Yazılım Yapısal Kapsama Analizinde Testlerin Önceliklendirilmesi

Tolga Ayav

İzmir Yüksek Teknoloji Enstitüsü
Bilgisayar Mühendisliği Bölümü
35430 Urla, İzmir
tolgaayav@iyte.edu.tr

Özet. Bu çalışma değiştirilmiş koşul/karar kapsama stratejisince türetilmiş test girişlerinin önceliklendirilmesi için bir teknik sunmaktadır. Teknik Fourier analizine dayanmaktadır ve testlerin hata ortaya çıkarma potansiyellerine göre sıralanabilmesini hedeflemektedir. Bu sayede yazılımın yüksek öncelikli testlerden başlayarak düşüğe doğru sınanması veya test kümesinin yüksek öncelikli testleri kapsayacak şekilde daraltılması mümkün olabilecektir.
Anahtar Kelimeler: Yazılım sınama, yapısal kapsama analizi, değiştirilmiş koşul/karar kapsama, Fourier analizi.

1 Giriş

Araştırmalar test maliyetinin toplam yazılım geliştirme maliyetinin yarısından fazla tutabileceğini göstermiştir [10]. Bir yazılımın hatasız olduğunu göstermek pratikte mümkün olamamaktadır ancak yazılım sınama metotlarıyla hatalarından arındırılan yazılımların kalite ve güvenilirliğini arttırmak mümkündür. Yazılımların büyüklüğüne, test maliyetine ve eldeki bütçeye göre tam kapsamlı bir sınama istenmeyebilir. Bu durumda, yazılımın hangi kısımlarının hangi test kümesiyle test edileceğine karar verilmesi kritik bir öneme sahiptir. Bu çalışma test durumlarının önceliklendirilmesi konusunu ele almaktadır. Test önceliklendirme yöntemleri testleri bazı öncelik kriterlerine göre sıralamaya dayalı tekniklerdir. Böylelikle testler hata bulma yetenekleri ölçüsünde sıralandığında, i) Hataların erken tespit edilme olasılığı artar, ii) Testlerin tamamı uygulanamadığında yüksek öncelikli testlerden oluşan bir alt küme seçilebilir.

Bu çalışma yazılım yapısal kapsama tekniklerinden biri olan değiştirilmiş koşul/karar kapsama (MCDC) analizi ile elde edilmiş testlerin önceliklendirilmesi üzerinedir. Federal Havacılık İdaresi'nin (FAA) şart koştuğu yazılım testlerinin başında MCDC kapsama analizi gelmektedir [1]. Elbaum çalışmasında endüstriden gelen bir bilgiye göre 20,000 satırdan oluşan bir kodun MCDC test kümesiyle

[1] FAA'nın onaylamış olduğu "DO-178C Software Considerations in Airborne Systems and Equipment Certification" standardı en kritik olan A seviyesi yazılımların kapsama analizi için MCDC'yi kullanmaktadır.

testinin yedi hafta sürdüğünü belirtmektedir [3]. Öncelik sırasına göre uygulanan testlerin hata bulma olasılığı daha yüksek olacağından test daha önce sonlandırılabilecektir ve özellikle regresyon testleri açısından düşünüldüğünde, bu konu daha da önemli olmaktadır [4].

Bu çalışmada testlerin öncelikleri mutasyon analizine göre hata ortaya çıkarma potansiyelleri üzerinden tanımlanmıştır. Öncelikler ile koşulların Fourier analiziyle elde edilen spektral katsayıları arasında tanımlanan bir negatif korelasyon kapsamlı denemelerle sınanmaya çalışılmıştır. Böylelikle test öncelikleri doğrudan spektral katsayılara bakarak belirlenebilmektedir.

Bölüm 2'de, önerilen metot detaylı olarak açıklanmaktadır. Öncelikle Kısım 2.1, MCDC analizinde kullanılan Boole işlevlerinin türevlerini, Kısım 2.2 Boole işlevlerinin Fourier analizini ve Kısım 2.3 değiştirilmiş koşul/karar kapsama analizini sunmaktadır. Bölüm 3 hata sınıflarını ve çalışmada yararlanılan mutasyon analizini anlatmaktadır. Bölüm 4 ise çeşitli işlevler için belirli hata sınıfları içerisinde ve 1,2,3-hata varsayımları altında spektral katsayılarla testlerin hata ortaya çıkarma potansiyelleri arasında negatif ilintiyi kapsamlı denemelerle göstermektedir. Çalışma Bölüm 5 ile sonuçlandırılmaktadır.

