Functional testing

Adapted from lessons 2/3 by Mohammad Mousavi – Eindhoven Univ. Of Technology, available from the web.

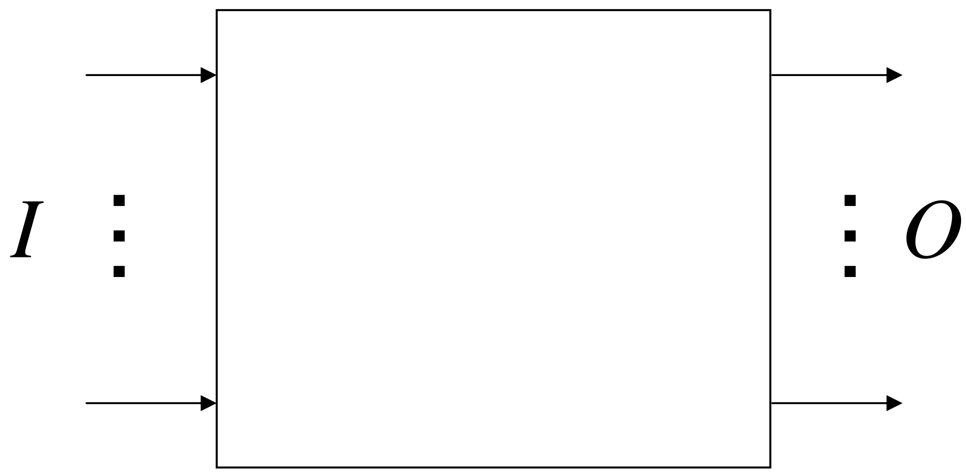
Objectives

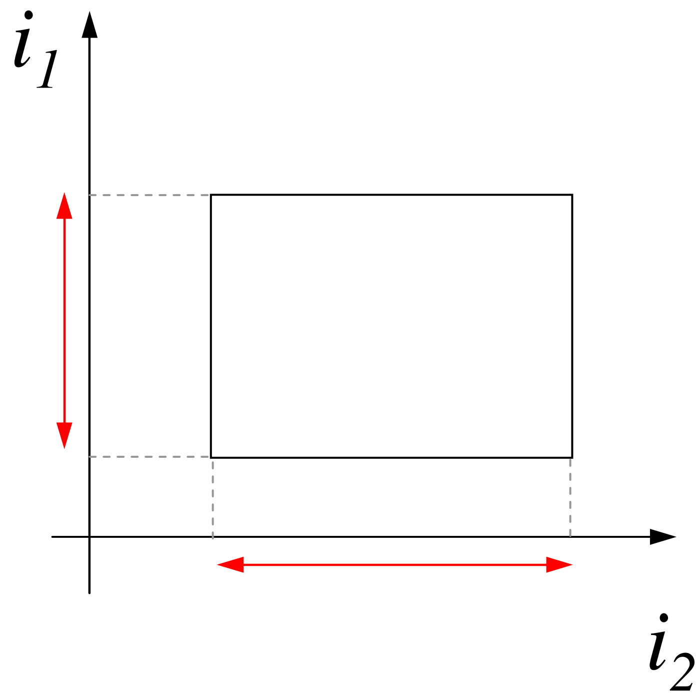
Define and use functional (black box) testing

- Nominal testing
- Boundary testing
- Robustness testing
- Equivalence testing
- Decision tables
- Classification trees

