Функционално тестване

доц. д-р Десислава Петрова-Антонова

Концепция на Халдън

- Дефиниция за функция: **(X_i, Y_i)**
- Функционалното тестване разглежда всяка програма P като функция, трансформираща входен вектор X_i в изходен вектор Y_i: **$Y_i = P(X_i)$**
- Примери
  - Програма P, изчисляваща корен квадратен Y_1 от променлива X_1
  - Програма P, компилираща C код X_2 в обектен код Y_2
  - Програма P, генерираща сортиран масив Y_3 от входен вектор $X_3 = \{A, N\}$
- Ключови елементи на функцията
  - Вход
  - Изход
  - Очаквана трансформация на входа в изход
- Програмата се интерпретира като функция без значение как входния вектор се трансформира в изходен вектор.

Типове променливи

Видове променливи

- Входна променлива
  - Получава входна стойност при стартирането на системата
- Изходна променлива
  - Съхранява изходна за системата стойност
- Променлива с двойно предназначение
  - Получава входна стойност при стартирането на системата и изходна стойност след това
- Променлива с множествен тип
  - Съхранява входни или изходни стойности от различни типове данни

Критерии за избор на стойности за входни променливи

- Тестови данни при входен домейн, представен с дискретно множество
  - Всички стойности от домейна
- Тестови данни при входен домейн, включващ един или няколко непрекъснати интервали
  - Минималната стойност от интервала
  - Максимална стойност от интервала
  - Представителна стойност за интервала
  - Стойности със специални математически свойства
  - Невалидни стойности
  - Стойности, представящи +∞ и -∞, когато максималната и минимална стойност са съответно +∞ и -∞

Критерии за избор на стойности за изходни променливи

- Тестови данни при изходен домейн, представен с малко дискретно множество
  - Избират се входни данни, които водят до получаване на всички стойности от изходния домейн
- Тестови данни при изходен домейн, представен с голямо дискретно множество
  - Избират се различни входни данни, които водят до получаване на различни стойности от изходния домейн
- Тестови данни при изходен домейн, представен с няколко непрекъснати интервали от стойности
  - Избират се входни данни, които водят до получаване на минималните стойности за интервалите
  - Избират се входни данни, които водят до получаване на максималните стойности за интервалите
  - Избират се входни данни, които водят до получаване на вътрешни за интервалите стойности

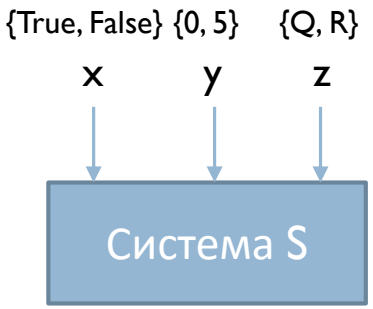
Критерии за избор на стойности за променливи с двойно предназначение и с множествен тип

- Променлива с двойно предназначение
  - Избират се тестови сценарии, при които входната стойност на променливата е различна от изходната стойност
  - Избират се тестови сценарии, при които входната стойност на променливата е еднаква с изходната стойност
- Променлива с множествен тип
  - При входните променливи се избират тестови сценарии, които подават входни стойности от всички типове
  - При изходни променливи се избират тестови сценарии, които подават входни стойности, които предизвикват получаване на изходни стойности от всички типове

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, 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.



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

- Обозначаване на ортогонален масив
  - $L_{Runs}(Levels^{Factors})$
    - Изпълнения (Runs): Брой на редовете в масива
    - Нива (Levels): Брой на възможните стойности на елементите в масива
    - Фактори (Factors): Брой на колоните в масива

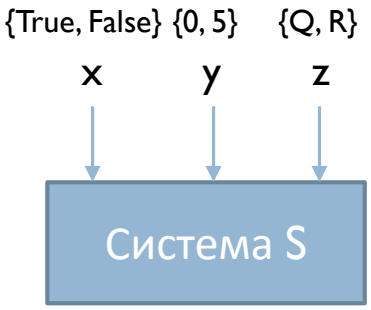
Често използвани ортогонални масиви

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
  - 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

