BLACK BOX TESTING (II)

Random Testing

Random testing (monkey testing)

Having difficulties to formulate equivalence classes (equivalence relation), random testing can be exercised

Random selection of inputs following some probability distribution function (uniform, normal, etc.)

Effectiveness of testing (number of test cases) depends upon the “size” of equivalence classes

Infinite monkey theorem

a monkey hitting keys *at random* on a typewriter keyboard for an infinite amount of time will almost surely type any given text, such as the complete works of William Shakespeare.

Random testing (monkey testing)

input domain $E = [0,1]^n$

Equivalence class:

E_1= hypercube: e - length of side

$E_2=E-E_1$

Prob (x in E_1) $=e^n$

Random testing (monkey testing)

input domain E = [0,2]^n

Equivalence class: hyperball of unit radius (r=1)

n	r^n	volume V(hyperball)
2	4	3.14 (πr^2)
3	8	4.18 ($4/3\pi r^3$)
5	32	5.26
10	1.024	5.26
20		2.55
40		$3.60 * 10^{-9}$
60		$3.09 * 10^{-18}$

$$V = \frac{\pi^{n/2}}{\Gamma(1+\frac{n}{2})} r^n$$

Fuzz testing (fuzzing)

automated software testing:
injecting invalid, random, unexpected inputs (data) to reveal software vulnerabilities

a spectrum of options: from random data to crafted inputs

Focus on anomalies:
-memory leaks, failing built-in assertions, crashes
-exception handling
-buffer overflows
-injection flows (SQL injections)
-denial of service (DOS)

Fuzz testing: applications

2012, Google **ClusterFuzz**,
a cloud-based fuzzing infrastructure for security-critical components of the Chromium web browser

2016, Microsoft **Project Springfield**,
a cloud-based fuzz testing service for finding security critical faults in software.

2020, Microsoft **OneFuzz**,
a self-hosted fuzzing-as-a-service platform automating the detection of software faults. Windows and Linux are supported.

Operational Profiles

Operational profiles

Some equivalence classes A_i s used more frequently

Test software as if it were used by customers

Operational profile (OP): list of disjoint operations and probabilities of occurrence

Set S
Collection of subsets $A_1, A_2, \ldots, A_c$
such that they satisfy the following conditions

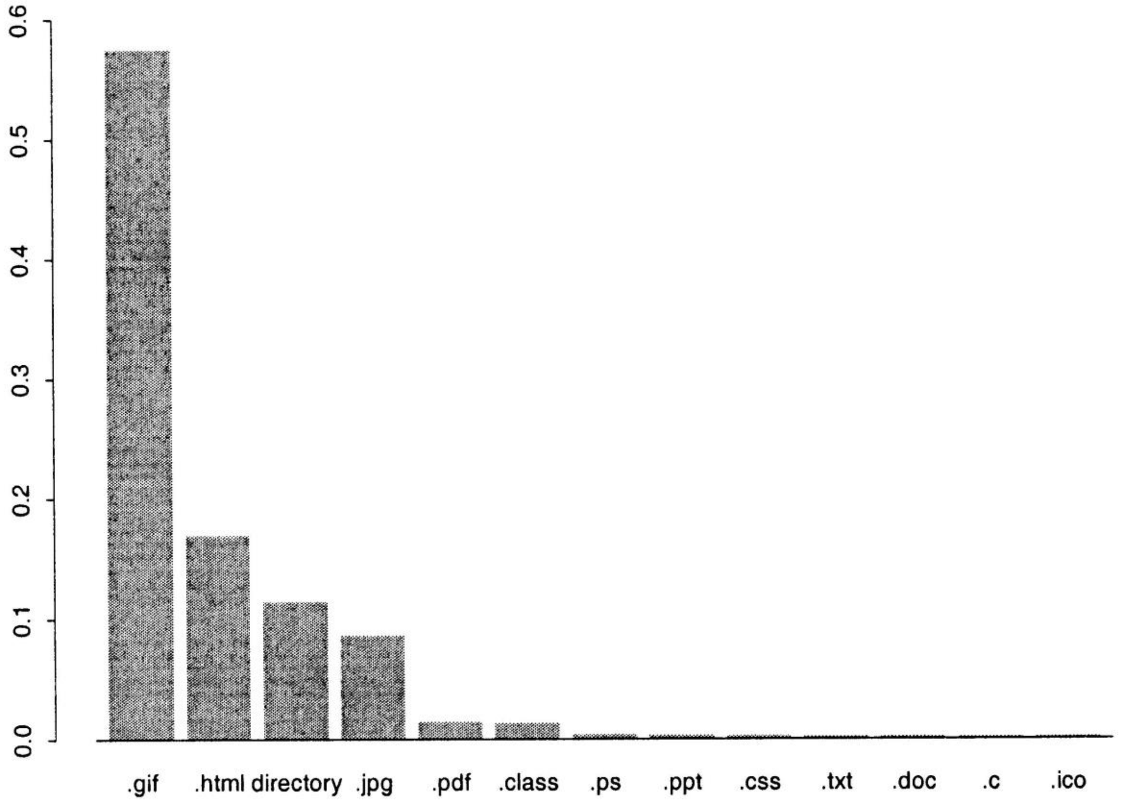
- Mutually exclusive
- Collectively exhaustive

Operational profile

Usage frequencies (hits) for different types of requested files for a given site

File type	Hits	% of total
.gif	438536	57.47%
.html	128869	16.89%
directory	87067	11.41%
.jpg	65876	8.63%
.pdf	10784	1.41%
.class	10055	1.32%
.ps	2737	0.36%
.ppt	2510	0.33%
.css	2008	0.26%
.txt	1597	0.21%
.doc	1567	0.21%
.c	1254	0.16%
.ico	849	0.11%
Cumulative	753709	98.78%
Total	763021	100%

Operational profile



Operational profiles

Progressive testing – start with testing operations with the higher probability of occurrence

Benefits:

- **Productivity improvement and schedule gains**
- **Faster introduction of new products by implementing highly used features quickly to capture market share**
- **Better communication with customers and better customer relations**
- **High return on investment (lower cost)**

Benefits of operational profiles

For AT&T – PBX switching system project:

•Customer-reported problems and maintenance costs by factor of 10

•System testing time by factor of 2

•Product introduction time by 10%

Definition- operational profile

Profile – a set of disjoint (only one can occur at a time) alternatives with the probability that each will occur.

A occurs 60% of time; B occurs 40% time

Profile (A, 0.6) (B, 0.4)

Profile

OP considered as a <u>prime candidate</u> for testing *large* scale software with many users and *diverse* usage environments

Once OP has been constructed, it supports statistical testing by some sampling procedure to select test cases according to the highest probability values

Profile

Economic gain

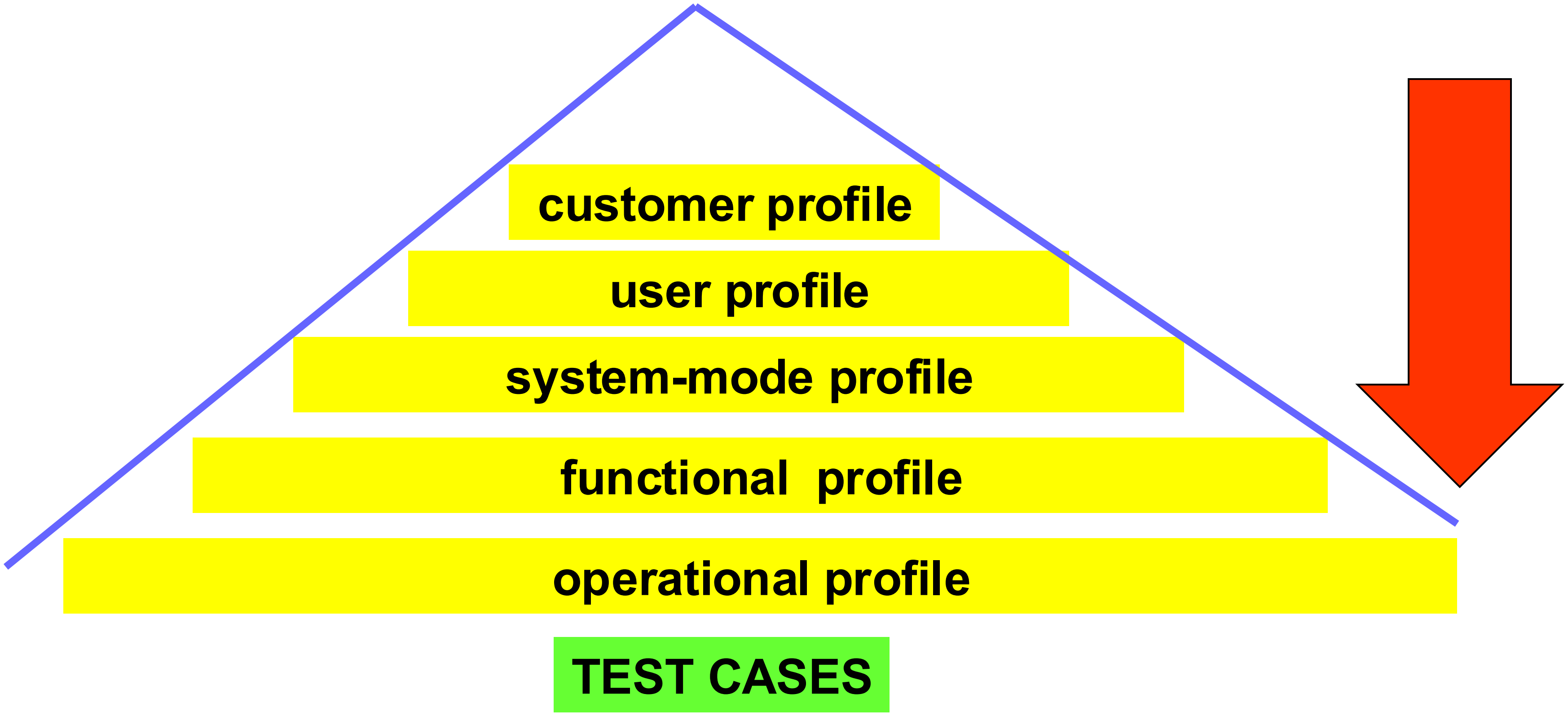
Engineering judgment

Frequently used functionality

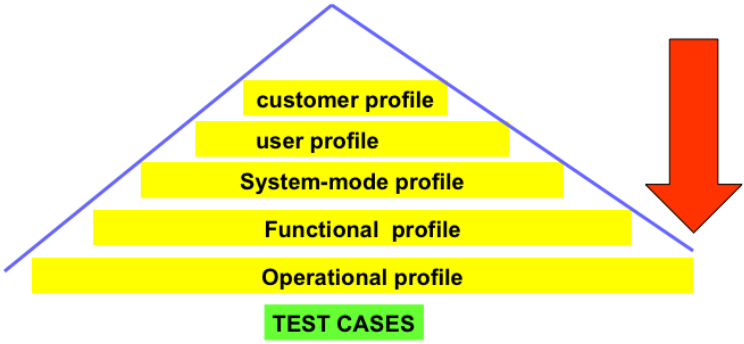
Rarely used functionality whose failures could lead to disastrous consequences

Development of operational profiles: top-down approach (Musa)

Looking at use of system from a progressively narrowing perspective- from customer down to operation



Profiles



Customer profile- person, group, institution that acquires the system

User profile – set of users who employ the system in the same way

System mode-profile - set of functions (operations) grouped for convenience in analyzing execution behavior. For instance, administrative mode, maintenance mode, overload mode, normal, initialization…

Functional profile – quantitative picture of the relative use of different functions (usually developed during requirement definition; a part of feasibility study)

Operational profile- consists of operations which represent a particular task with certain specific input variables

Development of operational profiles

customer profile

user profile

System-mode profile

Functional profile

Operational profile

TEST CASES

Acquisition of the software product

Example

Customer type	**weight**
Corporation	0.50
Government	0.40
Education	0.05
Others	0.05