Functional testing

- Program is an input from a certain domain to a certain range

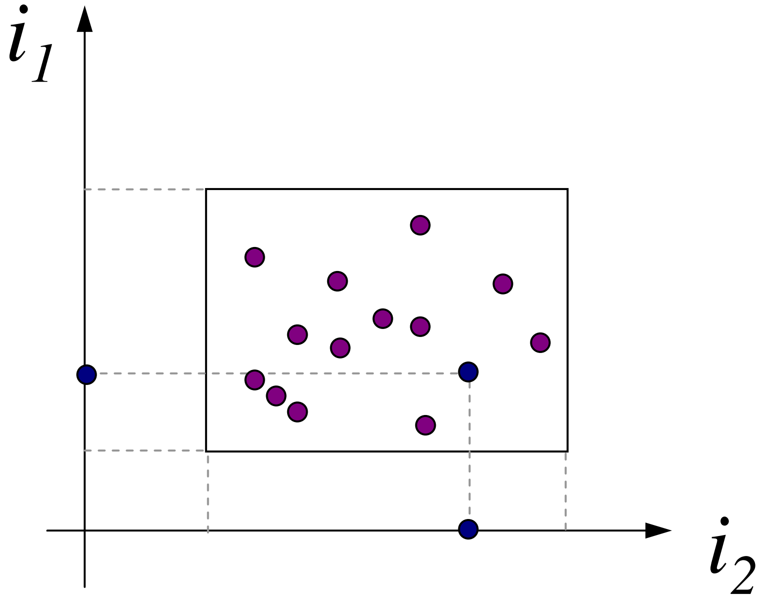




- $I=\{i_1, i_2, \ldots\ i_n\}$ with $D(i_k)$ being the range of the possible values of i_k
- impossible (impractical) to check all input/output combinations (range of system function): $Di_1 \times Di_2 \times \ldots \times Di_n$
  - need to choose some

Nominal testing

- For each input we select a value in the range of the admissible ones



- ... or possibly a set of randomly selected ones ...
- Problem
  - Very likely we tried a very small fraction of all the possible inputs
  - Can we select the “most meaningful ones”?

Assumptions of boundary testing

- program is an input from a certain domain to a certain range
- domain comprises (product of)
  - independent values
  - Continuous (not boolean/discrete) values (ordered, in an interval, taking all values in the interval)
- Rationale: most errors occur at extremes
  - (< instead of <=, counters off by one)
- also called: stress testing
- technique also applicable to range boundaries

Boundary testing

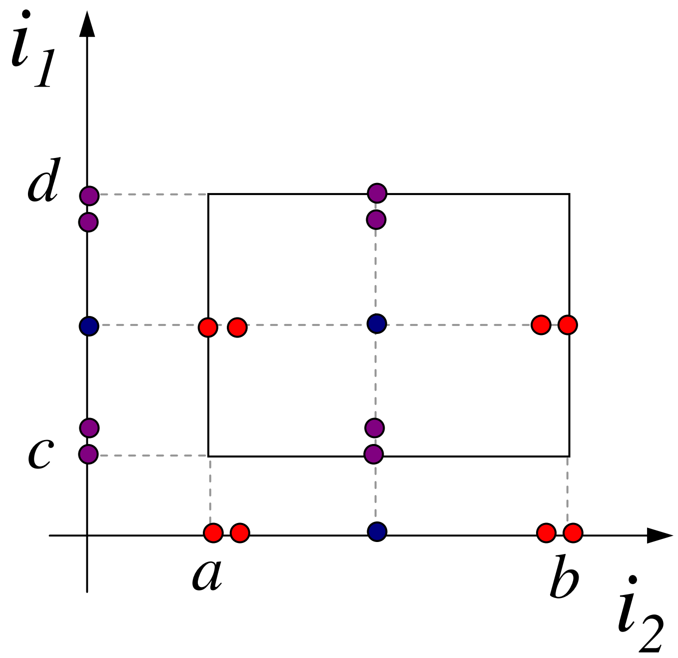
Choose 4 candidate values for each input in the range [a, b]:

- at the 2 extremes (a and b),
- near the 2 extremes (predecessor of a and successor of b).

Choose nominal values for all other variables.

Single-failure assumption: each failure is the result of a single bug (and a single error)

Assuming n input variables, 4n + 1 test-cases.



Boundary testing: mortgage example

Spec: Write a program that takes three inputs: gender (boolean), age([18-55]), salary ([0-10000]) and outputs the total mortgage for one person

Mortgage = salary * factor,

where factor is given by the following table.

Category	**Male**	**Female**
Young	(18-35 years) 75	(18-30 years) 70
Middle	(36-45 years) 55	(31-40 years) 50
Old	(46-55 years) 30	(41-50 years) 35

Boundary testing: mortgage example

Program solution

```
int mortgage (bool male, int age, int salary)
{
    if (male) then
      return ((18 age < 35)?(75  salary) : (31  age
      < 40)?(55  salary) : (30  salary))
    else                                // female
      return ((18  age < 30)?(75  salary) : (31  age
      < 40)?(50  salary) : (35  salary));
}
```

(12 bugs inside !!)

