論文 95-12-12-27

오류검출 능력을 갖는 효율적인 시험항목 생성 방법

正會員 金 光 鉉*, 李 東 豪**

An Effective Test Case Generation Method with Fault Detection Capability

Gwang-hyun Kim*, Dong-ho Lee** Regular Members

이 논문은 1994년도 한국과학재단의 연구비 지원에 의한 연구 결과임

要 約

통신 프로토콜의 적합성 시험은 상호 운용성과 효율적인 컴퓨터 통신을 위해서 매우 중요하다. 통신 프로토콜 명세가 주어진 경우에, 구현된 프로토콜과 주어진 명세사이의 일치 여부를 검증하는 과정을 적합성 시험이라 한다. 통신 프로토콜의 적합성 시험이 효율적으로 이루어지기 위해서는 시험 항목이 효율적으로 생성되어야 한다. 본 논문에서는 프로토콜 구현에서 주로 발생할 수 있는 오류를 네 가지로 분류하였고, 네가지 오류 모델에 따라 효율적 시험항목 생성을 위하여 오류 발견 함수를 정의하였다. 기존의 시험항목 생성 방법으로 오류를 발견하지 못하는 경우를 제시하였다. 또한 오류 발견함수를 사용한 방법은 미발견 오류를 찾아냄으로써 기존의 시험항목 생성 방법보다 오류 검출 능력이 향상된 시험 항목을 생성할 수 있다.

ABSTRACT

Protocol conformance testing is crucial to interoperability and effective computer communication. Given a protocol specification, the task of checking whether a protocol implementation conforms to the specification is called as conformance testing. For efficient conformance testing of communication protocol, test cases are generated efficiently. In this paper, we classified four types fault-model compromising in protocol implememtation and defined fault-detection function for effective test case generation according to fault model. And we presented fault which is not detected from existing test case generation method. Also the proposed method is able to generate test cases with high fault detection capability than existing test case generation method as detecting undetected faults.

*기전여자전문대학 사무자동화과
**광운대학교 전자계산학과
論文番號 : 95267-0808
接受日字 : 1995年 8月 8日

Ⅰ. 서 론

정보 기술의 발전으로 컴퓨터 통신과 분산처리가 가능하게 되었다. 그 결과 컴퓨터 통신망이 설계 구현되었고 이를 이용한 다양한 서비스가 제공되고 있다. 1980년대에는 전자기술과 정보기술의 급격한 발전으로 인하여 컴퓨터 통신과 분산처리 분야에 많은 영향을 받게된다. 그 결과로 인하여 1990년대 초에 새로운 통신분야인 High-Speed LAN, 멀티미디어 통신, ISDN, 다중통신, 데이타베이스/트랜잭션처리를 위한 응용 등 다양한 분야가 등장하게 된다. 이러한 다양한 서비스를 제공하기 위해서는 컴퓨터들 사이의 구조와 프로토콜이 정의되어 있어야 하며 이를 위해 ISO(International Standard Organization)에서는 OSI/RM(Open Systems Interconnection Reference Model)의 7계층 모델을 정의하였다. 7 계층 구조는 하위 계층에서 상위 계층으로 peer-to-peer 프로토콜에 의해 정보를 교환함으로써 지정된 서비스를 제공하는 계층적 구조이다. 물리적으로 분리된 두 컴퓨터 사이에 정보를 순서적으로 교환하려면 통신 요소간의 상호작용을 제어할 수 있는 일련의 규칙들이 필요하다. 이러한 규칙들을 총체적으로 통신 프로토콜이라 한다. 프로토콜은 컴퓨터 통신망에서 중요한 역할을 담당하며 컴퓨터 통신망 구축의 초석이 된다. 통신 프로토콜을 설계할 때 프로토콜 자체가 비정형화된 형태로 서술됨으로써 발생할 수 있는 모호성을 제거하고 신뢰성을 향상시키기 위해 형식 기술언어(FDT:Formal Description Technique)를 사용하는 것이 효율적일 것이다. 그러나 프로토콜 명세가 형식 명세 기법으로 명세되었다고 할지라도 여러 제조업자에 의해서 제공된 많은 정보통신 제품들 간의 상호운영이 가능해야만 신뢰성 있는 통신망이 구축될 수 있다. 효율적인 통신망 구축을 위해서 프로토콜 명세에서부터 구현시까지 발생될 수 있는 모든 문제점들을 제거하기 위해서는 통합적인 프로토콜 개발 과정이 필요하다[1,6]. 국제적으로 표준화된 통신 프로토콜은 매우 중요하기 때문에 표준 그 자체에 결함이 존재하지 않아야 하고 상호간에 서비스가 가능하도록 구현되어야 한다. 프로토콜을 구현함에 있어서 이미 표준안이 제시된 프로토콜에 대해서도 여러 가지의 서로 다른 형태의 독립된 구현 제품들이 가능하므로, 이렇게 서로 다른 제품들 간의 상호 운용성을 보장하기 위한 시험 절차가 필요하다. 형식 명세 언어를 사용하여 프로토콜이 서술되고 구현되었다고 할지라도 구현된 프로토콜이 표준에 맞도록 구현되었는지를 확인하는 적합성 시험 단계를 거쳐야만 한다. 적합성 시험은 OSI 통신 프로토콜의 표준화가 완료되고, 구현이 진행되어 감에 따라 최근에 연구가 집중되고 있는 분야이다[9,12]. 실제 프로토콜은 너무 방대하고 엔티티 사이에 다양한 상호작용을 하고 있기 때문에 모든 상태를 시험한다는 것은 불가능하다. 이러한 프로토콜 시험의 복잡성 때문에 프로토콜의 적합성 시험을 위한 시험항목 생성에 많은 시간이 필요하게 되고 시험목적에 부응하는 시험항목 생성에 어려움이 따르게 된다. 프로토콜의 시험은 구현 프로토콜의 반응을 관찰하여 표준사양으로부터 생성된 시험 항목의 출력을 비교함으로써 오류를 찾아내게 된다. 본 논문에서는 위에서 제기한 여러 문제점을 해결하기 위해 발생 가능한 오류 형태를 정의하고 각 오류의 형태에 따라 가장 효율적으로 오류를 찾아낼 수 있는 시험 항목을 생성하고자 한다. 또한 정의된 오류모델에 의하여 생성된 시험항목을 구현 프로토콜에 적용하여 얼마나 많은 오류를 찾아낼 수 있는가를 시험하게 된다. 오류모델을 결정적 유한 상태기계로 나타내어 다양한 오류를 잘 표현하도록 하여야 하며, 특히 오류모델은 구현과는 완전히 독립적으로 정의하고 현실적으로 프로토콜을 구현할 때 가장 잘 발생할 수 있는 오류를 고려하여 시험항목을 생성하면 오류 검출 능력을 높일 수 있는 하나의 방법이 될 것이다. 본 논문에서는 프로토콜 구현에서 주로 발생할 수 있는 오류를 네 가지로 분류하고 이러한 오류 모델에 따른 시험 항목 생성 방법을 제안하고, 높은 오류 검출 능력을 갖는 오류 발견 함수를 정의한다. 그리고 제안된 방법이 주어진 오류로부터 효율적인 시험 항목을 생성할 수 있음을 보이도록 한다.