Development of operational profiles

customer profile

user profile ← Usage (users) of the software product

System-mode profile

Functional profile

Operational profile

TEST CASES

Example

Customer type	**weight**
Corporation	0.50
Government	0.40
Education	0.05
Others	0.05

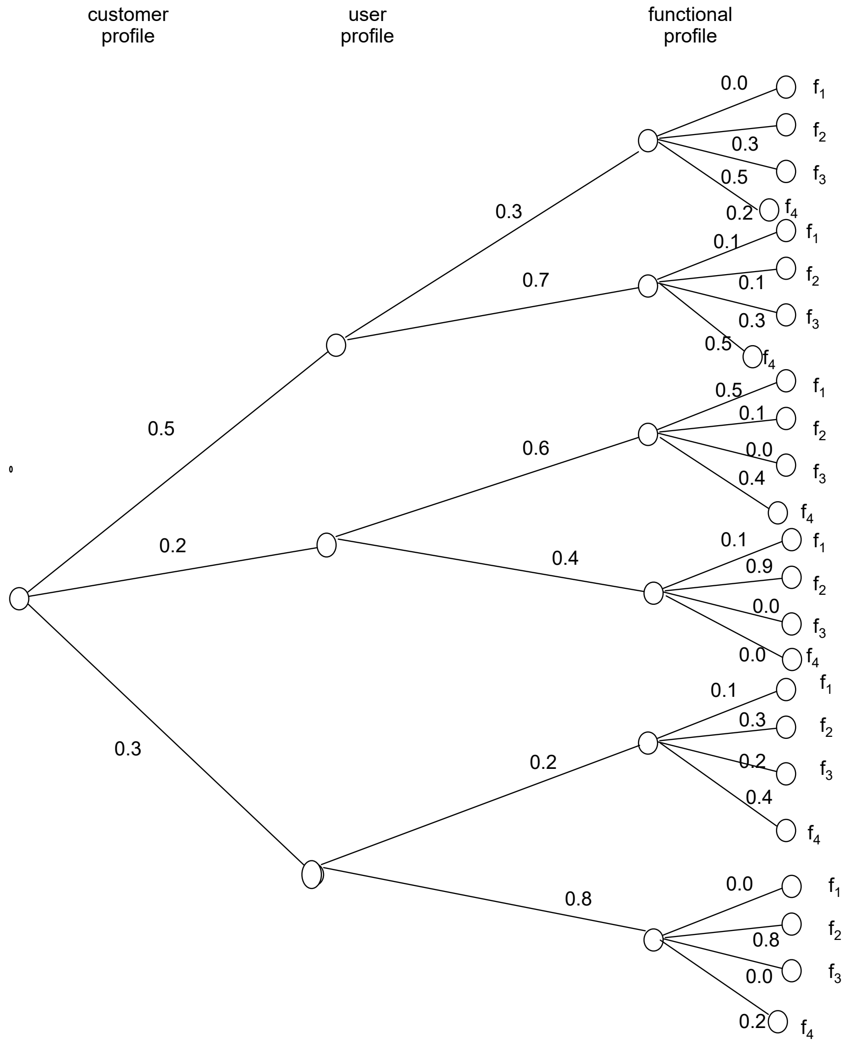
User profile

Example

Customer type	**weight**
Corporation	0.50
Government	0.40
Education	0.05
Others	0.05

User Type	Customer type				**overall user profile**
	corp	gov	edu	others	
	0.50	0.40	0.05	0.05	
End user	0.80	0.90	0.90	0.70	0.84
Datab. Admin	0.02	0.02	0.02	0.02	0.02
Programmer	0.18	0.00	0.00	0.28	0.104
3^{rd} party	0.00	0.08	0.08	0.00	0.036

User profile-graph representation



Construction of operational profiles

Measurements of usage
(at customer installations; business sensitive data)

Survey of target customers

Usage estimation based on expert opinions

Faults and minimal number of tests

A given test is aimed at discovering a collection of faults

Determine a minimal number of tests “covering” all faults

fault	t1	t2	t3	t4
f1		1		
f2	1	1	1	
f3	1		1	
f4			1	1
f5		1		1

Fault-test coverage matrix D=[dij]

Faults and minimal number of tests

Introduce binary variable:

ci =1 if test i^th ti is included in a collection of tests,

ci =0, otherwise

fault	t1	t2	t3	t4
f1		1		
f2	1	1	1	
f3	1		1	
f4			1	1
f5		1		1

Select a minimal number of tests (collection of tests) so that all faults are covered

Min {c1+c2+c3+c4}

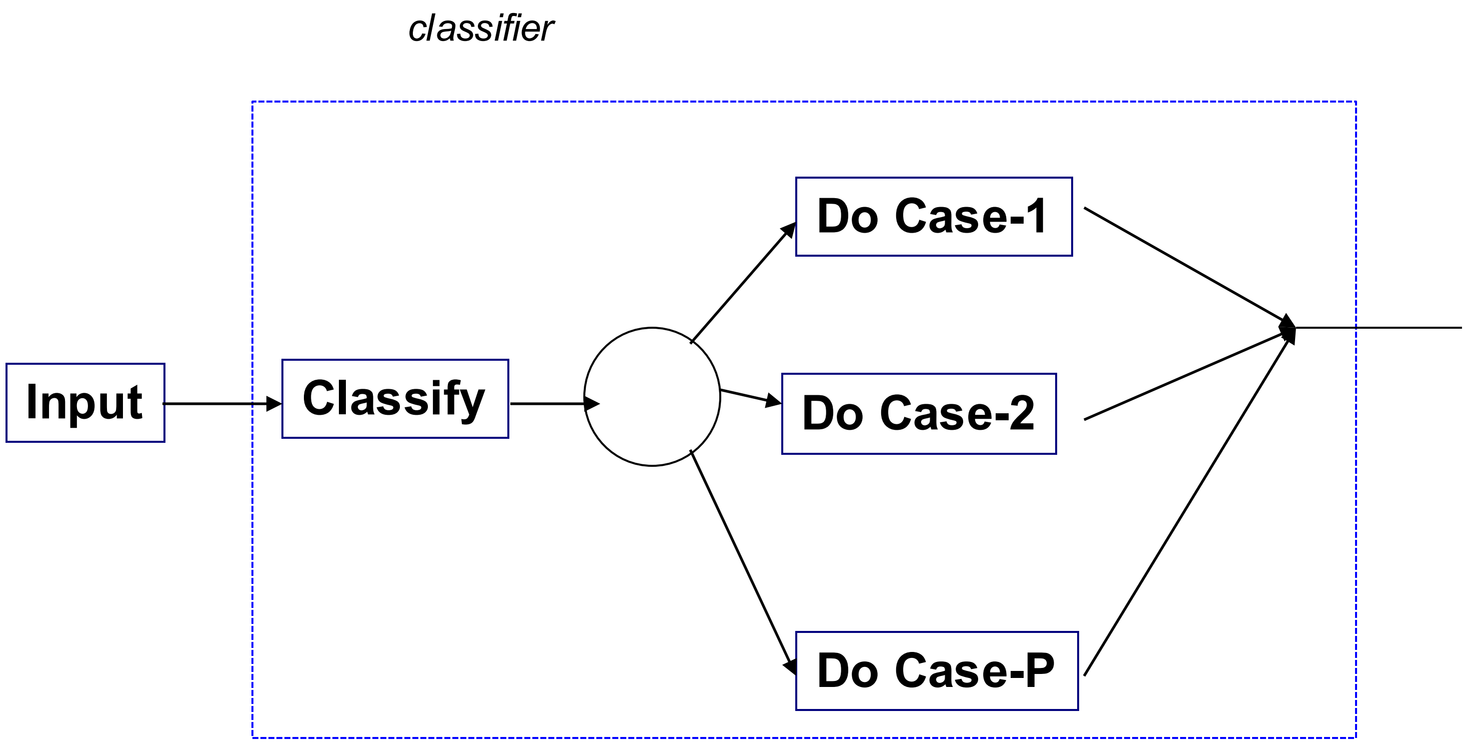
subject to coverage all faults

$$\sum_j d_{1j}c_j \geq 1 \quad \sum_j d_{2j}c_j \geq 1 \quad \sum_j d_{3j}c_j \geq 1$$

$$\sum_j d_{4j}c_j \geq 1 \quad \sum_j d_{5j}c_j \geq 1$$

Input Domain Testing

Input Domain Testing

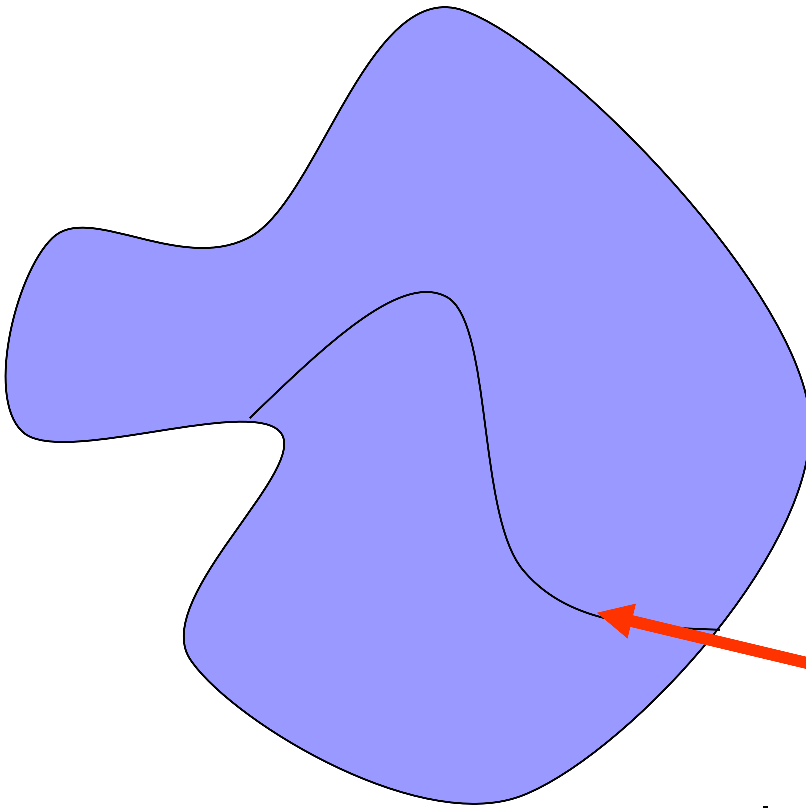


EXAMPLES – CATEGORIES OF COMPUTING

Parts of specifications given in terms of numerical inequalities

Heavy numeric processing with conditionals: payroll, taxes, financial computing

Input domain testing: Basic notation (1)



Input variables as vectors of numbers

$\mathbf{x} = [x_1 \;\; x_2 \;\;\; x_n]^T$

Input domain- all points representing all allowable inputs identified by the specifications

Input sub-domain- a subset of the input domain

$f(x_1, x_2, \ldots, x_n)$ rel K

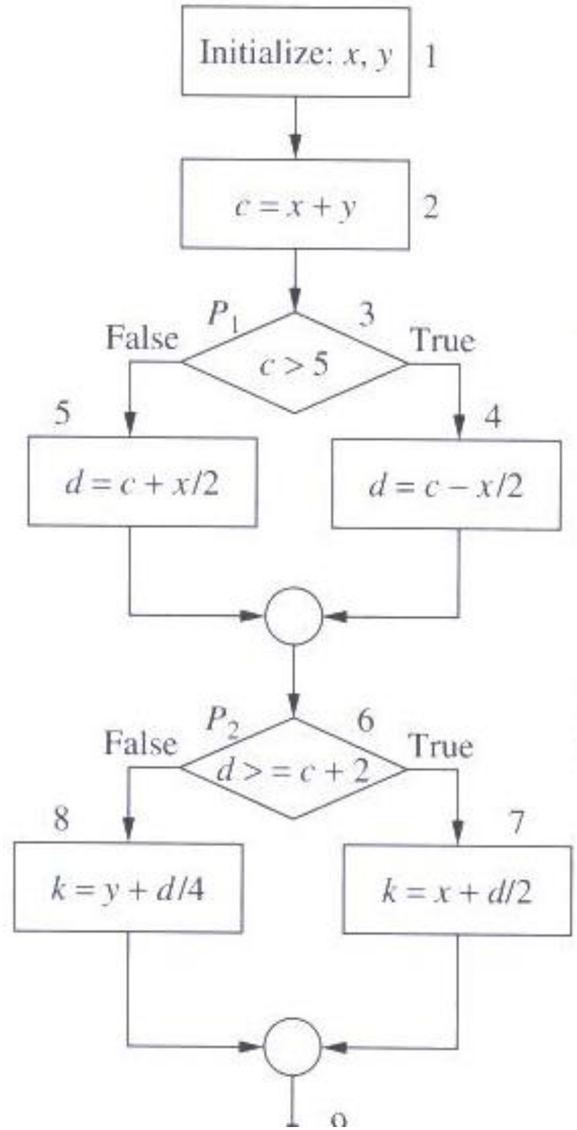
rel={ less than, greater than, equal,…..}

