



РАЗДЕЛЯНЕ НА ВХОДНИЯ ДОМЕЙН И ТЕСТВАНЕ НА ГРАНИЦИТЕ

проф. д-р Десислава Петрова-Антонова

Съдържание

- ❖ Разделяне на входния домейн
- ❖ Анализ и тестване с разделяне на входния домейн
- ❖ Стратегии за гранично тестване
- ❖ Разширение на стратегиите за гранично тестване и перспективи за приложение
- ❖ Тестване по двойки

Разделяне на входния домейн



*Тестване, базирано на употреба с оперативен профил: Приоритетно тестване на поддомейни със сложни граници

❖ Анализ на входния домейн

- Генериране на тестови сценарии посредством присвояване на специфични стойности на входните променливи въз основа на анализ на входния домейн

❖ Тестване с разделяне на входния домейн

- Покриване на **малък брой** входни ситуации посредством **систематичен избор** на определени входни стойности

❖ Характеристики

- Тестване на входно/изходните зависимости
 - ✓ Осигуряване на стойности за всички входни променливи
- Изходните променливи не се специфицират експлицитно
 - ✓ Допуска се, че има начин да се направи проверка за очаквания изход от всеки вход
- Прилага се **главно** за **функционално тестване**
- Вътрешните детайли, свързани с реализацията могат да се използват за анализ на входните променливи, което е предпоставка за извършване на **структурно тестване**

Дефиниции

❖ Входно пространство

- n -размерно пространство от входни променливи x_1, x_2, \dots, x_n

❖ Входна променлива

- Единичен елемент или вход, на който може да се присвои стойност (променливи и константи)
- Сложните структури (например масиви) могат да се представят с множество входни променливи

❖ Входен вектор

- Представяне на входното пространство с вектор $X = [x_1, x_2, \dots, x_n]$

❖ Тестов сценарий

- Входен вектор с присвоени стойности на променливите x_1, x_2, \dots, x_n (съответства на точка в n -мерно входно пространство)

❖ Входен домейн

- Обхваща всички **допустимите входни комбинации**, описани в продуктовата спецификация

Дефиниции

❖ Входен поддомейн

- Множество от неравенства $f\{x_1, x_2, \dots, x_n\} < K$, където “<” може да се замени с други релационни оператори като “>”, “=”, “≠”, “≤”, “≥”

❖ Разделяне на входен домейн

- Разделяне, при което входните поддомейни са **взаимоизключващи** се и **взаимно изчерпващи** се

❖ Граница

- Когато за определяне на поддомейните се използват неравенства, то дадена граница се дефинира като $f\{x_1, x_2, \dots, x_n\} = K$

❖ Границата е линейна, ако се дефинира по следния начин:

- $a_1x_1 + a_2x_2 + \dots + a_nx_n = K$

❖ Линеен поддомейн

- Поддомейн само с линейни граници

❖ Гранична точка

- Точка върху границата

Дефиниции

❖ Затворена граница

- Граничните точки принадлежат на поддомейна

❖ Отворена граница

- Нито една от граничните точки не принадлежи на поддомейна

❖ Отворен, затворен и смесен поддомейн

- Поддомейн, на който всички граници са отворени е отворен
- Поддомейн, на който всички граници са затворени е затворен
- Всички останали поддомейни са смесени

❖ Вътрешна точка

- Точка, която принадлежи на даден поддомейн, но не принадлежи на границата му

❖ Външна точка


- Точка, която не принадлежи на даден поддомейн, нито на границата му

❖ Върхова точка

- Точка, в която две или повече граници се пресичат

Основни стъпки при тестването на домейна


Идентифициране на входната променлива, входния вектор, входното пространство и дефиниране на входния домейн




Разделяне на входния домейн на поддомейни



Анализ на поддомейните с цел определяне на границите им по всички измерения



Избор на тестови точки (сценарии), покриващи на поддомейните



Тестване с избраните тестови точки, проверка на резултатите, решаване на проблеми и анализ на резултатите

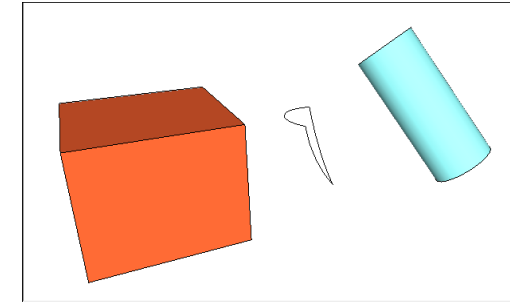
Базова идея на домейн тестването

- ❖ Стъпка 1 и стъпка 2 се определят въз основа на спецификации (black box testing) или детайлите по реализацията (white box testing)
- ❖ Вариациите при тестване на домейна се определят от стъпка 3
- ❖ Начинът на избор на тестови точки при стъпка 4 определя тестовата стратегия
 - Пример: Осигуряване на пълно покритие посредством избор на вътрешна точка от всеки поддомейн
- ❖ Стъпка 5 е стандартна за всички техники за тестване

Проблеми при разделяне на входния домейн

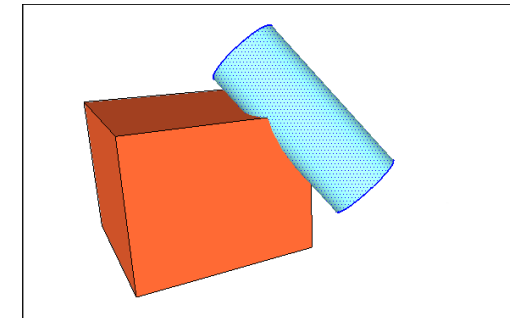
❖ Неопределеност за даден вход

- Тестваната програма не обработва някои входни стойности или тестови точки
- Причина: **Липса на изчислителни процедури за определени поддомейни** от общия входен домейн
 - ✓ „Дупки“ във входния домейн



❖ Противоречивост за даден вход

- Повреда в системата или производство на различни изходи при един и същ вход
- Причина: **Дефиниране на изчислителни процедури за поддомейни, които се припокриват**



Проблеми с границите на домейните

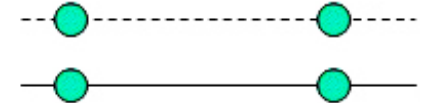
❖ Проблем със затвореността на границите

- Отворена граница се специфицира или реализира като затворена



❖ Изместване на граница

- Разминаване между очакваната и действително специфицирана или реализирана граница