2 Testlerin Önceliklendirilmesi

$\mathbf{c} = (c_1, \ldots, c_i, \ldots, c_n)$ n adet Boole değişkeninden/koşulundan oluşan bir vektör olsun. Boole işlevi $f(\mathbf{c})$ aşağıdaki şekilde ifade edilebilir:

$$f : \mathbb{B}^n \to \mathbb{B}, \quad \mathbb{B} = \{0, 1\}.$$

Bu çalışmada Boole değişkenleri ve operasyonları için aşağıdaki tanım kullanılmıştır:

$$x ::= 0|1|x'|x_1 \Theta x_2, \quad \Theta = \{\cdot, +\}.$$

"$\cdot$" ve "$+$" sırasıyla VE ve VEYA mantık operasyonlarını, x' ise x değişkeninin olumsuzunu ifade etmektedir. Diğer operatörler bu temel olanlardan türetilebilir. Boole işlevleri yazılımların yapısal kapsama analizinde yazılımın veya kod parçasının yapısını ifade etmekte de kullanılabilir. Örnek olarak aşağıdaki kod parçasını ele alalım:

```
if(((curTemp < dTemp-thresholdDiff) ||
   (override && curTemp < overTemp - thresholdDiff)) &&
   (timeSinceLastRun > minLag) )
{...}
```

(Örnek, NASA/TM-2001-210876 A Practical Tutorial on Modified Condition/Decision Coverage raporundan alınmıştır.)

İfade içerisindeki koşulların her birini aşağıda görüldüğü şekilde Boole değişkenlerle ifade edersek,

c_0: `curTemp < dTemp-thresholdDiff`
c_1: `override`
c_2: `curTemp < overTemp - thresholdDiff`
c_3: `timeSinceLastRun > minLag`

Test vektörü $\mathbf{c} = [c_0, c_1, c_2, c_3]^T$ olmak üzere Boole işlevi aşağıdaki gibi yazılabilir:

$$f(\mathbf{c}) = (c_0 + (c_1 \cdot c_2)) \cdot c_3$$

Testlerin önceliklendirilmesi ilk kez formel olarak Elbaum vd. tarafından yapılmıştır [3]. Buradan uyarlayarak gerekli tanımlar aşağıdaki şekilde yapılabilir:

Tanım 1 *(Hata Ortaya Çıkarma Potansiyeli). $f : \mathbb{B}^n \to \mathbb{B}$ olarak verilen bir n girişli Boole işlevi için maskeleyen değiştirilmiş koşul/karar kapsama kriterine göre elde edilmiş test ikilileri $T = \{t_0, t_1, \ldots, t_{n-1}\}$ olsun. Boole işlevine ilişkin m adet mutant yaratılmış olsun. Test ikililerinin hata bulma potansiyeli $HP(t_i)$, ayırt edebildikleri mutant sayısının toplam mutant sayısı m'e bölünmesiyle elde edilir.*

Burada mutantlar Bölüm 3'te daha detaylı anlatılacağı üzere işlevin belirli hata prototipleri varsayımı altında mutasyona uğratılmasıyla elde edilirler.

Tanım 2 *(Testlerin Öncelikleri). $t_0, t_1, \cdots, t_n$, c_0, c_1, c_{n-1} koşullarına ilişkin üretilmiş test ikilileri olsun. Test ikililerinin öncelikleri $Pr(t_i)$, hata ortaya çıkarma potansiyelleriyle orantılıdır:*

$$Pr(t_i) \sim HP(t_i), \quad i \in \{0, \ldots, n-1\}.$$

2.1 Boole İşlevlerinin Türevi

Boole işlevleri için literatürde çeşitli türev ve integral hesaplamaları tanımlanmıştır. En kabul görmüş türev tanımı uyarınca bir Boole işlevinin c_i'ye göre türevi aşağıdaki gibi hesaplanır [11]:

$$\frac{\partial f(\mathbf{c})}{\partial c_i} = f(c_1, \ldots, c_i, \ldots, c_n) \oplus f(c_1, \ldots, c_i', \ldots, c_n) \tag{1}$$

Burada $\oplus$ dışlayan-VEYA operatörüdür. Buna göre, diğer değişkenler sabit kalmak kaydıyla c_i'nin 0 ve 1 değerleri için işlevin sonucu değişmiyorsa, bu işlevin c_i'ye göre türevi 0'dır. Eğer her iki durumda sonuçlar farklıysa türev 1'dir.

