.
- If misused, may cause more harm than good.



## Consistency

- Good heuristics for decision making.

## Consistency with ...

- other functions of the product, similar products, history, producer image, specifications, standards, user expectations, the purpose of the product, etc.

## Advantages

- Consistency is objective enough.
- Easily described in bug report.

## Unintentional Blindness

- Human tester does not consider any test outputs that he/she does not pay attention to.
- Similarly, mechanical tester does not consider test outputs that it is not told to include into decision.

## Uncertainty Principle

- The presence of observer may affect what is observed.

## Consequence

- It is impossible to observe all possible outputs from a single test.

## Motivation

- Automation process eliminates human errors.
- Automation leads to repeatable procedures.
- Automation allows faster test evaluation.

## Problems of Automation

- It necessary to automate the decision making (oracle) principle.
- Can we do it? Only partially.

## Standard Way of Oracle Automation

- A file of expected outputs, which is required to match precisely with the outputs of a test being executed.
- Example: MS Word could be used to define a the file of expected outputs for testing WordPad.

## Amount of Agreement

- Assume MS Word to serve as the file of expected outputs for testing WordPad.
- How exactly is the expected output stored?
- Is 99% agreement still agreement?
- How is the percentage of agreement defined?

## False Alarms

- Using outdated expected output.
- Consequence of simplification of decision making.

## Undiscovered Errors

- Expected file exhibits the same error as test output.
- Unintentional Blindness.

## Measure Methods in Testing

## Coverage

- A set of source code entities that has been checked with at least one test.
- Source-code entities: lines of code, conditions, function calls, branches, etc.
- Used to identify parts that have not been tested yet.

## Coverage as a Measure

- Possible test plan is to achieve a given percentage of coverage.
- The percentage that expresses how much of the final product has been tested.
- Numeric expression for managers to see how much of the product remains to be tested.

## Problems

- Could avoid testing of interesting input data.
- Does not properly test parts of the product that rely on external services.

## Using Coverage as a Measure

- The mission is to test all entities of the product, is that OK?
- Complete coverage does not guarantee quality of the product.
- Stimulates to prefer quantity rather than quality.
- Misleading satisfaction (shouldn't feel safe).

## Example

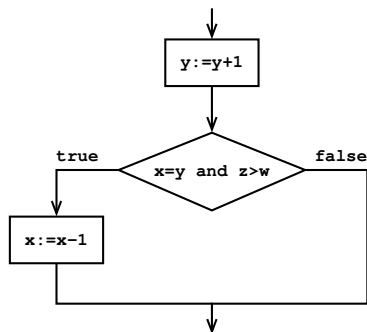
```
Input A      // program accepts any
Input B      // integer into A and B
Print A/B
```

## Observation

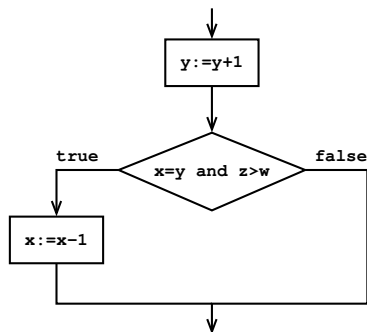
- Complete coverage is easy achievable.
- For example:
  - input: 2,1
  - output: 2
- There is of course a hidden bug in the program!



# Coverage Criteria for Control-Flow Graphs

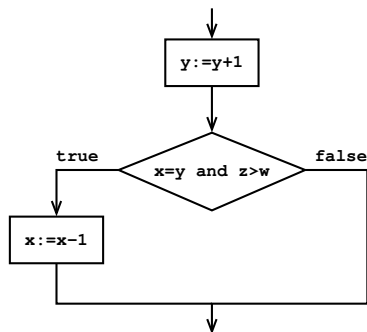


There are various criteria for control-flow graph coverage.