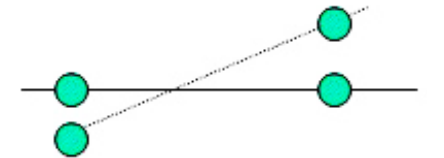


$$f\{x_1, x_2, \dots, x_n\} = K, \text{ малка промяна в } K$$

❖ Изкривяване на граница

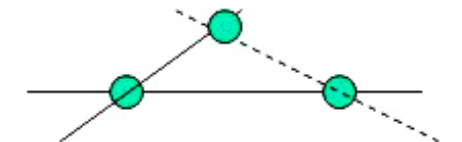
- Промяна на някои от параметрите на границата

$$f\{x_1, x_2, \dots, x_n\} = K$$



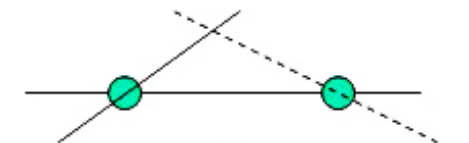
❖ Липсваща граница

- Сливане на два различни поддомейна в общ поддомейн



❖ Излишна граница

- Некоректно разделяне на поддомейн



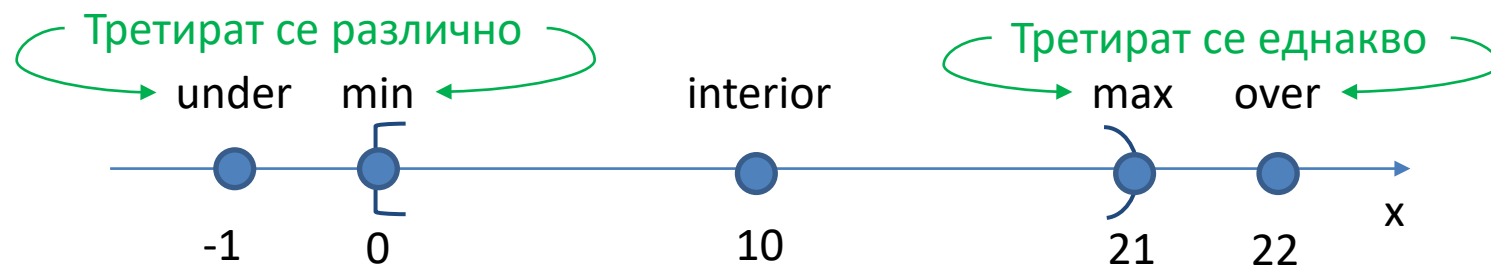
Групи проблеми: обобщение



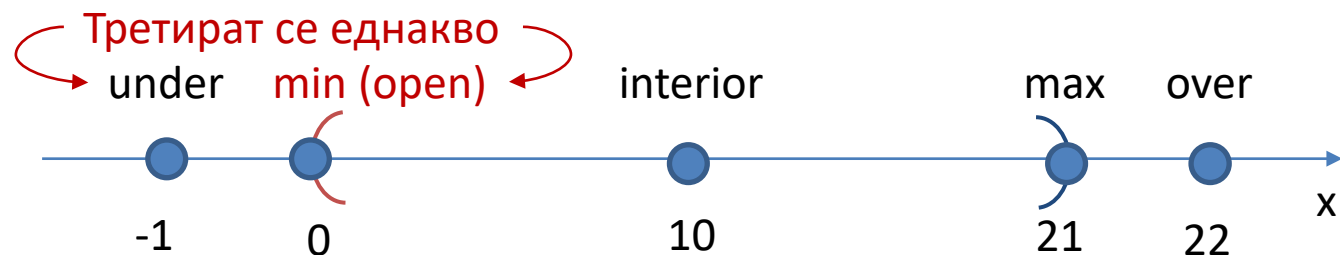
Тестване с комбинация на екстремни точки: идея

- ❖ Тестването на екстремните точки споделя идеята при тестване на капацитет, стрес тестване и тестване на устойчивост
 - Наблюдаване на поведението на системата при необичайни гранични ситуации
- ❖ Стратегия за тестването на екстремните точки
 - Асоцииране на множество от тестови точки с всеки поддомейн
 - Анализирание на всеки поддомейн с цел определяне на граничните му стойности по всички измерения
 - ✓ Намиране за всяка променлива x_i минималната ѝ min_i и максималната ѝ max_i стойност
 - ✓ Определяне на стойност $under_i$ малко по-малка от min_i и стойност $over_i$ малко по-голяма от max_i
 - Използване на стойностите min_i , max_i , $under_i$, $over_i$ и $interior$ за създаване на всички възможни комбинации за всяка променлива x_i на входа
 - ✓ Всяка комбинация представлява тестов сценарий в n-размерното пространство
 - ✓ Броят на тестовите сценарии е $4^n + 1$

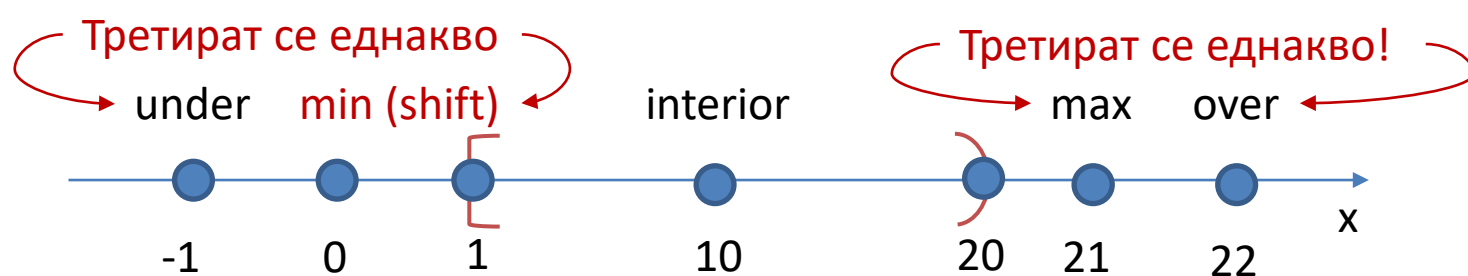
Приложение при тестване на едномерен домейн