Boundary testing: mortgage example

Spec: Write a program that takes three inputs: gender (boolean), age([18-55]), salary ([0-10000]) and outputs the total mortgage for one person

```
int mortgage (bool male, int age, int salary)
{
      if (male) then
         return ((18 age < 35)?(75  salary) : (31  age < 40)?(55  salary) : (30
         salary))
      else                                          // female
         return ((18  age < 30)?(75  salary) : (31  age < 40)?(50  salary) : (35
         salary));
}
```

age: extremes: 18, 55(?). near extremes: 19, 54. nominal: 25.

salary: extremes: 0, 10000. near extremes: 1, 9999. nominal: 5000.

male: *true, false. nominal: true*.

No boundaries: define type-specific boundaries (e.g., 0 and MAXINT for integers).

Boundary testing: mortgage example

Gender	Age	Salary	Output	Correct	Pass/Fail
male	18	5000	75*5000	75*5000	P
male	19	5000	75*5000	75*5000	P
male	25	5000	75*5000	75*5000	P
male	54	5000	30*5000	30*5000	P
male	55	5000	30*5000	30*5000	P
male	25	0	75*0	75*0	P
male	25	1	75*1	75*1	P
male	25	9999	75*9999	75*9999	P
male	25	10000	75*9999	75*10000	P
female	25	5000	75*5000	70*5000	F

Boundary testing

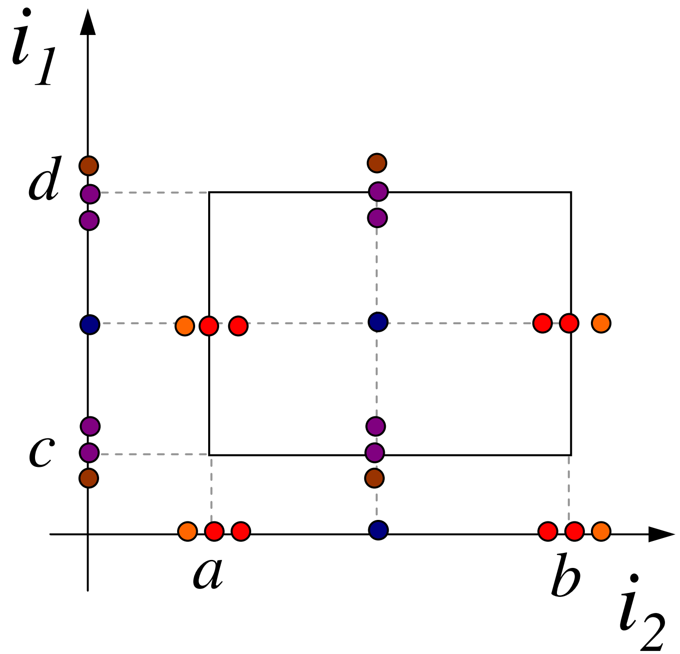
- Observations
  - strange technique for booleans: decision-table-based technique (yet to come)
  - not suitable due to the dependency between age and gender
  - more combinations to be tested: wait for a few slides!
  - finer partitioning needed: wait till next session

Robustness BV testing

In addition to the 4 candidates, choose 2 more candidates just beyond the extremes

- Predicting the output: tricky
- Suitable for PL's with weak typing (testing exception handling)

Assuming n input variables, 6n + 1 test-cases.