2.2 Boole İşlevlerinin Fourier Analizi

Boole işlevlerinin spektrum analizinde Fourier, Walsh, Walsh-Hadamard dönüşümlerinin hepsi aynı anlama gelmektedir ve literatürde birbirinin yerine kullanılmaktadır [6, 12]. Bir $x(t)$ sinyalinin Fourier (Walsh) açılımı aşağıdaki gibi yazılır:

$$x(t) = a_0 + \sum_{i=1}^{\infty} (a_i sal(i,t) + b_i cal(i,t)) \tag{2}$$

Burada a_0 doğru akım bileşeni, a_i ve b_i ise Walsh spektrum katsayılarıdır. *sal* ve *cal* işlevleri ise Fourier açılımındaki sinüs ve kosinüs işlevlerine olan benzerlikleri

nedeniyle bu şekilde isimlendirilmiş olup aşağıda gösterilen Walsh işlevlerine denk düşerler:

$$sal(\omega, x) = wal(2\omega - 1, x) \tag{3}$$
$$cal(\omega, x) = wal(2\omega, x) \tag{4}$$

Walsh işlevleri $[0, 1]$ zaman aralığında tanımlanmıştır ve doğal sıralı Walsh işlevleri aşağıda verilen Walsh dönüşüm dizeyince de ifade edilebilirler:

$$\mathbf{W}(n) = \bigotimes_{i=1}^{n} \mathbf{W}(1), \quad \mathbf{W}(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

Burada $\bigotimes$ Kronecker çarpımını ifade etmektedir. Dizeyin her bir satırı ayrı bir Walsh işlevine aittir. Örneğin, $\mathbf{W}(2)$ aşağıdaki şekilde hesaplanır:

$$\mathbf{W}(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{matrix} 1 \\ x_1 \\ x_2 \\ x_1 \oplus x_2 \end{matrix}$$

Dizeyin her bir satırının denk geldiği Walsh işlevi/bileşeni dizeyin yanında belirtilmiştir. $\oplus x_{1,2}$ şeklindeki ifade $x_1 \oplus x_2$ ile aynıdır. Walsh dizeyi görüldüğü gibi elemanları sadece $+1$ ve -1'den oluşan bir kare dizeydir ve satırlar karşılıklı dikgendir. Öyle ki, I^n bir $n \times n$ birim dizey olmak üzere $\mathbf{W}(n) \times \mathbf{W}(n)^T = n\mathbf{I}^n$ diyebiliriz. Walsh dönüşümü ve bunun ters dönüşümü aşağıdaki gibi tanımlanır:

$$S_f(\omega) = 2^{-n} \sum_{x=0}^{2^n-1} (1 - 2f(x)) wal(\omega, x) \tag{5}$$

$$f(x) = \frac{1}{2} - \frac{1}{2} \sum_{\omega=0}^{2^n-1} S_f(\omega) wal(\omega, x) \tag{6}$$

Dizey simgeleminde dönüşümler şu şekilde ifade edilebilirler:

$$\mathbf{S}_f = 2^{-n} \mathbf{W}(n)(\mathbf{1} - 2\mathbf{F}) \tag{7}$$

$$\mathbf{F} = \frac{1}{2}(\mathbf{1} - \mathbf{W}(n)\mathbf{S}_f) \tag{8}$$

Formül (7)'de görülen $\mathbf{1} - 2\mathbf{F}$ ifadesi dönüşümden önce Boole değişkenlerinin aldığı $\{0, 1\}$ değerlerini $\{1, -1\}$ değerlerine dönüştürmeye yarar. Benzer şekilde ters dönüşümde de $\mathbf{W}(n)\mathbf{S}_f$, $\{1, -1\}$ değerini üretir ve bu değer $\{0, 1\}$'e dönüştürülür. Herhangi bir $f(x)$ işlevinin Walsh açılımı aşağıdaki şekilde yazılır:

$$\begin{aligned} f(x) = S_f(0) + S_f(1)x_1 + S_f(2)x_2 + S_f(3)(x_1 \oplus x_2) + \\ S_f(4)x_3 + \cdots + S_f(2^n - 1)(x_1 \oplus \cdots \oplus x_n) \end{aligned} \tag{9}$$

Walsh katsayıları sinyal çıkışıyla ilgili bileşene ait Walsh işlevi arasındaki ilintiyi gösterirler. Örneğin, $k \in \{1, 2, \ldots, n\}$ olmak üzere $wal(2^{k-1}, t)$ devrenin x_k girişine karşılık gelen bileşendir ve $S_f(2^{k-1})$ sinyal ile, diğer bir deyişle Boole işlevinin doğruluk yöneyi ile bu bileşen arasındaki ilintidir.