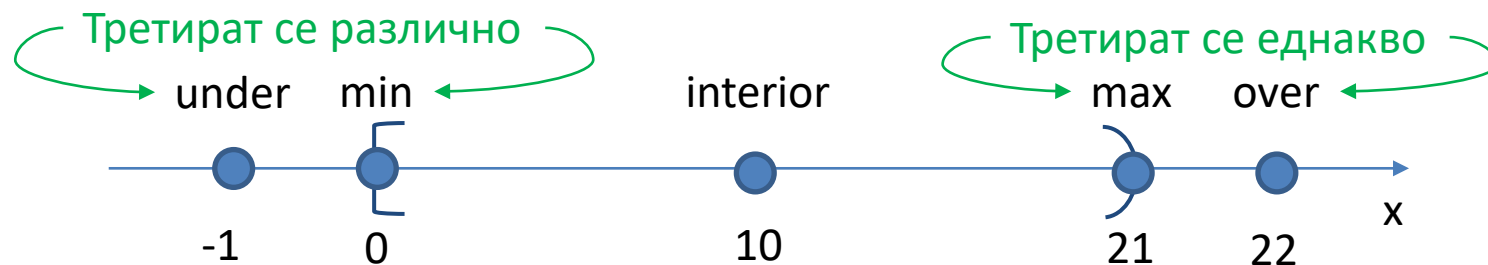
Затвореност на граница



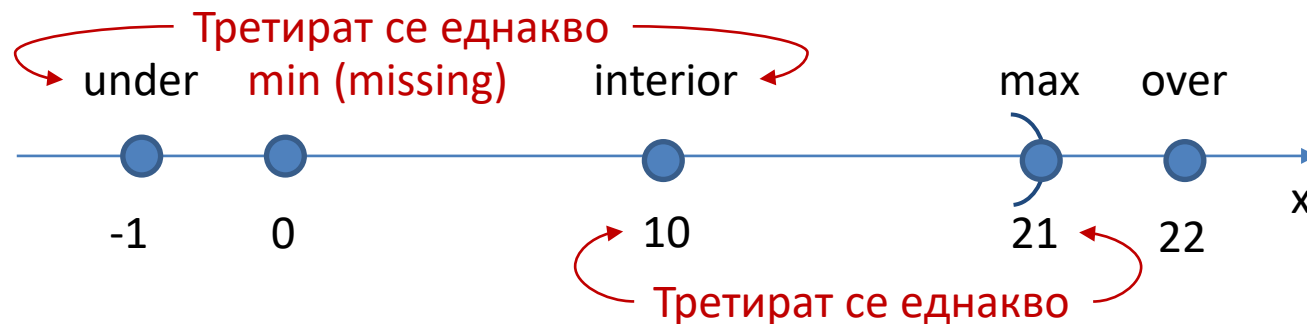
Изместване на граница



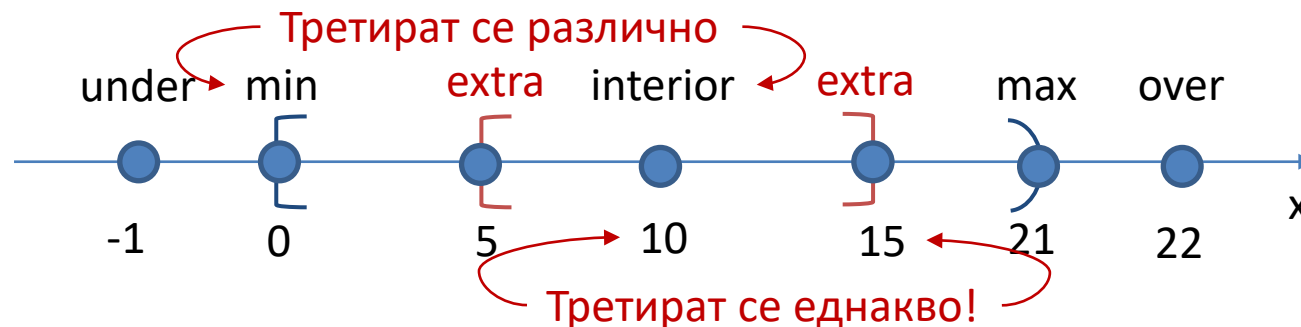
Приложение при тестване на едномерен домейн



Липсваща на граница



Излишна на граница

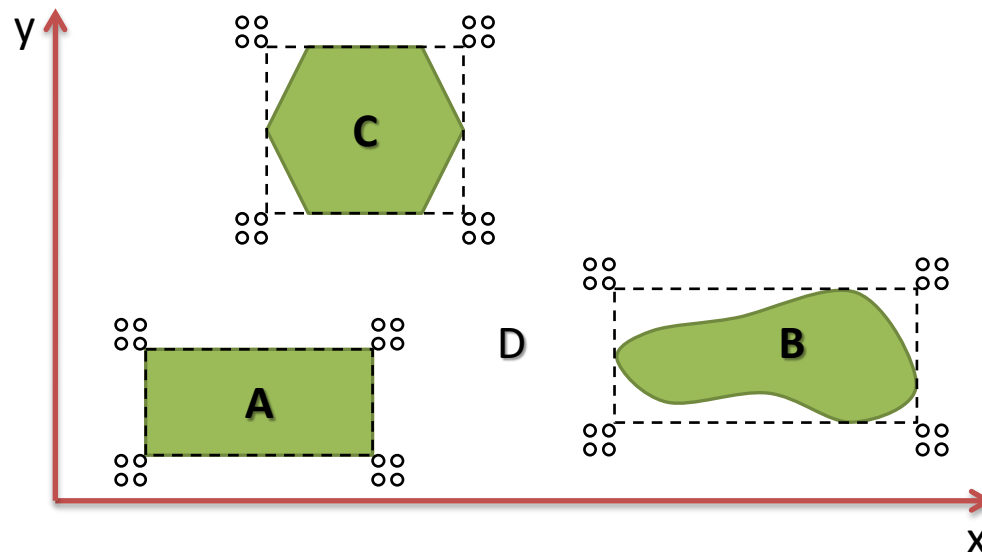


Приложение при тестване на едномерен домейн

- ❖ Използване на стойностите *min*, *max*, *under*, *over* и една вътрешна стойност *interior* за тестване
 - Пример: домейн $0 \leq x < 21$, тестови точки -1, 0, 10, 21, 22
- ❖ Проблем със затвореността на граница
 - Реализиране на затворената граница $0 \leq x$ на поддомейна $[0, 21)$ като отворена, при което се открива проблем
- ❖ Изместена граница
 - Реализация на поддомейна $[0, 21)$ като $[1, 20)$
 - ✓ Точката $x = 0$ ще се третира като външна точка и ще се открие проблем
 - ✓ При дясната граница няма да се открие проблем (*max* и *over* се третират еднакво)
- ❖ Липсваща граница
 - При липса на затворената граница $0 \leq x$ точката -1 ще се третира като вътрешна за поддомейна, при което се открива проблем
- ❖ Излишна граница
 - Ако се добави излишна граница $x = 5$, то тестовите точки $x = 0$ и $x = 10$ ще се обработват по различен начин
 - Ако се добави излишна граница $x = 15$, то тестовите точки $x = 10$ и $x = 15$ ще се обработват еднакво и няма да се открие проблем

Приложение при тестване на многомерен домейн

- ❖ Проблем с определяне на екстремните точки за поддомейна D
- ❖ Ефективност на тестването за поддомейните A, B и C
 - Екстремните точки за поддомейна A съвпадат с върховете на региона, което води до усложняване на логиката при комбинирането на екстремните точки
 - ✓ Ефективността при откриване на проблеми с границите е същата както при едномерен домейн
 - Всички екстремните точки за поддомейните B и C са външни и неизползваеми за откриване на проблеми



Стратегия “weak N x 1”



Базова идея за откриване на гранични измествания

- ❖ При n -размерно пространство са необходими n линейно независими точки, за да се дефинира граница от вида
 - $f\{x_1, x_2, \dots, x_n\} = K$
- ❖ Точките, които са на границата се наричат “ON” точки, а точките, които не са на границата се наричат “OFF” точки
- ❖ При **отворена** граница “ON” точките се обработват като **външни**
 - Като “OFF” точка се избира вътрешна точка, която е много близо до границата
 - Вътрешната точка ще получи външна обработка при свиване на границата
 - “ON” точките ще получат вътрешна обработка при разширяване на границата
- ❖ При **затворена** граница “ON” точките се обработват като **вътрешни**
 - Като “OFF” точка се избира външна точка, която е много близо до границата
 - Откриването на проблеми, свързани с изместването на границите е огледално