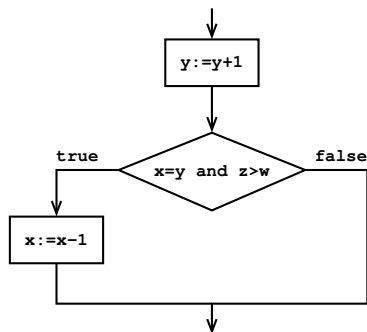
## Statement coverage

- Every statement (assignment, input, output, condition) is executed in at least one test.
- Set of tests to achieve full coverage:  
( $x = 2, y = 1, z = 4, w = 3$ )



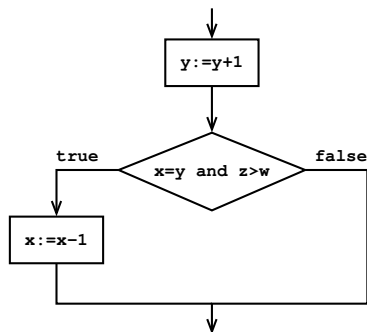
## Edge coverage

- Every edge of CFG is executed in at least one test.
- Set of tests to achieve full coverage:  
( $x = 2, y = 1, z = 4, w = 3$ ), ( $x = 3, y = 3, z = 5, w = 7$ )



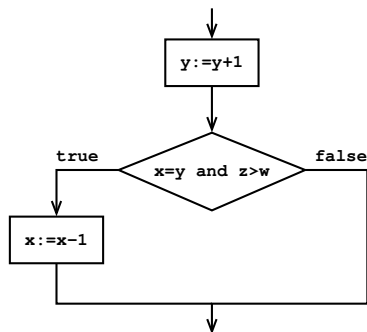
## Condition coverage

- Every condition is a Boolean combination of **elementary conditions**, for example  $x < y$  or  $\text{even}(x)$ .
- If it is possible, every elementary condition is evaluated in at least one test to TRUE and in at least one test to FALSE.



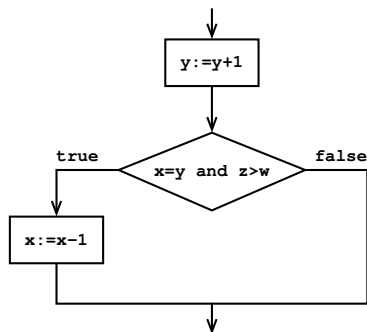
## Condition coverage

- Set of tests to achieve full coverage:  
( $x = 3, y = 2, z = 5, w = 7$ ), ( $x = 3, y = 3, z = 7, w = 5$ )
- In both cases, only the FALSE branch of IF statement is taken.



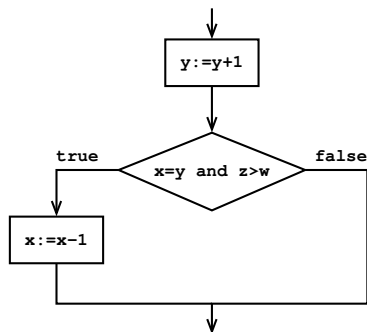
## Edge/Condition coverage

- Edge and Condition coverage at the same time.
- Set of tests to achieve full coverage:  
( $x = 2, y = 1, z = 4, w = 3$ ), ( $x = 3, y = 2, z = 5, w = 7$ ),  
( $x = 3, y = 3, z = 7, w = 5$ )
- Is the set the smallest possible one?



## Multiple condition coverage

- Every Boolean combination of TRUE/FALSE values that may appear in some decision condition must occur in at least one test.

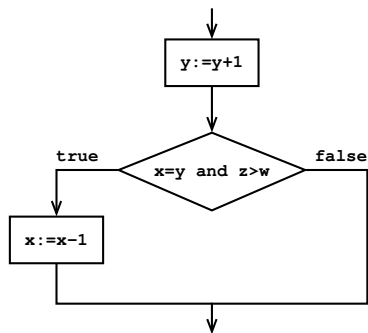


## Multiple condition coverage

- Set of tests to achieve full coverage:  
( $x = 2, y = 1, z = 4, w = 3$ ), ( $x = 3, y = 2, z = 5, w = 7$ ),  
( $x = 3, y = 3, z = 7, w = 5$ ), ( $x = 3, y = 3, z = 5, w = 6$ )
- Exponential growth in the number of tests.



