論文 96-21-8-9

통신 프로토콜 시험항목의 오류 발견 능력 평가 방법

正會員 김 광 현*, 허 기 택**, 이 동 호***

Fault Coverage Evalulation Method of Test Case for Communication Protocol

Gwang-Hyun Kim*, Gi-Taek Hur**, Dong-Ho Lee*** *Regular Members*

요 약

통신 프로토콜의 적합성 시험은 구현된 프로토콜이 표준 규격과 동일하게 구현되었는지를 검사하는 과정을 말한다. 생성된 시험 항목이 어느 정도의 오류를 발견해 낼 수 있는지를 평가함으로써 적합성 시험의 효율성을 평가하는 하나의 기준으로 사용될 수 있다. 시험 항목의 오류 발견 능력의 평가 방법은 주로 수학적 평가 방법과 시뮬레이션을 이용한 연구가 이루어져 왔다.

본 본문에서는 기존 평가 방법의 문제점을 지적하고 오류 모델을 사용하여 생성된 시험항목에 대한 새로운 오류 발견 평가 모델을 제시하였다. 그리고 제안된 평가 모델을 기존의 방법과 비교, 분석하여 타당성을 입증하였다.

ABSTRACT

The conformance testing of communication protocol is the process to evaluate whether the protocol implemented is identified with standard specification. By evaluating how generated test cases detect many faults, it can be used with standard estimating efficiency of conformance testing. The method that evaluates the capability of fault coverage for test cases, has been researched by mathematical analysis and simulation.

In this paper, we pointed out the problem of existing method and proposed new evaluation model of fault coverage for test case which generated by fault model. Also, we analyzed the results comparing to the existing evaluation method and proved its validity.

*기전여자전문대학 사무자동화과
Dept. of Office Automation, Kijeon Women's Junior College
**동신대학교 전자계산학과
Dept. of Computer Science, Dong Shin University
***광운대학교 전자계산학과, 기초과학연구소
Dept. of Computer Science, Kwangwoon University
論文番號:96131-0429
接受日字:1996年 4月 29日

Ⅰ. 서 론

최근 급속한 정보통신 기술의 발달로 인하여 다양한 형태의 컴퓨터 통신망이 설계 구현되고 있으며 이를 이용한 서비스가 요구됨에 따라 여러 가지 정보통신 제품들이 개발되고 있다. 프로토콜을 구현함에 있어 이미 표준안이 제시된 프로토콜에 대해서도 서로 다른 형태의 독립된 구현 제품들이 가능하므로 이렇게 서로 다른 제품들간의 상호 운용성을 보장하기 위한 사전 절차로써 적합성 시험이 필요하게 된다. 적합성 시험의 가장 중요한 절차의 하나는 구현 프로토콜을 시험하기 위해 시험항목을 생성하는 것이다. 시험항목의 효율적인 생성에 관한 많은 연구가 진행되어 왔고[3, 4], 시험 목적에 부응하는 시험항목 생성에는 상당한 어려움이 존재한다. 시험항목의 기본적인 생성 방법은 프로토콜이 유한상태기계 모델로 표현된 각각의 천이에 대해 테스트 세그먼트를 생성, 이것을 조합하여 시험항목을 생성하게 된다[1, 4]. 구현 프로토콜의 오류 발견 방법은 표준 사양으로부터 생성된 시험항목을 구현된 프로토콜에 적용하고, 출력을 관찰하여 표준 사양으로부터 생성된 시험항목의 출력과 비교함으로써 오류를 찾아내게 된다. 현재까지 시험항목 생성에 관한 연구는 주로 시험항목의 크기를 최소화하는 방법에 집중되어 왔다[5, 6]. 그러나 앞으로는 최대의 오류 발견 능력을 갖는 시험항목을 생성하는 방법을 제시함으로써 효과적인 적합성 시험을 수행할 수 있게 될 것이다.

시험항목에 대한 오류 발견 능력의 평가 방법은 주로 수학적 평가 방법과 시뮬레이션 방법을 이용한 연구가 이루어져 왔다[3, 6, 7]. 본 본문에서는 기존 평가 방법의 문제점을 지적하고 오류 모델을 사용하여 생성된 시험항목이 어느 정도의 오류 발견 능력을 갖는지를 평가하는 새로운 방법을 제시하였다. 이로써 새로운 오류 발견 평가 모델은 효율적인 시험항목 생성에 대한 기준으로 이용될 수 있을 것이다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 시험항목의 오류를 검출해 내는 방법을 살펴보고, 3장에서는 생성된 시험항목으로 어느 정도의 오류 검출 능력을 갖는지를 평가하기 위한 방법과 문제점을 제시한다. 4장에서는 본 논문의 주된 연구 결과로 오류 모델에 따라 시험항목의 오류 평가 방법을 제안하고 분석 결과를 보여준다. 5장에서는 결론을 맺고 앞으로의 해결 과제를 제시한다.

Ⅱ. 시험항목의 오류 발견 방법

시험항목의 오류 검출 방법은 표준 사양으로부터 생성된 시험항목을 구현된 프로토콜에 입력 순서열을 적용하여 나타난 출력을 관찰하여 표준 사양으로부터 생성된 시험항목의 출력과 비교함으로써 오류를 찾아내게 된다. 프로토콜의 시험항목 생성시 구현에서 잘 발생할 수 있는 오류를 고려하여 시험항목을 생성하면 오류 검출 능력을 높일 수 있는 방법이 된다. 여기서는 적합성 시험의 기본이 되는 시험항목 생성 방법과 오류 발견 방법을 살펴보도록 한다.