Стратегия “weak N x 1”: формални дефиниции

- ❖ Избор на една “OFF” точка за всяка граница и определяне на разстоянието ѝ ε от границата
 - $\varepsilon = 1$ за цели числа, $\varepsilon = 1/2^n$ за числа с точност n след десетичната запетая
- ❖ Откриване на измествания на границата със стратегията “weak N x 1”
 - За всяка границата на поддомейн в n -размерно входно пространство като “ON” точки се избират n линейно независими гранични точки
 - “OFF” точката винаги получава различна обработка от “ON” точките
 - ✓ Ако границата е затворена, то “OFF” точката ще бъде извън поддомейна
 - ✓ Ако границата е отворена, то “OFF” точката ще бъде в поддомейна
 - ✓ Във всички случаи “OFF” точката е на дистанция ε от границата
 - Произволна вътрешна точка се взима като представител на еквивалентния клас, представящ разглеждания поддомейн
 - Броят на тестовите точки е $(n + 1) \times b + 1$ за всеки домейн с b на брой граници

Стратегия “weak N x 1”: проблеми

❖ Проблем със затвореността на границите

- “ON” и “OFF” точките получават еднаква обработка, при което проблемите се откриват лесно

❖ Изкривяване на граница

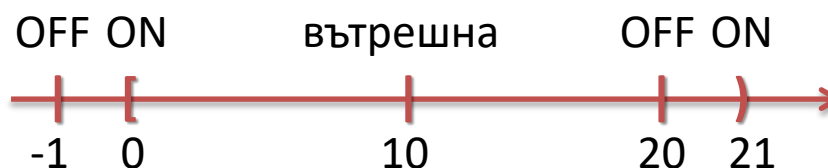
- Някои или всички “ON” точки престават да бъдат на границата и откриването на проблемите е аналогично на изместване на границата

❖ Липсваща граница

- “ON” и “OFF” точките получават еднаква обработка

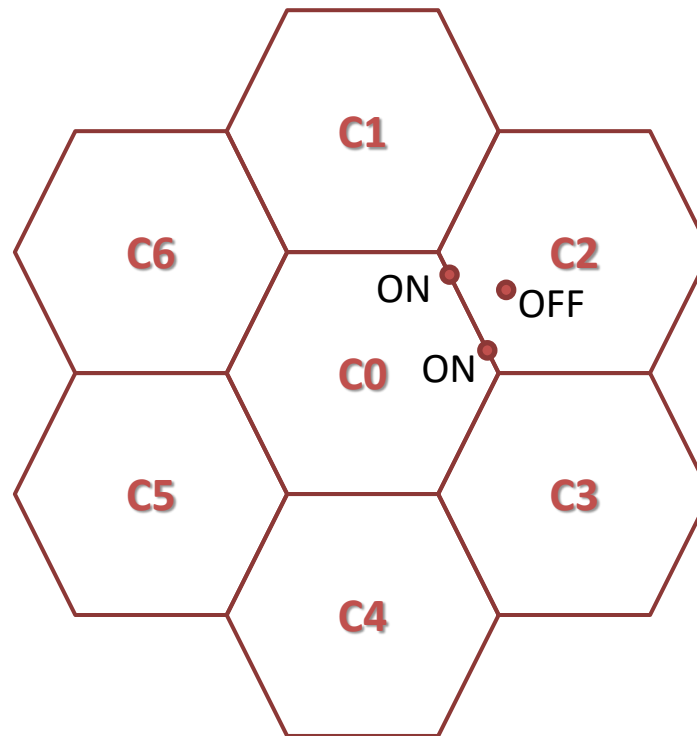
❖ Излишна граница

- За всяка граница съществуват n на брой “ON” точки или 1 “OFF” точка, и произволна вътрешна точка, получаващи вътрешна обработка (“IN” точки)
 - ✓ Някои “IN” точките получават различна обработка при наличие на излишна граница
- Излишната граница няма да бъде открита, ако точката, която я определя е много отдалечена от “ON”, “OFF” или вътрешната точки



“ON” и “OFF” точки при двумерни поддомейни

- ❖ Избор на „OFF“ точка
 - „Централна“ спрямо „ON“ точките
- ❖ Избира се средна точка между две “ON” точки
- ❖ “OFF” точката се поставя на дистанция ε от външната страна за затворена или вътрешната страна за отворена граница



Комбинация от екстремни точки vs. “Weak N x 1”



❖ Тестови точки при стратегията “weak N x 1”

- -1, 0, 10, 20 и 21



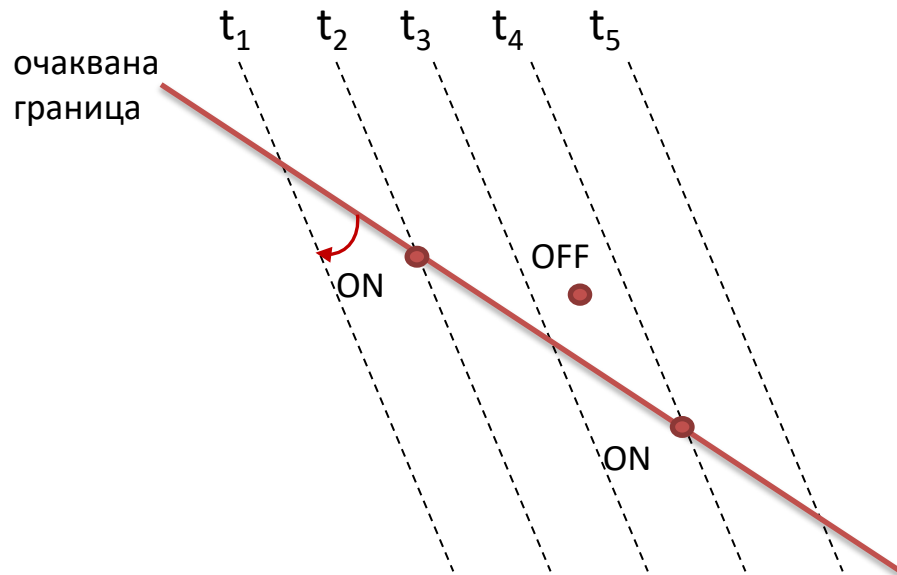
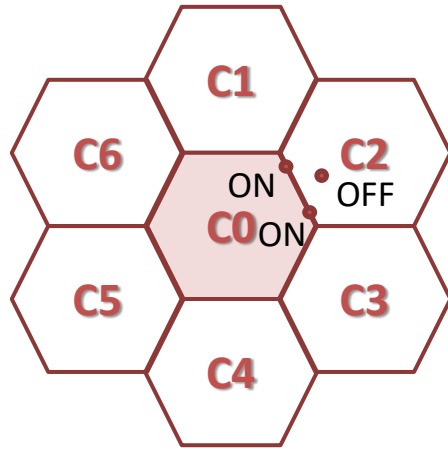
❖ Тестови точки при комбинация от екстремни точки

- -1, 0, 10, 21, 22

❖ Изместване на границата в [1, 20)