본 논문는 다음과 같이 구성된다. 2장에서는 적합성 시험에 관한 일반적인 개념을 살펴보고, 3장에서 프로토콜 구현에서 잘 발생하는 오류를 표현하기 위하여 구현 오류를 4가지 형태로 분류하고, 정의된 오류 모델에 따라 오류 발견 함수를 정의하여 효율적인 시험항목을 생성하는 방법을 제시한다. 4장에서는 기존의 방법으로 오류를 찾아내지 못하는 미발견 오류를 살펴보고, 오류 발견 함수를 사용하여 적용사례를 분석한다. 5장에서는 연구된 결과에 대한 결론을 맺고 앞으로의 해결 과제를 제시한다.

Ⅱ. 적합성 시험

적합성 시험의 관점에서 구현이란 해당 표준안에서 명시한 요구사항들을 실현한 하드웨어와 소프트웨어의 제품을 말한다. 통신 프로토콜을 구현할 때 표준안에서 제시된 프로토콜에 대해서도 여러 가지의 서로 다른 형태의 독립된 구현 제품들이 가능하므로, 이렇게 서로 다른 제품들간의 상호 운용성을 높여주기 위한 시험 절차가 필요하게 된다. 적합성 시험에서는 형식 기술 언어로 묘사된 프로토콜의 동작을 분석하여 구현 프로토콜의 외부 행위가 프로토콜 명세에서 정의된 행위와 일치하는 것을 검증하고, 구현된 프로토콜이 표준에서 정의한 것과 똑같이 동작하면 명세와 일치한다고 말한다. 또한 모든 경우를 포함할 수 있는 시험 스위트를 주어진 테스트 목적에 따라 찾아내고 이를 적용하여 적합성을 체크한 후 오류가 없는 완전한 프로토콜을 구현할 수 있도록 한다.

2.1 적합성 관계

구현된 프로토콜의 적합성을 검사하기 위해서는 프로토콜이 정형모델에 기초한 명세언어로 표현되도록 하고 구현하기 전에 논리적인 정확성을 검증하기 위한 여러 방법을 사용한다[9-12]. 프로토콜은 우리가 예측할 수 없을 만큼 매우 복잡한 통신 개체간의 상호작용을 표현하고 있다. 따라서 프로토콜의 기능적인 동작이 설계자의 의도를 따른다는 것을 확인하려면 먼저 프로토콜을 정형모델에 바탕을 둔 명세언어를 사용하여 기술하고 이를 바탕으로 구문 및 기능 분석을 행하여야 한다. 적합성 시험은 두 가지 형태의 요구사항으로 나눌 수 있다. 먼저 정적 적합성 요구 사항은 시험을 받고자 하는 IUT(Implementation Under Test)에 대해 구현자가 해당 프로토콜의 모든 구성 요소들을 어떻게 구현하였는지를 기술한 프로토콜 구현 적합성 명세 (PICS : Protocol Implementation Conformance Statement)를 검토하여 프로토콜의 정적 적합성 요구사항과의 일치여부를 검사한다. 두번째의 동적 적합성 요구사항(Dynamic Conformance Requirement)이란 시험대상의 관측가능한 행위들이 표준사양에서 허용된 행위인지에 관한 요구사항을 의미한다. 다시 말하자면 표준사양 S에서 정의된 동적 적합성 요구사항을 $R=\{r_1, r_2, \ldots, r_n\}$이라 하고, 구현 I의 외부관측 행위를 B_I라 한다면, 동적 적합성 시험이란 다음과 같이 표현된다.

$$\forall\ r \subset R : B_I \ \underline{sat}\ r$$

여기서 sat는 요구사항을 만족한다는 관계를 나타낸다.

그림 1은 표준사양과 구현 및 구현모델 사이의 관계를 나타낸 것이다. 먼저 표준사양(Specifications : SPECS)은 선택사항들이 특정 값으로 고정된 여러 사양들의 집합으로 볼 수 있으며, 표준사양에 따라 구현될 수 있는 구현의 집합은 IMPS(Implemenations)로 표시할 수 있다. IMPS의 집합에서의 구현들은 가상적인 표기법으로 표현될 수 있다고 가정하고 이들의 집합을 MODS(Models)라고 하면, 표준사양에서 선택사항들이 특정 값으로 부여된 사양 S는 SPECS의 한 원소이며, 사양 S에 따라 구현된 구현들의 집합을 I_s라 하고 이 집합은 IMPS의 부분집합이다. I_s의 추상모델을 M_s라 한다. 그림 1에서 표시된 "implements correctly"관계는 "적합성"관계를 의미한다. 이 적합성 관계를 형식적으로 표현하기 위하여 실제 구현물(IUT : Implementation Under Test)를 형식표기법을 사용하여 모델화한 것을 m_{IUT}라 한다.

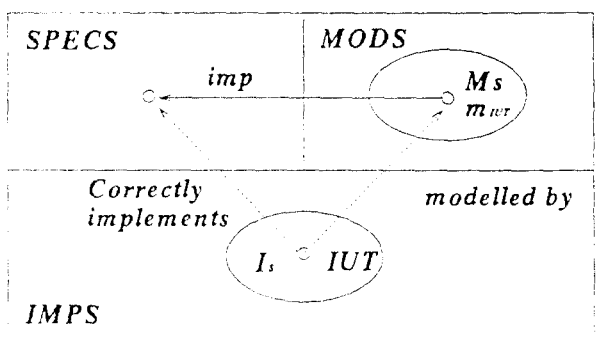

그림 1. IMPS, MODS와 SPECS의 관계
Fig. 1. Relations between IMPS, MODS and SPECS

FMCT(Formal Method in Conformance Testing)에서는 구현의 적합성을 정의하기 위하여 구현들은 형식기법으로 모델화 된다고 정의하고 있다[9]. 실질적으로 형식 기술기법으로 기술된 프로토콜 사양에는 적합성 요구사항들이 명시적으로 기술되어 있는 것은 아니므로 주어진 사양을 시험항목 생성에 적절한 형태로 변환하게 된다.

2.2 시험 항목 생성 조건

시험 수행이란 주어진 시험 사양을 실제로 구현하여 그 시험구현을 시험대상에 적용하여 시험결과를 얻는 과정을 말한다. 본 절에서는 시험 사양으로부터 구현된 시험 스위트를 시험대상에 적용하고 그 결과를 분석하여 적합성 판정을 내리는 과정을 형식기법의 관점에서 정의하고자 한다. 가능한 구현들의 유형 IMPS는 적합성 시험에 통과한 I_{pass}와 적합성 시험에 실패한 I_{fail}로 나누어지고 실제 적합한 구현 $I_{conform}$은 반드시 I_{pass}에 속해야 한다. 다시 말하면 잘못된 구현이 적합성 시험을 통과하는 항목이 발생해서는 안된다는 것이다. 시험 스위트와 관련하여 두 가지 중요한 성질은 다음과 같다.

(1) Soundness : 시험 스위트가 표준사양 S에 적합한 구현 I를 주어진 시험환경 C에서 시험한 결과는 반드시 "pass"이어야 한다.