2.1 FSM에 기초한 시험항목 생성

프로토콜은 정보를 처리하거나 복잡한 계산을 하기보다는 사용자 요구나 메시지의 도착 등과 같은 사건들에 대응하는 특성을 갖고 있으므로 통신 프로토콜의 제어 부분의 표현을 위해 유한상태기계(FSM : finite state machine) 모델이 주로 사용되어 왔다. FSM 모델에 적절한 제약을 가한 결정적 유한상태기계(DFSM : deterministic finite state machine) 모델이 시험 항목 생성을 위해 현재까지는 많이 사용되고 있다. 유한 상태 기계를 기초로 한 시험 항목 생성 방법은 프로토콜의 자료 부분중 출력값을 제외한 흐름만을 나타내므로 실제의 프로토콜에 적용하기에는 많은 문제점들을 내포하고 있다. 이러한 문제점이 존재하고 있지만 현실적으로 프로토콜을 명세하고 기술하는데 유한 상태 기계 모델을 사용하고 있다.

[정의 1] : 결정적 FSM 모델

결정적 FSM 모델은 6개의 쌍인 $\langle S, I, O, \lambda, \delta, S_o \rangle$으로 표현되는 transition 시스템이다.

여기서 S는 유한 상태의 집합, I와 O는 입력(input)과 출력(output)을 나타내는 유한 집합, δ는 상태천이(transition) 함수로 $S \times I \rightarrow S$, λ는 상태 출력 함수로 $S \times I \rightarrow O$로 정의되며 S0는 초기상태로 $S_0 \in S$이다. 결정적 유한 상태기계 모델의 상태 천이는 방향 그래프 $G = (V, E)$에 의해 결정적 FSM으로 표현될 수 있다. 여기서 V는 유한개의 non-empty인 버틱스의 집합이고 E는 에지의 집합을 나타내며 V_i에서 V_j로의 에이

지는 $(V_I, V_J; a_k/o_k)$로 표현될 수 있으며 a_k와 o_k는 각각 입력과 출력 동작을 나타낸다. 입력은 하위층과 상위층의 인터페이스로부터 받은 메세지이고 출력은 외부에서 관측 가능한 이벤트이며 null일 수도 있다. 프로토콜을 결정적 유한 상태 기계 모델로 표현된 예가 그림 1에 나타나 있다.

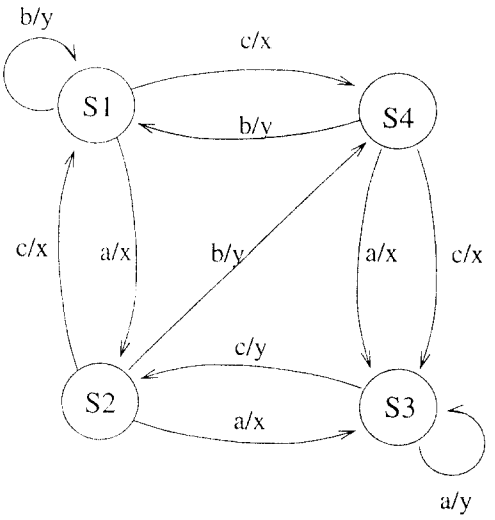

그림 1. 결정적 유한 상태 기계
Fig 1. Deterministic finite state machine

프로토콜 적합성 시험을 위해 시험항목 생성은 규격에서 정의된 FSM의 각 상태 천이에 대해 IUT의 동작을 확인할 수 있는 Checking experiment를 고려할 수 있는데, 이것은 FSM에서 임의의 시험대상이 되는 상태에서 각각의 입력에 대한 출력과 함께 천이까지도 올바르게 이루어졌는지를 확인하는 과정을 포함해야 한다. 현재까지 시험항목 생성 방법들은 주로 제어흐름을 쉽게 나타낼 수 있는 결정적 유한 상태 기계를 기초로 한 기법인 Transition Tour, W-method, Distingushing 순서, RCP(Rural Chinese Postman tour), UIO(Unique Input Output) 순서 등이 있다. UIO를 사용한 시험항목 생성 방법은 다른 상태에서는 발생되지 않아서 특정 상태를 유일하게 구분할 수 있는 입출력 순서의 쌍인 UIO 순서를 이용하여 시험항목을 생성하는 방법이다. 한 상태에 여러 개의 UIO 순서가 존재할 수 있으므로 이들 중에서 시험순서의 길이를 최소화할 수 있는 UIO를 선택하는 것이 바람직하다. UIO 방법을 사용한 시험 항목의 생성 과정은 다음과 같다.

1) 시스템을 시험하고자 원하는 상태로 이동시키는 순서열.
2) 입력과 출력의 검증.
3) 목적 상태를 확인하는 상태 signature.
4) 시스템을 초기 상태로 이동시키는 reset 입력.

2.2 시험항목의 오류 발견 방법

시험항목을 구현된 프로토콜에 적용하여 오류를 발견하기 위한 방법은 오류의 형태에 따라 약간 다를 수 있다. 오류 모델은 가장 잘 발생할 수 있는 오류의 형태를 규정함으로써 임의적으로 생성된 FSM에 적용하는 것보다 효율적인 오류 발견 방법으로 사용될 수 있다. 오류의 형태는 [12]에서 정의한 것처럼 보통 네가지로 분류할 수 있다.

1) 출력 오류(output faults): 구현 프로토콜(M_I)의 천이에서 발생하는 출력과 사양 머신(M_S)에서 정의하고 있는 출력이 서로 다른 경우를 말한다. 즉 $M_S\{\lambda(S_i, \alpha)\} \neq M_I\{\lambda(S_i, \alpha)\}$의 성질을 갖는 오류이다.

2) 천이 오류(transfer faults): 구현 프로토콜(M_I)에서 발생하는 천이가 사양 머신(M_S)에서 정의하고 있는 상태로 천이가 발생하지 않고 상태함수에 의하여 다른 상태로 천이가 발생하는 오류이다.