2.3 Maskeleyen Değiştirilmiş Koşul/Karar Kapsama (Masking MCDC) Analizi

Maskeleyen Değiştirilmiş Koşul/Karar Kapsama (Masking MCDC) tekniği, yapısal kapsama analizinde kullanılan yöntemlerden biridir. MCDC'nin üç farklı formu bulunmaktadır. Bu çalışmada kullanılan "maskeleyen" tipinin diğer yöntemlerle kıyaslamalı olarak zayıflıkları ve üstünlükleri hakkında detaylı bilgi için [2] ve [5]'e başvurun. MCDC stratejisine göre test kümesi her bir koşul için elde edilecek test vektör ikililerinden oluşur $(\mathbf{x}, \mathbf{y})$ ve aşağıdaki şekilde tanımlanır:

$$T = \{t_i = (\mathbf{x}, \mathbf{y}) : \quad \forall i \in \{1, \ldots, n\} : \\ x_i = y_i' \wedge f(\mathbf{x}) = f'(\mathbf{y}) \wedge \frac{\partial f(\mathbf{x})}{\partial x_i} = 1 \wedge \frac{\partial f(\mathbf{y})}{\partial y_i} = 1\} \tag{10}$$

İfadedeki her koşul için öyle test ikilileri bulunmalıdır ki her bir test ikilisinde:

1. Koşul i her iki testte farklı olmalıdır $(x_i = y_i')$
2. İfade her iki test için farklı değer üretmelidir $(f(\mathbf{x}) = f'(\mathbf{y}))$
3. Birinci test $\mathbf{x}$ için Koşul i'nin sonuç üzerinde etkisi olmalıdır $(\frac{\partial f(\mathbf{x})}{\partial x_i} = 1)$
4. İkinci test $\mathbf{y}$ için Koşul i'nin sonuç üzerinde etkisi olmalıdır $(\frac{\partial f(\mathbf{y})}{\partial y_i} = 1)$

Örnek işlevin her bir değişkene göre kısmi türevi Formül 1'e göre analitik olarak aşağıdaki gibi hesaplanabilir:

$$\frac{\partial f(\mathbf{c})}{\partial c_0} = (c_1' + c_2')c_3 \tag{11}$$

$$\frac{\partial f(\mathbf{c})}{\partial c_1} = c_0' c_2 c_3 \tag{12}$$

$$\frac{\partial f(\mathbf{c})}{\partial c_2} = c_0' c_1 c_3 \tag{13}$$

$$\frac{\partial f(\mathbf{c})}{\partial c_3} = c_0' + (c_1 c_2) \tag{14}$$

Kısmi türevlerin yardımıyla test girişleri kolaylıkla bulunabilir. Örneğin, c_1 koşulu için test ikilisini oluştururken $\partial f(\mathbf{c})/\partial c_1 = 1$ şartını sağlayan tek (c_0, c_2, c_3) değerinin $(0, 1, 1)$ olduğu görülmektedir. Bu durumda ilgili test ikilisi $\mathbf{c^{(1)}} = [0, 1, 1, 1]^T$ ve $\mathbf{c^{(2)}} = [0, 0, 1, 1]^T$ olarak yazılabilir. Aynı zamanda $\mathbf{f}(\mathbf{c^{(1)}}) = 1$ ve $\mathbf{f}(\mathbf{c^{(2)}}) = 0$ şartlarının yerine getirildiğine de dikkat edilmelidir. Benzer şekilde hesaplandığında Tablo 1'de görüldüğü şekilde dört test ikilisi, diğer bir deyişle sekiz test vektörü elde edilir. İçlerinden ikisi tekrar ettiğinden elenebilir, böylelikle toplam altı test yeterli olur.

Bir önceki kısımda açıklandığı şekliyle Boole işlevinin girişlerine ilişkin spektral katsayıları hesaplamak üzere öncelikle işlevin doğruluk yöneyi aşağıdaki gibi yazılır:

$$\mathbf{F} = [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]^T$$

Test No	c_0	c_1	c_2	c_3	$f(\mathbf{c})$
1	**1**	1	0	1	1
2	**0**	1	0	1	0
3	0	**1**	1	1	1
4	0	**0**	1	1	0
5	0	1	**1**	1	1
6	0	1	**0**	1	0
7	1	1	1	**1**	1
8	1	1	1	**0**	0

Tablo 1. $f(\mathbf{c})$ için bulunan test vektörleri

Formül (7)'de verilen dönüşüm hesaplaması yapıldığındaysa aşağıdaki spektral katsayılar elde edilir:

$$\mathbf{S}_f = [0.375, 0.375, 0.125, 0.125, 0.125, 0.125, -0.125, -0.125, 0.625, \\ -0.375, -0.125, -0.125, -0.125, -0.125, 0.125, 0.125]^T \quad (15)$$

Buna göre, işlevin c_0, c_1, c_2 ve c_3 ile ilintilerinin sırasıyla $S_f(1) = 0.375$, $S_f(2) = 0.125$, $S_f(4) = 0.125$ ve $S_f(8) = 0.625$ olduğu görülmektedir. Yazının devamında sözkonusu ilintiler sırasıyla s_0, s_1, s_2 ve s_3 olarak adlandırılacaktır.