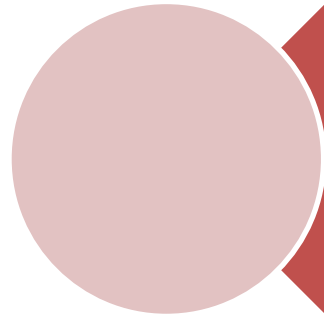
- Открива се само при стратегията “weak N x 1” ($x = 20$ получава външна обработка)

Стратегия “weak N x 1”: изкривяване на граница



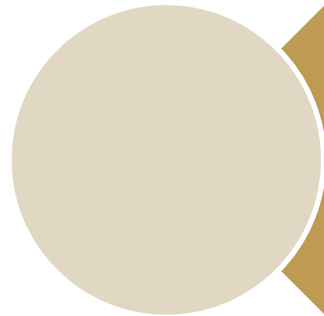
- ❖ Двумерен домейн за представяне на мобилна мрежа
- ❖ Откриване на изкривена граница в няколко точки при двумерен домейн
 - Всяко изкривяване извън сегмента между двете “ON” точки води до обработването им по начин, аналогичен на “OFF” точката (t_1 и t_5)
 - Всяко изкривяване в сегмента между двете “ON” точки води до обработването им по различни начини (t_3)
 - Всяко изкривяване в една от “ON” точките води до еднакво обработване на “OFF” точката и другата “ON” точка (t_2 и t_4)

Стратегия “weak 1 x 1”



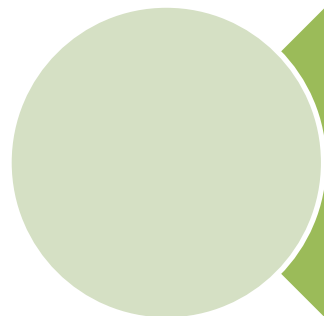
Екстремни точки

- $(4^n + 1)$



Weak N x 1

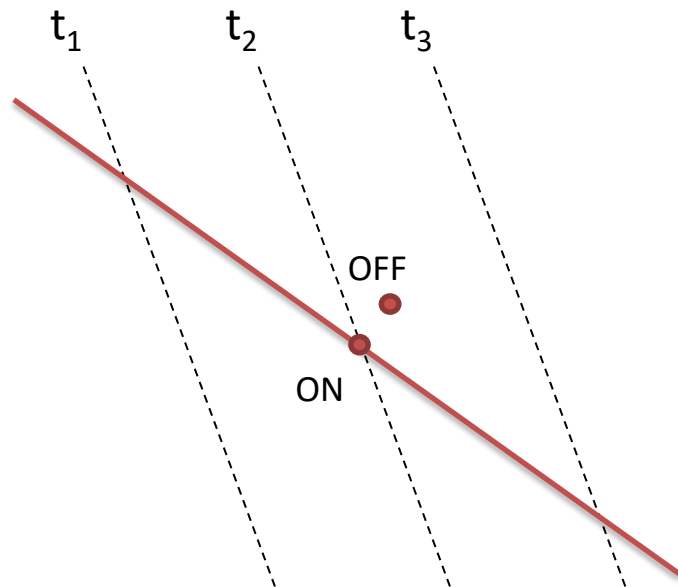
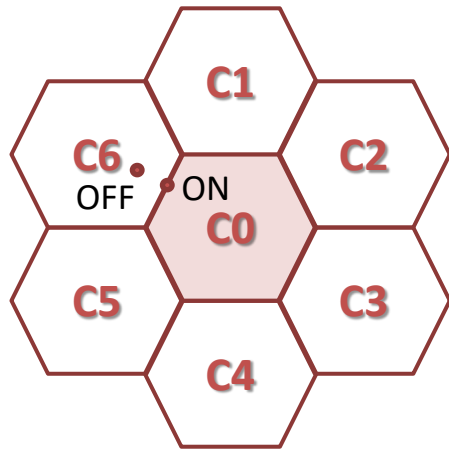
- $(n + 1) \times b + 1$



Weak 1 x 1

- $2b + 1$

Стратегия “weak 1 x 1”



❖ Базова идея

- Избира се само една “ON” точка за всяка граница
- Броят на тестовите сценарии е $2b + 1$ при b на брой граници на поддомейна
- “OFF” точката се избира на дистанция ε от “ON” точката, перпендикулярно на границата

❖ Откриваемост на проблеми при стратегията “weak 1 x 1”

- Успешно откриване на проблеми, свързани със затвореността на границата, изместването и липсваща граница
- Излишна граница (разделя „ON“ точките)
 - ✓ По-големия брой “ON” точки повишава вероятността за откриване на проблеми (weak N x 1)
- Изкривяване на граница
 - ✓ Стратегията “weak N x 1” винаги открива проблем
 - ✓ Стратегията “weak 1 x 1” може да пропусне проблем

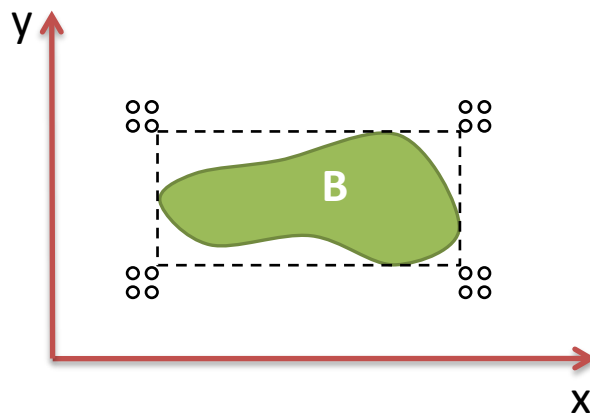
Строги и апроксимиращи стратегии

❖ Строги стратегии

- Подходящи са при неконсистентност на границите
 - ✓ Промяна на затвореността по границата, в единични точки или наличие на „дупки“
- Всеки сегмент се разглежда отделно, при което се определят множества от “ON” и “OFF” точки за всеки сегмент

❖ Апроксимиращи стратегии

- Подходящи са за нелинейни граници, за които използването на n на брой точки не е достатъчно
- Възможно е границата да се апроксимира със серия от линейни сегменти, за всеки от които се дефинира множество от “ON” и “OFF” точки



Приложение на стратегиите за гранично тестване

- ❖ Ограничения или граници, свързани със сложни структури от данни и техните реализации
 - Реализация на масиви
- ❖ Тестване на капацитет и стрес тестване
 - При тестването на капацитета обикновено се тества горната граница и се пренебрегва долната
- ❖ Изпълнение на цикли
- ❖ Критични системи, за които безопасността е от първостепенно значение
 - Пример система за отопление, в която температурата е изходен параметър, а управляващите променливи на устройството са входни променливи
 - Техниките за гранично тестване могат да се приложат върху изходния домейн на системата
 - ✓ Трудности при инициализирането на тестовите сценарии
 - Разделяне на системните повреди на такива, които предизвикват инциденти и такива, които не предизвикват
 - ✓ Извършване на анализи за определяне на условията и входовете за възникване на инциденти



Опашки: основни понятия

❖ Дефиниция

- Тип даннова структура, при който обработката или реда на премахване на елементи следва реда на пристигане или добавяне на елементи