즉, $\forall I \in IMPS : I$ conforms $S \Rightarrow C[I]$ passes T

(2) Exhaustiveness : 시험스위트가 표준사양 S에 부적합한 구현 I를 주어진 시험환경 C에서 시험한 결과는 반드시 "fail"이어야 한다.

즉, $\forall I \in IMPS : C[I]$ passes $T \Rightarrow I$ conforms S

시험항목 t는 실제로 시험기의 일부로서 구현되며, 시험기는 주어진 시험환경 C에서 시험대상 I와 함께 수행되므로 시험항목의 수행함수 exec은 아래와 같이 정의된다.

exec(t,C[I]) → {pass, fail, inconclusive}

시험항목의 수행과 관련한 함수는 시험항목을 시험대상에 적용하여 그 행위를 관찰하는 apply 함수와 관찰결과를 바탕으로 판정을 내리는 verd 함수로 나눌 수 있다. 먼저 apply 함수는 아래와 같이 정의된다.

apply(t,C[I]) → OBS

여기서, OBS는 시험 환경에서 구현물의 관찰(OBServation)상태를 나타낸다. 그리고 apply 함수와 시험항목

수행함수 exec와의 관계는 다음과 같다.

exec(t,C[I]) = verd(t,apply(t,C[I]))

시험 스위트의 수행은 그 시험 스위트에 속한 모든 시험항목들을 수행하는 과정이다. 시험 스위트의 수행에서는 개별적인 시험항목 수행결과를 바탕으로 전체 시험 스위트에 대한 적합성 판정을 하게 된다. 즉, 시험스위트 T가 시험대상 I에 시험환경 C에서 적용되었다면 시험 스위트에 대한 판정 부여 함수 verd_ts(T,C[I])는 아래와 같다.

verd_ts(T,c[I])={ pass if $\forall_t \in T$:exec(t,C[I])=pass,
fail if $\exists_t \in T$:exec(t,C[I]) = fail}

주어진 사양에 적합한 구현 I는 시험환경 C에서 시험 스위트 T에 의해 fail 판정을 받을수 없으며, 부적합한 구현이 pass 판정을 받을 수 없다는 사실은 앞절에서 언급한 시험 스위트의 두가지 성질에 기인한다.

즉, C[I] passes T ⇔ verd_ts(T,C[I]) ≠ fail

시험 항목을 수행할 때 같은 시험 스위트에 속한 시험항목들의 수행순서를 어떻게 결정할 것인가의 문제가 발생한다. 서로 독립적인 시험항목인 경우 수행순서는 시험수행에 영향을 주지 않지만 한 시험항목이 다른 시험항목의 일부이거나 한 시험항목의 결과가 다른 시험항목의 수행에 영향을 줄 항목에는 수행순서의 선택이 시험수행의 성능에 영향을 끼칠 수 있다. 예를 들어 어떤 시험항목이 프로토콜의 연결 설정에 관한 시험을 수행하고, 다른 시험항목은 데이타의 전송을 시험하는 경우, 앞의 시험항목를 먼저 수행함으로써 시험대상이 부적합함을 빨리 알 수 있을 것이다. 시험대상이 연결 설정 과정에서 문제를 발생할 경우 뒤의 시험 항목은 "inconclusive"로 판정될 것이기 때문이다. 따라서 시험 스위트에는 그 스위트 시험에 포함된 시험항목들의 수행순서가 명시되어야 한다. 수행순서에 따라 시험을 수행하다가 어느 시험항목이 "fail" 판정을 하게되면 즉시 시험을 중단함으로써 시험시간을 단축할 수 있다.

2.3 적합성 시험의 수행 과정

적합성 시험의 수행과정은 제어 관찰점(PCO:point of control and observation)을 통하여 IUT에 이벤트를 인가하고 IUT로부터 발생된 이벤트를 관찰하여 IUT의 상태를 진단하게 된다. IUT를 FSM(finite state machine)이라고 해석할 때 FSM의 완벽한 테스트는 초기상태로부터 도달가능한 모든 상태공간을 확인하는 것이다. 통신 프로토콜의 적합성을 시험하기 위해서는 Black Box 시험을 하게 된다. Black Box 시험은 내부 정보를 사용치 않고 외부 정보만을 이용하여 시험을 수행하는 방법이다. Black Box 시험시 IUT가 유한 상태기계로 표현된 프로토콜의 동작을 직접 제어하거나 관찰할 수 없으므로 IUT의 행위는 제어 관찰점에서의 입출력 동작을 제어하거나 관찰함으로써만 가능하다. 따라서 IUT가 외부적으로 관찰 가능한 동작에 의

해서 시험되기 위해서는 제어 가능성(Controllability)과 관찰 가능성(Observability)이 정의되어야 한다.

ISO/IEC 및 ITU-T에서 규정한 적합성 시험 절차는 크게 5가지의 단계로 구분된다. 첫번째 단계인 정적 적합성 분석 단계에서는 시험을 받고자 하는 IUT에 대해 구현자가 해당 프로토콜의 모든 구성 요소들을 어떻게 구현하였는지를 기술한 프로토콜 구현 적합성 명세(PICS)를 검토하여 프로토콜의 정적 적합성 요구 사항과의 일치여부를 검사한다. 정적 적합성의 검사는 보통 수동적인 방법을 통하여 수행되며 시험 의뢰자가 작성하여 제출한 PICS의 내용이 장비간의 연동성을 위한 최소한의 기능 요구사항을 규정한 정적 적합성 요구사항과 일치하는지를 검토한다. PICS의 내용이 정적 적합성 요구사항과 일치하지 않으면, 더 이상 시험은 진행되지 않는다. 두번째 단계는 선택(Selection) 및 파라미터화(Parameterization) 단계로서 PICS의 내용중 구현된 항목에 따라 시험 스위트에서 동적 적합성 요구사항을 만족하는 경우에 시험에 사용될 시험항목(Test Case)들을 선택한다. 특정 프로토콜의 적합성 시험을 위한 시험 스위트(Test Suite)는 프로토콜의 모든 기능을 시험할 수 있는 시험항목을 포함하고 있으나 특정 구현 제품이 일부 선택기능을 구현하지 않을 수도 있으므로 시험항목은 해당 구현제품의 PICS에 따라서 선별적으로 수행하게 된다. 그리고 매개변수화란 동일한 기능을 구현했다고 할지라도 특정 IUT에 따라서 변할 수 있는 네트워크의 주소, 카운터, 타이머 값 등을 포함하는 PIXIT(protocol Implementation eXtra Information for Testing)를 시험항목 실행 전에 입력하는 과정을 말한다.

세번째 단계는 파라미터화된 실행 시험 스위트(EST : Executable test suite)를 실행하는 단계이다. 이 단계에서는 세가지의 시험 즉, 기본 상호연결 시험(Basic Interconnection Test), 능력시험(Capability Test)과 행위시험(Behavior Test)을 하게 된다. 네번째 단계는 시험결과 분석단계로 각 시험항목의 시험목적과 관찰된 행위의 타당성을 분석하여 합격(Pass), 불합격(Fail) 또는 미확정(Inconclusive)의 판결로 표시된다. 마지막 단계는 최종 적합성 고찰 단계로서 이 단계에서는 정적 적합성 분석 결과와, 능력 및 행위시험의 결과를 분석·종합하여 표준화된 형식의 적합성 시험 보고서를 작성한다.