Regresyon testi için testlerin önceliklendirilmesi konusunda literatürde çalışmalara rastlanmaktadır [7–9]. Jones ve Harrold ise çalışmalarında MCDC test kümesinin küçültülmesi ve testlerin önceliklendirilmesi için iki yeni teknik sunmuştur [4]. Test önceliklendirme çalışmalarında önerilen tekniklerin çoğu APFD (Average Percentage of Faults Detected) metriğine dayanan sezgisel algoritmalar kullanmaktadır. Bu çalışmadaysa literatürdeki diğer çalışmalardan farklı olarak Boole işlevlerinin spektral analizine dayanan matematiksel bir yöntem sunulmaktadır.

Testlerin spektral katsayılara bağlı olarak önceliklendirilmesi aşağıda verilen teoremle ele alınmıştır.

Teorem 1 *$f : \mathbb{B}^n \to \mathbb{B}$, n girişli Boole işlevi ve $s_0, s_1, \ldots, s_{n-1}$ sırasıyla $c_0, c_1, \ldots, c_{n-1}$ koşullarına ilişkin Walsh işlevlerine ait spektral katsayılar olsun. Testlerin öncelikleri ile spektral katsayılar arasında negatif korelasyon vardır.*

Teoremin ispatı burada yapılmayacak olup doğruluğu Bölüm 4'te verilen kapsamlı denemelerle sınanmaya çalışılacaktır.

3 Hata Prototiplerinin Sınıflandırılması

n koşula sahip bir Boole işlevi bu koşullara bağlı olarak 2^n farklı çıkış üretebilir. Öte yandan n koşula sahip 2^{2^n} farklı Boole işlevi yazmak mümkündür. Örneğin, dört koşul olması durumunda, $2^{2^4} = 65536$ farklı Boole işlevi oluşturulabilir. Tablo 3 bu işlevlerden bir kısmını göstermektedir. Orijinal işlevimiz

olan $f_{43648} = (c_0 + (c_1c_2))c_3$ bunlardan sadece biridir ve testin amacı orijinal işlevi diğerlerinden ayırabilmektir. İşlevi diğer $2^{2^n} - 1$ işlevden ayırabilmek için 2^n adet test gerekir. Çoklu-koşul kapsama testi olarak adlandırılan bu test koşulların fazla olması durumunda imkansızlaşır. Eğer $m < 2^n$ adet test uygulayacak olursak, $2^n - m$ kadar kombinasyon denenmemiş olur. Bu durumda m adet testin n koşulu kapsama oranı aşağıdaki formülle hesaplanabilir:

$$P_{(n,m)} = 1 - \frac{2^{(2^n - m)} - 1}{2^{2^n}} \tag{16}$$

Bir test kümesinin hata kapsama oranını incelerken yukarıda verilen formülden yola çıkmak çok sağlıklı olmayabilir. Yazılım geliştirilirken koşullarda yapılabilecek hatalar sonucunda istenenden farklı Boole işlevleri oluşturmak olasıdır ancak her bir hatalı Boole işlevinin ortaya çıkma olasılığı birbirinden farklıdır ve $2^{2^n} - 1$ işlevin çok büyük bir kısmının hatalı yazılımla elde edilme olasılığı çok düşüktür. Bu nedenle hata kapsama oranı hesaplanırken yazılım geliştiricilerin sık yaptığı hatalara dayalı bilgiden faydalanarak mutasyon metoduna dayalı bir hesaplama yapmak daha gerçekçi olacaktır. Testlerin veya test oluşturma yöntemlerinin per-

$c_0c_1c_2c_3$	f_0	f_1	f_2	f_3	$\cdots$	f_{32768}	$\cdots$	f_{43648}	$\cdots$	f_{65514}	$\cdots$	f_{65535}
0000	0	1	0	1		0		0		0		1
0001	0	0	1	1		0		0		1		1
0010	0	0	0	0		0		0		0		1
(0011)	0	0	0	0		0		0		1		1
0100	0	0	0	0		0		0		0		1
(0101)	0	0	0	0		0		0		1		1
0110	0	0	0	0		0		0		1		1
(0111)	0	0	0	0		0		1		1		1
1000	0	0	0	0		0		0		1		1
1001	0	0	0	0		0		1		1		1
1010	0	0	0	0		0		0		1		1
1011	0	0	0	0		0		1		1		1
1100	0	0	0	0		0		0		1		1
(1101)	0	0	0	0		0		1		1		1
(1110)	0	0	0	0		0		0		1		1
(1111)	0	0	0	0		1		1		1		1

Tablo 2. Dört değişkenli Boole işlevlerine ilişkin örnek gösterim

formansları genellikle mutasyon analizleriyle değerlendirilir. Bir mutant, orijinal Boole işlevinde küçük bir değişiklik yaparak elde edilir ve bir hata prototipini temsil eder. Bir testin mutantlarını orijinalinden ayırabilme yeteneği arttıkça hata kapsama oranı da artar. Literatürde önerilen ve genel kabul görmüş hata prototipleri aşağıda kısa açıklamalarla verilmiştir [1].
İşlem Hataları:

İRH İşlem Referans Hatası. "VE" işlemi "VEYA" ile veya "VEYA" işlemi "VE" ile değiştirilir. Örn. $(c_0 + (c_1c_2)) + c_3$.