❖ Последователност на обработка

- Приоритизиране на ред на постъпване (FIFO, FCFS)
- Създаване на приоритизирани класове (FIFO обработка в класа)
- Случайно обработване без приоритети

❖ Капацитет

- Повечето опашки имат горна граница за броя на елементите
- Долната граница за броя на елементите също е ограничена
 - ✓ Не може да има отрицателен брой елементи

❖ Особености при обработка на елементите

- Изместване на текущо обработвания елемент от нов елемент с по-висок приоритет
- Групова обработка на елементи (Batching)
- Синхронизация
 - ✓ Определени елементи изчакват други елементи преди да бъдат обработени заедно

Приложение на стратегиите за гранично тестване при опашки

- ❖ Тестване на долната граница при 0, 1 или 2 елемента в опашката
 - При празна опашка, нов елемент може да бъде обработен незабавно (server idle) или да се добави в опашката (server busy)
 - При 1 или 2 елемента в опашката се тестват функциите за добавяне в опашката
- ❖ Тестване на горна граница B при B и $B \pm 1$ елемента в опашката
 - При текущ капацитет $B-1$ тестовете проверяват добавянето на нов елемент и запълването на капацитета
 - При текущ капацитет B тестовете проверяват добавянето на нов елемент в запълнена опашка
 - Текущ капацитет $B+1$ съответства на “OFF” точка за затворен поддомейн, при което могат да възникнат проблеми
- ❖ Дефиниране на тестов сценарий за тестване на поведението на опашката при нормални обстоятелства

❖ Последователност на тестване

- Тестване на долна граница преди тестването на нормалното поведение на опашката и горната граница
- Съчетаване на тестовите за вмъкването и премахването на елемент в комбинация с тестовите за границите

❖ Приложение на тестването с опашки

- Оценка на производителността и анализ
 - ✓ Обслужване на опашки в единични сървъри или мрежи
- Използват се генератори на трафик, симулиращи оперативния трафик върху мрежата или сървъра
 - ✓ Оценка на надеждността подобна на тази при използване на оперативен профил

ТЕСТВАНЕ ПО ДВОЙКИ

Същност на тестването по двойки

❖ Тестване на всички комбинации

- Тества се всяка възможна комбинация от стойности за всички входни променливи
- Недостатък
 - ✓ Броят на тестовите сценарии е прекалено голям: При n на брой входни променливи, всяка от които има k на брой стойности, които трябва да се тестват, броят на тестовите сценарии е k^n .

❖ Тестване по двойки

- Всяка възможна комбинация от стойности за всяка двойка входни променливи се покрива най-малко от един тест.

❖ Големината на тестовия пакет е пропорционална на броя на входните променливи, които участват при създаването на тестови комбинация.

❖ Откриване на проблеми

- Медицински устройства и разпределени бази от данни – 90% (по двойки)/100% (по четворки)
- Уеб и сървърни приложения – 70% (по двойки)/100% (по шесторки)

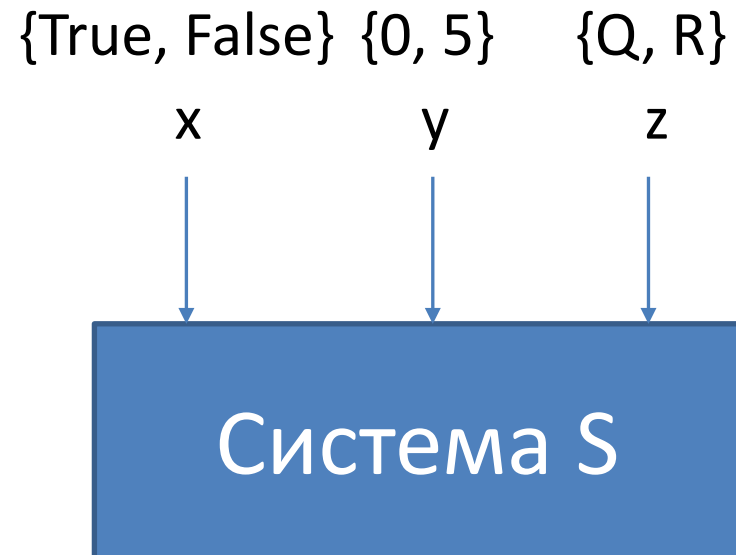
Пример

❖ Тестване на всички комбинации

- $2 \times 2 \times 2 = 8$ тестови сценария

❖ Тестване по двойки

- 4 тестови сценария



Тестов сценарий	X	Y	Z
TC ₁	True	0	Q
TC ₂	True	5	R
TC ₃	False	0	Q
TC ₄	False	5	R

Ортогонален масив

❖ Пример: ортогонален масив

- Масив с размерност 4 x 3 и елементите със стойности 1 или 2
- Налице са всички възможни комбинации на стойностите на двойки елементи
- От 8 възможни комбинации на стойностите на всички елементи са налице само 4

$L_4(2^3)$			
Factors			
Runs	1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

(1,1), (1,2), (2,1), (2,2)

(2,2,2)

► Обозначаване на ортогонален масив

► $L_{\text{Runs}}(\text{Levels}^{\text{Factors}})$

- Изпълнения (Runs): Брой на редовете в масива
- Нива (Levels): Брой на възможните стойности на елементите в масива
- Фактори (Factors): Брой на колоните в масива

Често използвани ортогонални масиви

Масив	Изпълнения	Максимален брой фактори	Максимален брой на колоните при определени Levels			
			2	3	4	5
L ₄	4	3	3			
L ₈	8	7	7			
L ₉	9	4	-	4		
L ₁₂	12	11	11			
L ₁₆	16	15	15			
L ₁₆ '	16	5	-	-	5	
L ₁₈	18	8	1	7		
L ₂₅	25	6	-	-	-	6
L ₂₇	27	13	-	13		
L ₃₂	32	31	31			
L ₃₂ '	32	10	1	-	9	
L ₃₆	36	23	11	12		
L ₃₆ '	36	16	3	13		
L ₅₀	50	12	1	-	-	11
L ₅₄	54	26	1	25		
L ₆₄	64	63	63			
L ₆₄ '	64	21	-	-	21	
L ₈₁	81	40	-	40		

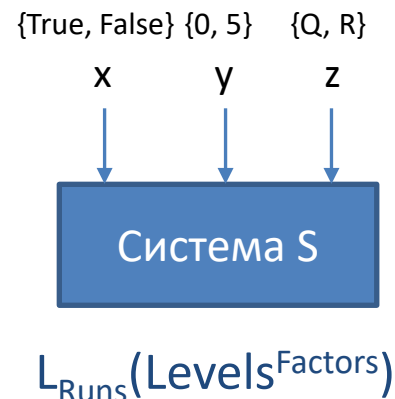
Техника за тестване с ортогонален масив

❖ Свойства на техниката за тестване с ортогонален масив

- Гарантирано тестване на двойки комбинации от всички избрани променливи
- Генериране на по-малък брой тестови сценарии в сравнение с подхода за тестване на всички възможни комбинации
- Генериране на тестов пакет с равномерно разпределение на двойките комбинации
- Възможност за автоматизиране