Ⅲ. 오류 모델을 이용한 시험 항목 생성

프로토콜은 네트워크 엔티티가 상호 통신하도록 하는 규칙들의 집합이다. 프로토콜의 사건 중심 특성 때문에 상태 천이 시스템 모델인 FSM 모델이 프로토콜 명세시 주로 사용되었다. 이는 Peer 엔티티로부터 사용자의 요구나 메시지 도착과 같이 처리 형태가 단순하기 때문에 광범위하게 사용되었고, 프로토콜의 제어흐름을 쉽게 나타낼 수 있어서 주로 사용되고 있다[1,10].

3.1 DFSM에 대한 오류모델

프로토콜의 제어부분은 보통 FSM으로 모델링한다. FSM은 방향 그래프 G = (V,E)로 표현되는데 여기서 V는 버텍스의 유한 집합이고 E는 에이지의 집합이다. 버텍스는 FSM의 상태에 대응되고 각 에이지는 가능한 상태 전이에 대응된다. 또한 FSM은 입력(I)과 출력(O)동작의 쌍으로 구성되는 레이블을 갖는다. 입력 동작은 FSM에게 상태 전이를 발생시키는 이벤트이고 출력동작은 외부에서 관찰 가능한 동작이다. FSM은 하나의 입력에 대해 서로 다른 상태로 천이가 발생할 수 있는데 이러한 경우 비결정성 때문에 상태 천이의 정확한 표현이 불가능하므로 결정적인 경우로 제한하였다. 이러한 FSM을 결정적 유한 상태기계(DFSM : deterministic finite state machine)라고 한다. DFSM의 예는 그림 2에 나타나 있다. 지금까지의 프로토콜 적합성 시험에 관한 연구는 주로 FSM에 적용할 수 있는 제어 흐름만을 고려한 사항들이 주로 이루어졌다. 프로토콜의 시험 항목 생성 방법에 DFSM에 기초한 모델이 광범위하게 사용되고 있다[8-13].

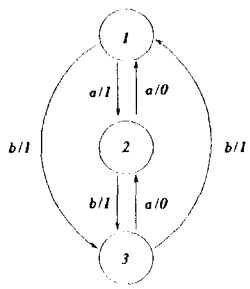

그림 2. 결정적 유한 상태 기계
Fig. 2. Deterministic finite state machine

[정의 1] : DFSM은 6개의 쌍인 (S, I, O, λ, δ, S_0)으로 구성된다.

여기서 S : set of states

I : set of inputs

O : set of outputs

λ : output function S×I →O

δ : transition function S×I →S

S_0 : initial states

DFSM에 기초한 블랙 박스 시험은 주어진 DFSM 사양에 대하여 DFSM 구현을 시험하는 것이다. 그러나 유한상태기계를 기초로 한 시험항목 생성방법은 프로토콜의 자료 부분중 출력값을 제외한 나머지 부분, 즉 상호작용 파라미터, 천이를 수행하기 위한 술어(Predicate), 명세에 포함된 우선순위나 지연(delay), 내부변수와 같은 프로토콜의 자료부분을 무시하고 프로토콜의 제어흐름만을 나타내므로 실제의 프로토콜에 적용하기에는 많은 문제점들을 내포하고 있다. 이러한 여러 문제점이 존재하고 있지만 프로토콜을 명세하고 기술하는데 유한 상태 기계 모델을 사용하고 있다. 실제적인 프로토콜을 시험할 때 시험되어져야 할 구현된 프로토콜의 상태 수는 정확하게 알려져 있지 않다. 이러한 문제를 극복하기 위하여 발생할 수 있는 오류의 형태를 적절하게 제한하는 방법을 사용하게 된다[2,3]. 오류 모델은 모든 상태를 검사하는 것이 불가능하기 때문에 사용하는 방법이 되며, 오류는 구현 프로토콜의 행위와 사양의 행위가 일치하지 않고 차이가 발생하는 경우를 말한다. 이러한 오류는 구현 과정에서 발생할 수 있다는 것을 가정한다. 이렇게 발생가능한 오류를 적절하게 제한하는 방법이 오류 모델을 사용하는 목적이 된다. 오류 모델에 관한 일반적인 내용을 기술하고 있고[5,6], 본 논문에서는 가장 잘 발생할 수 있는 오류의 형태를 네 가지 형태로 분류한다.

(가) 오류 모델 1

어떤 한 상태와 대응한 입력에 대하여 천이가 발생하는 동안 출력 오류가 발생하는 항목을 말하며, 즉 구현 프로토콜이 출력함수에 의해 정의된 것과 다른 출력을 발생하게 된다. DFSM으로 표현된 에이지 E가 사양과 다른 출력을 발생시키면 그러한 오류의 종류를 오류 모델 1이라고 분류한다. 그림 3을 보면 그림 2의 경우와 비교하여 상태 1에서 3으로 천이되는 출력이 1에서 0으로 변경되었음을 알 수 있다. 오류 모델 1은 천이하는 동안 출력이 변경되므로 이를 출력 오류(output fault) 라고도 한다.

(나) 오류 모델 2

오류 모델 2는 어떤 한 상태와 대응한 입력에 대하여 천이가 발생할 때 다음 상태함수에 의하여 구현 프로토콜이 사양과는 다른 상태로 천이가 되는 오류를 말한다. DFSM으로 표현된 에이지 E의 상태가 상태 i로부터 i′로 변경되면 에이지 E에서 에러가 발생했다고 한다. 이러한 오류는 상태가 다른 상태로 되었기 때문에 상태 천이 오류(transfer fault)라 하고 오류 모델 2로 정의한다. 그림 4를 보면 오류가 없는 그림 2의 유한상태기계와 비교하여 상태 1에서 3으로 천이가 발생하지 않고 상태 1에서 다시 상태 1로 천이가 발생하였음을 알 수 있다. 즉 천이 (1,3 : b/1)가 (1,1 : b/1)로 되었으므로 상태 천이 오류가 발생한 것이다.

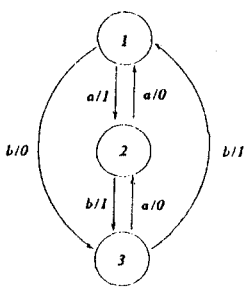

그림 3. 오류모델 1
Fig. 3. Fault model 1

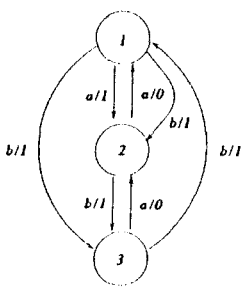

그림 4. 오류모델 2
Fig. 4. Fault model 2

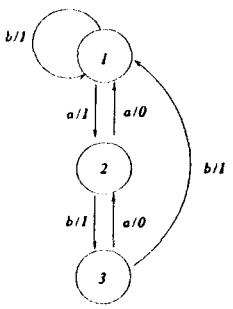

그림 5. 오류 모델 3
Fig. 5. Fault model 3

(다) 오류모델 3

어떤 특정한 상태와 입력에 대하여 하나 이상의 상태가 추가되거나 제거되는 경우를 말한다. 오류 모델 3은 두가지 형태의 오류로 나눌 수 있다. 상태가 두개 이상으로 분류되는 상태 분리 오류(state split fault)와 상태가 두개 이상에서 하나로 합쳐지는 상태 병합 오류(state merge fault)로 나눌 수 있다. 그림 5가 그러한 예를 보여주고 있다.