Robustness BV testing: mortgage example

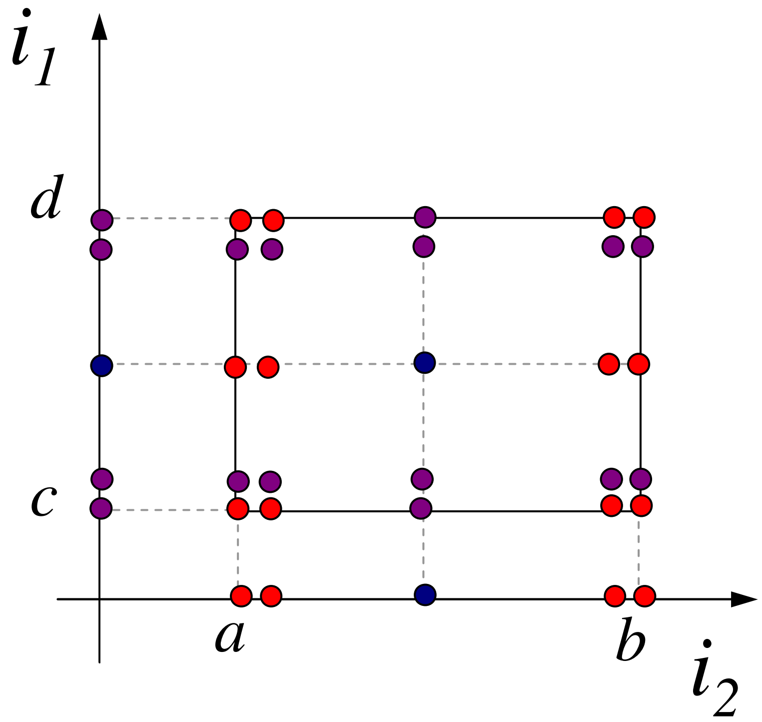
Gender	Age	Salary	Output	Correct	Pass/Fail
male	17	5000	30*5000	niet	F
male	56	5000	75*5000	too late	F
male	25	-1	75*-1	invalid salary	F
male	25	10001	75*10001	75*10000(?)	F