3) 상태 병합·분리오류(state merge and split fault): 구현 프로토콜(M_I)에서 발생하는 천이가 사양 머신(M_S)에서 정의하고 있는 상태에 대하여 하나 이상의 상태가 추가되거나 제거되는 경우이며, 한 상태에서 두 상태로 나누어지는 상태 분리 오류(state split fault)와 두 상태에서 하나의 상태로 합쳐지는 상태 병합 오류(state merge fault)가 있다.

4) 혼합 오류(hybrid faults): 구현 프로토콜(M_I)에서 발생하는 오류가 사양 머신(M_S)에서 정의하고 있는 상태 천이와 출력의 한 곳에서만 발생하지 않고 이 두가지 오류가 동시에 발생하는 경우를 말한다. 혼합 오류는 앞에서 정의한 오류 중에서 하나 이상이 복합적으로 발생하는 오류이다.

[12]에서는 발생 가능한 오류를 네가지로 분류하였고 오류 모델에 따라 오류 발견 방법이 다르다는 것을 보였다. 또한 [12]에서 정의한 오류 발견 함수를 이

용하여 시험 항목에 대한 오류를 검출하는 방법을 제시하였다.

그림 2은 오류 발견 과정을 간단하게 보여주고 있다. 표준 사양으로부터 생성된 시험 항목을 구현된 프로토콜에 입력 순서열을 적용하여 나타난 출력과 표준 사양으로부터 생성된 시험 항목의 출력을 비교함으로써 오류를 찾아내게 된다.

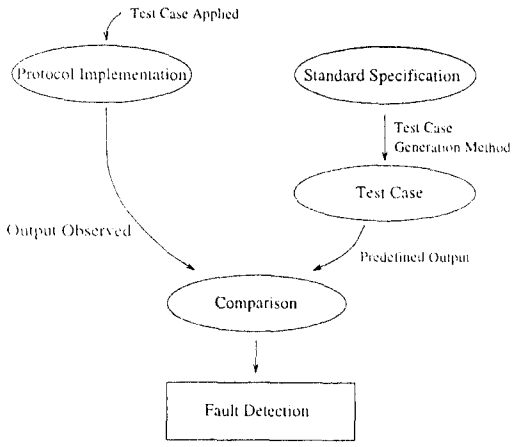

그림 2. 시험 항목의 오류 발견 과정
Fig 2. Fault detection procedure of test case

Ⅲ. 오류 발견 능력 평가 방법

생성된 시험항목이 어느 정도의 오류를 발견해 낼 수 있는지를 평가함으로써 적합성 시험의 효율성을 평가하는 하나의 기준으로 사용될 수 있다. 시험항목의 오류 발견 능력의 평가 방법은 주로 수학적 분석 방법과 시뮬레이션을 이용한 연구가 이루어져 왔다. 첫번째는 시험항목 생성과 관련한 여러 조건들을 고려하여 수학적인 분석을 통해 평가하는 방법이다[2, 8]. 두번째 방법은 시뮬레이션 기법을 이용하여 오류를 갖는 FSM을 생성하여 시험항목을 오류 FSM에 적용하여 통과한 FSM 수와 통과하지 못한 FSM을 계산하여 오류 발견 능력을 평가하는 방법이다[3]. 여기서는 두가지 기법을 간단하게 살펴보고 이들의 문제점을 제시하도록 한다.

3.1 수학적 평가 방법

실질적으로 프로토콜을 시험할 때 구현된 프로토콜의 상태 수는 정확하게 알려져 있지 않기 때문에 생성된 시험항목의 오류 발견 능력을 평가한다는 것은 매우 어려운 작업 중의 하나이다. 이는 기본적으로 시험항목의 수가 너무 많고, 프로토콜의 시험 상태 수가 많아질수록 시험 시간은 기하 급수적으로 증가하기 때문이다. 실제로 FSM으로 표현된 프로토콜을 시험할 때 시험되어야 할 FSM의 상태 수가 매우 많기 때문에 모든 상태를 시험한다는 것은 거의 불가능하다. FSM으로 표현된 프로토콜이 n개의 상태, m개의 입력, p개의 출력상태를 갖고 있으면 $(np)^{mn}$ 개의 시험 상태가 존재하게 된다. 현실적으로 이렇게 많은 FSM의 상태를 시험한다는 것은 불가능하므로 부분적인 시험 방법이 사용되고 있다. 이러한 제한적인 시험 때문에 시험항목의 오류 발견 능력을 정확하게 평가할 수는 없다[9]. 이러한 제약 조건을 고려한 방법으로 다음과 같은 조건들을 사용한 수학적인 평가 방법이 사용되고 있다. M_S을 사양 기계라 하고, TS는 테스트 스위트 그리고 **Impl**(m, Ms)는 사양을 구현한 것이라고 정의한다. 이것을 기본으로 하여 오류 발견 능력(FC: Fault Cverage)을 평가하기 위한 식은 다음과 같다.

$$FC(m, M_S, TS) = \frac{N_t(m, M_S) - N_p(m, M_S, TS)}{N_t(m, M_S) - N_c(m, M_S)}$$

- $\mathbf{N_t}(m, M_S)$: **Impl**(m, M_S)에 있는 기계의 총 수;
- $\mathbf{N_c}(m, M_S)$: M_S와 일치하는(conforming) **Impl**(m, M_S)에 있는 기계의 수;
- $\mathbf{N_p}(m, M_S, TS)$: 주어진 테스트 스위트를 통과한 **Impl**(m, M_S)에 있는 기계의 수;
- $\mathbf{N_t}(m, M_S) - \mathbf{N_c}(m, M_S)$: M_S와 일치하지 않는(not-onforming) **Impl**(m, M_S)에 있는 기계의 수;
- $\mathbf{N_t(m, M_S) - N_p(m, M_S, TS)}$: 주어진 테스트 스위트를 통과하지 못하는 **Impl**(m, M_S)에 있는 기계의 수;

시험항목의 오류 발견 범위는 0에서 1의 범위에 존재하고 오류 발견 능력이 1에 근접할수록 효율적인 시험항목이 생성되었다고 말할 수 있다. 이러한 확률적 정의는 모든 구현 프로토콜에서 발생 가능한 오류를 갖는 비적합한 구현을 찾아내는 것으로써 이해될

수 있으며, 오류 발견 능력이 최대가 되는 시험항목을 생성하여 적합성 시험의 효율성을 평가하는 기준으로 사용된다.