Domain analysis: example (1)

```
int codedomain(int x, int y){
  int c, d, k
  c = x + y;
  if (c > 5) d = c - x/2;
  else       d = c + x/2;
  if (d >= c + 2) k = x + d/2;
  else            k = y + d/4;
  return(k);
}
```

Domain analysis: example (2)

```
int codedomain(int x, int y){
  int c, d, k
  c = x + y;
  if (c > 5) d = c - x/2;
  else       d = c + x/2;
  if (d >= c + 2) k = x + d/2;
  else            k = y + d/4;
  return(k);
}
```



P_1: $x + y > 5$

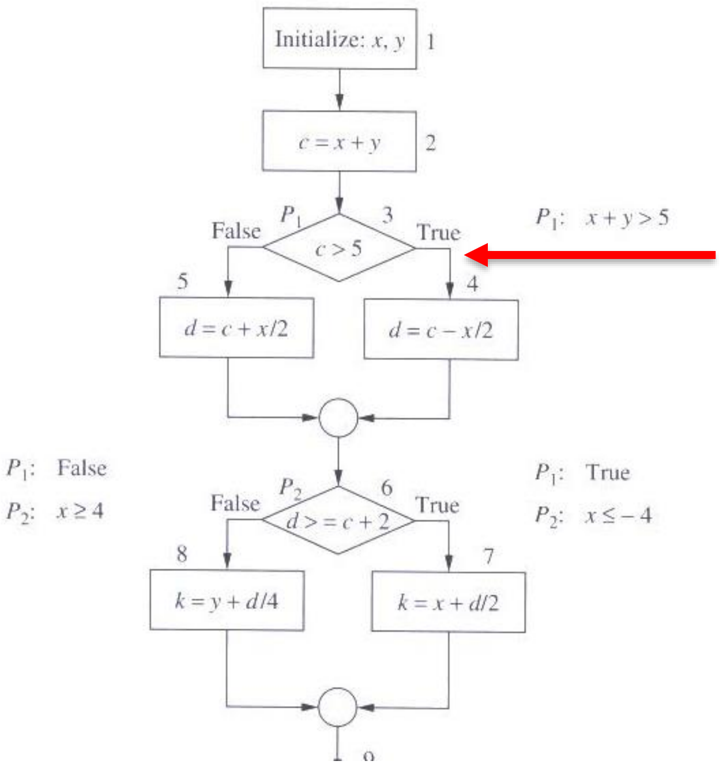
P_1: False

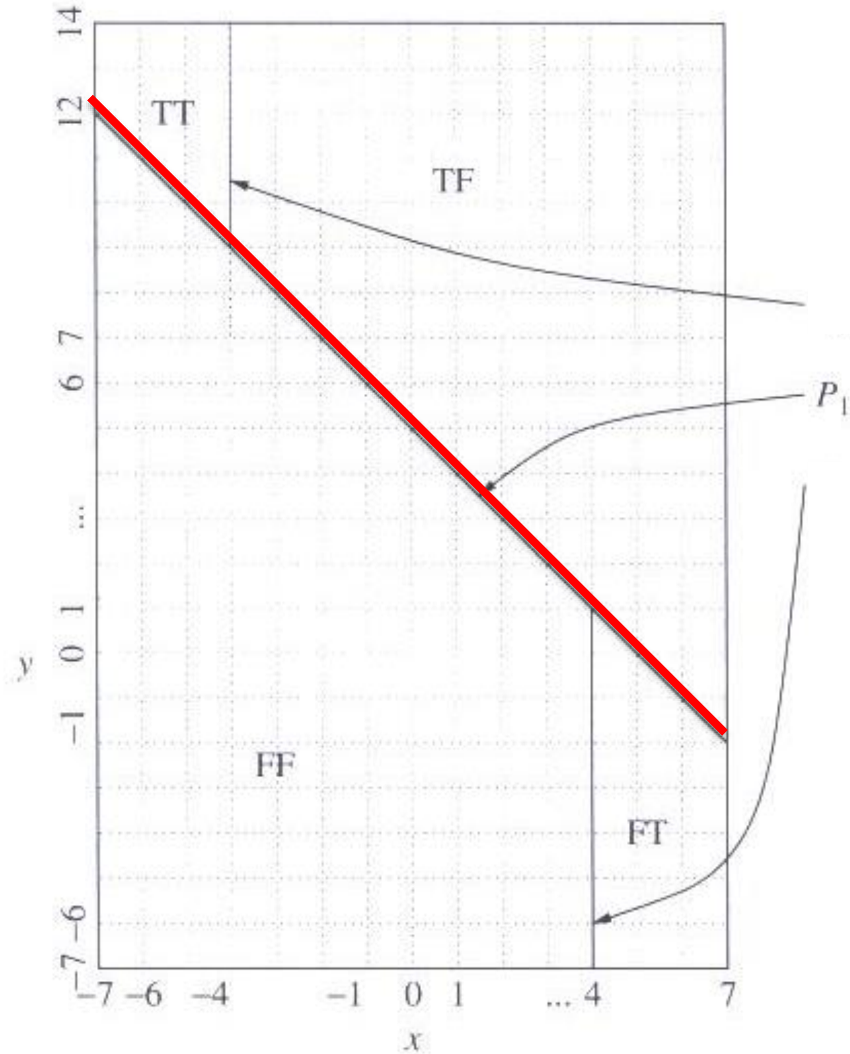
P_2: $x \geq 4$

P_1: True

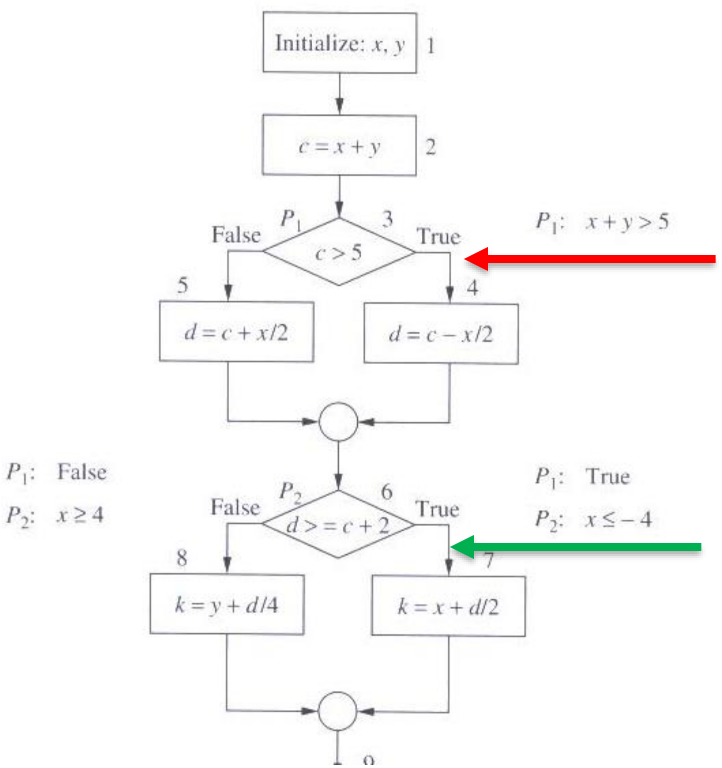
P_2: $x \leq -4$

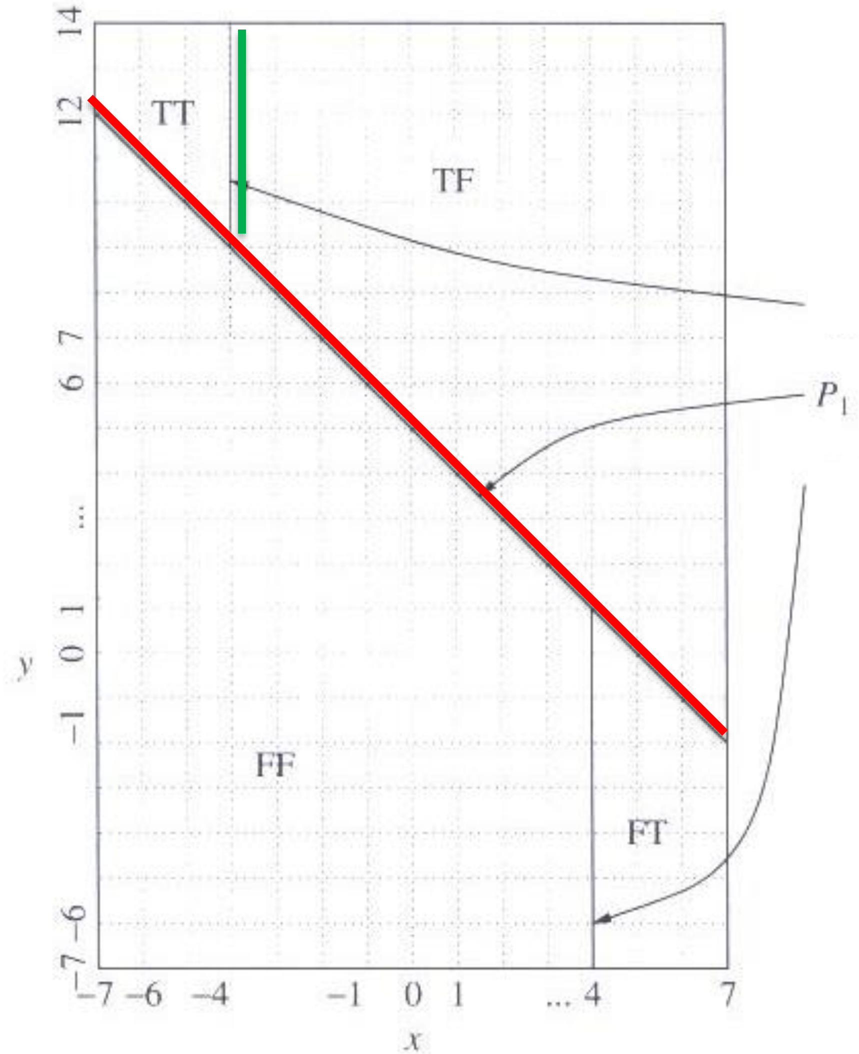
Domain analysis: example (3)





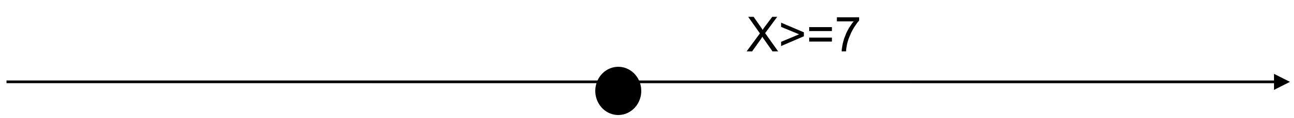
Domain analysis: example (4)