❖ Пример

$L_4(2^3)$				
	Factors			
Runs	1	2	3	
1	1	1	1	
2	1	2	2	
3	2	1	2	
4	2	2	1	



$L_4(2^3)$				
	Фактори			
Изпълнения	1	2	3	
1	True	0	Q	
2	True	5	R	
3	False	0	R	
4	False	5	Q	

Стъпки за генериране на ортогонален масив

- ❖ **Стъпка 1:** Определя се максималният брой независими входни променливи, с които ще се тества системата (Factors)
- ❖ **Стъпка 2:** Определя се максималният брой стойности, които всяка независима променлива може да приеме (Levels)
- ❖ **Стъпка 3:** Избира се подходящ ортогонален масив с възможно най-малък брой изпълнения (Runs)
- ❖ **Стъпка 4:** Променливите се асоциират с факторите, а стойностите за всяка променлива с нивата
- ❖ **Стъпка 5:** Запълват се неасоциирани нива
- ❖ **Стъпка 6:** Редовете на масива се трансформират в тестови сценарии

Пример: Тестване с ортогонален масив в уеб 1-2

- ❖ Уеб сайт се тества при различни браузъри, плъгини, операционни системи и мрежови връзки
- ❖ **Стъпка 1:** Независимите променливи са 4 (Factors)
- ❖ **Стъпка 2:** Всяка променлива може да има най-много 3 стойности (Levels)
- ❖ **Стъпка 3:** Избира се ортогонален масив $L_9(3^4)$

Променливи	Стойности
Браузър	Netscape, IE, Mozilla
Плъгин	Realplayer, Mediaplayer
ОС	Windows, Linux, Macintosh
Мрежова връзка	LAN, PPP, ISDN

$L_9(3^4)$				
	Фактори			
Изпълнения	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Пример: Тестване с ортогонален масив в уеб 2-2

- ❖ **Стъпка 4:** Свързване на променливите с факторите и стойностите с нивата на масива
- ❖ **Стъпка 5:** Фактор 2 има три специфицирани нива в масива, но съответстващата му променлива (Плъгин) има само 2 възможни стойности (Realplayer и MediaPlayer). Неасоциираните нива се запълват със стойностите на променливата, като се редуват отгоре надолу
- ❖ **Стъпка 6:** Генерират се 9 тестови сценарии

Тестов сценарий	Браузър	Плъгин	ОС	Мрежова връзка
1	Netscape	Realplayer	Windows	LAN
2	Netscape	2 (Realplayer)	Linux	PPP
3	Netscape	MediaPlayer	Macintosh	ISDN
4	IE	Realplayer	Linux	ISDN
5	IE	2 (MediaPlayer)	Macintosh	LAN
6	IE	MediaPlayer	Windows	PPP
7	Mozilla	Realplayer	Macintosh	PPP
8	Mozilla	2 (Realplayer)	Windows	ISDN
9	Mozilla	MediaPlayer	Linux	LAN

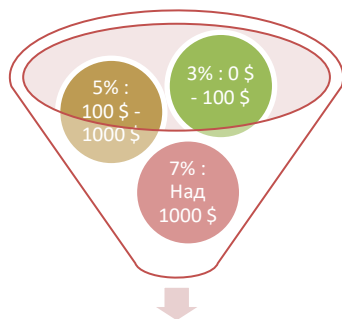
ПРИМЕРИ

Пример: Банкови спестовни сметки

❖ Спестовни сметки в банка с различни лихвени проценти

- Валидни класове
 - ✓ 0 \$ – 100 \$: 3%
 - ✓ 100 \$ - 1000 \$: 5%
 - ✓ Над 1000 \$: 7%
- Невалидни класове (обикновено липсват в спецификацията)
 - ✓ Въвеждане на отрицателни стойности
 - ✓ Въвеждане на нечислени стойности
- Допускания
 - ✓ Числените стойности са точност до втория знак след десетичната запетая (100.00 \$)
- Прилагане на техника за тестване
 - ✓ Класове на еквивалентност върху входа
 - ✓ Класове на еквивалентност върху изхода

Пример: Банкови спестовни сметки



Лихвени проценти

Критерии за сравнение	Класове на еквивалентност	“Наивен” подход
Брой тестовани класове	4	2
Брой тестове	4	16

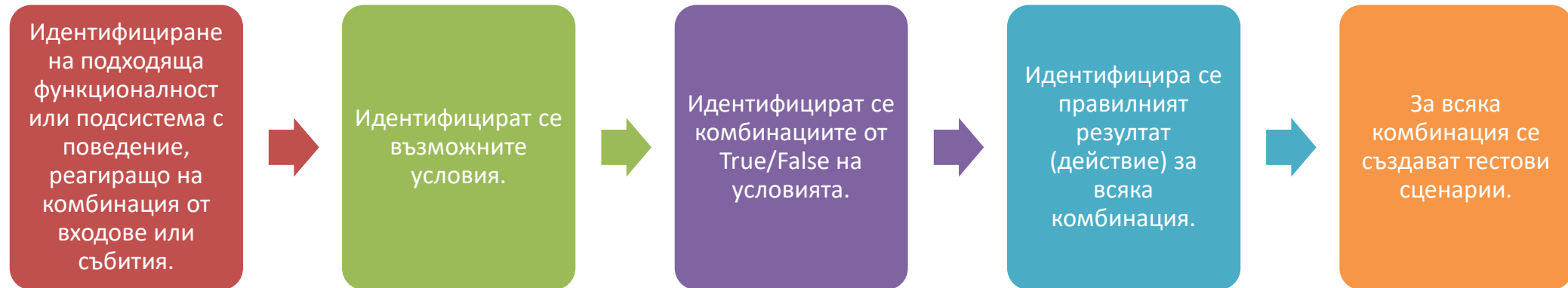
Невалиден клас		Валиден клас (3%)		Валиден клас (5%)		Валиден клас (7%)		Входни тестови данни	
	-0.01\$	0.00 \$	100.00 \$	100.01 \$	999.99 \$	1000.00 \$			
Входни тестови данни									100.00 \$
-10.00 \$		50.00 \$		260.00 \$		1348.00 \$			150.00 \$
Изходни данни									200.00 \$
Съобщение за грешка		3%		5%		7%		...	
								800.00 \$	

Таблица на решенията

❖ Същност

- Предоставят систематичен подход за изследване на ефекта от комбинация на различни входове и други състояния на софтуера, които трябва правилно да имплементират бизнес правила.
- Осигуряват систематичен подход за избор на ефективни тестови сценарии, когато комбинациите на различните входове са прекалено много.