3.2 시뮬레이션에 의한 방법

일반적으로 프로토콜 적합성 시험은 상태 천이를 가능한 모두 시험해야 완전한 테스트가 이루어질 수 있다. 그러나 현실적으로 모든 상태를 시험항목으로 평가하는 것은 불가능함에 따라 수학적인 평가 방법과 함께 Monte Carlo 시뮬레이션 기법을 사용하여 오류 발견 능력을 평가하고 있다. 이 방법은 발생 가능한 오류의 형태를 10개의 클래스로 한정하여, 임의의 오류를 갖는 FSM을 생성하여 시험항목이 어느 정도까지 오류를 발견하는지를 평가하는 방법이다. 10개의 클래스로 오류의 형태를 적절하게 제한하는 이유는 생성 가능한 모든 구현 기계를 시험하는 것은 불가능하기 때문이다. 오류 클래스의 분류는 시험하고자 하는 FSM에 대해 출력과 천이 상태가 변경되도록 한다. 첫째, 출력 오류는 상태 천이가 발생할 때 나타나는 출력을 변경하여 오류 기계를 생성한다. 두 번째 상태 천이 오류는 상태 천이가 발생할 때 정상적인 천이가 아닌 다른 상태로 천이가 일어나도록 하여 오류 기계를 생성한다. 10개의 클래스로 분류하는 기준은 출력과 상태 천이 오류가 결합되어 하나 이상 발생하는 경우를 순서대로 분류한다.

앞에서 정의한 오류를 갖는 임의의 FSM를 사용하여 생성된 시험 항목으로 오류를 찾아내기 위한 절차는 다음과 같다.

1) FSM 사양을 읽어 들인다.
2) 사용될 시험항목을 읽어 들인다.
3) 위에서 정의한 10개의 클래스에 대하여 오류 FSM을 생성한다.
4) 단계 3에서 생성된 FSM에 시험항목을 적용한다.
5) 단계 4에서 테스트를 통과한 FSM은 오류가 없는 것으로 분류되고, 통과하지 못한 FSM은 오류가 있는 것으로 판단한다.

임의로 생성된 FSM에 대해 생성된 시험항목으로 오류를 발견하기 위해 다음과 같은 알고리즘을 적용한다. 상태 기계 M_1, M_2에 대하여 containment을 정의하여 equivalence를 검사함으로써 두 상태 기계가 일치하는지를 판정한다. 두 상태 기계 M_1, M_2의 출력 순서열이 M_1에 대한 입력 순서열과 동일하게 되면 M_2는 M_1를 contain 한다고 한다. is_1을 상태 기계 M_1의 초기 상태라 하고 is_2를 상태 기계 M_2의 초기 상태라고 한다. tail(e_1)와 tail(e_2)는 에지 e_1과 e_2로부터 도착하는 상태를 나타낸다.

알고리즘 3.1(오류 발견 알고리즘)

```
Set1 = { }
Set2 = {is1, is2}
while (Set2 ≠ { }) {
  Pick an element(s1, s2) ∈ Set2
  for (each outgoing edge e1 from s1) {
    if (there exists an outgoing edge e2 from s2
    with the same label as e1)
      tuple = {tail(e1), tail(e2)}
      if(~(tuple ∈ Set1) and ~(tuple ∈ Set2))
       Set2 = Set2 ∪ tuple
       Set2 = Set2 - {(s1, s2)}
       Set1 = Set1 ∪ {(s1, s2)}
      else M2 does not conform to M1
    }
  }
  M2 conforms to M1
```

위 알고리즘에서 FSM이 시험항목을 통과하면 그것이 사양 FSM과 equivalence를 검사하여 올바르게 구현되었는지를 평가한다. 즉 생성된 FSM이 equivalence 하지 않으면 시험항목으로 사양 FSM과 구현 FSM을 구별하지 못하게 된다. 임의로 생성된 상태 기계의 상당수는 오류 기계가 아닐 수 있으므로 오류 발견 능력의 평가는 실제 오류에 대하여 발견된 오류 비율로 표시할 수 있다. 여기에서 r을 오류 머신의 갯수, c는 사양과 동일하다고 하다고 판단된 상태 머신의 갯수, 그리고 d를 시험 항목에 의해 오류라고 판단된 상태 머신의 갯수라 하면 오류 발견 능력은 d/(r-c)로 정의할 수 있다[6].

오류를 갖는 FSM의 수(Nf)를 계산하기 위해 다음과 같은 수식을 사용한다. S개의 상태와 I개의 입력, O개의 출력을 갖는 FSM은 SI개의 에지를 갖게 된다. 위에서 정의했던 각 클래스별 오류를 갖는 FSM의 수는 다음과 같다. 여기서 C는 조합의 갯수, P는 순열의

갯수를 의미한다.