(라) 오류 모델 4

마지막으로 오류 모델 4는 오류가 상태나 에이지의 한 부분에서만 발생하지 않고 이 두가지 오류가 동시에 발생하는 경우를 말한다. 즉, 출력 오류와 상태 천이 에러가 복합적으로 발생하는 경우가 발생할 수 있다. 이러한 오류가 발생한 경우 오류를 찾아내기가 쉽지 않다. 오류모델 4는 앞에서 정의한 오류 중에서 하나 이상이 혼합되어 발생하는 오류로 정의되며, 이러한 오류를 혼합 오류(hybrid fault)라고 한다.

3.2 DFSM에 기초한 시험항목 생성

DFSM 모델에 근거한 시험항목 생성 방법이 광범위하게 연구되어 왔고[1,4,10], 프로토콜의 제어 흐름을 쉽게 모델화 할 수 있으므로 시험 항목 생성 방법들이 DFSM 모델을 기초로 하여 이루어지고 있다. 프로토콜을 명세하기 위해서는 프로토콜이 무엇을 수행하고 서비스 프리미티브나 타임 아웃과 같은 사건에 대해서 어떻게 대응해야 되는가를 기술해야 한다. 그런데 프로토콜은 일반적으로 매우 복잡하므로 프로토콜 설계 단계에서부터 완전하고, 정확하며, 간결하면서도 모호성을 포함하지 않게 명확하게 기술되어야만 한다. 이를 위해서 형식 명세기법이 사용되고 있다. 프로토콜을 명세하기 위해서 형식 명세기법을 사용했다고 할지라도 구현자가 프로토콜을 잘못 구현하면 구현된 제품들 간의 상호 운용이 이루어지지 않게 된다. 본 절에서는 지금까지 연구된 시험항목 생성방법들이 주로 제어흐름을 쉽게 나타낼 수 있는 결정적 유한 상태 기계를 기초로 하여 이루어진 Transition Tour, W-method, Distingushing 순서, RCP(Rural Chinese Postman tour) 방법, UIO(Unique Input Output) 순서 등과 같은 방법중에서 가장 많이 사용되는 UIO 순서에 대해서만 기술하도록 한다.

3.2.1 UIO 생성 방법

FSM에서는 서로 다른 상태에서 천이할 때 같은 입력에 대해 같은 출력을 생성하는 경우가 많다. 따라서 하나의 이벤트만을 관찰하여서는 IUT가 원하는 상태에 있는지를 확인할 수가 없다. 그러므로 IUT에서 관찰되는 연속 이벤트로부터 상태의 변화를 확인할 수 있는 방법이 요구된다. 이를 해결한 방법이 UIO 기법이다. UIO 순서란 어떤 특정 상태로부터 출발하여 하나 이상 몇개의 연속된 I/O 순서를 조사하여 이 연속된 이벤트가 프로토콜 규격에 따르면서 다른 어떤 상태에서도 발생하지 않는 이벤트일 때 이를 특정 상태의 유일한 I/O (UIO) 순서라고 한다. 이 방법은 다른 상태에서는 발생되지 않아서 특정 상태를 유일하게 구분할 수 있는 입출력 순서의 쌍인 UIO 순서를 이용하여 시험 항목을 생성하는 방법이다. 한 상태에 여러 개의 UIO 순서가 존재할 수 있으므로 이들 중에서 시험순서의 길이를 최소화할 수 있는 UIO를 선택하는 것이 바람직하다. 이 방법은 프로토콜에서 정의된 모든 천이를 적어도 한번은 순회하고, 천이의 목적상태의 UIO 순서를 적용함으로써 모든 상태의 천이를 검증할 수 있다. 그리고 DS 방법처럼 오류 발견 능력도 좋고 대부분의 경우에 시험순서의 길이가 DS 방법보다 짧아서 시험항목 생성시 주로 UIO 방법이 사용되고 있다.

3.2.2 UIO 방법의 문제점

한 상태에 대하여 여러개의 UIO 순서가 존재할 때 어떤 특정 UIO 순서를 가지고 시험 항목을 생성하였다면 오류를 갖는 IUT에서 UIO 순서의 중복이 발생하게 된다. 이러한 경우에 어떤 상태에 오류가 존재하고 있는지 구별할 수 없는 문제가 발생한다. 즉 UIO 순서의 기본 개념은 어떤 특정 상태에 유일할 때에만 상태의 인식이 가능하기 때문이다. IUT가 명시된 FSM과 같은 UIO 순서의 집합을 가진다면, 사양으로부터 생성된 UIO 순서는 IUT의 한 상태를 인식할 수가 있다. 그러나 IUT가 잘못 구현되었고 사양으로부터 생성된 UIO 순서가 다수 개 존재할 때 UIO 순서는 IUT에 유일하지 않을 수도 있기 때문에 특정 상태를 찾아낼 수가 없는 경우가 존재한다. UIO 순서의 또 다른 문제점은 어떤 상태에 대해 UIO 순서가 존재하지 않는 경우이다. 어떤 상태에 대하여 UIO 순서가 존재하지 않는다면 오류가 있는 특정 상태를 찾아낼 수가 없다. 이렇게 UIO

순서가 존재하지 않는 경우에는 UIO 순서 대신 signature 라는 새로운 개념을 사용한다. signature는 상태가 다른 I/O에 대한 순서의 연속으로 정의한다. 이러한 signature를 사용하여 UIO 순서가 존재하지 않는 경우에 오류를 갖는 상태를 밝혀내는데 사용된다. [8]은 이와 같이 UIO 순서를 갖지 않는 상태를 다른 상태와 하나씩 구분하기 위해서 확인자(signature)를 사용하였고, [4][8]등은 원시상태와 목적상태에 대해서 유일한 접속된 상태 확인자(concatenated state signature)를 사용하였다. 그리고 [1]는 임의의 상태가 자신과 다른 모든 상태와 구분할 수 있는 UIO 순서를 갖지는 않았지만 다른 상태들의 일부와 구분할 수 있는 순서들을 가졌으면 부분적으로 구분 가능한 상태들만을 따로 모아서 별도의 배타적 집합(exclusion Set)을 구성할 수 있는 PUIO을 사용하여 처리하였다.

3.3 중간 모델을 이용한 시험 항목 생성

FMCT에서는 자연어로 프로토콜을 서술할 경우에 제기되는 애매성을 제거하고, 자동화를 쉽게 하기 위해서는 FDT를 사용하여 프로토콜 서술, 구현 및 시험하는 통합 환경을 구축하기 위한 연구가 진행중이다. 시험항목 자동 생성기법은 적합성 시험에 형식기법을 도입함으로써 얻어지는 가장 큰 장점중 하나이다. 따라서 형식기술기법(FDT)으로 작성된 프로토콜 사양으로부터 시험항목을 생성하는 방식에 대한 많은 연구가 이루어져 왔다. 시험 항목의 효율적인 생성을 위하여 중간 모델을 이용한 방법도 한 기법이 될 수 있다. 이 방법은 주로 LOTOS를 중심으로 이루어지고 있는 이론적 연구로써, FDT를 이용한 시험 항목 생성 방법은 크게 두가지로 분류한다. 하나는 형식 명세로부터 직접 시험 항목을 생성하는 경우이고 다른 하나는 형식명세를 중간모델로 변환하고 중간모델로부터 시험 항목을 생성하는 간접 시험 항목 방법이다. 간접 시험 항목 생성 방식에서 주로 사용하고 있는 중간모델(intermediate model)로는 다음과 같은 것들이 있다.