Worst-case BV testing

multiple-fault assumption: a fault may be the result of a combination of errors

all combinations of 5 values for all variables

5^n test-cases

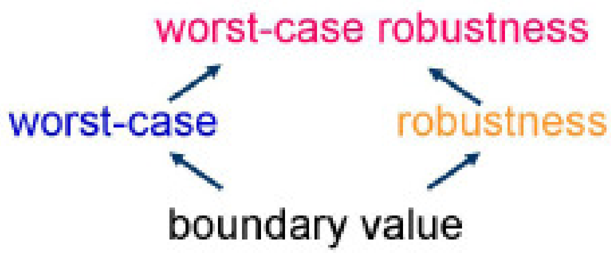


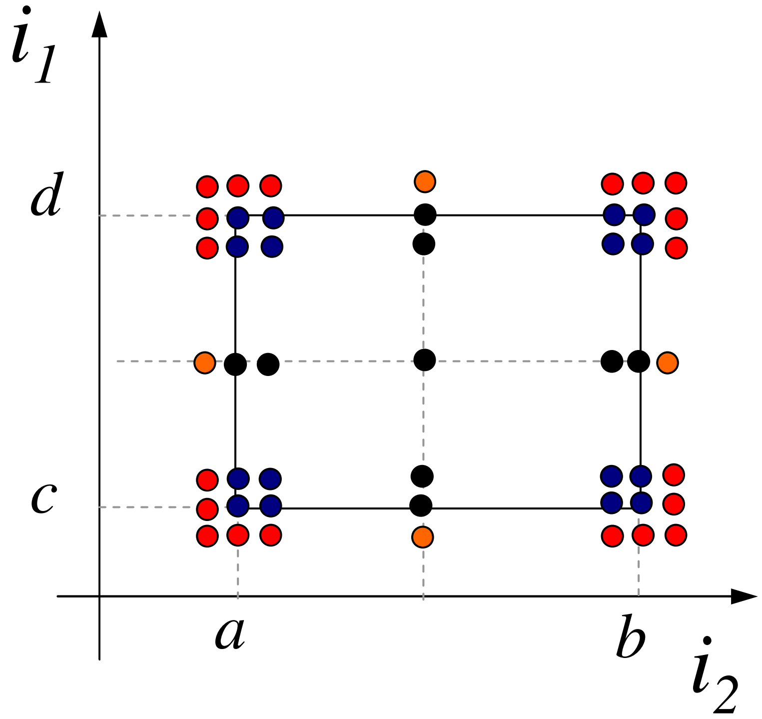
Worst-case+ robustness testing

combination of worst case and robustness BV Testing

all combinations of 7 values for all variables

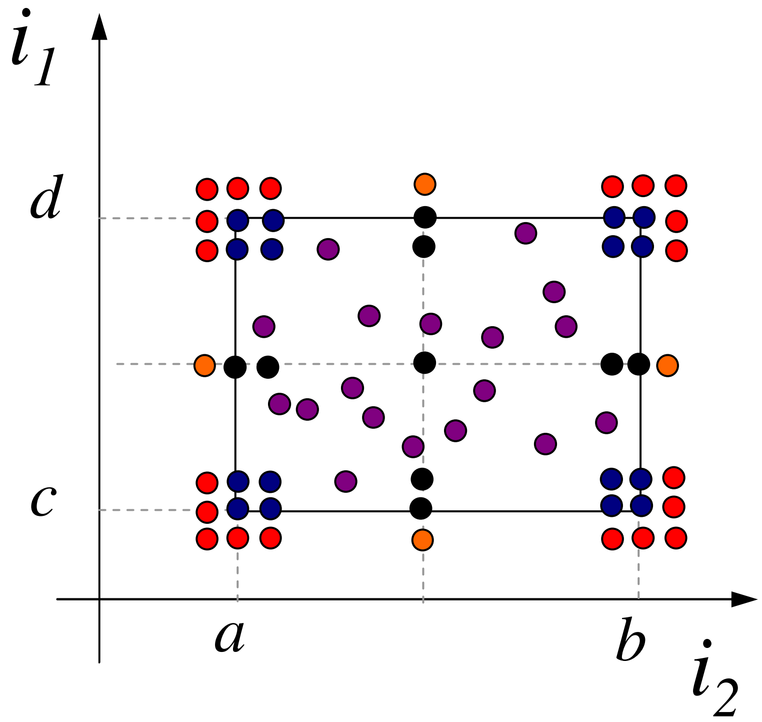
7^n test-cases





Combine w. Random …

combination with randomly selected values



Special values

using domain knowledge

finding corresponding boundaries for internal variables

in combination with the techniques mentioned before

Gender	Age	Salary	Output	Correct	Pass/Fail
male	18	1	75*1	75*1	P
male	35	1	55*1	75*1	F
male	36	1	55*1	55*1	P
male	45	1	30*1	55*1	F
male	46	1	30*1	30*1	P
male	55	1	30*1	30*1	P
female	18	1	75*1	70*1	F
female	30	1	35*1	70*1	F
female	31	1	50*1	50*1	P
female	40	1	35*1	50*1	F
female	41	1	35*1	35*1	P
female	50	1	35*1	35*1	P

Equivalence classes (weak)

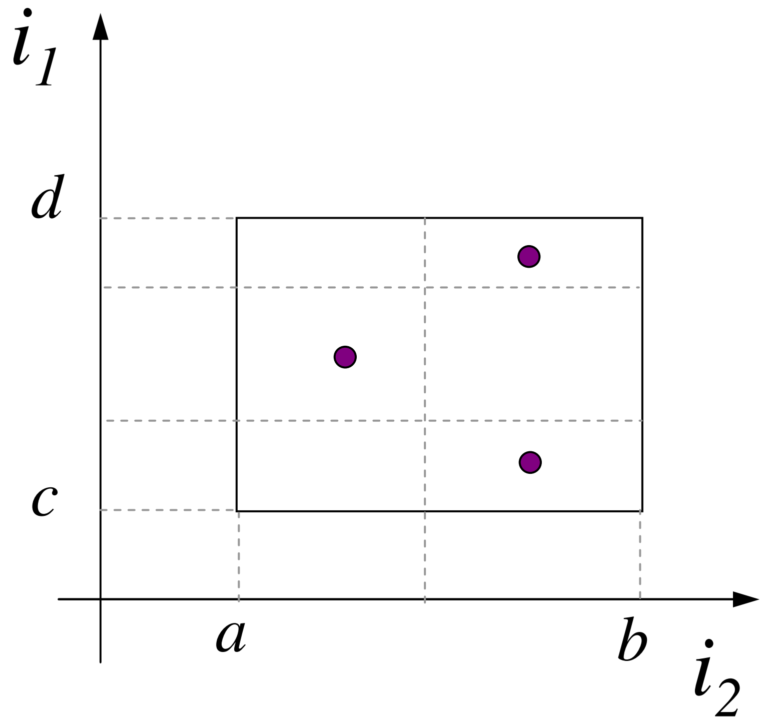
Define equivalence classes on the domain (range) of input (output) for each variable: (independent input)

(weak) cover equivalence classes for the domain of each variable: single fault assumption

how many test-cases are needed?