클래스 1 : $Nf = SI \times (O-1)$
클래스 2 : $Nf = SI \times (S-1)$
클래스 3 : $Nf = C(SI, 2) \times (O-1)^2$
클래스 4 : $Nf = C(SI, 2) \times (S-1)^2$
클래스 5 : $Nf = P(SI, 2) \times (O-1) \times (S-1)$
클래스 6 : $Nf = SI \times (O-1) \times (S-1)$
클래스 7 : $Nf = C(SI, 2) \times (O-1)^2 \times (S-1)^2$
클래스 8 : $Nf = C(SI, 2) \times C(3, 1) \times (S-1) \times (O-1)^2$
클래스 9 : $Nf = C(SI, 4) \times C(4, 2) \times (S-1)2 \times I(O-1)^2$
클래스 10 : $Nf = C(SI, 5) \times C(5, 3) \times (S-1)^3 \times (O-1)^2$

Ⅳ. 오류 모델을 이용한 평가 모델

프로토콜 시험항목에 대한 성능 평가의 두가지 측면은 시험항목이 시험 시스템에서 얼마나 빨리 수행될 수 있는가(efficiency)와 생성된 시험항목에 의해 얼마나 많은 오류를 찾아낼 수 있느냐의 문제(effectiveness)를 다루는 것으로 구분할 수 있다. 현재까지 시험항목 생성에 관한 모든 연구는 최소의 길이를 갖는 시험항목의 생성에 집중되어 왔다[11]. 본 논문에서는 최대의 오류 발견 능력을 갖는 시험항목의 생성을 위해 사용되는 새로운 평가 방법을 제시하도록 한다.

4.1 오류 모델을 이용한 평가 방법

프로토콜이 구현될 때 여러가지 이유로 인하여 구현 오류가 발생하게 된다. 발생된 구현 오류를 모두 시험한다는 것은 현실적으로 불가능하므로, 오류 모델을 사용하여 발생 가능한 오류를 적절하게 제한하게 되면 적합성 시험의 효율성이 증가하게 된다. 오류 발견 능력의 정확한 평가에 관한 기준은 아직까지 알려져 있지 않다. 3장에서 기술했던 두가지 평가 방법은 실질적인 적용 가능성과 정확한 평가 기준으로는 사용하기에는 부족하다. 본 논문에서 제안하고 있는 오류 모델에 따라 각각의 평가 기준을 정의하면 오류에 따른 정확한 평가 기준을 제공할 수 있다. 새로운 평가 모델을 제시하기 전에 오류 모델에 따른 시험항목 생성의 장점을 살펴보면 다음과 같다.

1) 모든 시험항목은 잘 정의된 시험 목적을 갖고 있다.
2) 시험항목을 찾는 complexity를 감소시킨다.
3) 사용자가 중요한 오류를 선택할 수 있다.
4) 고정된 오류 모델을 사용한 기존의 방법보다 상당한 융통성을 갖고 있다.

표준으로 정의한 프로토콜 사양의 구현 오류에 대한 오류 발견 능력의 정확한 측정은 어렵더라도 오류 발견 능력에 대한 다양한 방법들이 사용되고 있다. 3장에서 기술한 수학적인 분석 방법과 시뮬레이션을 통한 기법이 현실적으로 사용되고 있는 대표적인 오류 평가 방법이다. 본 논문에서는 구현 오류의 정확한 평가를 위하여 새로운 방법으로 오류 모델을 이용한 평가 방법을 제시한다. 오류 평가 방법은 [12]에서 정의한 오류 모델 4가지에 따라 서로 다르게 정의된다. 본 연구에서는 주로 발생 가능한 출력 오류와 상태 천이 오류에 대한 오류 평가 방법을 제안한다.

4.1.1 출력 오류의 오류 발견 능력 평가 방법

[12]에서 정의하고 있는 첫번째 모델에 대한 오류는 오류 발견함수를 사용하여 출력 오류를 발견한다. 출력 오류에 대한 오류 발견 함수는 $Ms\{\lambda(S_i, \alpha)\} = M_I\{\lambda(S_i, \alpha)\}$를 만족하면 오류 미발견 상태가 되고, $M_S\{\lambda(S_i, \alpha)\} \neq M_I\{\lambda(S_i, \alpha)\}$가 되면 출력 오류를 발견했다고 할 수 있다. 출력 오류의 평가 방법은 3.2에서 제시한 알고리즘을 이용하여 기대된 출력과 적용된 결과를 비교하여 오류 발견 능력을 평가하면 된다. 예를 들면, 3.2절에서 분류한 오류 클래스에서 단순하게 하나의 출력 오류를 갖는 경우(클래스 1)에 임의의 FSM을 생성하여 시험항목이 오류를 찾아내는 FSM 수와 발견하지 못하는 갯수를 평가하면 된다.

4.1.2 상태 천이 오류의 오류 발견 능력 평가 방법

두번째 오류 모델인 상태 천이 오류에 대한 오류 발견 능력을 평가하기 위해서는 상태 천이의 오류 발견 함수를 사용하여 천이 오류를 찾아내야 한다. 상태 천이 오류를 찾기 위해 출력 오류와 비슷한 방법으로 구현 프로토콜에 시험항목을 적용하여 관찰되는 출력을 이용하게 된다. 상태 천이 오류(transfer fault)를 찾기 위해서는 DFSM의 한 상태(S_i)에서 다른 상태(S_k)로 출발하는 에지의 집합에서 하나의 천이($T_{m,n}$)를 선택한다. 선택된 천이에 대해 올바르게 다음 상태(NextStates)로 이동되는 것을 확인하기 위해 시험 스위트(TS) 중에서 시험항목(tc)를 적용하여 기

대된 출력과 비교함으로써 천이 오류가 발견된다. 다음은 DFSM으로 정의된 프로토콜에 대해 천이 오류를 발견하기 위한 알고리즘이다.

