7. Exploiting dependencies

Dependency Islands
Structure of the Island
Island within Island
Island with Gates

Invalid inputs

Dependency Islands

- A technique for reducing the number of test cases
- Relies on visualisation
- Helps to see and exploit these cases where it is possible to reduce the number of tests further

- Introduced by Bob Stahl

Example problem

14.3.5 Calculating the Correction

The Correction is determined by the Base, by whether the Base is new, and by the value of Type.

If the Base is less than 10,000, use Method 1 to calculate the Correction. If the Base is at least 10,000 but not greater than 50,000 use Method 2. Otherwise, use Method 3.

Each Method will use procedures that depend on whether the Base is new or is a previous Base. (That is, "New" = Y or N).

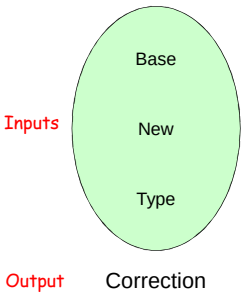
The value of Type can be A, B, C, or X. Each Method uses the Type to determine what process to use for computing the Correction.

Building the Dependency Island

Output variable: Correction

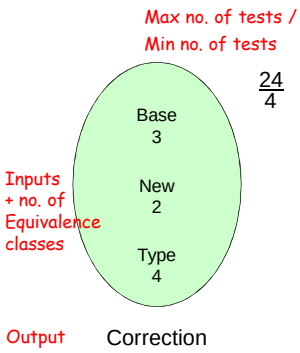
Input variables:

Base	New	Type
<10,000	Y	A
10,000 – 50,000	N	B
>50,000		C
		X



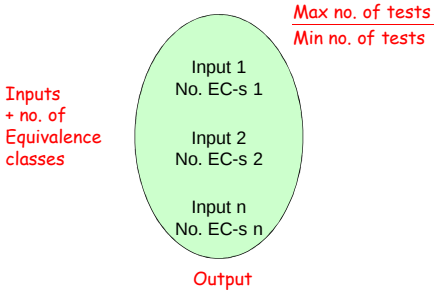
Building the Dependency Island

Base	New	Type
<10,000	Y	A
10,000 – 50,000	N	B
>50,000		C
		X

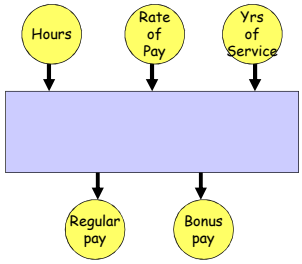


Building the Dependency Island

Generic graphical representation



Salary_and_bonus_system



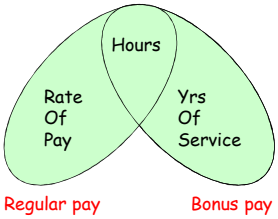
Regular pay = g(Hours, RateOfPay)

Bonus pay = f(Hours, YrsOfSevice)

Number of equivalence classes (worked out from the specification):

- Hours: 3
- RateOfPay: 4
- YrsOfService: 5

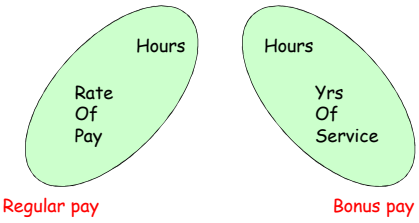
Case 1: Multiple overlapping islands



How do we treat multiple outputs related to the same input(s)?

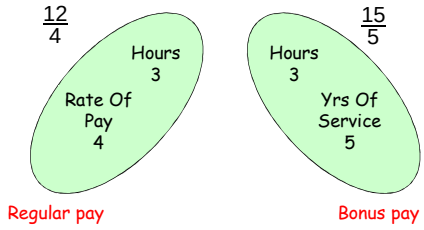
One island per output

Case 1: Multiple overlapping islands



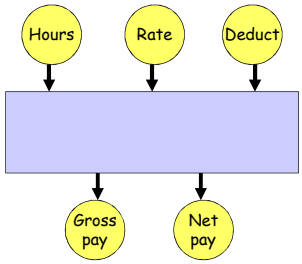
Each output must be treated separately

Case 1: Multiple overlapping islands



Number of tests:
min 9 (4+5)
max 27 (12+15)