# Coverage Criteria for Control-Flow Graphs



## Path coverage

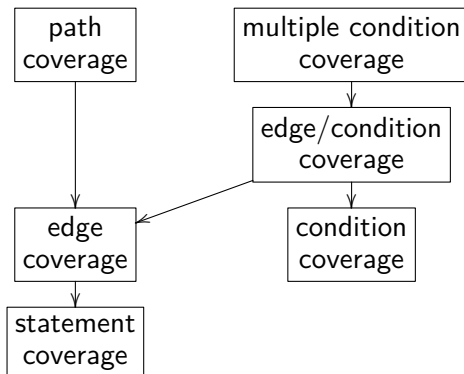
- Every executable path is executed in at least one test.
- The number of paths is big, even infinite in case there is an unbounded cycle in the control-flow graph.

# Hierarchy of Coverage Criteria

- Criterion A **includes** criterion B, denoted with  $A \rightarrow B$ , if after full coverage of type A we guarantee full coverage of type B.

# Hierarchy of Coverage Criteria

- Criterion A **includes** criterion B, denoted with  $A \rightarrow B$ , if after full coverage of type A we guarantee full coverage of type B.



## Coverage and Number of Cycle Iterations

- All criteria except the path criterion does not reflect the number of iterations over a cycle body.
- In case of nested cycles, systematic testing of all possible executable paths become complicated.

## Ad hoc Strategy for Testing Cycles

- Check the case when the cycle is completely skipped.
- Check the case when the cycle is executed exactly once.
- Check the case when the cycle is executed the expected number of times.
- If a boundary  $n$  is known for the number of cycle executions, try to design tests where the cycle is executed  $n - 1$ ,  $n$ , and  $n + 1$  times.

## Motivation

- Detect usage of undefined variables.
- On some paths, a variable may be set for a specific purpose and later on its value misused for other purpose.
- Control Flow criteria do not guarantee inclusion of tests for above mentioned or likewise situations.

## Data Flow Coverage

- Cover paths through control flow graph that go through a location where a variable is used but it is not defined along all incoming paths through control-flow graph.

## C/C++, Linux

- Tools `gcov` and `lcov`.

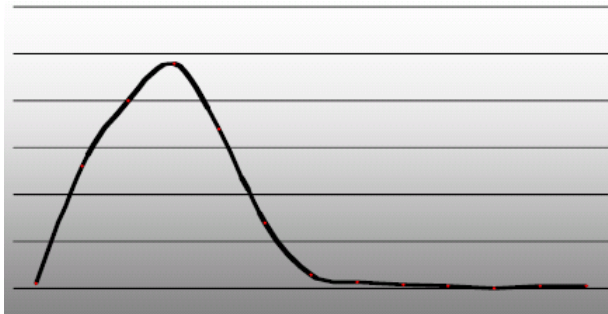
### Example: `lcov`

- `gcc -fprofile-arcs -ftest-coverage foo.c -o foo`  
`lcov -d . -z`  
`lcov -c -i -d . -o base.info`  
`./foo`  
`lcov -c -d . -o collect.info`  
`lcov -d . -a base.info -a collect.info -o result.info`  
`genhtml result.info`

## Week statistics

- The number of newly discovered errors.
- The number of fixed errors.
- The ratio of found versus fixed errors.

## Visualisation



## Observation

- The number of discovered errors per time unit exhibits Weibull Distribution.
- Can be used as a measure for the remaining amount of testing.
- Software Engineering Method to set the release date.

## Using Weibull Distribution

- At the moment the curve reaches the peak, the remaining part of the curve may be predicted, hence, a moment in time may be set, when expected number of errors discovered per week drops below a given threshold.
- Parameters of Weibull distribution influence the “width” and “height/slope” of the peak.
- $F(x) = 1 - e^{-ax^{-b}}$  for  $x > 0$



## Vague Precision

- Testing does not follow the typical usage of the product.
- The probability of error discovery is different for different errors.
- Fix may cause other new errors.
- Bugs are dependent.
- The number of errors in the product changes over time.
- Error insertion exhibits Weibull distribution itself.
- Testing epochs (various testing procedures) are independent.
- ...