İOH İfade Olumsuzlama Hatası. İfadenin bir kısmı (alt ifade) olumsuzlanır. Örn. $(c_0 + (c_1c_2)')c_3$.

DOH Değişken Olumsuzlama Hatası. Boolean değişkenlerinden biri olumsuzlanır. Örn. $(c_0 + (c_1'c_2))c_3$.

İKH İlişkisel Kaydırma Hatası. İfadedeki koşullar arasındaki ilişkilerde hata yapılmasından kaynaklanır.

- **PUH** Parantez Unutma Hatası. İfade içerisindeki bir parantez çifti unutulursa oluşur. Örn. $(c_0 + c_1c_2)c_3$.
- **PEH** Parantez Ekleme Hatası. Bir parantez çifti yanlışlıkla ifadeye eklendiğinde ortaya çıkar. Örn. $(c_0 + (c_1c_2))c_3$.

Terim Hataları:

EDH Eksik Değişken Hatası. Koşullardan birinin unutulması durumudur. Örn. $(c_0 + (c_1c_2))$.

DRH Değişken Referans Hatası. Koşullardan biri ifadede yer alan başka bir koşul ile yer değiştirilir. Örn. $(c_1 + (c_1c_2))c_3$.

YBH Yantümce Birleşme Hatası. Koşul a yerine ab yazılması durumudur. Örn. $(c_0 + (c_1c_3c_2))c_3$.

YAH Yantümce Ayrılma Hatası. Koşul a yerine $a + b$ yazılması durumudur. Örn. $(c_0 + c_3 + (c_1c_2))c_3$.

0-T-H 0'a Takılma Hatası. Koşullardan birinin hep 0 olması durumudur. Örn. $(0 + (c_1c_2))c_3$.

1-T-H 1'e Takılma Hatası. Koşullardan birinin hep 1 olması durumudur. Örn. $(c_0 + (1 \cdot c_2))c_3$.

4 Deneyler

Bu bölümde çalışmanın konusu olan, spektral katsayılarla testlerin hata ortaya çıkarma potansiyelleri arasındaki ilişki deneysel olarak verilecektir. İlk olarak dört girişli sekiz farklı Boole işlevi oluşturulmuş, bunlar için spektral katsayılar ve test girişleri hesaplanmıştır. Testlerin hata ortaya çıkarma potansiyellerini ölçebilmek için mutasyon yönteminden yararlanılmıştır. Bölüm 3'te tanımlanmış olan hata sınıflarından İRH, İOH, DOH, DRH, 0-T-H ve 1-T-H'a göre olası tüm mutantlar oluşturulmuştur. Mutantlar 1-hata, 2-hata ve 3-hata varsayımlarına göre yaratılmıştır. 1-hata varsayımı sistemde tek bir hata oluşabileceğini söyler. Bu durumda, İRH sınıfı için 3 mutant, İOH ve DOH sınıfları için 6 mutant, DRH sınıfı için 12 mutant, 0-T-H ve 1-T-H sınıfları için de 6'şar mutant olmak üzere toplam 33 mutant elde edilmektedir. Tablo 4 1-hata durumuna göre dört girişli sekiz farklı işlevin spektral katsayıları (s_0, s_1, s_2 ve s_3) ile her bir koşula bağlı test ikililerinin hata ortaya çıkarma potansiyellerini (c_0, c_1, c_2 ve c_3) göstermektedir. Bu değerlerle spektral katsayılar arasındaki r ile gösterilen ilinti, aşağıdaki korelasyon formülüyle hesaplanmıştır. Buna göre ilinti -1 ile +1

arasında değişmekte olup, -1 güçlü bir negatif ilişkiyi, +1 ise güçlü bir pozitif ilişkiyi ifade eder.

$$r = \frac{n\sum_{i=0}^{n-1} c_i s_i - \sum_{i=0}^{n-1} c_i \sum_{i=0}^{n-1} s_i}{\sqrt{\left(n\sum_{i=0}^{n-1} c_i^2 - (\sum_{i=0}^{n-1} c_i)^2\right)\left(n\sum_{i=0}^{n-1} s_i^2 - (\sum_{i=0}^{n-1} s_i)^2\right)}} \tag{17}$$

Tablo 4'e göre, r tanımsız olduğu 1 ve 8 numaralı işlevlerin dışında -1'e çok yakın olup koşullara ilişkin spektral katsayılar ile yine koşullara bağlı testlerin hata ortaya çıkarma potansiyelleri arasında kuvvetli bir negatif ilinti olduğunu söylemektedir. 1 ve 8 numaralı işlevlerdeki tüm koşullar denk olup spektral katsayılar birbirine eşittir. Bu durumda Formül 17'nin sonucu tanımsız olduğundan testler arasında bir önceliklendirme yapılamamaktadır.