Very_simple_payroll_system



Gross pay = g(Hours, Rate)

Net pay = f(Gross pay, Deduct)

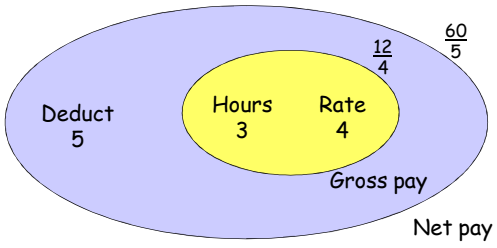
Number of equivalence classes (worked out from the specification:)

- Hours: 3
- Rate: 4
- Deduct: 5

How many islands?

Draw them

Case 2: Island within Island

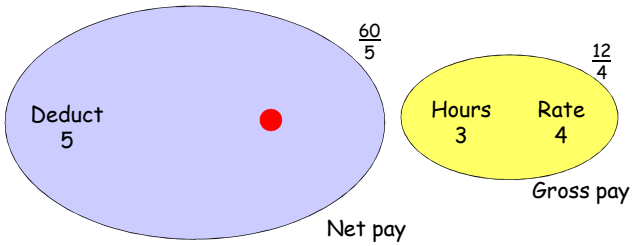


- How many tests are needed to get the (naïve) Path Coverage (Strong Equivalence Testing)?

  12 + 60 = 72

Case 2: Island within Island

- Can we reduce the number of tests but still retain the some coverage?

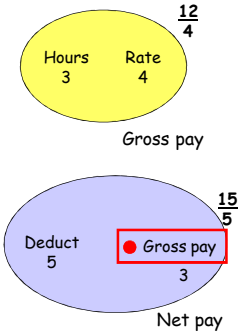


Case 2: Island within Island

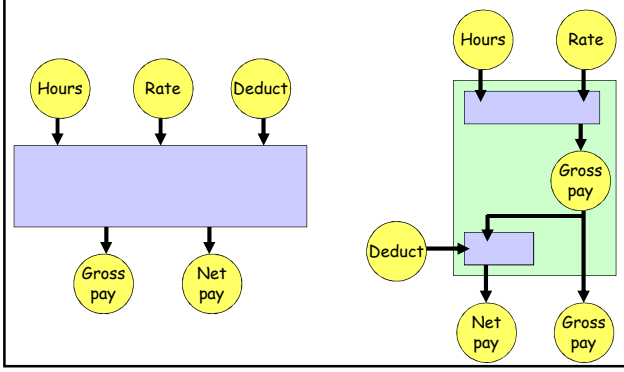
- Simplification
  - Work out that Gross Pay works correctly (12 tests)

    Gross Pay is now a single input to Net Pay
  - Find out how many equivalence classes should be assigned to Gross Pay (e.g. 3)

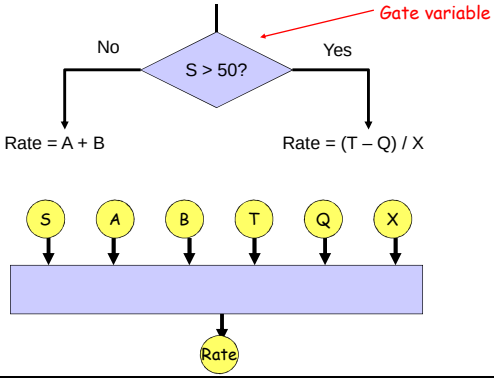
    Extra work!!!
  - Work out that Net Pay works correctly (15 tests)
- Total: 12 + 15 = 27 tests
  - Approx 1/3 of the original number of tests (72)