알고리즘 4.1(transfer fault finding 알고리즘)

```
Transferfaultfind( )
  for (tc_m ∈ TS) {
    for (T_m,n ∈ tc_m) {
      for (T_k in SIT ) {
        EndSatesk = 0;
        for (s ∈ S and s != NextStates(T_k)) {
            flag = true; /* transitions are correct */
            if (T_m,n == T_k) THEN
              if O(s, I_m,n+1 != O_m,n+1) {
                  flag = false; /* transitions are fault */
                  exit;
              }
        if (flag == true) then
            EndStatesk = EndStatesk ∪ {s};
        }
      }
  }
```

다양하게 발생할 수 있는 상태 천이 오류를 발견하기 위해서 상태 확인 트리(State Identification Tree)를 정의한다. 상태 확인 트리(SIT)는 시험항목 구성시 사용되는 상태 확인 부분을 트리 구조 형태로 표현함으로써 상태 천이를 쉽게 확인할 수 있도록 한다. 상태 확인 트리를 정의하기 위해 다음과 같은 개념이 필요하다.

[정의 4.1]

$\delta(S_1, \alpha)=S_i$이고 $\delta(S_1, \alpha x)=S_j$를 만족하면 M_S에서 천이 $\langle S_i; i/o; S_j \rangle$는 TS에 의해 포함된다고 한다.

여기서 α, x는 시험항목의 일부분이다.

[증명]

상태 S_1에 입력 순서열 α를 적용했을 때 S_i 상태로 천이가 발생했고 또한 상태 S_1에서 S_j로의 천이는 상태 S_i에서 입력 순서 x를 추가하여 발생했다. 여기서 x란 입력 순서열을 i/o로 표현한 것이다. 즉 상태 S_i에서 S_j로 천이는 초기 상태 S_1에 입력 순서열 α를 적용한 후 다시 x(i/o)를 적용하면 된다.

[정의 4.2]

$\delta(S_1, \alpha)=S_i$ $\delta(S_1, \alpha x)=S_j$, $\delta(S_1, \beta)=S_k$ 그리고 $\lambda(\delta(S_1, \alpha x), \gamma) \neq \lambda(S_1, \beta), \gamma)$를 만족하면 Ms에서 천이 $\langle S_i; i/o; S_j \rangle$의 tail 상태 S_j는 TS(Test Suite)에 의해 다른 상태 S_k와 식별된다고 한다. 이것을 Tail_Dis라고 한다. 여기서 α, β, γ는 테스트 스위트(TS)에 속하는 시험항목의 일부분이다.

[증명]

상태 천이 함수 δ는 상태 S_i에 입력 순서 α를 적용하여 상태 S_i으로 천이가 이루어지는 것은 정의 4.1과 동일하다. $\lambda(\delta(S_1, \alpha x), \gamma) \neq \lambda(\delta(S_1, \beta), \gamma)$에서 $\delta(S_1, \alpha x)$는 S_j가 되고 여기에 입력 순서열 γ를 적용했을 때 출력값과 $\delta(S_1, \beta)$를 수행했을 때 천이는 S_k가 되고 여기에 입력 순서열 γ를 적용했을 때의 출력값이 서로 다르게 되면 S_j는 S_k와 서로 다른 상태에 있게 된다.

[6]에서는 미발견 오류를 찾아내기 위해 상태를 추가하여 시험항목을 다시 생성하여 오류를 발견하는 방법을 제시하고 있다. 그러나 이 방법은 상태를 추가한 만큼 시험항목의 크기가 증가됨에 따라 시험 비용과 수행 시간이 길어지는 단점을 갖고 있다. 이러한 문제점을 극복하기 위해 다양한 형태로 발생하는 프로토콜 오류를 발견하기 위해서 상태 확인 트리(SIT: state identification tree)개념을 이용하도록 한다.

표 1. 그림 1에 대한 테스트 세그먼트
Table 1. Test segment for Fig. 1

transition	test segment
$T_1(S_1 \rightarrow S_1)$	b/y c/x a/x c/y
$T_2(S_1 \rightarrow S_4)$	c/x c/x c/y
$T_3(S_4 \rightarrow S_1)$	b/y c/x a/x c/y
$T_4(S_2 \rightarrow S_1)$	c/x c/x a/x c/y
$T_5(S_1 \rightarrow S_2)$	a/x b/y a/x c/y
$T_6(S_2 \rightarrow S_4)$	b/y c/x c/y
$T_7(S_2 \rightarrow S_3)$	a/x c/y
$T_8(S_4 \rightarrow S_3)$	c/x c/y
$T_9(S_3 \rightarrow S_2)$	c/y b/y a/x c/y
$T_{10}(S_4 \rightarrow S_3)$	a/x c/y
$T_{11}(S_3 \rightarrow S_3)$	a/y c/y

상태 확인 트리를 구성하기 위해서는 정의[4.1]과 [4.2]를 이용하여 각 상태 천이에 대한 확인 과정을 거친다. 그리고 각 상태 천이에 대한 테스트 세그먼트(test segment)를 구하여 부분적인 천이 확인에 이용되도록 한다. 테스트 세그먼트는 상태 확인을 위해 기본적으로 사용되는 UIO 순서와 천이에 대한 레이블을 결합함으로써 구성된다. 그림 1에 대한 테스트 세그먼트는 표 1에 나타낸 것과 같다.

상태 확인 트리를 구성하기 위해서는 먼저 각 상태에 대한 테스트 세그먼트를 구한다. 초기 상태에서 다음 상태의 천이에 대한 테스트 세그먼트를 선택하여 초기 상태에 선택된 테스트 세그먼트를 추가한다. 이러한 과정을 모든 상태에 도달할 때까지 반복한다. 다음 알고리즘은 오류 발견을 위한 상태 확인 트리를 구성하는 과정을 간단하게 기술하고 있다.