Önceki bölümde ele alınan 3 numaralı işlev $f = (c_0 + (c_1 c_2))c_3$ için tablodan bakıldığında aşağıdaki ilişki görülebilir:

$$s_3 > s_0 > s_1 = s_2 \implies Pr(c_1) = Pr(c_2) > Pr(c_0) > Pr(c_3)$$

Buna göre test girişleri Tablo 3'te görüldüğü şekilde sıralanabilir.

Öncelik Değeri	c_0	c_1	c_2	c_3	$f(\mathbf{c})$
1	0	1	1	1	1
1	0	0	1	1	0
2	0	1	0	1	0
3	1	1	0	1	1
4	1	1	1	1	1
4	1	1	1	0	0

Tablo 3. $f(\mathbf{c})$ için sıralanmış test vektörleri

Tablo 5 ve 6 aynı hesaplamaların 2-hata ve 3-hata varsayımlarına göre sonuçlarını göstermektedir. 2-hata varsayımı sistemde aynı anda 2 hatanın olabileceğini söyler ve bu durumunda toplam 414 mutant yaratılabilir. 3-hata durumundaysa en fazla 2484 mutant yaratılabilmektedir. Negatif korelasyon her iki durumda da geçerliliğini korumuştur.

5 Sonuç

Bu çalışmada yazılım sınama için maskeleyen değiştirilmiş koşul/karar kapsama stratejisince üretilen testlere öncelik değerlerinin atanması için bir yöntem sunulmuştur. Önceliklendirme Fourier analizine dayanmaktadır ve öncelikler test ikililerinin hata ayıklama yeteneklerine göre verilmektedir. Önerilen yöntem sayesinde önceliklendirme doğrudan Fourier analiziyle hesaplanan spektral katsayılar üzerinden yapılabilmektedir. Yöntem, testlerin öncelik sıralamasına göre uygulanmasında ve test kümelerinin küçültülmesinde kullanılabilir.

No	İşlev	s_0	s_1	s_2	s_3	c_0	c_1	c_2	c_3	r
1	$f = (c_0(c_1c_2))c_3$	0.125	0.125	0.125	0.125	0.485	0.545	0.545	0.424	tanımsız
2	$f = (c_0(c_1 + c_2))c_3$	0.375	0.125	0.125	0.375	0.576	0.667	0.667	0.515	-0.943
3	$f = (c_0 + (c_1c_2))c_3$	0.375	0.125	0.125	0.625	0.576	0.667	0.667	0.364	-0.978
4	$f = (c_0 + (c_1 + c_2))c_3$	0.125	0.125	0.125	0.875	0.606	0.667	0.667	0.364	-0.980
5	$f = (c_0(c_1c_2)) + c_3$	0.125	0.125	0.125	0.875	0.606	0.667	0.667	0.273	-0.989
6	$f = (c_0(c_1 + c_2)) + c_3$	0.375	0.125	0.125	0.625	0.576	0.667	0.667	0.364	-0.978
7	$f = (c_0 + (c_1c_2)) + c_3$	0.375	0.125	0.125	0.375	0.485	0.667	0.667	0.424	-0.980
8	$f = (c_0 + (c_1 + c_2)) + c_3$	0.125	0.125	0.125	0.125	0.485	0.545	0.545	0.424	tanımsız

Tablo 4. Tek hata olması durumunda test ikililerinin hata ortaya çıkarma potansiyelleriyle spektral katsayıları arasındaki korelasyon (Toplam mutant sayısı: 33).