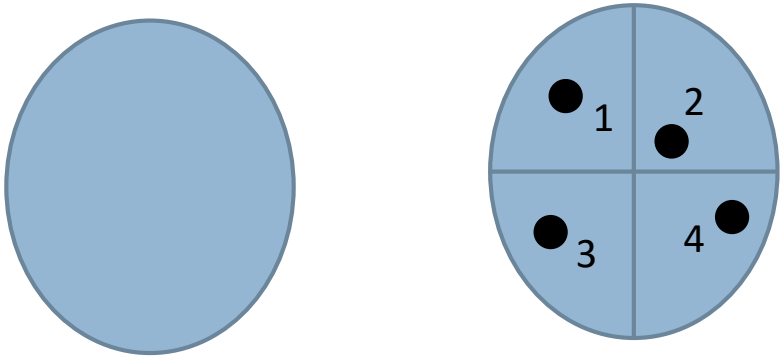
- **Стъпка 4:** Свързване на променливите с факторите и стойностите с нивата на масива
- **Стъпка 5:** Фактор 2 има три специфицирани нива в масива, но съответстващата му променлива (Плъгин) има само 2 възможни стойности (Realplayer и Mediaplayer). Неасоциираните нива се запълват със стойностите на променливата, 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

Разделяне на класове на еквивалентност

Основни понятия

- Клас на еквивалентност
  - Множество от входове, които системата третира по един и същ начин по време на тестване
- Входно условие
  - Предикат върху стойностите на входния домейн
- Валиден вход
  - Елемент от входния домейн, за който системата се очаква да върне стойност, която не е грешка
- Невалиден вход
  - Вход, за който системата се очаква да върне грешка



Идентифициране на класове на еквивалентност

- Входно условие, дефиниращо интервал [a, b]
  - 3 класа за a ≤ X ≤ b, X< a и X > b
- Входно условие, дефиниращо множество от стойности
  - Класове за всички елементи от множеството
  - 1 клас за невалиден елемент
- Входно условие, дефиниращо индивидуални стойности
  - Класове за всички валидни входове (Пример, меню)
- Входно условие, дефиниращо определен брой валидни стойности
  - Клас за валиден брой стойности
  - Клас за брой стойности по-голям от валидния
  - Клас за брой стойности равен на 0 (Пример, масив)
- Входно условие, дефиниращо стойност от тип “must-be”
  - Клас за стойност от тип “must-be”
  - Клас за стойност, която не е от тип “must-be” (Пример, парола)
- Допълнително разделяне на класове на еквивалентност
  - Ако елементите от даден клас предизвикват различно поведение на системата, то е необходимо ново разделяне

Тестване с класове на еквивалентност

- Последователност на създаване на тестови сценарии от класове на еквивалентност
  - **Стъпка 1:** На всеки клас се присвоява уникален идентификатор
  - **Стъпка 2:** За всеки валиден клас на еквивалентност, който все още не е покрит, се създава тестов сценарий, който покрива възможно най-много непокрити валидни класове
  - **Стъпка 3:** За всеки невалиден клас на еквивалентност, който все още не е покрит, се създава тестов сценарий, който покрива само един невалиден клас

Пример: Начисляване на данък

- Система начислява данък в зависимост от брутен доход както следва
  - 22%, ако брутният доход е от 1 до 29 500
  - 27%, ако брутният доход е от 29 501 до 58 500
  - 36%, ако брутният доход е от 58 501 до 100 билиона
- Входни условия
  - **Условие 1:** 1 ≤ брутен доход ≤ 29 500
  - **Условие 2:** 29 501 ≤ брутен доход ≤ 58 500
  - **Условие 3:** 58 501 ≤ брутен доход ≤ 100 билиона
- Класове на еквивалентност
  - **Клас 1:** 1 ≤ брутен доход ≤ 29 500, валиден вход
  - **Клас 2:** брутен доход < 1, невалиден вход
  - **Клас 3:** 29 501 ≤ брутен доход ≤ 58 500, валиден вход
  - **Клас 4:** 58 501 ≤ брутен доход ≤ 100 билиона, валиден вход
  - **Клас 5:** брутен доход > 100 билиона, невалиден вход