알고리즘 4.2(상태 확인 트리 구성 알고리즘)

Step 1 : Generate test segment for transition of each state;

Step 2 ; Select test segment of transition from initial to next state;
Attach selected test segment to initial state;

Step 3 : Repeat Step 2 until select test segment for all state;

Step 4 : Check cycle for attached test segment;
If it exists cycle then fault occurs else fault not occurs;

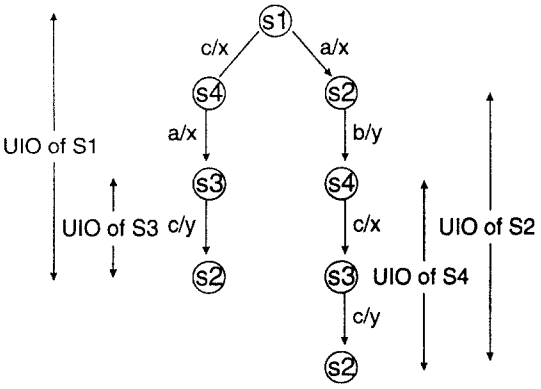

그림 3. 그림1에 대한 상태 확인 트리
Fig 3. State Identification Tree for fig. 1

그림 1의 DFSM을 상태 천이 〈S1;b/y;S1〉가 상태 천이 오류 〈S1;b/y;S3〉를 갖도록 변경했을 때의 상태 확인 트리는 구조는 그림 4와 같다. 그림 4에서 상태 천이 오류를 발견하는 방법은 상태 확인 트리 생성 알고리즘에서 사이클이 존재하게 되면 상태 천이 오류가 발생했다고 한다. 즉 상태 천이 $S_1 \rightarrow S_3 \rightarrow S_1 \rightarrow S_3$는 사이클이 생성되었음을 알 수 있다. 그러므로 그림 1의 상태 기계 〈S1;b/y;S1〉를 〈S1;b/y;S3〉를 변경하였을 때의 상태 천이 오류를 발견하게 된다. 그러나 상태 천이 오류를 발견하도록 생성된 시험 항목이 출력 오류와 결합되어 천이 오류가 발생되는 경우에 천이 오류를 발견하지 못하는 경우가 존재한다. 이러한 경우는 앞으로 계속적인 연구가 필요한 부분이다.

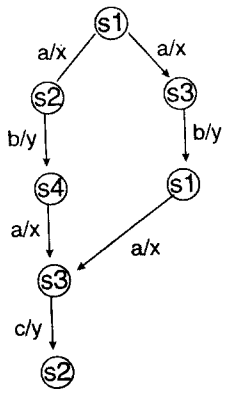

그림 4. 천이 오류를 갖는 상태 확인 트리
Fig 4. State identification tree with transfer fault

4.2 평가 모델 분석

[6]에서는 미발견 오류를 찾아내기 위해 상태를 추가하여 시험항목을 다시 생성하여 오류를 발견하는 방법을 제시하고 있다. 그러나 이 방법은 상태를 추가한 만큼 시험항목의 크기가 증가됨에 따라 시험비용과 수행시간이 커지는 단점을 갖고 있다. 오류 모델에 따라 제안된 평가 방법을 분석함으로써 시험 항목 생성에 효율적인 기준을 제공하는 기준으로 사용될 수 있는지를 분석한다. [정의 1]에서 정의하고 있는 DFSM 모델의 구성요소에 따라 다음과 같은 전제조건을 정의한다.

[정의 4.3]

tail 상태 S_i와 S_k를 상태 천이 $\langle S_i; i/o; S_j \rangle$에 식별될 수 없을때 Tail_NDis라 하고, Tail_NDis($\langle S_i; i/o; S_j \rangle$, TS)는 $\{S_1, S_2, ..., S_n\}$-Tail_Dis($\langle S_i; i/o; S_j \rangle$, TS)와 같다.

[증명]

tail 상태 S_i와 S_k를 입력 순서열 α, β, γ에 의해 두 상태를 구별하지 못하면 동일한 상태로 인식하게 된다. 즉 $\lambda(\delta(S_1, \alpha x), \gamma) = \lambda(\delta(S_1, \beta), \gamma)$와 같은 관계를 갖게 된다. 그러므로 Tail_NDis($\langle S_i; i/o; S_j \rangle$, TS)는 $\{S_1, S_2, ..., S_n\}$-Tail_Dis($\langle S_i; i/o; S_j \rangle$, TS)와 같게 된다.

[정리 4.3]으로부터 손쉽게 Tail_Dis와 Tail_NDis 수를 계산할 수 있다.

- $|\text{Tail\_NDis}(\langle S_i; i/o; S_j \rangle, TS)| \times |O| = n|O|$
- $M_t(M_S) = |Impl(Ms)| = (n|O|)^{n|I|}$: 구현 기계의 전체 수.
- $N_c(M_S) = (n-1)!(n|O|)^{n|I|-|D_s|}$: Ms와 적합한(conforming) 구현 기계 수.