- max(n,m) = the minimal number of tokens in an m × n grid such that each row and column contains at leats one token?

also called: (equivalence, category) partition method



Equivalence classes: mortgage example

Spec: Write a program that takes three inputs: gender (boolean), age([18-55]), salary ([0-10000]) and outputs the total mortgage for one person

Mortgage = salary * factor,

where factor is given by the following table.

Category	**Male**	**Female**
Young	(18-35 years) 75	(18-30 years) 70
Middle	(36-45 years) 55	(31-40 years) 50
Old	(46-55 years) 30	(41-50 years) 35

age: [18-30], [31-35], [36-40], [41,45], [46-50], [51-55]

salary: [0-10000]

male: as strange as boundary value! true, false

Equivalence classes: mortgage example

Gender	Age	Salary	Output	Correct	Pass/Fail
male	20	1000	75*1000	75*1000	P
female	32	1000	50*1000	50*1000	P
male	38	1000	55*1000	50*1000	P
female	42	1000	35*1000	35*1000	P
male	48	1000	30*1000	30*1000	P
female	52	1000	35*5000	too late!	F

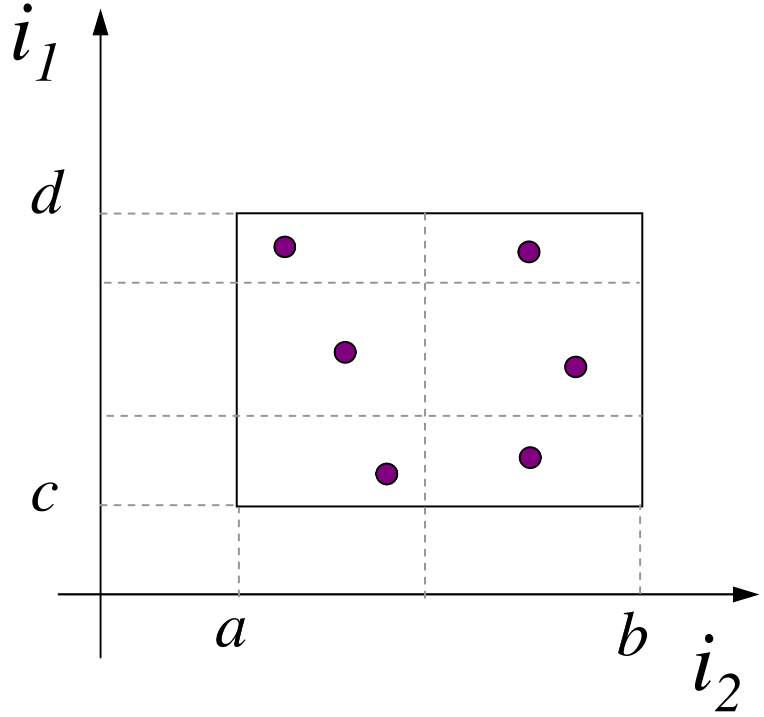
Equivalence classes (strong)

cover all combinations of equivalence classes for the domain of all variables:

multiple fault assumption

number of test-cases

$\Pi_x(S_x)$



Equivalence classes: mortgage example

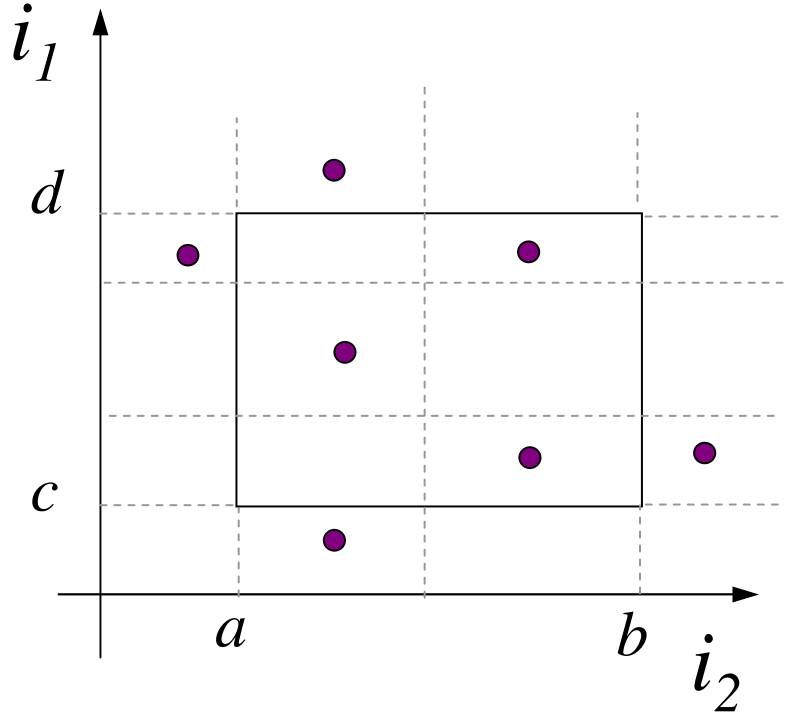
Gender	Age	Salary	Output	Correct	Pass/Fail
female	20	1000	75*1000	70*1000	F
female	32	1000	50*1000	50*1000	P
female	38	1000	50*1000	50*1000	P
female	42	1000	35*1000	35*1000	P
female	48	1000	35*1000	35*1000	P
female	52	1000	35*5000	too late!	F
male	20	1000	75*1000	75*1000	P
male	32	1000	50*1000	75*1000	F
male	38	1000	55*1000	50*1000	P
male	42	1000	30*1000	55*1000	F
male	48	1000	30*1000	30*1000	P
male	52	1000	30*1000	30*1000	P

Weak Robust EC

includes weak normal; adds out of range test-cases for each variable

number of test-cases

$(\max_x |S_x|) + 2n$



Strong robust EC: mortgage example

Gender	Age	Salary	Output	Correct	Pass/Fail
male	17	1000	30*1000	too young!	F
female	56	1000	35*1000	too late	F
male	36	-1	55*-1	0	F
female	36	10001	50*10001	50*10000	F

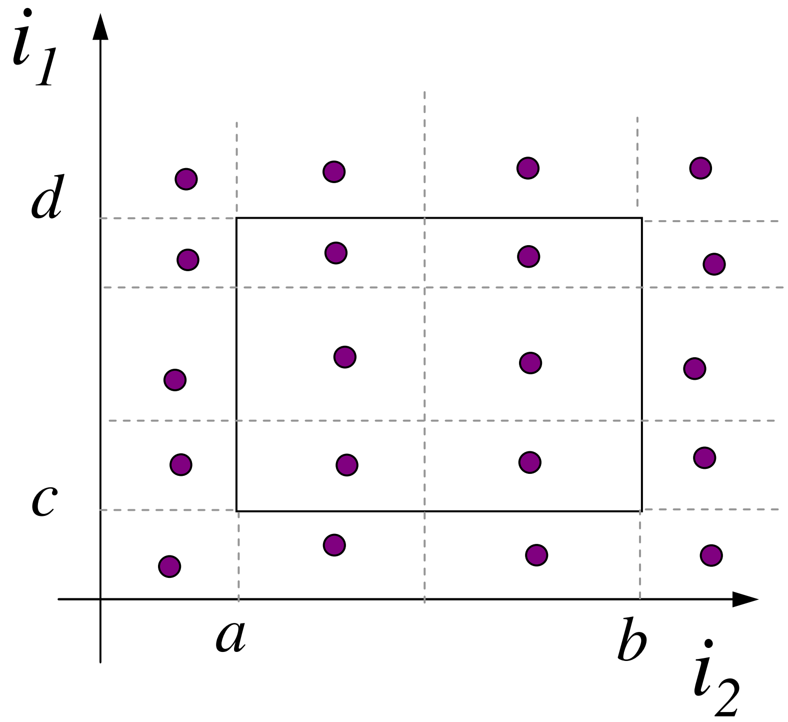
Strong Robust EC

Same as strong normal but also checks for all out of range combinations

Number of test-cases

$\prod_x (|S_x| + 2)$

$A \longrightarrow B$: Test-cases of A (faults detected by A) is a subset of those of B.