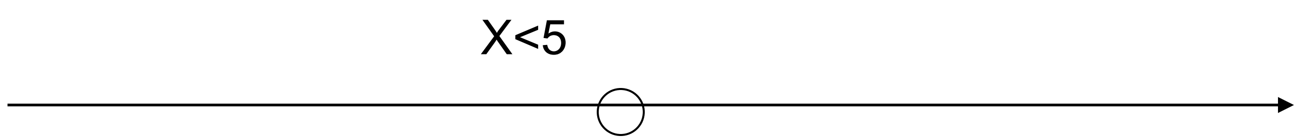


Input domain testing: Basic notation (2a)

Closed boundary (with respect to a specific sub-domain) if all boundary points belong to this sub-domain



Open boundary (with respect to a specific sub-domain) none of the boundary points belongs to the sub-domain



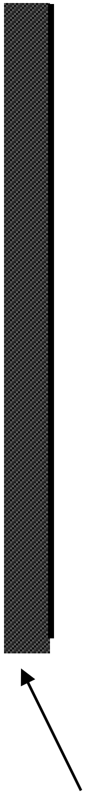
Input domain testing: Basic notation (2b)

Open sub-domain – a sub-domain with *all* open boundaries

closed sub-domain – a sub-domain with *all* closed boundaries

Interior point – a point belonging to a sub-domain but not on the boundary
Exterior point – a point not belonging to a sub-domain and not on its boundary

Input domain testing: Basic notation (2c)



side on which the domain is closed

Input domain testing: Basic notation (3)

Domain partition – partition of the input domain into a number of sub-domains (sub-domains mutually exclusive and exhaustive)

Boundary – where two sub-domains meet

$f(x_1, x_2, \ldots, x_n) = K$ (if inequalities used for sub-domains)

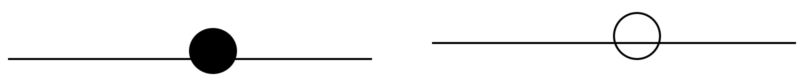
Linear and nonlinear boundaries (domains)

Boundary point – point on the boundary

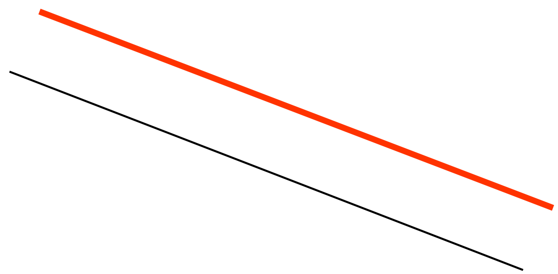
Boundary problems

Specification – implementation gap

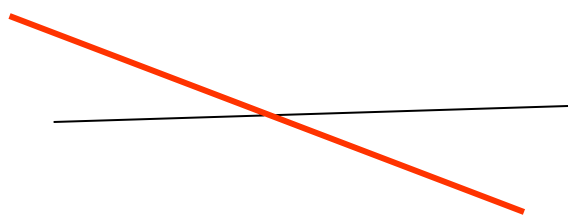
Closure problem
(open – closed)



Boundary shift
$f(x_1, x_2, \ldots, x_n) = K + \delta$

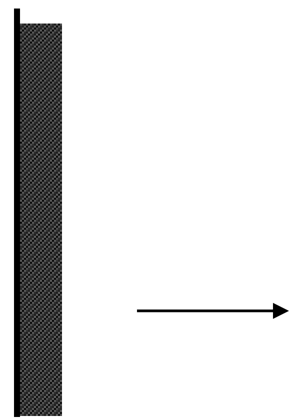


Boundary tilt
$f(x_1, x_{2\varepsilon}, \ldots, x_n) = K$



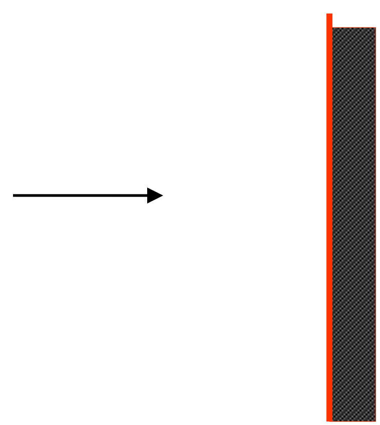
Boundary problems

Missing boundary



No boundary-
All points receive the same treatment

Extra boundary



extra boundary-
different treatment of points

Extreme Point Combination (EPC) Strategy

Domain testing strategy similar to capacity testing, stress testing or robustness testing (extreme input values, other limits are contested)

Heuristics: usage of extreme values (extreme points) combination

Extreme Point Combination (EPC) Strategy

For sub-domain, complete a simple domain analysis to identify the domain limits in each dimension (variable)

Choose for x_i: max x_i, min x_i, slightly under min x_i, slightly over max x_i

Produce all possible combinations of inputs with each of its variables taking on one of the four values shown above. In n-dim space we end up with 4^n points +1 -one point added inside the domain (to assure domain coverage strategy)

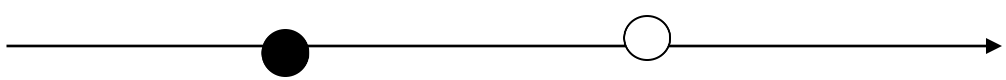
n =1   4+1 =5 points
n = 2   17 points

Extreme Point Combination (EPC) Strategy: Examples (1)

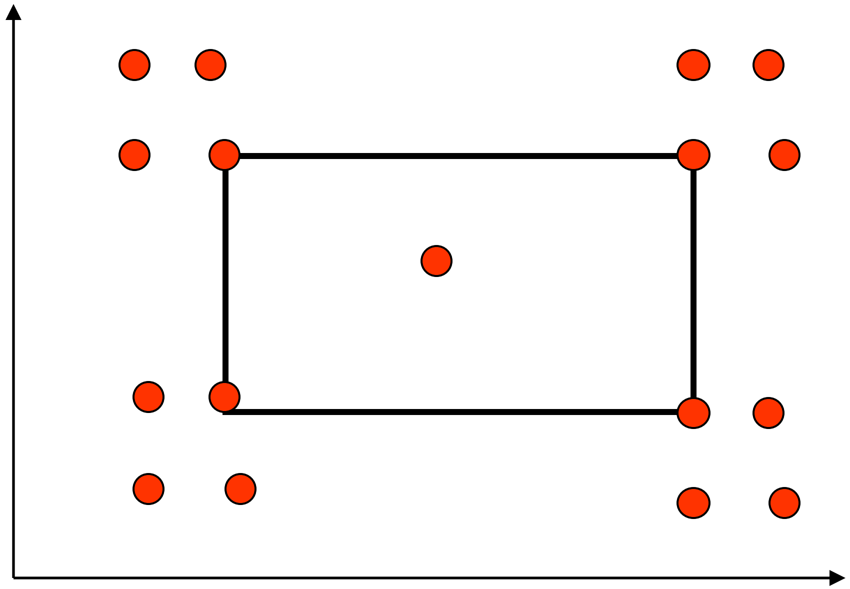
For sub-domain, complete a simple domain analysis to identify the domain limits in each dimension

Choose for x_i: max x_i, min x_i, slightly under min x_i, slightly over max x_i

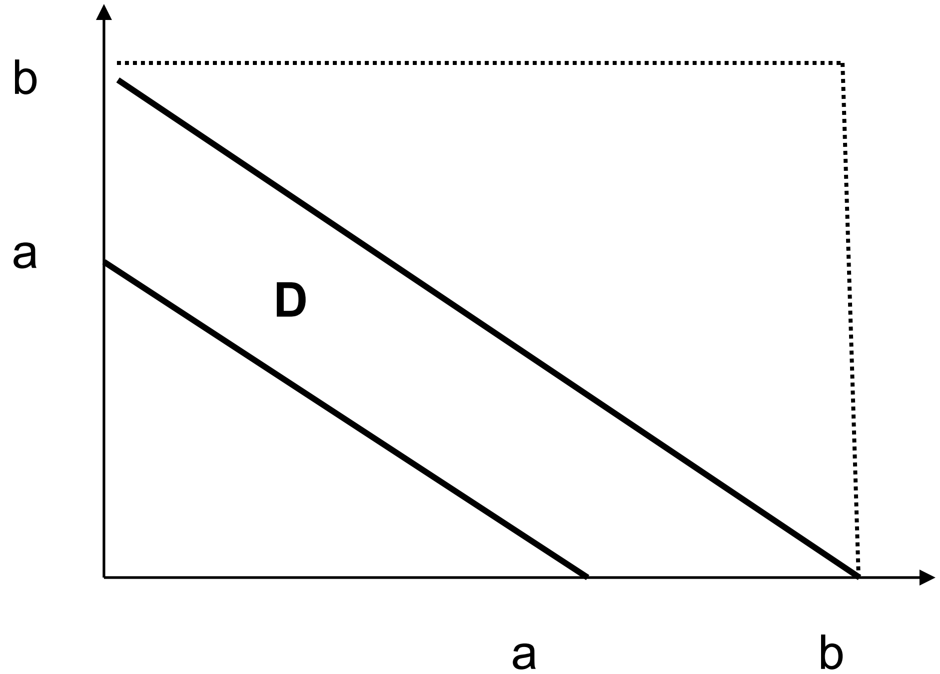
Produce all possible combinations of inputs with each of its variables taking on one of the four values shown above. In n-dim space we end up with 4^n points +1 -one point inside the domain (domain coverage strategy)



Extreme Point Combination (EPC) Strategy: Examples (2)

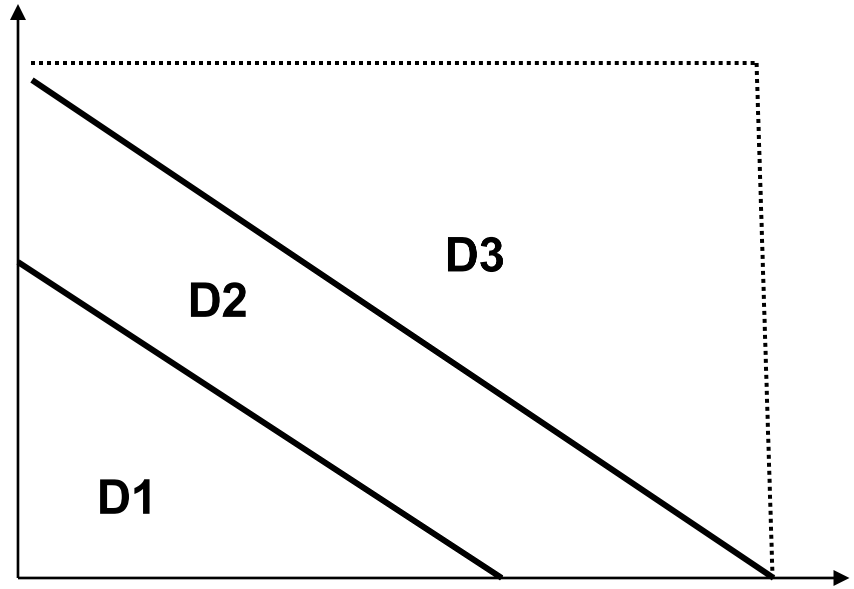


Extreme Point Combination (EPC) Strategy: Examples (2a)