(1) Asynchronous Communication Tree(ACT)
(2) Chart
(3) IO-Extended FSM
(4) Labelled Transition System(LTS)

현재는 FDT로 정의된 ESTELLE, LOTOS 및 SDL로 표현된 프로토콜 명세를 중간 모델로 변환하고 있는 연구 결과가 나와 있고, 중간 모델로부터 시험 항목을 생성하기 위한 연구들이 진행되고 있다[9].

3.4 오류 모델에 의한 시험항목 생성

적합성 시험에 있어서 시험되어져야 할 구현된 프로토콜 상태 수는 정확하게 계산할 수가 없다[7]. 이러한 부정확한 상태 수 문제를 해결하고 시험목적에 적합할 수 있도록 하기 위하여 구현 프로토콜에서 발생할 수 있는 오류 형태를 정의하여 시험 항목을 생성하는 방법을 사용하면 상당히 효율적인 방법이 될 수 있을 것이다. 구현 프로토콜의 행위와 사양의 행위가 일치하지 않고 차이가 발생하는 경우를 구현 프로토콜 오류라고 하며, 이러한 경우에 시험 항목이 구현 프로토콜의 오류를 발견해 낸다면 적합성 시험의 기본 목표에 부응하게 된다. 오류 모델은 모든 상태를 검사하는 것이 불가능하기 때문에 사용하는 방법이 되며, 이러한 오류는 구현 과정에서 발생할 수 있다는 것으로 가정한다. 오류 모델을 사용하여 시험항목을 생성하는 방법은 3.1절에서 정의한 DFSM에 대한 오류 모델을 기초로 하고 있다[7,11].

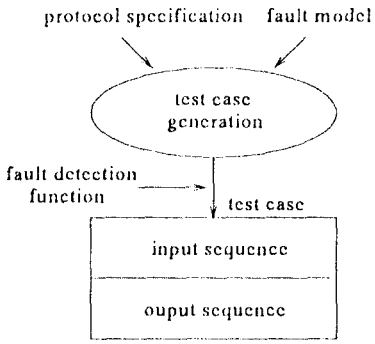

그림 6. 오류 모델을 이용한 시험항목 생성
Fig. 6. Test case generation using fault model

오류 발견함수를 이용하여 시험항목을 생성하기 위한 전단계로써 적합성과 관련하여 몇가지 개념을 결정적 유한 상태 기계(DFSM)를 사용하여 정의한다. 먼저 DFSM의 구조와 형식에 관련한 구문 관계는 다음과 같이 정의한다.

〈구문관계 1〉 $F_1 \leq F_2 =_{def} s^0_1 = s^0_2$ and $S_1 \subseteq S_2$ and $T_1 \subseteq T_2$.

여기서, S는 상태의 집합이고 T는 천이의 집합이다.

DFSM F_2가 F_1으로부터 하나 이상의 상태와 천이를 부가하여 생성되면 F_2은 F_1을 확장하였다고 한다

〈구문관계 2〉 $F_1 \subseteq F_2 =_{def} \exists f : S_1 \rightarrow S_2 : f$ is 1-1 and $f(s^0_1) = s^0_2$

and $((s,a,x,s') \in T_1 \Rightarrow (f(s),a,x,f(s')) \in T_2)$

DFSM F_1와 F_2에 대해 F_2가 상태의 특성을 그대로 유지하게 되면 F_1의 실현이라고 한다. 다음으로 DFSM의 행위에 기초한 문맥 관계에 대한 정의는 다음과 같다.

〈문맥관계〉 $F_1 \triangle F_2 = def$ accept(F1) $\subseteq$ accept(F_2) and $\forall \in$ accept(F_1) :

output(F_1, α) = output(F_2, α)

F_2가 F_1의 모든 입력 순서열(α)을 받아들이고 그에 따른 출력도 동일하다면 F_1은 F_2에 포함된다고 한다.

본 논문에서는 오류 모델 1인 출력 오류에 대해서만 오류 발견 함수를 정의하였다. 오류 발견 함수(fault detection function)를 이용하여 시험항목을 생성하는 방법은 오류모델에 따라 서로 다르고 또한 다른 오류 발견 함수를 정의하여야 한다. 오류 발견함수를 정의하기 위해 3장에서 정의한 결정적 유한 상태 기계를 하나의 참조 DFSM F = (S, I, O, λ, δ, S_0)와 위에서 정의한 구문관계와 문맥관계를 이용하여, DFSM으로부터 IUT의 오류 행위를 판단하기 위하여 오류 생성 함수 G를 다음과 같이 정의된다.

〔정의 2: 오류 생성 함수〕

$G : S' \times I \rightarrow P(S' \times O)$, if $\forall (s,i)$

$\in S \times O$ ($\{ \lambda(s,i), \delta(s,i) \} \in F(s,i)$

여기서, S' S이고 $P(S' \times O)$는 $S' \times O$의 부분 집합을 의미한다. 오류 생성 함수를 정의하기 위해 구문관계 1,2를 이용하였다. 구문관계 1에서는 상태나 천이를 추가함으로써 오류 상태가 계속적으로 변화되는 과정을 의미하고 구문관계 2는 서로 다른 DFSM F_1와 F_2에 대해 입력 순서를 적용했을때 포함 관계를 나타낸다. 오류 생성 함수의 의미는 어떤 입력에 대해 수없이 많은 다른 출력 상태를 표현하게 하여 사양과 비교했을때 출력 오류를 갖는 구현물을 생성하게 된다.

〔정의 3: 오류 발견 함수〕

$M_s \{\lambda(S_s, \alpha)\} = M_i \{\lambda(S_i, \alpha)\}$이면 오류 미발견 상태이고, $M_s \{\lambda(S_s, \alpha)\} \neq M_i \{\lambda(Si, \alpha)\}$가 되면 오류 발견 상태이다. 여기서 λ는 정의 1에서 정의한 것처럼 DFSM의 출력함수를 나타내고, M_s와 M_i는 사양 머신과 구현물을 의미한다. 그리고 오류 발견함수의 의미는 사양 머신(M_s)의 상태(S_s)에 입력 순서(α)를 적용한 결과와 오류 생성 함수에 의해 생성된 구현물(M_i)의 상태(S_i)에 동일한 입력 순서를 적용했을 때의 출력이 동일하면 오류 미발견 상태가 되고, 서로 다르면 오류 발견 상태라고 할 수 있다. 위의 문맥관계에서 정의된 것처럼 오류 발견 함수는 서로 다른 상태에서 동일한 입력 순서를 인식(accept)하게 되면 두개의 상태 기계 F_1과 F_2를 동일한 DFSM으로 인식하여 오류를 발견하지 못하게 된다. 2장에서 기술한 적합성 시험의 판정은 시험스위트 T가 시험대상 I에 시험환경 C에서 적용되었다면 시험 스위트에 대한 판정 부여 함수와 오류 발견함수와의 관계는 다음과 같다.