## Conclusions

- Weibull Distribution is not very reliable.
- Can be used only with large projects for very rough estimation.

## Assumption

- Software developers are aware of being measured.

## Phase one

- Reach the peak as quickly as possible.
- Double reporting of errors.
- Avoid fixing known errors.
- ...

## Phase two

- Stick to expected shape of the curve.
- Delay reporting of errors.
- Reporting outside bug-tracking system.
- ...

## Incompleteness of Testing

## Observation

- The amount of tests to be run is extremely large.
- Resources for testing are always limited.

## What Is Not Complete Testing

- Complete Coverage
  - Every line of code.
  - Every branching point.
  - ...
- When testers do not find more errors.
- Testing plan is finished.

## What Is Complete Testing

- There are no hidden or unknown errors in the product.
- If new issue is reported, testing could not be complete.

**The number of tests is too large (infinitely many).**

**To perform all tests means:**

- To test all possible input values of every input variable.
- To test all combinations of input variables.
- To test every possible run of a system.
- To test every combination of HW and SW, including future technology.
- To test every way a user may use the product.

## Data Bus-Width

- The number of tests grows exponentially with respect to bit used for data representation.
- $n$ -bits requires  $2^n$  tests.

## Other Reasons

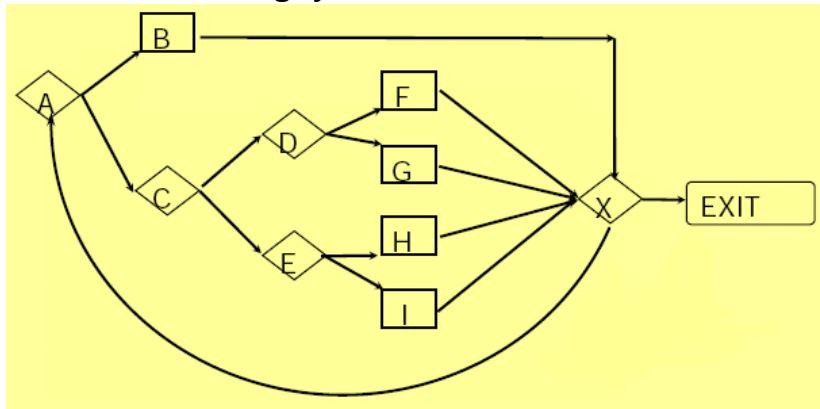
- Timing of actions.
- Invalid or unexpected inputs (buffer overflow).
- Edited inputs
- Easter egg [<http://j-walk.com/ss/excel/eastereg.htm>]

## Common Argumentation

- “This is not what the customer would do with our product.”

# Incapability to Test All Runs

Assume the following system



## Questions

- How many different ways it is possible to reach `EXIT` ?
- How many different ways it is possible to reach `EXIT` , if `A` can be visited at most  $n$ -times?

## Example

- In [F] is a memory leak, in [B] garbage collector.
- System will reach an invalid state, if [B] is avoided long enough.

## Observation

- Simplified testing of paths may not discover the error.
- The error manifests in circumstances that cannot be achieved with a simple test.



## Incompleteness

- Testing cannot prove absence of error.
- It is impossible to test all valid inputs.
- Existence of testing plan inhibits testing creativity.

## Measure

- There are methods to measure progress in testing phase.
- but they are unreliable.
- Focusing strongly on a selected measure may influence the effectiveness of testing.

## Practicals and Homework

## Practicals

- Analyse a simple `triangle.cpp` program and develop a test suit fulfilling given criterion of coverage.
- Automate regression testing (make test), include
  - Hand-written functional tests
  - Coverage of code (LCOV/GCOV)

## Homework

- Reading on MC/DC:  
[http://www.faa.gov/aircraft/air\\_cert/design\\_approvals/air\\_software/cast/cast\\_papers/media/cast-10.pdf](http://www.faa.gov/aircraft/air_cert/design_approvals/air_software/cast/cast_papers/media/cast-10.pdf)
- List, and briefly describe as many black-box testing approaches as you can find or are aware of.  
<http://www.testingeducation.org/BBST/>
- Optional: Look at CMAKE and CTEST systems.