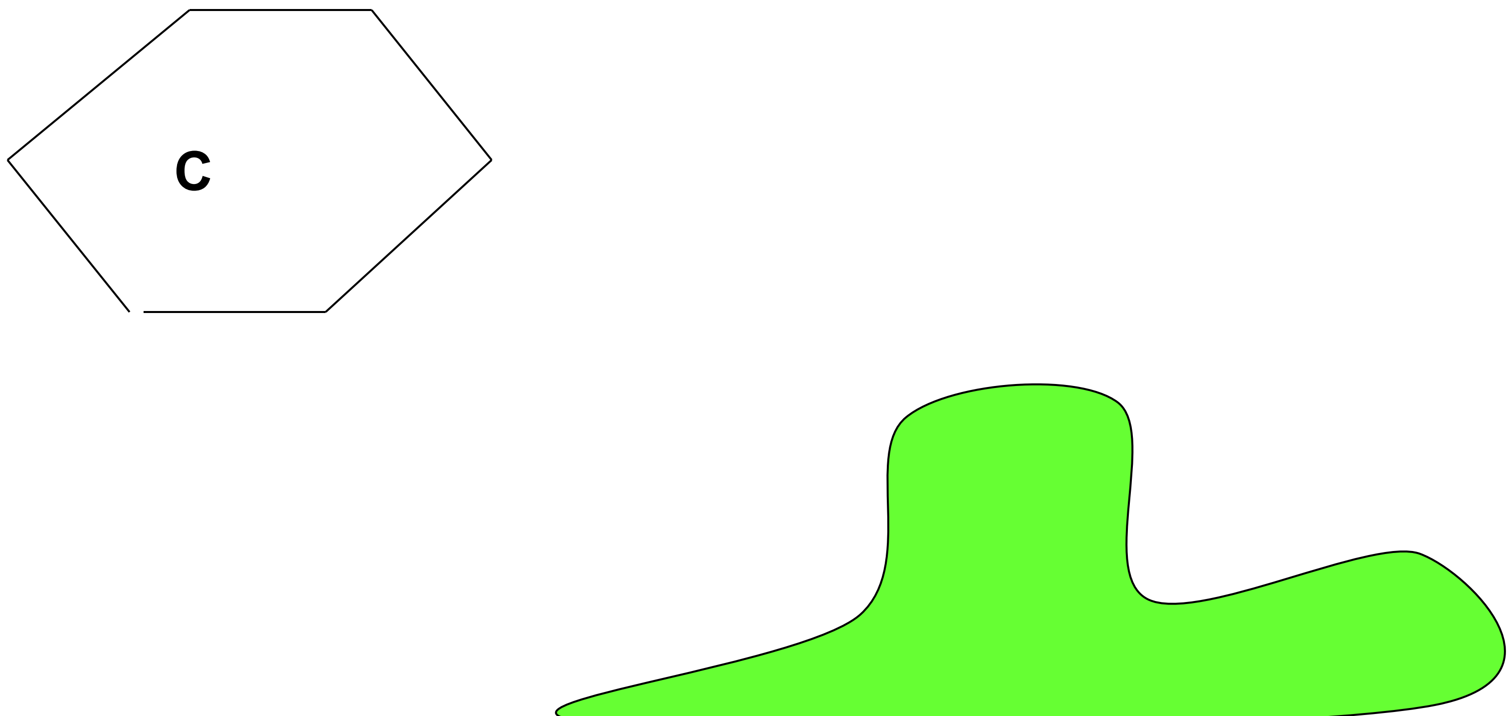
Subdomain: D

Extreme Point Combination (EPC) Strategy: Examples (2b)



Subdomains: D1, D2, D3

EPC in higher dimensional sub-domains



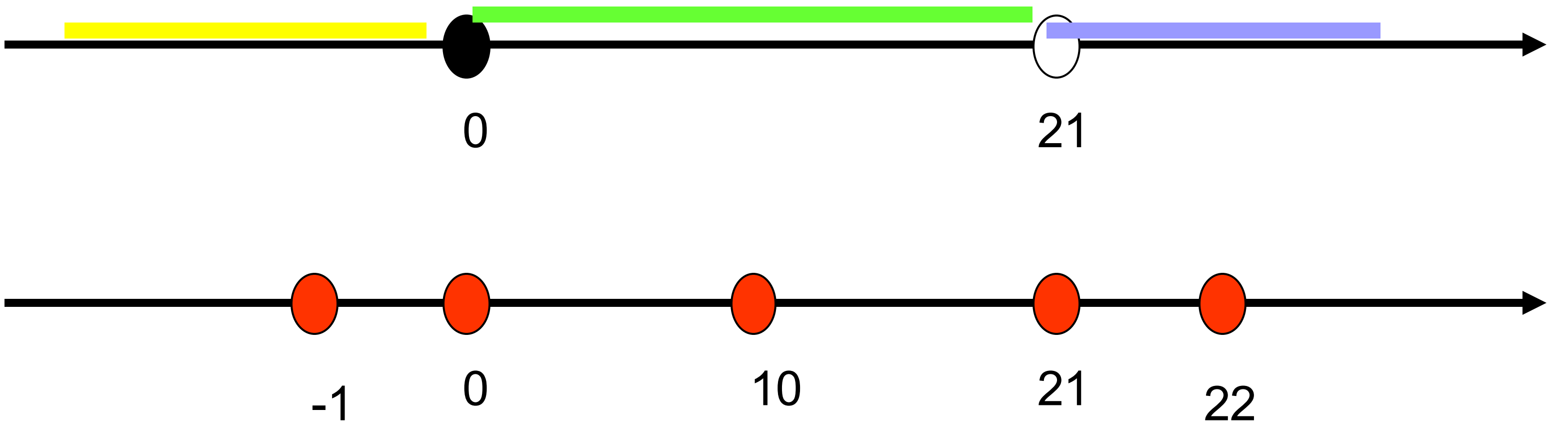
Extreme Point Combination (EPC) for 1-dimensional sub-domain

Integers [0, 21)

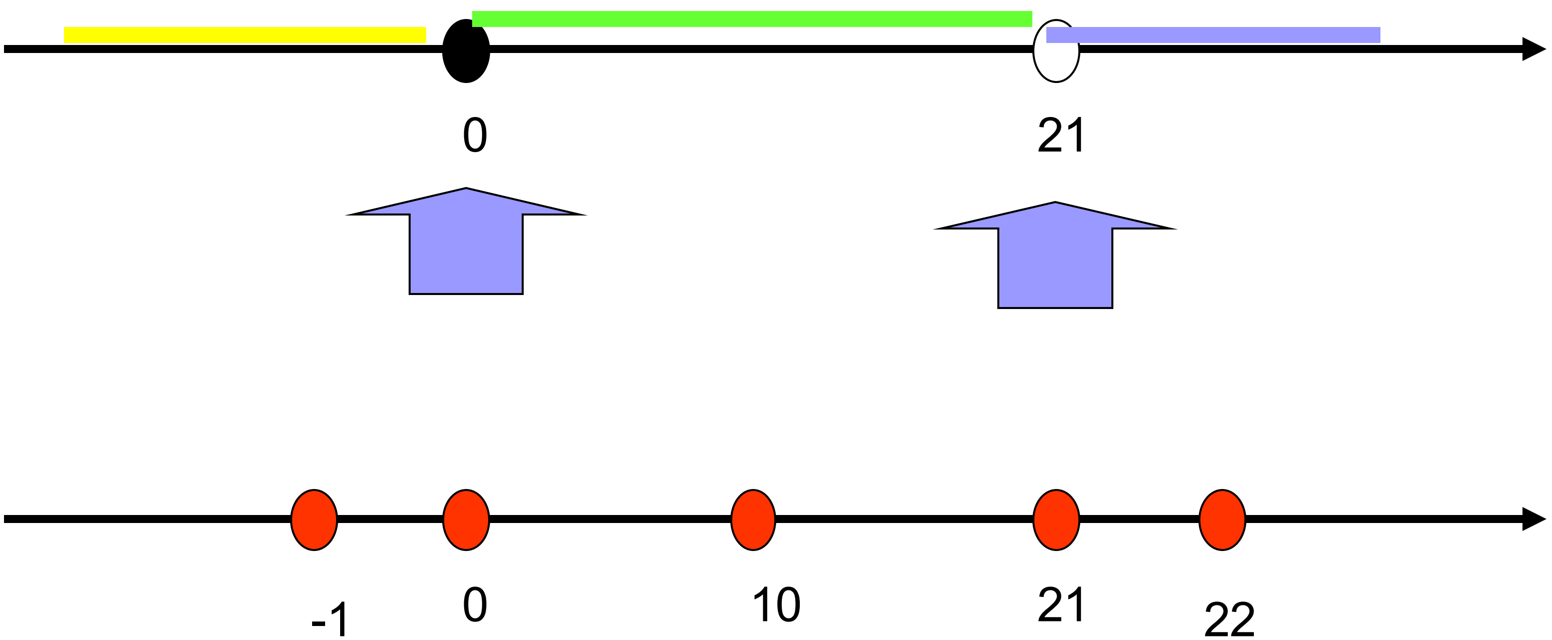
EPC :

0, 21,	// min and max
-1, 22	// below min and above max
10	// interior point

Extreme Point Combination (EPC) for 1-dimensional sub-domain

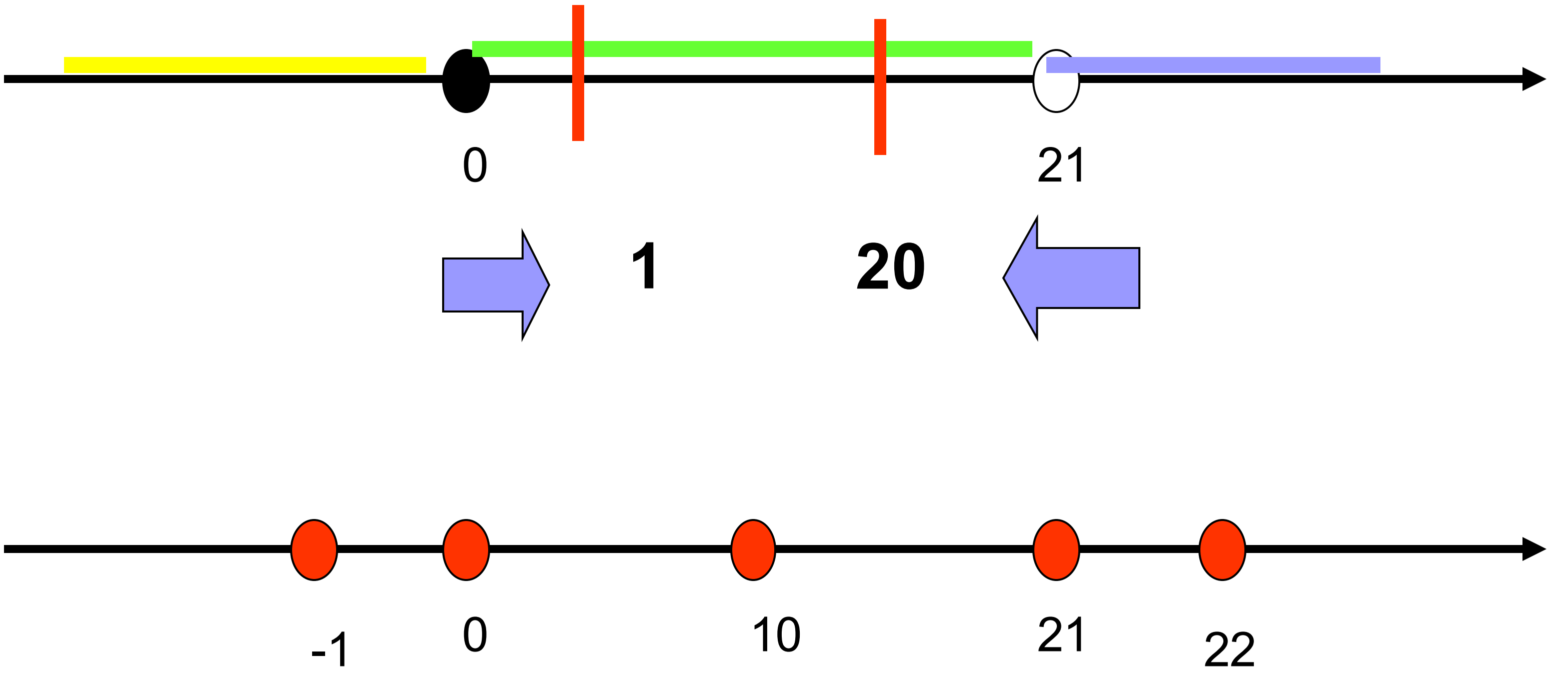


Extreme Point Combination (EPC) for 1-dimensional sub-domain closure problem (open-closed)