오류 발견함수를 적용하여 출력 결과가 동일하면 "$\forall_t \in$ T:exec(t,C[I]) = pass"가 되고, 출력 결과가 서로 다르게 되면 "$\exists_t \in$ T:exec(t,C[I]) = fail"로 판정할 수 있다. 오류 발견 함수의 사용은 오류 모델에 따라 다르게 정의되어 사용될 수 있으며, 이러한 오류 발견 함수를 이용하여 시험항목을 생성하여 적합성 시험에 적용하게 되면 오류 발견 능력을 높이게 될 것이다. 4장에서는 기존 FSM 모델에 근거한 방법에서 오류를 찾지 못하는 경우을 살펴보고, 실제 적용 예를 보이도록 한다.

Ⅳ. 적용사례

적합성 시험이 효율적으로 이루어지기 위해서는 시험항목이 효율적으로 생성되어야 한다. 실제 프로토콜은 너무 방대하고 엔티티 사이에 다양한 상호작용을 하고 있기 때문에 프로토콜의 시험항목 생성을 위하여 오류모델을 정의하고 정의된 오류모델에 의하여 생성된 시험항목을 적용하여 얼마나 많은 오류를 찾아낼 수 있는가를 시험하게 된다. 따라서 최적의 시험항목을 생성하고 오류검출 능력이 최대가 되는 시험항목을 생성하는 방법이 필요하게 된다. 오류 발견 함수를 이용하여 시험 항목을 생성하고, 적용 예를 보임으로써 제안된 방법이 기존의 미발견 오류를 찾아냄으로써 오류 발견 능력이 우수함을 보이도록 한다.

4.1 미발견 오류

FSM 모델에 기초한 여러 시험 항목 생성 방법을 적

용했다 할지라도 시험 항목의 구성에 따라 오류를 찾아내지 못하는 경우가 발생할 수가 있다. 실제 프로토콜 구현시 오류가 발생했지만 기존의 시험 항목 생성 기법으로 오류를 찾아내지 못하는 경우가 발생할 수 있다. 즉 오류가 실제로 존재하고 있는데 생성된 시험항목으로 오류를 찾아내지 못하게 되면 그만큼 시험 항목의 오류 발견 능력이 떨어지므로 효율적인 적합성 시험을 수행하지 못하게 된다. 여기서는 UIO 방법을 사용하여 생성된 시험항목으로 오류를 발견하지 못하는 예를 살펴보도록 한다.

그림 7의 유한 상태 기계를 오류가 있는 상태 기계로 변경하기 위하여 상태 V1에서 V2로의 천이를 V1에서 V3으로 변경시켜 상태 천이 오류를 발생하도록 한다. 실제로 그림 7의 유한 상태 기계를 변경하여 오류가 존재하도록 한 상태 테이블이 표 1에 나타나 있다.

표 1의 오류를 갖는 상태 변환 테이블을 이용하여 시험항목을 생성하기 위해 UIO 순서를 적용하게 되면 그림 8과 같이 한 상태를 구분할 수 있는 UIO 순서가 중복되어 오류를 찾아내지 못하게 된다. 실제로 그림 8의 V2와 V3의 UIO 순서가 b/y a/x가 되어 동일한 UIO가 만들어지게 되어 이를 기본으로 하여 시험 항목을 생성하게 되면 오류를 발견하지 못하게 되는 문제점이 발생한다. 이러한 문제점을 해결하기 위해 3장에서 정의한 오류 발견함수를 사용하여 시험항목을 생성하도록 한다.

4.2 적용 사례

기존의 UIO 방법을 사용하여 오류를 발견하지 못하는 예를 앞절에서 살펴보았는데, 여기서는 이러한 경우를 포함하여 기존의 다른 방법으로 발견하지 못하는 경우에 오류 발견 함수를 사용하여 시험 항목을 생성하는 것이 효율적으로 오류를 찾아 낼 수 있음을 보이도록 한다. 먼저 3장에서 제안한 오류 발견함수를 다시 살펴보도록 한다. $M_s\{\lambda(S_s, \alpha)\} = M_i\{\lambda(S_i, \alpha)\}$이면 오류 미발견 상태이고, $M_s\{\lambda(S_s, \alpha)\} \neq M_i\{\lambda(S_i, \alpha)\}$가 되면 오류 발견 상태이다. 오류 모델 1의 출력오류는 동일한 입력에 대해 사양과 구현 프로토콜의 출력이 서로 다르게 나타나는 경우를 말한다. 이렇게 출력이 서로 다른 오류를 찾기 위하여 오류발견함수를 적용한다. 그림 7에 대한 UIO 방법을 적용하기 위하여 그림 7의 UIO 순서는 그림 9(a)와 같다. 그림 7의 UIO 순서를 기본

표 1. 그림 7에 대한 오류 상태 변환 테이블
Table 1. Fault state transition table for Fig. 7

state \ input	a	b	c
v1	v3/x	v1/y	v4/x
v2	v3/x	v4/y	v1/y
v3	v3/y	v1/y	v2/y
v4	v3/y	v1/y	v3/x

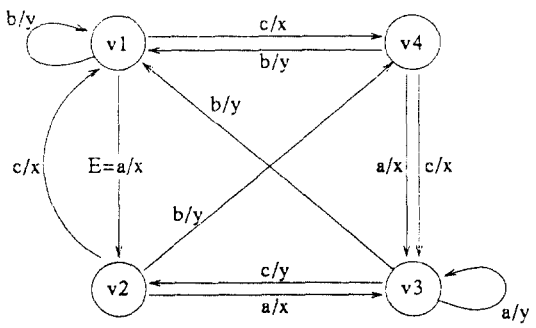

그림 7. 미발견 오류를 위한 유한 상태 기계
Fig. 7. Finite state machine for undetected fault

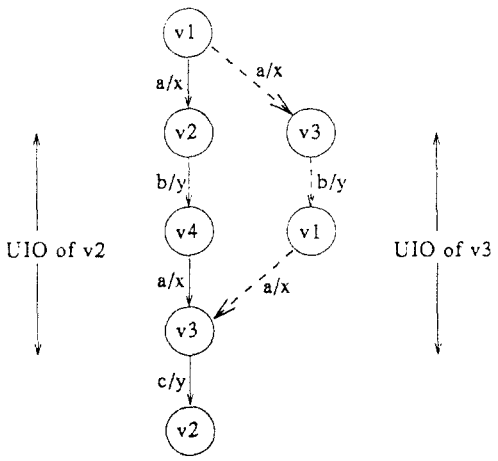

그림 8. UIO 방법에 의한 미발견 오류
Fig. 8. Undetected fault for UIO method

상 태	UIO 순서
v1	c/x a/x c/y
v2	b/y a/x c/y
v3	c/y
v4	c/x c/y

(a)

a/x b/y a/x c/y-a/x c/y-b/y c/x c/y-c/x c/x a/x c/y c/x
b/y c/x a/x c/y-c/x-c/x c/x c/y-a/x-a/y c/y a/x
b/y c/x a/x c/y-a/x-c/y b/y a/x c/y-b/y a/x c/y
b/y-b/y c/x a/x c/y-b/y-c/x c/y