Weak robust EC: mortgage example

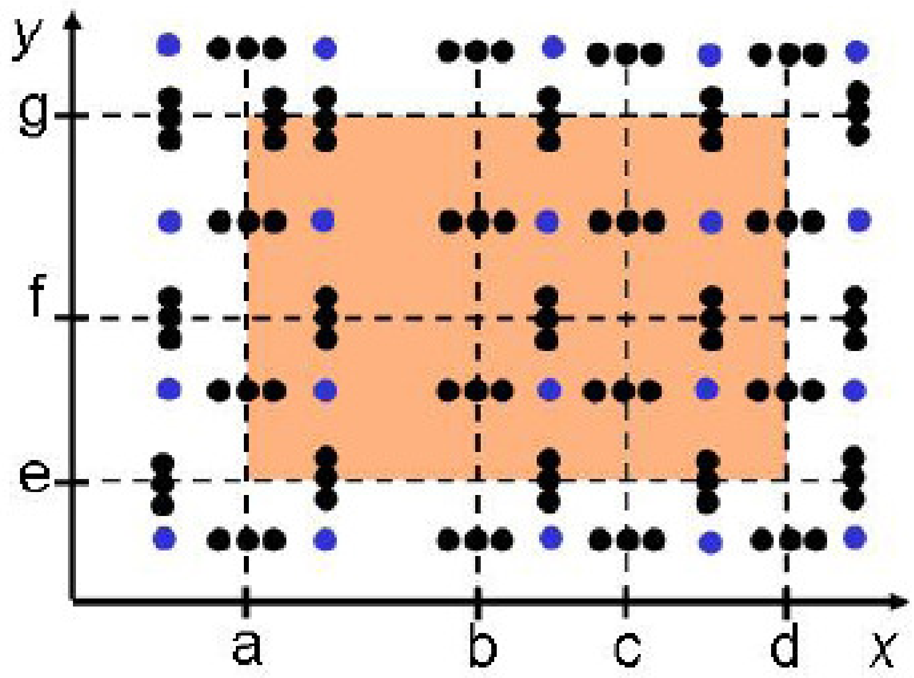
Gender	Age	Salary	Output	Correct	Pass/Fail
male	17	1000	30*1000	too young!	F
female	56	1000	35*1000	too late	F
female	17	1000	35*1000	too young!	F
male	56	1000	30*1000	too late	F
male	36	-1	55*-1	0	F
female	36	10001	50*10001	50*10000	F

. . .

Combined with WCT

Techniques can be combined

Es: Robust WCT + Robust EC



Combined with BV

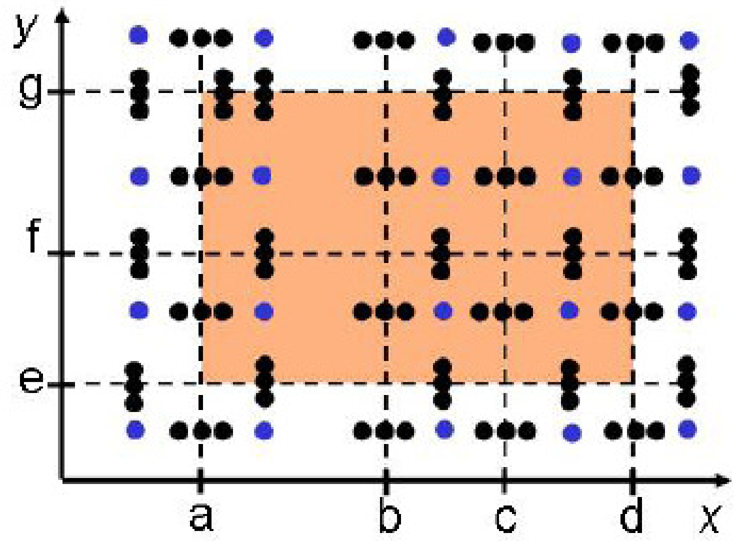
Strong EC + Robust BV

number of test-cases:

$\prod_x 4(|S_x| + 1)$, whopping

>100 test-cases for the mortgage example (it catches all 12 bugs!)

too many for any real-life program
e.g., 5 vars., each 5 partitions:
8 million test-cases
1 sec. for each test-case:
3 months testing!



Problems

- No constraints on the equivalence classes
- Dependencies among different variables not taken into account
- No choice among relevant classes (e.g., apply worst-case testing on some and boundary values on others)