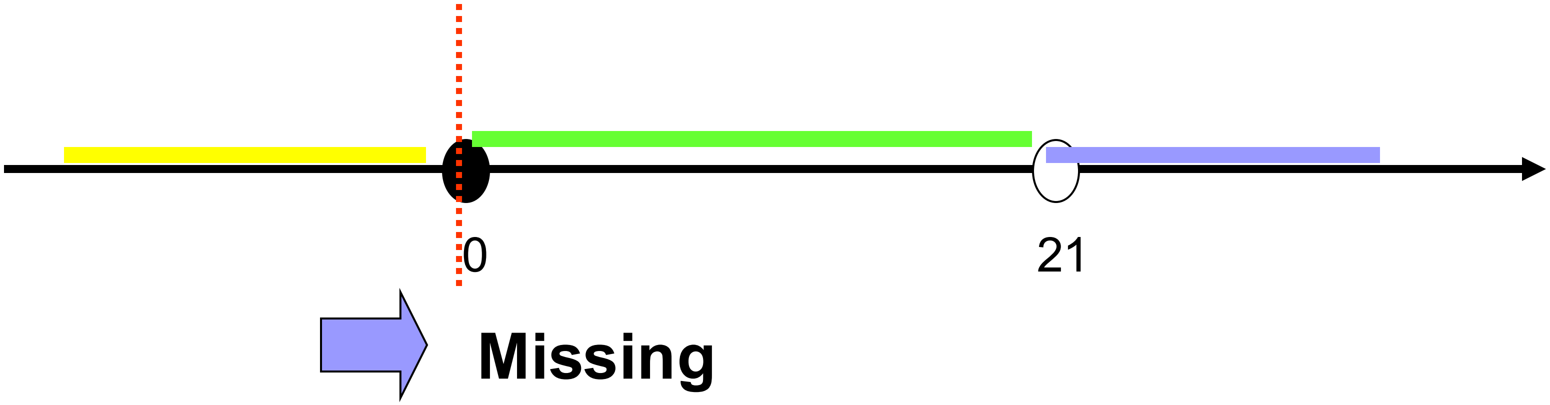
Closure problem is identified

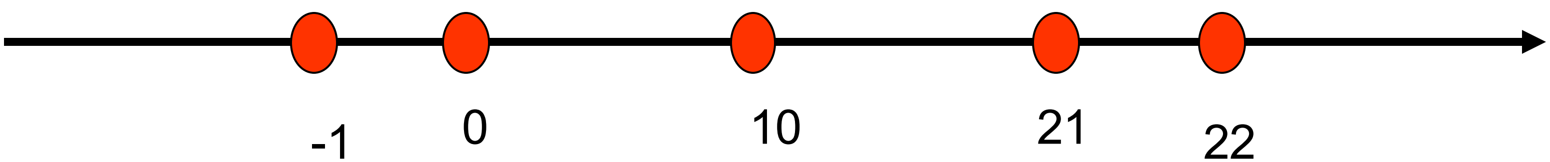
Extreme Point Combination (EPC) for 1-dimensional sub-domain boundary shift [1, 20)



Boundary shift problem *could* be identified

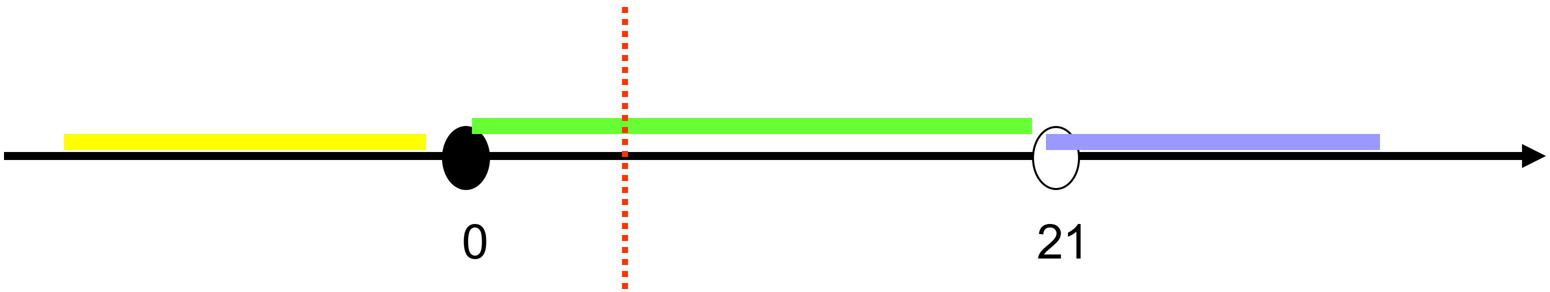
Extreme Point Combination (EPC) for 1-dimensional sub-domain missing boundary



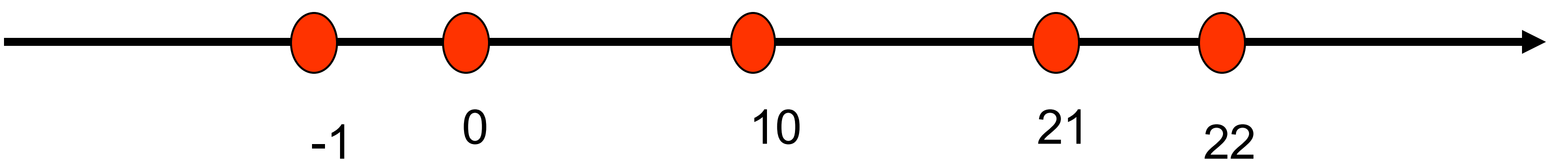


Missing boundary problem is identified

Extreme Point Combination (EPC) for 1-dimensional sub-domain extra boundary

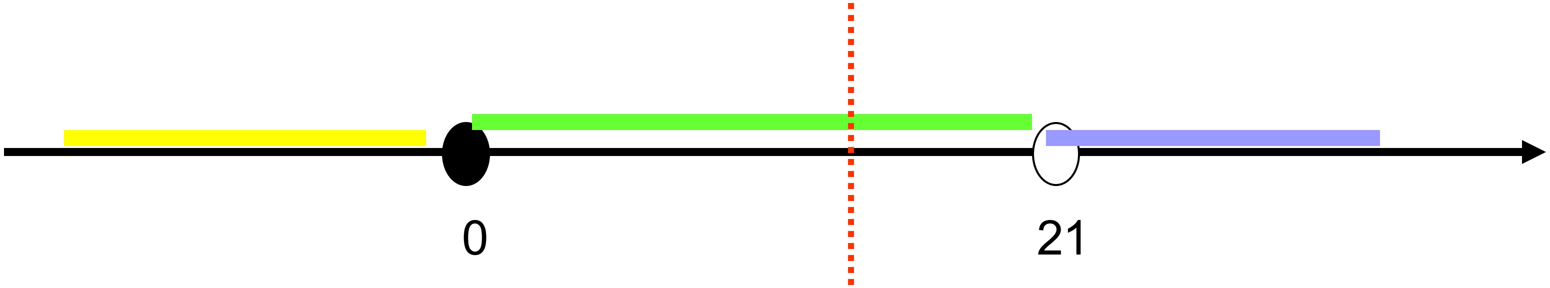


Extra at 5

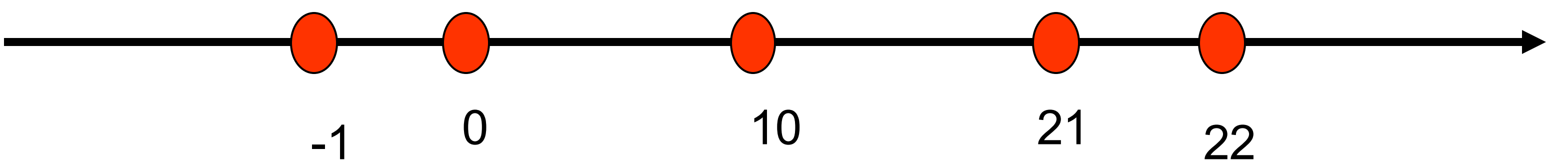


Extra boundary problem is identified

Extreme Point Combination (EPC) for 1-dimensional sub-domain extra boundary



Extra at 15



Extra boundary problem is <u>not</u> identified

Extreme Point Combination (EPC) for 1-dimensional sub-domain summary

Closure	YES
Missing boundary	YES

Boundary shift	maybe
Extra boundary	maybe

Weak n x 1 strategy

“n” points located <u>on</u> the boundary -- ON points
1 point that is OFF

Linear boundary $f(x_1, x_2, ..x_n) = K$

‘n” <u>independent</u> points fully defines this boundary – locate ON points on the boundary

OFF point: if <u>open</u> boundary then all **ON** points receive <u>exterior</u> processing
Choose **OFF** point so it receives <u>interior processing</u>; keep close to boundary
if <u>closed</u> boundary then all **ON** points receive <u>interior</u> processing
Choose **OFF** point so it receives <u>exterior processing</u>; keep close to boundary

Weak n x 1 strategy

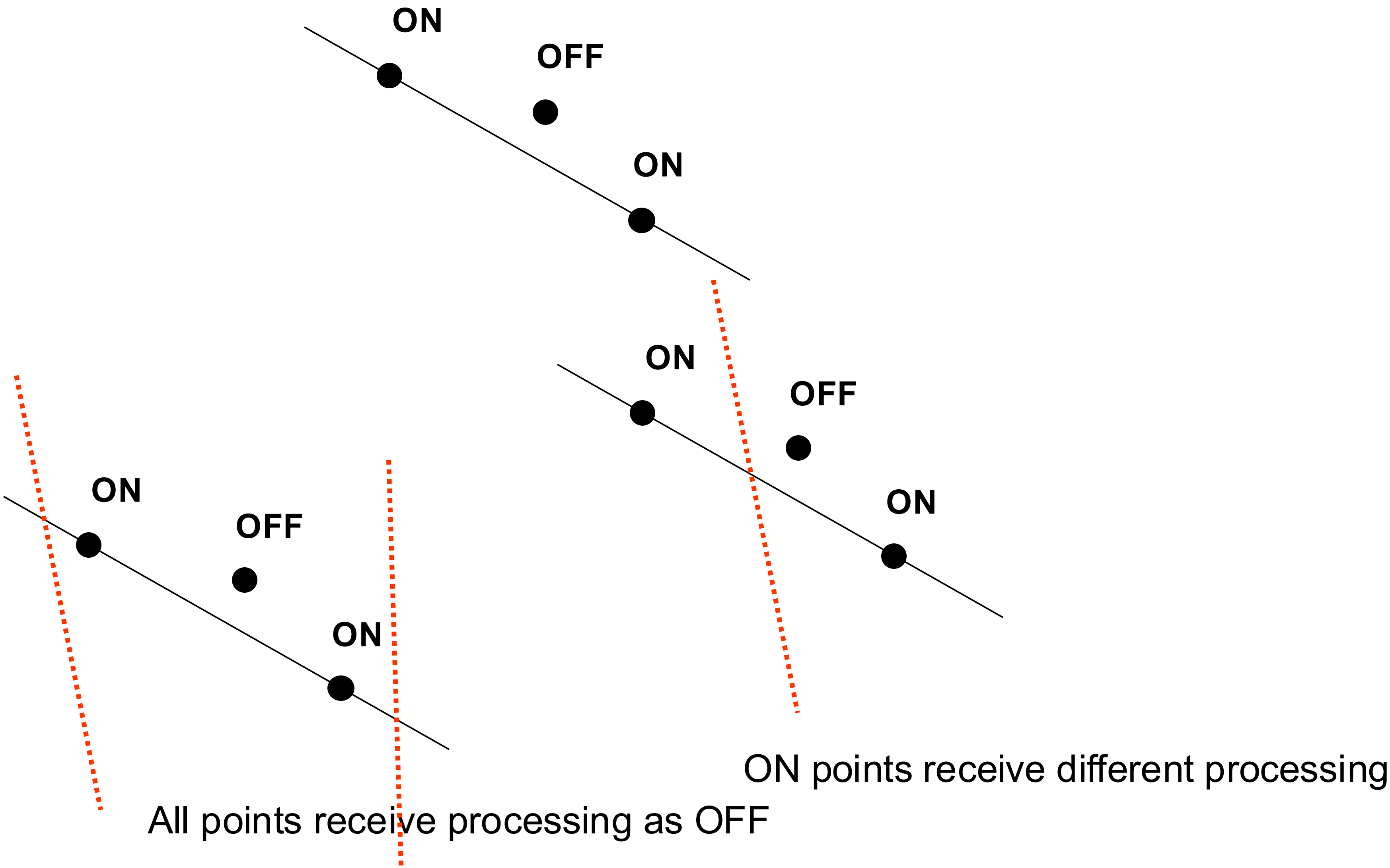
"n" points located <u>on</u> the boundary -- ON points
1 point that is OFF