(b)

그림 9. 그림 7의 UIO 순서와 시험 항목
Fig. 9. UIO sequence and test case of Fig. 7

으로 하여 시험항목을 생성하게 되면 그림 9(b)와 같는 시험 순서가 생성된다. 그림 7의 유한 상태 기계의 오류는 상태 V1에서 V2로의 천이가 V1에서 V3으로 변경되어 유한 상태 기계는 오류를 포함하고 있다. 즉 표 1의 오류를 갖는 상태 변환 테이블을 이용하여 시험항목을 생성하기 위해 UIO 순서를 적용하게 되면 되면 그림 8과 같이 V2와 V3의 UIO 순서가 b/y a/x가 되어 동일한 UIO가 만들어지게 되어 오류를 발견하지 못하게 되는 문제점이 발생한다. 이러한 경우에 상태를 추가하여 시험 항목을 다시 생성하여 적용하면 오류를 발견하게 되지만 상태를 추가한 만큼 시험항목의 크기를 증가시키고 시험비용과 수행시간이 증가되는 단점을 갖고 있다. 그러나 시험비용과 오류 발견 능력을 높이는 것과의 상관관계에 대한 정확한 연구결과가 나와 있지 않으며 앞으로의 연구가 이루어져야 할 분야이다. UIO 방법으로 찾지 못하는 오류의 경우에 오류 발견 함수를 적용하면 오류를 찾아낼 수가 있다. 그러나 오류 모델 4개마다 서로 다른 오류 발견함수를 정의하여야 한다. 본 연구는 먼저 오류 모델 1의 출력오류에만 적용하였다.

오류 발견함수를 적용하여 출력 오류를 발견하기 위해 그림 7의 상태 V1에 대하여 오류 생성함수를 이용하여 구현물의 상태 V1이 V2로 천이할 때의 출력을 a/x를 a/y로 변경한다. 상태 V1에 대한 UIO 순서는 그림 9(a)에서 나타나 있는 c/x a/x c/y이다. 상태 V1에 대한 사양 머신(M_s)에 대한 UIO 순서 입력 순서열(α)은 c, a, c가 되고 출력 순서는 x, x, y가 된다. 상태 V1에서 V2로 천이할때 구현물(M_i)에 대한 입력 순서열은 c, a, c가 되고 출력 순서열은 y, x, y가 된다. 오류 발견함수의 정의에 따라 사양(M_s)에 입력 순서열을 적용한 결과는 x, x, y가 되고, 구현물(M_i)에 입력 순서를 적용한 결과는 y, x, y가 되어 출력결과가 일치하지 않게 된다. 이렇게 시험항목 생성 전에 오류 발견 함수를 미리 적용하게 되면 오류 발견 능력도 높이게 되고 불필요한 시험시간도 단축할 수 있게 된다.

Ⅴ. 결 론

적합성 시험이 효율적으로 이루어지기 위해서는 시험항목이 효율적으로 생성되어야 하며 시험항목의 오류 검출 능력이 뛰어나야 한다. 따라서 최적의 시험항목을 생성하고 오류 검출 능력이 최대가 되는 시험항목을 생성하는 방법이 필요하게 된다. 효율적인 시험항목 생성을 위하여 오류를 네가지 모델로 분류하였고, 오류 모델에 따라 오류 발견 함수를 정의하였다. 본 논문에서는 모든 오류 모델에 대해 오류 발견함수를 정의하지 않았고 오류 모델 1에 대해서만 정의한 적용 결과를 보였다. 점진적으로 모든 오류 모델에 대해 오류 발견함수를 정의하고 적용 예를 보일 예정이다. 오류 발견 함수를 사용하여 시험 목적에 적합한 시험항목을 생성함으로써 효율적인 적합성 시험을 수행할 수 있고, 오류 발견 능력도 향상시킴을 알 수 있었다. 앞으로의 해결과제는 본 논문의 연구에서 제시한 오류 발견 함수를 이용한 시험항목 생성 방법을 실제 프로토콜에 적용하는 문제와 자동적으로 오류 모델을 생성하여 오류 발견 함수를 적용하여 시험항목을 생성하는 방법이 연구되어져야 할 것이다.

참고문헌

1. W. J. Chun, "Test Case Generation for Protocols Specified in ESTELLE", Ph.d Thesis,

computer and infomation science, University of Delware, 1992.
2. C-J WANG and M. T. LIU, "Generating Test Cases for EFSM with Given Fault Models", IEEE INFOCOM '93, pp.774-781, 1993.
3. D. P. Sidhu, H. Motteler and R. Vallurupalli, "On Testing Hierarchies for Protocols", IEEE/ACM Transactions on Networking, VOL. 1, NO. 5, October 1993.
4. 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.
5. H. Mottler, A. Chung and D. Sidhu, "Fault Coverage of UIO-based Methods for Protocol Testing", Protocol test system, IFIP 1994.
6. Paul W. King, "Formalization of Protocol Engineering Concepts," IEEE Transactions on Computers, Vol. 40, No. 4, April, 1991.
7. Pim Kar, "Test Coverage Estimation by Explict Generation of Faulty FSMs", IWPTS VII, pp.323-330, 1994.
8. David Lee, K. Sabnani, David M. Kristol, Sanjoy Paul, "Conformance Testing of Protocols Specified as Communicating FSMs", IEEE INFOCOM'93, pp.115-127, 1993.
9. Dieter Hogrefe, "Status Report on the FMCT Project", IWPTS VII, pp.165-176, 1994.
10. D. P Sidhu, and Leung. T, "Formal Methods for Conformance Testing: A detailed Study", IEEE Transactions on Software Engineering, VOL. 15, NO. 4, April 1990.
11. A. Petrenko, N. Yevtushenko, R. Dssouli, "Testing Strategies for Communicating FSMs", IWPTS VII, pp.181-196, 1994.
12. 전우직, "적합성 시험의 형식 기법-이론과 전망", 하계 컴퓨터 통신 Workshop, 1994.
13. 허기택, "비결정성 제거에 근거한 통신 프로로토콜의 효율적인 시험 항목 생성", 광운대학교 박사학위 논문, 1994.

金 光 鉉(Gwang-hyun Kim) 정회원

1962년 2월 10일생
1989년 2월 : 광운대학교 전자계산학과 졸업(이학사)
1991년 2월 : 광운대학교 대학원 전자계산학과 졸업(이학석사)
1992년 3월~현재 : 광운대학교 대학원 전자계산학과 박사과정 수료
1991년 3월~1993년 2월 : 기전여자전문대학 전자계산과 전임강사
1993년 3월~현재 : 기전여자전문대학 사무자동화과 조교수
※주관심 분야 : 컴퓨터 네트워크, 고속 통신망, 분산 시스템

李 東 豪(Dong-ho Lee) 정회원

현재 : 광운대학교 전자계산학과 교수
통신학회논문지 제20권 제3호 참조