3.1의 수학적 평가 방법에서 FC의 값을 구하기 위해서 $N_p(M_S, TS)$의 정확한 값을 필요로 한다. 이값을 계산하기 위해서는 TS가 사양 기계에 대해 통과하는지를 시험하여야 하는데 이는 현실적으로 너무 많은 비용이 들어 불가능하다. 그러므로 N_p를 $\overline{N_p}$로 대치하여 식 4.1과 같은 새로운 평가식을 생성한다.

$$FC_e(M_S, TS) = \frac{N_t(M_S) - \overline{N_p}(M_S, TS)}{N_t(M_S) - N_c(M_S)} \quad (4\text{-}1)$$

식 (4-1)에서 M_S에 적합한 구현 머신($N_c(M_S) = (n-1)!(n|O|)^{n|I|-|D_s|}$)과 테스트 스위트를 통과한 구현 기계 중에서 큰값을 선택하면 식 (4-2)가 된다.

$$FC_e(M_S, TS) = \frac{N_t(M_S) - \max(N_c(M_S), \overline{N_p}'(M_S, TS))}{N_t(M_S) - N_c(M_S)} \quad (4\text{-}2)$$

식 (4-3)은 3.1절의 평가식에서 상용 대수를 취한 식으로써 적절한 오류 발견 능력의 평가식으로 사용된다.

$$FCo(m, M_S, TS) = \frac{\log(N_t(m, M_S) - N_p(m, M_S, TS))}{\log(N_t(m, M_S) - N_c(m, M_S))} \quad (4\text{-}3)$$

일반적으로 시험항목의 길이와 오류 발견 능력 사이에는 trade-off 관계를 갖고 있다. 어느정도까지 시험항목의 길이를 증가시켜 최대의 오류 발견 능력을 갖는지를 평가하는 기준이 필요하게 된다. 시험 환경에 따라서 시험항목의 길이를 증가시켜도 시험 수행에 영향을 미치지 않으면 최대의 오류 발견 능력을 갖도록 시험항목의 길이를 증가시키는 것이 필요하다. 그렇지만 어느 정도까지 시험항목의 길이를 증가시켜야 fault converage를 만족 시키면서 효율적인 적합성 시험이 될 것인가를 정량화될 수 있는 방법이 필요하다.

V. 결 론

적합성 시험이 효율적으로 이루어지기 위해서는 최소 길이의 시험항목 생성과 오류 검출 능력이 최대가 되는 시험항목을 생성하는 방법이 필요하다. 현재까지는 주로 오류를 효율적으로 찾아낼 수 있는 시험항목에 대한 연구는 많았지만 이를 정량화하여 수식으로 표현하고 분석하는 연구는 상대적으로 부족하였다. 본 논문에서는 생성된 시험항목에 대하여 어느 정도의 오류 발견 능력을 갖는지를 평가할 수 있는 모델을 제안하였고, 제안된 모델이 어느 정도 오류 발견 능력 평가를 정확하게 할 수 있는지 수식을 통해 이를 분석하였다. 적합성 시험 과정에서 시험항목의 크기와 오류 발견 능력은 비례함을 알 수 있었다. 따라서 오류발견 능력도 최대가 되고 시험항목의 길이가 최소가 되는 시험항목 생성 방법이 가장 이상적이라고 할 수 있다. 앞으로 시험 비용과 오류 발견 능력과의 상관 관계에 대한 정확한 연구가 필요하다.

참 고 문 헌

1. David Lee, K. Sabnani, David M. Kristol, Sanjoy Paul, "Conformance Testing of Protocols Specified as Communicating FSMs", IEEE INFOCOM'93, pp. 115-127, 1993.
2. G. v. Bochmann, A. Petrenko and M. Yao, "Fault Coverage of Tests Based on Finite State Models", IWPTS VII, pp. 55-74, 1994.
3. A. Petrenko, N. Yevtushenko, R. Dssouli, "Testing

Strategies for Communicating FSMs", IWPTS VII, pp. 181-196, 1994.
4. K. Naik and B. Sarikaya, "Testing Communication Protocols", IEEE Software pp. 27-37, Jan. 1992.
5. D. P. Sidhu, H. Motteler and R. Vallurupalli, "On Testing Hierarchies for Protocols", IEEE/ACM Transactions on Networking, Vol. 1, NO. 5, October 1993.
6. F. Lombardi and Y. U. Shen, "Evaluation and Improvement of Fault Coverage of Conformance Testing by UIO Sequences," IEEE Trans. Communications, Vol. 40, No. 8, pp. 1288-1293, August 1992.
7. Kshirasagar Naik, "Fault-tolerant UIO Sequences in Finite State Machines", IWPTS VIII, pp. 207-220, 1995.
8. Raymond E. Miller, Sanjoy Paul, "Structural Analysis of Protocol Specifications and Generation of Maximal Fault Coverage Conformance Test Sequences", IEEE/ACM Trans. on Networking, VOL. 2, NO. 5, pp. 457-470, 1994.
9. M. Yao, A. Petrenko and G. v. Bochmann, "A Structural Analysis Approach to the Evaluation of Fault Coverage for Protocol Conformance Testing", FORTE'94, pp. 389-404.
10. T. Ramalingam, Anindya Das and K. Thulasiraman, "Fault detection and diagnosis capabilities of test sequence selection methods based on the FSM model", Computer Communications, Vol. 18, NO. 2, pp. 113-122, February 1995.
11. T. Ramalingam, Anindya Das and K. Thulasiraman, "On testing and diagnosis of communication protocols based on the FSM model", Computer Communications, Vol. 18, NO. 5, pp. 329-337, May 1995.
12. 김광현, 이동호 "오류 검출 능력을 갖는 효율적인 시험 항목 생성 방법", 한국 통신학회 논문지, VOL. 20, NO. 12, pp. 3540-3551, 1995.

김 광 현(Gwang-Hyun Kim) 정회원
1989년 2월 : 광운대학교 전자계산학과 졸업(이학사)
1991년 2월 : 광운대학교 대학원 전자계산학과 졸업(이학석사)
1992년 3월~현재 : 광운대학교 대학원 전자계산학과 박사과정
1991년 3월~현재 : 전주 기전여자전문대학 사무자동화과 조교수
※주관심 분야 : 컴퓨터 네트워크, 초고속 통신망, 분산 시스템

허 기 택(Gi-Taek Hur) 정회원
한국통신학회 논문지 제 20권 3호 참조

이 동 호(Dong-Ho Lee) 정회원
한국통신학회 논문지 제 *20*권 *3*호 참조