No	İşlev	s_0	s_1	s_2	s_3	c_0	c_1	c_2	c_3	r
1	$f = (c_0(c_1c_2))c_3$	0.125	0.125	0.125	0.125	0.792	0.855	0.855	0.700	tanımsız
2	$f = (c_0(c_1 + c_2))c_3$	0.375	0.125	0.125	0.375	0.800	0.872	0.872	0.703	-0.870
3	$f = (c_0 + (c_1c_2))c_3$	0.375	0.125	0.125	0.625	0.795	0.872	0.872	0.546	-0.962
4	$f = (c_0 + (c_1 + c_2))c_3$	0.125	0.125	0.125	0.875	0.816	0.874	0.874	0.575	-0.982
5	$f = (c_0(c_1c_2)) + c_3$	0.125	0.125	0.125	0.875	0.812	0.872	0.872	0.435	-0.991
6	$f = (c_0(c_1 + c_2)) + c_3$	0.375	0.125	0.125	0.625	0.792	0.874	0.874	0.548	-0.957
7	$f = (c_0 + (c_1c_2)) + c_3$	0.375	0.125	0.125	0.375	0.720	0.874	0.874	0.635	-0.957
8	$f = (c_0 + (c_1 + c_2)) + c_3$	0.125	0.125	0.125	0.125	0.802	0.862	0.862	0.713	tanımsız

Tablo 5. Çift hata olması durumunda test ikililerinin hata ortaya çıkarma potansiyelleriyle spektral katsayıları arasındaki korelasyon (Toplam mutant sayısı: 414).

No	İşlev	s_0	s_1	s_2	s_3	c_0	c_1	c_2	c_3	r
1	$f = (c_0(c_1c_2))c_3$	0.125	0.125	0.125	0.125	0.833	0.882	0.882	0.738	tanımsız
2	$f = (c_0(c_1 + c_2))c_3$	0.375	0.125	0.125	0.375	0.844	0.886	0.886	0.745	-0.796
3	$f = (c_0 + (c_1c_2))c_3$	0.375	0.125	0.125	0.625	0.836	0.882	0.882	0.716	-0.973
4	$f = (c_0 + (c_1 + c_2))c_3$	0.125	0.125	0.125	0.875	0.845	0.885	0.885	0.699	-0.977
5	$f = (c_0(c_1c_2)) + c_3$	0.125	0.125	0.125	0.875	0.843	0.884	0.884	0.537	-0.993
6	$f = (c_0(c_1 + c_2)) + c_3$	0.375	0.125	0.125	0.625	0.835	0.886	0.886	0.721	-0.980
7	$f = (c_0 + (c_1c_2)) + c_3$	0.375	0.125	0.125	0.375	0.853	0.894	0.894	0.763	-0.805
8	$f = (c_0 + (c_1 + c_2)) + c_3$	0.125	0.125	0.125	0.125	0.853	0.899	0.899	0.759	tanımsız

Tablo 6. Üç hata olması durumunda test ikililerinin hata ortaya çıkarma potansiyelleriyle spektral katsayıları arasındaki korelasyon (Toplam mutant sayısı: 2484).

Kaynaklar

1. Badhera, U., Purohit, G.N., Taruna, S.: Fault based techniques for testing boolean expressions: A survey. CoRR abs/1202.4836 (2012), http://arxiv.org/abs/1202.4836
2. Chilenski, J.J.: An investigation of three forms of the modified condition decision coverage (mcdc) criterion. Tech. rep., Office of Aviation Research (2001)
3. Elbaum, S., Malishevsky, A.G., Rothermel, G.: Prioritizing test cases for regression testing. SIGSOFT Softw. Eng. Notes 25(5), 102–112 (Aug 2000), http://doi.acm.org/10.1145/347636.348910
4. Jones, J.a., Harrold, M.J.: Test-suite reduction and prioritization for modified condition/decision coverage. IEEE International Conference on Software Maintenance, ICSM pp. 92–103 (2001)
5. Kelly J., H., Dan S., V., John J., C., Leanna K., R.: A practical tutorial on modified condition/decision coverage. Tech. rep. (2001)
6. O'Donnell, R.: Some topics in analysis of boolean functions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 569–578. STOC '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1374376.1374458
7. Rothermel, G., Untch, R.H., Chu, C., Harrold, M.J.: Test Case Prioritization: An Empirical Study pp. 1–10 (1999)
8. Rothermel, G., Untch, R.H., Chu, C., Harrold, M.J., Society, I.C.: Prioritizing Test Cases For Regression Testing Prioritizing Test Cases For Regression Testing (August), 102–112 (2001)
9. Srivastava, P.: Test case prioritization. Journal of Theoretical and Applied Information ... (December), 1–32 (2008)
10. Tassey, G.: The economic impacts of inadequate infrastructure for software testing. Tech. rep. (2002), http://www.nist.gov/director/prog-ofc/report02-3.pdf
11. Thayse, A., Davio, M.: Boolean differential calculus and its application to switching theory. IEEE Trans. Comput. 22(4), 409–420 (Apr 1973), http://dx.doi.org/10.1109/T-C.1973.223729
12. de Wolf, R.: A Brief Introduction to Fourier Analysis on the Boolean Cube. No. 1 in Graduate Surveys, Theory of Computing Library (2008), http://www.theoryofcomputing.org/library.html