Тестов сценарий	Тестова стойност	Очакван резултат	Клас на еквивалентност
1	22 000	4 840	Клас 1
2	46 000	12 420	Клас 3
3	68 000	24 480	Клас 4
4	-20,000	Съобщение за грешка	Клас 2
5	150 билиона	Съобщение за грешка	Клас 5

Анализ на граничните стойности

Основни понятия

- Гранична стойност
  - Стойност, намираща се близо до границата на входния домейн
- Гранични условия
  - Предикати, които се прилагат на или близо до границите на входните и изходните класове на еквивалентност
- Предназначение
  - Разширява техниката за тестване с класове на еквивалентност, откривайки дефекти близо до границите на входния домейн

Идентифициране на гранични стойности

- Клас на еквивалентност, дефиниращ интервал
  - Създават се тестови сценарии за стойностите, които са на и до границата на входния домейн
  - Пример: $-10.0 \leq X \leq 10.0$
- Клас на еквивалентност, дефиниращ определен брой стойности
  - Създават се тестови сценарии за максималния и минималния брой стойности и по един тестов сценарии за брой стойности по-малък от минималния и брой стойности по-голям от максималния (Пример, брой студенти в стая на общежитие)
- Клас на еквивалентност, дефиниращ подредено множество
  - Създават се тестови сценарии за първия и последния елемент от множеството (Пример, линеен списък)

Пример: Начисляване на данък

- Класове на еквивалентност
  - **Клас 1:** 1 ≤ брутен доход ≤ 29 500; валиден вход
  - **Клас 2:** брутен доход < 1; невалиден вход
  - **Клас 3:** 29 501 ≤ брутен доход ≤ 58 500; валиден вход
  - **Клас 4:** 58 501 ≤ брутен доход ≤ 100 билиона; валиден вход
  - **Клас 5:** брутен доход > 100 билиона; невалиден вход
- Тестови данни
  - **Клас 1:** 0.99; 1; 1.01; 29 499.99; 29 500; 29 500.01
  - **Клас 2:** 0.99; 1; -100 билиона
  - **Клас 3:** 29 500.99; 29 501; 58 499.99; 58 500; 58 500.01
  - **Клас 4:** …
  - **Клас 5:** …
- Излишните стойности се премахват

Таблица на решенията

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

Създаване на тестови сценарии с таблица на решенията

- **Стъпка 1:** Идентифицират се условията и следствията от тях
- **Стъпка 2:** Всички условия и следствия се записват в таблица на решенията като списък. Записват се и стойностите, които условието може да приеме
- **Стъпка 3:** Изчислява се броя на възможните комбинации
- **Стъпка 4:** Колоните на таблицата се попълват с възможните комбинации. За всеки ред се изпълнява следното:
  - Определя се факторът на повторение (RF) посредством разделяне на оставащия брой комбинации на броя възможни стойности за условието
  - Първата стойност за условието се записва толкова пъти колкото е изчислената стойност за фактора на повторение. След това толкова пъти се записва и втората стойност за условието. Повтаря се до запълване на реда.
- **Стъпка 5:** Редуцират се правилата
- **Стъпка 6:** Проверяват се покритите правила
- **Стъпка 7:** Попълват се следствията от изпълнението на правилата
- **Стъпка 8:** Колоните на таблицата се трансформират в тестови сценарии

Пример: клиенти на авиокомпания

- Притежател на златна карта 10%
- Ползване на услуги на авиокомпанията за пръв път 5%
- Закупуване на билет за бизнес класа 2%
- Отстъпката за ползване на услугите на авиокомпанията за пръв път е валидна само за икономичната класа.

		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 (10%)				1	1				
E_2 (5%)							1		
E_3 (2%)				2		1		1	
E_4 (0%)									1
Чек-сума	8	1	1	1	1	1	1	1	1

Въпроси?

d.petrova@fmi.uni-sofia.bg