ON points on the boundary, say $\mathbf{x}_1$, $\mathbf{x}_2,\ldots,\mathbf{x}_n$

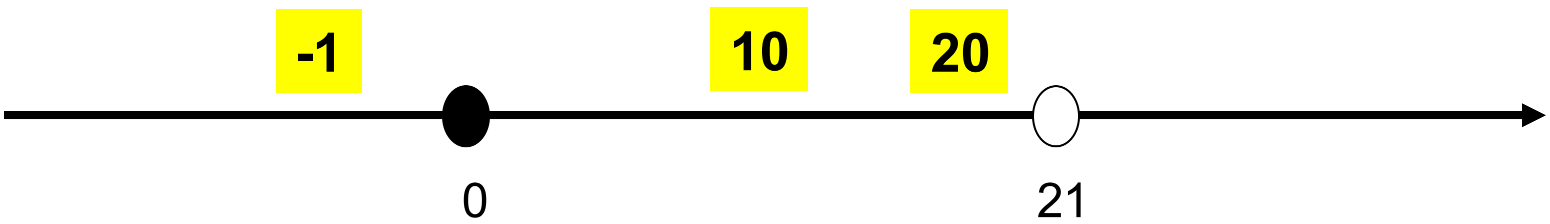
OFF point:

Midpoint between **ON** points, $(\mathbf{x}_1+\mathbf{x}_2+\ldots+\mathbf{x}_n)/n$, move it small distance (ε) off the boundary (outward or inward)

Weak n x 1 strategy- boundary tilt



Weak n x 1 strategy- 1 dim example



ON points 0 and 21

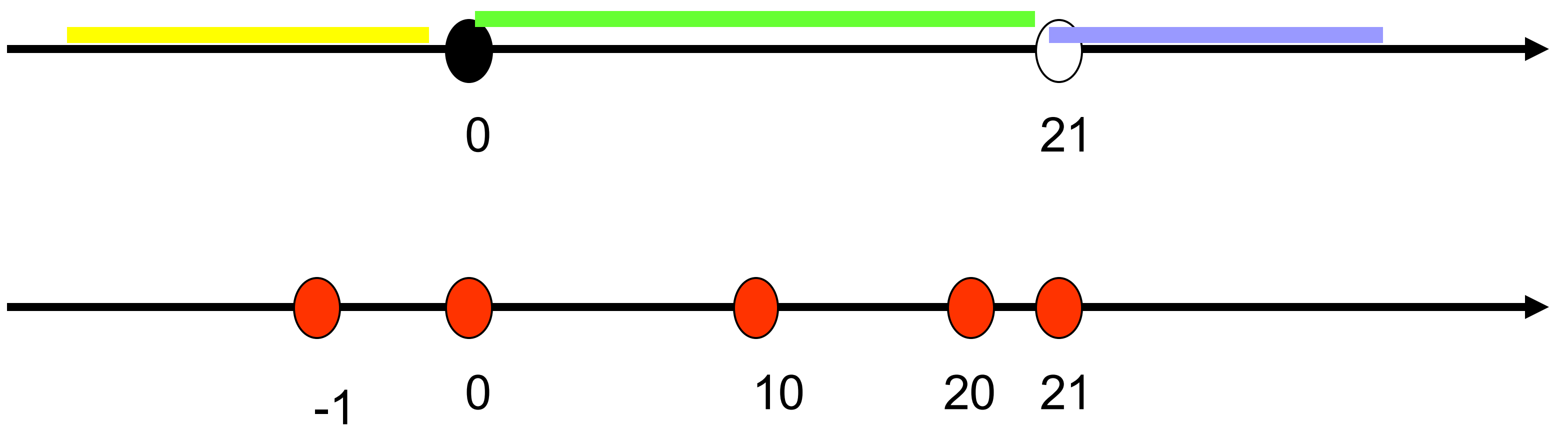
OFF points

0 closed boundary, receives interior processing, choose OFF as exterior, say -1

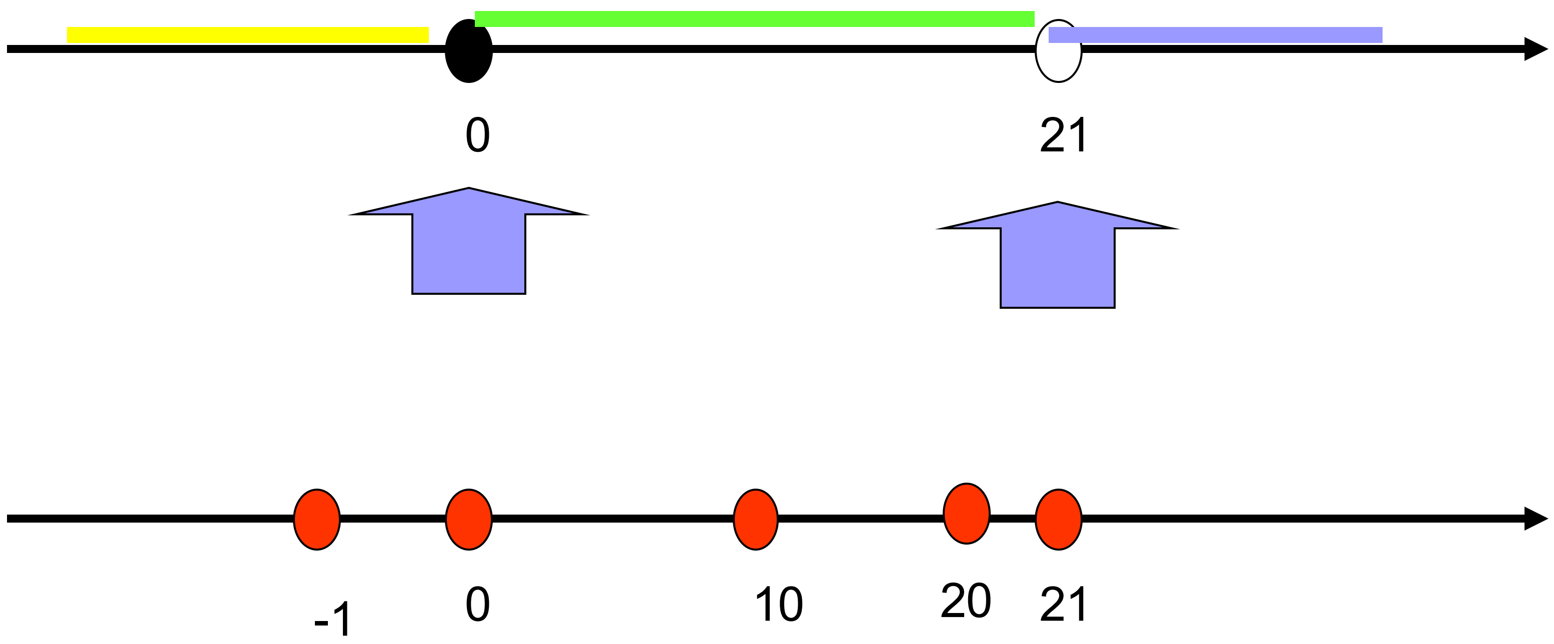
21 open boundary, receives exterior processing, choose OFF as interior, say 20

Also one point in the interior of subdomain, say 10

Weak nx1 strategy for 1-dimensional sub-domain

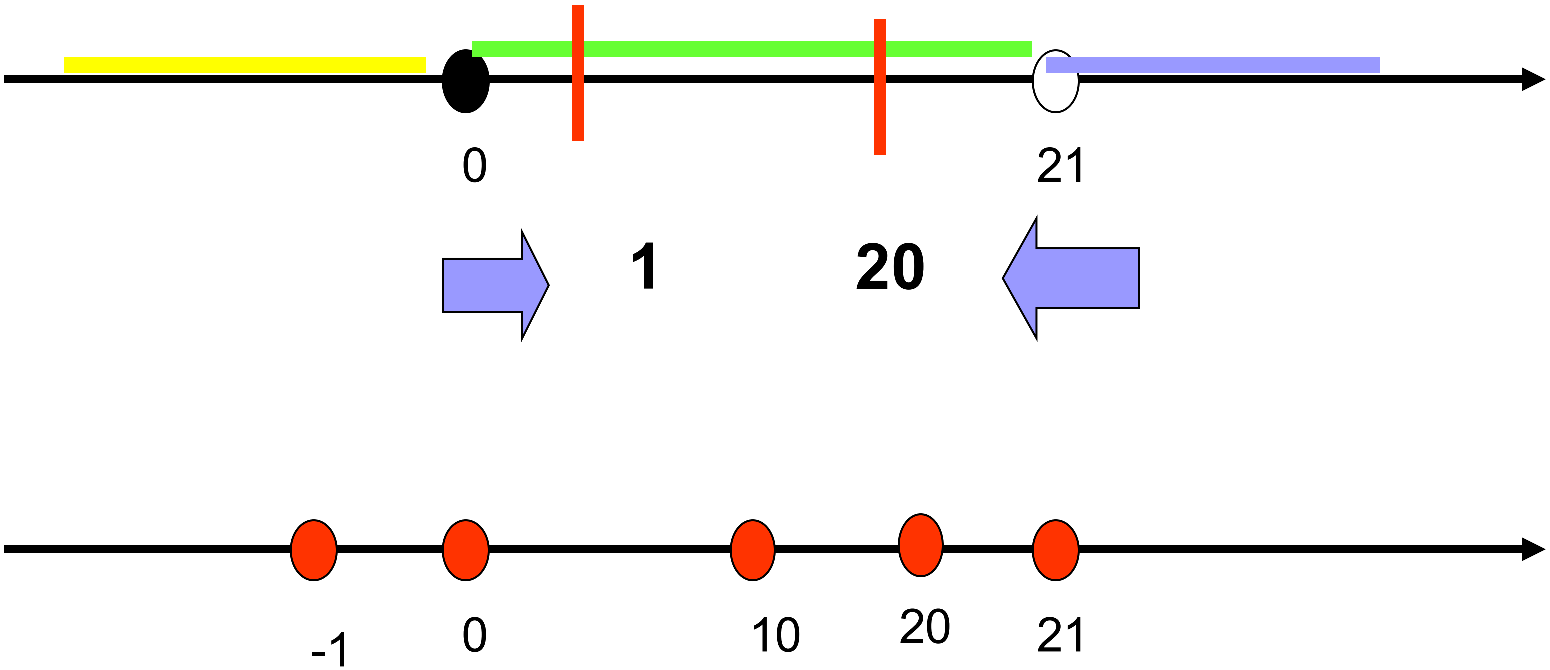


Weak nx1 strategy for 1-dimensional sub-domain closure problem (open-closed)