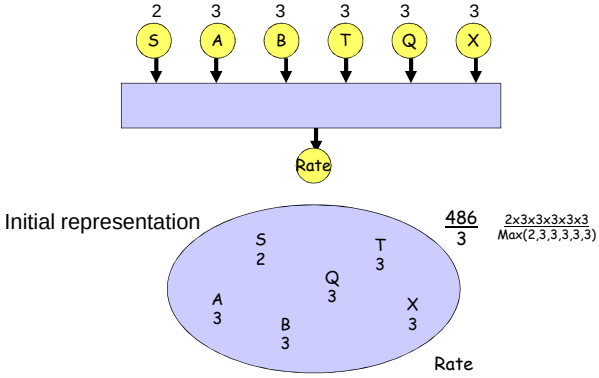
Re-representing dependencies



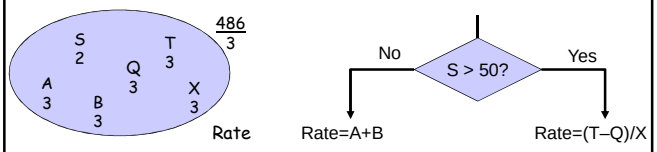
Case 3: Islands with Gates



Case 3: Islands with Gates

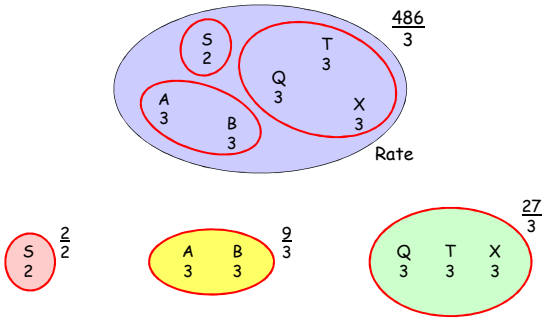


Case 3: Islands with Gates



- Observations
  - Rate depends on all the variables, but not all at the same time
  - The gate variable S partitions the variable set into two equivalence classes
- Need to test
  - Whether the gate variable S works correctly
  - Case with input variables A and B
  - Case with input variables T, Q and X

Case 3: Islands with Gates



Maximum total number of tests: 2 + 9 + 27 = **38**

Dependency Islands - summary

- The use of Dependency Islands helps to identify dependencies between input variables, but also mutually-independent variable sets
- If there is something totally enclosed in the Island, take it out and test it
- Only THEN decide whether it is an Island-within-Island or a Gate
- If it is an Island, put it back as a single input to the output represented by the larger Island. Find out the number of Equivalence Classes that that input has in the context of the larger Island
- If it is a Gate, the large Island will split and you don't have to put anything back.
- Gates can be single or compound; and can be found inside the Islands-within-Islands.

Invalid inputs and dependencies

- So far we have been considering only valid cases
- Invalid inputs can also be divided into Equivalence Classes
- Example: Invalid Equivalence Class for alphanumeric strings for the variables that should be numeric
- This reduces the number of test cases to be prepared
- This does NOT necessarily reduce the number of test cases to be carried out – they have to be tested, one at a time, for each output variable

Cross-field dependencies

- Cross-field dependency occurs when an input is not valid in its own right …
- … but whether it is valid or not depends on the value(s) of some other variable(s)
- In these cases, the value of one input variable imposes constraints on the possible values of another variable, e.g.
  - It cannot be there
  - It must be there
  - It must be in the certain range
  - . . .

Car insurance example

A car insurance company "Simplicity" provides insurance cover for motor vehicles.

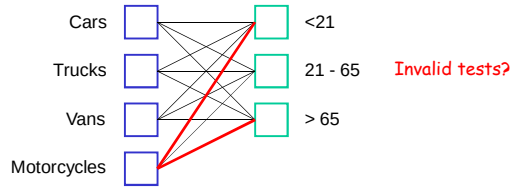
When a customer applies for a quote on insurance, the agent requests the following information:

1. Vehicle type
2. Age of the driver

The system outputs a quote for insurance

Car insurance example

Car insurance example



The company no longer accepts applications from people under 21 and over 65 who drive a motorcycle

Cross-field dependencies

- Handling cross-field dependencies
  - Move the combinations that are illegal under the cross-field constraints to the Invalid set of tests
  - Test the Invalid combinations one at a time
  - The expected test result should be “reject”

How do we find Dependencies and Equivalence Classes?

- From specifications
- From users

If this fails use

- Manuals
- Code
- Anything that may give you clues

Next lecture

Decision Table based testing

Homework

- If Island A is totally enclosed in Island B, what does this mean?
- Using the Dependency Islands try to reduce the number of test cases for the Triangle Problem.