❖ Последователност при създаване на тестове



Дефиниране на таблица на решенията

- ❖ Условия: C_1, C_2, C_3
- ❖ Следствия: E_1, E_2, E_3
- ❖ Възможни стойности при решаване на предикатите от условията: Y, N, _
- ❖ Правила: $R_1 \div R_8$
 - Представят възможните комбинации на условията и следствията от изпълнението им
 - За всяко правило посредством индекс се задава и последователността на възникване на следствията
- ❖ Чек-сума
 - Използва се за верификация на комбинациите, които таблицата на решенията представя

		Правила							
Условия	Стойности	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
C_1	Y, N, _	Y	Y	Y	Y	N	N	N	N
C_2	Y, N, _	Y	Y	N	N	Y	Y	N	N
C_3	Y, N, _	Y	N	Y	N	Y	N	Y	N
Следствия									
E_1		1		2	1				
E_2			2	1			2	1	
E_3		2	1	3		1	1		
Чек-сума	8	1	1	1	1	1	1	1	1

Пример: Отпускане на кредити

Стъпка 1: Идентифицират се възможните условия.

Условия	Правило 1	Правило 2	Правило 3	Правило 4
Размер на месечната вноската				
Продължителност на кредита в години				

Стъпка 2: Идентифицират се комбинациите от True/False на условията.

Условия	Правило 1	Правило 2	Правило 3	Правило 4
Размер на месечната вноската	T	T	F	F
Продължителност на кредита в години	T	F	T	F

Пример: Отпускане на кредити

Стъпка 3: Идентифицира се правилният резултат (действие) за всяка комбинация.

Условия	Правило 1	Правило 2	Правило 3	Правило 4
Размер на месечната вноската	T	T	F	F
Продължителност на кредита в години	T	F	T	F
Резултати/Действия				
Обработка на размера на кредита	Y	Y		
Обработка на срока на кредита	Y		Y	

Допуска се въвеждане и на двата входни параметъра

Пример: Отпускане на кредити

Стъпка 3а: Идентифицират се липсващите в спецификацията резултати/действия.

Условия	Правило 1	Правило 2	Правило 3	Правило 4
Размер на месечната вноската	T	T	F	F
Продължителност на кредита в години	T	F	T	F
Резултати/Действия				
Обработка на размера на кредита	Y	Y		
Обработка на срока на кредита	Y		Y	
Съобщение за грешка				Y

← Липсва в спецификацията

Пример: Отпускане на кредити

Стъпка 4: Идентифицира се правилният резултат (действие) за всяка комбинация.

Условия	Правило 1	Правило 2	Правило 3	Правило 4
Размер на месечната вноската	T	T	F	F
Продължителност на кредита в години	T	F	T	F
Резултати/Действия				
Обработка на размера на кредита		Y		
Обработка на срока на кредита			Y	
Съобщение за грешка	Y			Y

Не се допуска въвеждане на двата входни параметъра

Пример: Отпускане на кредити

Стъпка 5: Идентифицира се правилният резултат (действие) за всяка комбинация.

Условия	Правило 1	Правило 2	Правило 3	Правило 4
Размер на месечната вноската	T	T	F	F
Продължителност на кредита в години	T	F	T	F
Резултати/Действия				
Резултат	Съобщение за грешка	Обработка на размера на кредита	Обработка на срока на кредита	Съобщение за грешка

На всяка комбинация се съпоставя единичен резултат

Пример: Кредитна карта

Условия	Правило 1	Правило 2	Правило 3	Правило 4	Правило 5	Правило 6	Правило 7	Правило 8
Нов клиент (15%)	T	T	T	T	F	F	F	F
Лоялен клиент (10%)	T	T	F	F	T	T	F	F
Ваучер за отстъпка (20%)	T	F	T	F	T	F	T	F
Действия								
Отстъпка (%)	X	X	20	15	30	10	20	0

↖ ↗
Неосъществими
комбинации

↖
Ваучерът за отстъпка не е
валиден заедно с
отстъпката за нов клиент

- ❖ Всяка комбинация представлява отделен тест.
- ❖ При голям брой комбинации, комбинациите се приоритизират и се тестват само най-важните от тях.

Пример: Калкулиране на работна заплата

- ❖ Консултант, работещ повече от 40 часа на седмица, получава заплащане на час, като първите 40 часа се заплащат регулярно, а следващите над 40 се заплащат двойно.
- ❖ Консултант, работещ по-малко от 40 часа на седмица, получава заплащане на час като изработените часове се заплащат регулярно и се създава доклад за отсъствие.
- ❖ Служител на щат, работещ по-малко от 40 часа на седмица, получава заплата и се създава доклад за отсъствие.
- ❖ Служител на щат, работещ повече от 40 часа на седмица, получава заплата.

Пример: Тестване с таблица на решенията 1-3

❖ **Стъпка 1:** Идентифицират се следните условия и следствия

- C_1 : Служител на щат
- C_2 : Работещ по-малко от 40 часа
- C_3 : Работещ 40 часа
- C_4 : Работещ повече от 40 часа
- E_1 : Изплащане на заплата
- E_2 : Създаване на доклад за отсъствие
- E_3 : Регулярно заплащане на час
- E_4 : Двойно заплащане на час

❖ **Стъпка 2:** Условията и следствията се изброяват в таблица на решенията

❖ **Стъпка 3:** Броят на комбинациите е $2^4 = 16$

❖ **Стъпка 4:** Колоните на таблицата се попълват като се има предвид, че $RF_1 = 16/2 = 8$, $RF_2 = 8/2 = 4$, $RF_3 = 4/2 = 2$, $RF_4 = 2/2 = 1$

Пример: Тестване с таблица на решенията 2-3

Условия	Стойности	Правила															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C ₁	Y, N	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
C ₂	Y, N	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
C ₃	Y, N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
C ₄	Y, N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Следствия																	
E ₁																	
E ₂																	
E ₃																	
E ₄																	

► Стъпка 5: Редуцират се следните правила

- Ако C₁ = True и C₂ = True, то стойностите на условията C₃ и C₄ нямат значение. Обединяват се правила 1, 2, 3 и 4
- Ако C₁ = True и C₂ = False, то стойностите на условията C₃ и C₄ нямат значение. Обединяват се правила 5, 6, 7 и 8
- Ако C₁ = False и C₂ = True, то стойностите на условията C₃ и C₄ нямат значение. Обединяват се правила 9, 10, 11 и 12
- Ако C₁ = False, C₂ = False и C₃ = True, то стойността на условието C₄ нямат значение. Обединяват се правила 13 и 14

Пример: Тестване с таблица на решенията 2-3

- ❖ **Стъпка 6:** Изчисляване на чек-сумата за колоните на таблиците
- ❖ **Стъпка 7:** Определяне на следствията от всяко правило
 - Ако $C_1 = \text{True}$ и $C_2 = \text{True}$, то са налице следствия E_1 и E_2 , които се индексират съответно с 1 и 2 според реда на настъпването им.
- ❖ **Стъпка 8:** Генерират се 6 тестови сценарии, съответстващи на правилата в таблицата

		Правила					
Условия	Стойности	1	2	3	4	5	6
C_1	Y, N	Y	Y	N	N	N	N
C_2	Y, N	Y	N	Y	N	N	N
C_3	Y, N	-	-	-	Y	N	N
C_4	Y, N	-	-	-	-	Y	N
Следствия							
E_1		1	1				
E_2		2		2			
E_3				1	1	1	
E_4						2	
Чек-сума	16	4	4	4	2	1	1