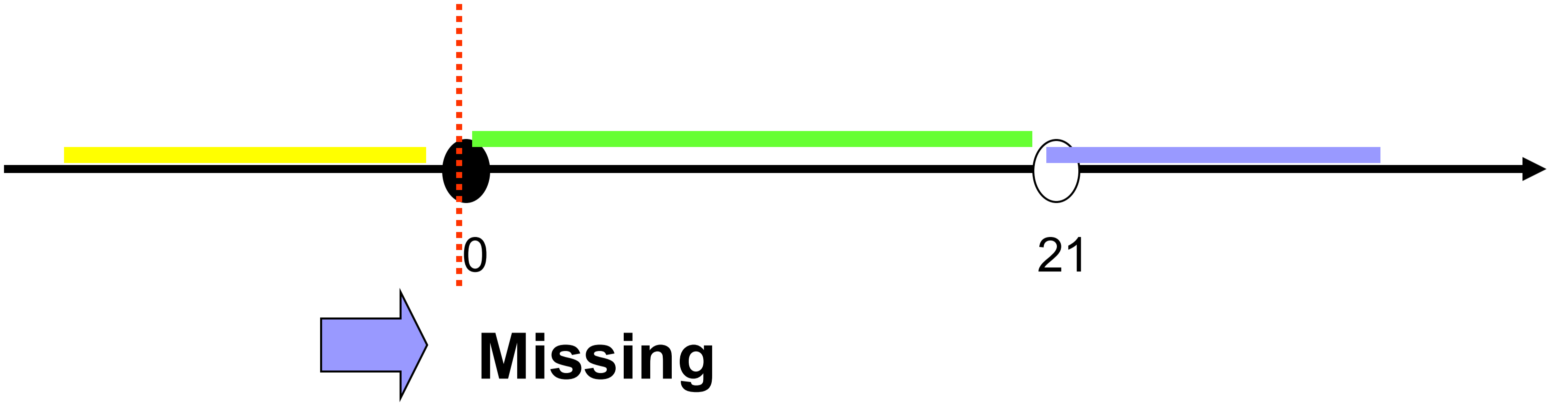
Closure problem is identified

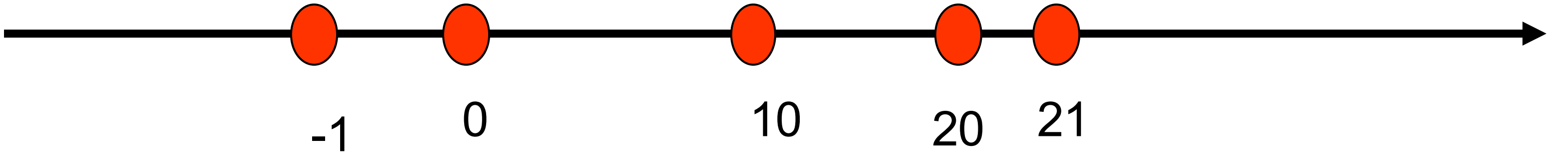
Weak nx1 strategy for 1-dimensional sub-domain boundary shift [1, 20)



Boundary shift problem *is* identified

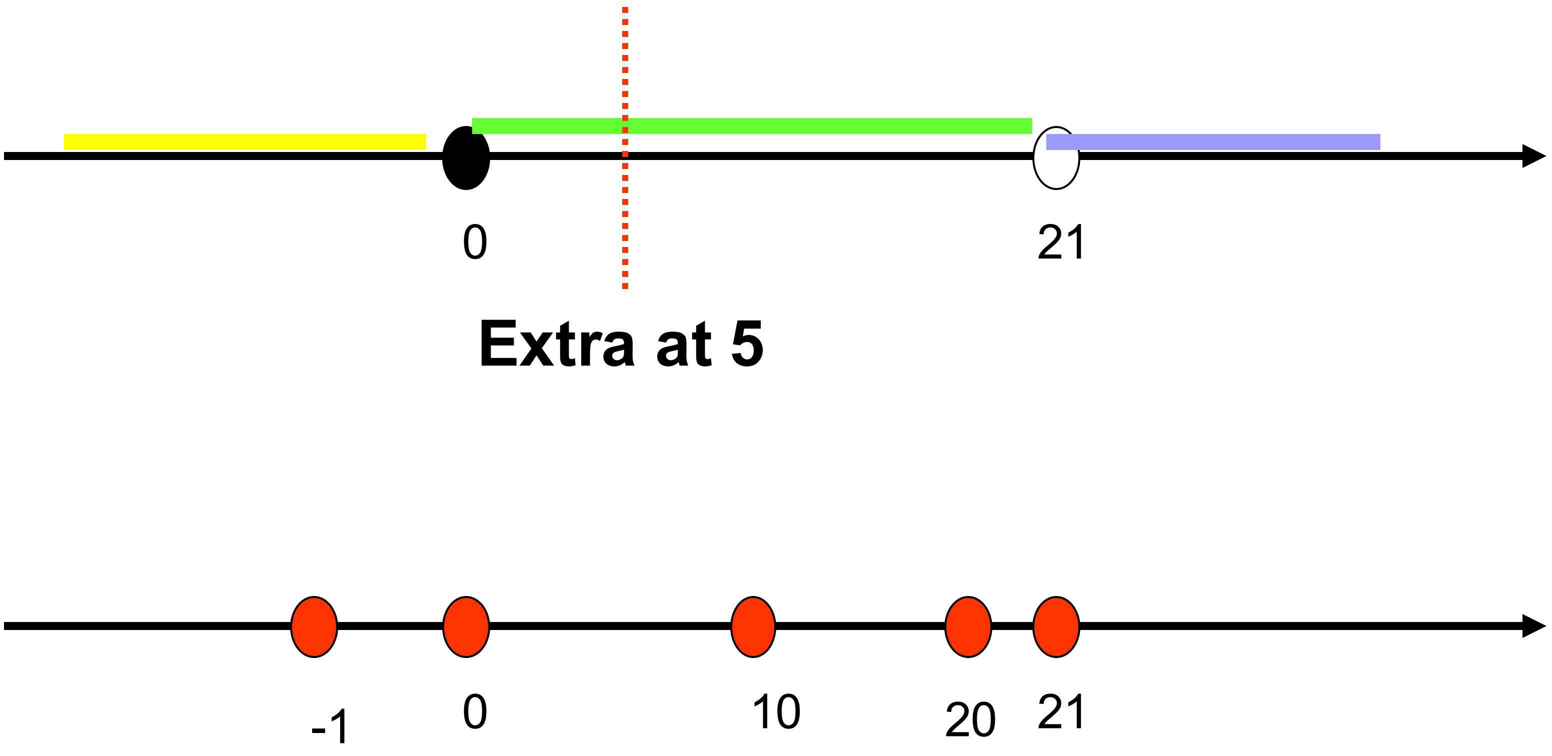
Weak nx1 strategy for 1-dimensional sub-domain missing boundary





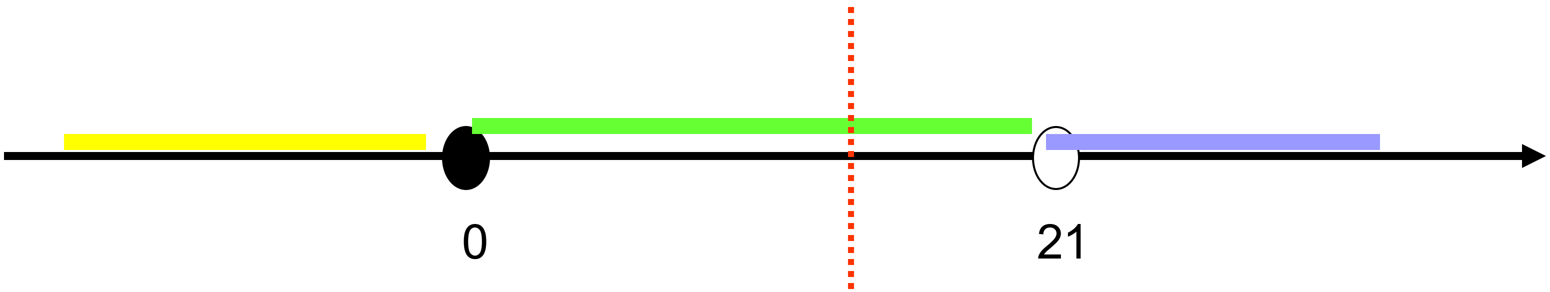
Missing boundary problem is identified

Weak nx1 strategy
for 1-dimensional sub-domain
extra boundary

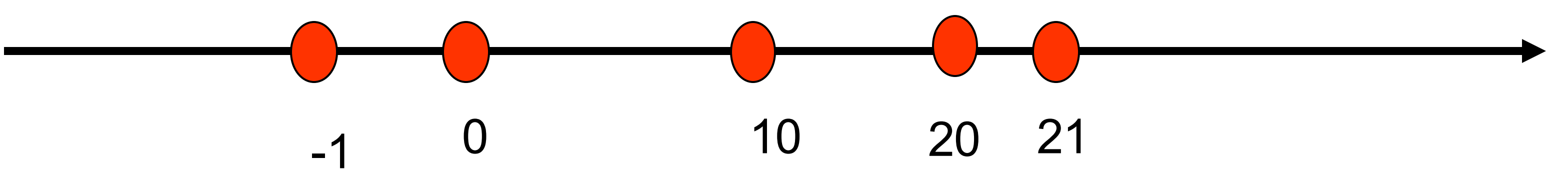


Extra boundary problem is identified

Weak nx1 strategy for 1-dimensional sub-domain extra boundary



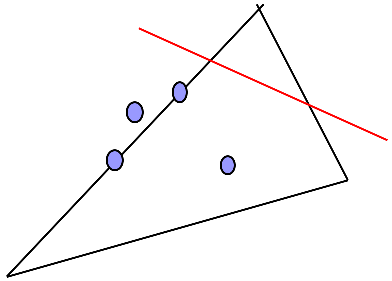
Extra at 15



Extra boundary problem is identified

Weak n x 1 strategy

Closure problem



Missing boundary

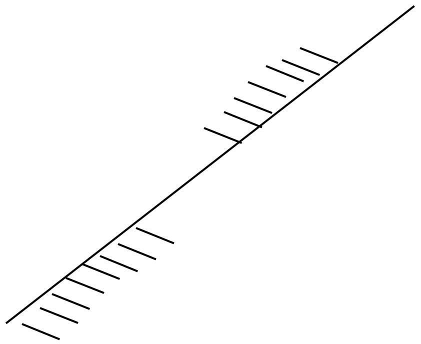
Boundary tilt/shift

Extra boundary (in general- extra boundary near a vertex point being far away from any of the ON and OFF and the selected interior point)

strong n x 1 strategy

boundaries with inconsistencies

e.g., changing closure along the boundary



Weak n x 1 strategy and EPC

<u>Consider sub-domain with "b" linear boundaries</u>

Weak nx1 strategy

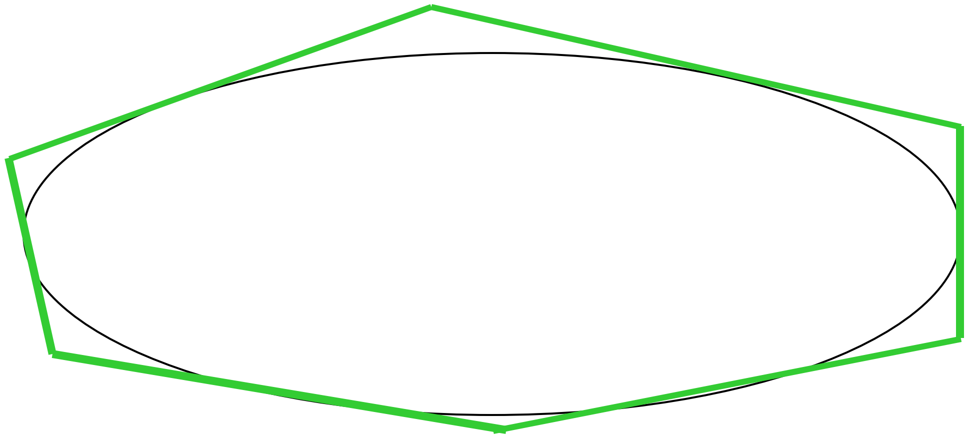
$$\text{No of points} = (n+1)b + 1$$

EPC

$$\text{No of points} = 4^n + 1$$

Nonlinear boundaries

Approximate boundary by a series of linear segments and formulate test cases for each segment (ON- OFF points)



Testing and specifications

Inconsistencies in specifications

can be detected through

input domain testing

equivalence classes and boundary testing