TTCN으로 기술된 시험 항목 검증 도구의 설계 및 구현

이 옥 빈† · 이 현 정† · 진 병 문†† · 이 상 호†††

요 약

프로토콜 적합성 시험을 효과적으로 하기 위하여 시험항목을 올바로 기술하는 것은 매우 중요하다. 시험 항목 검증을 자동화한다면 수 작업으로 할 때의 번거로움과 오류를 피할 수 있다. 이 연구에서는 이러한 수 작업으로 생성된 시험 항목의 검증을 자동화함으로써 오류를 줄이고, 적합성 시험의 생산성을 향상시키기 위한 시험 항목 검증 도구를 설계하고 이를 구현하였다. 검증대상 시험 항목은 TTCN형태이고 검증의 핵심기반은 reachability tree analysis 방법이다.

Design and implementation of validation tool for TTCN test case

Ok-Bin Lee† · Hyun-Jeong Lee† · Byoung-Moon Chin†† · Sang-Ho Lee†††

ABSTRACT

For effective protocol conformance testing, it is very important to write test cases correctly. If there is an automated test case verification procedure, we can avoid the complexity and errors-prone manual method. In this study, we propose a test case verification tool and realize it to reduce the above problems and improve quality of conformance testing. This tool is based on reachability tree and deals with TTCN form test case.

1. 서 론

통신 프로토콜은 통신 엔티티간의 정보교환을 관장하는 규칙의 집합이다. 새로운 프로토콜의 개발에 있어, IUT(Implementation Under Test)가 규격서(Specification)와 일치한다는 것을 확신하는 것은 매우 중요하며, 규격에 대해 IUT를 검사하는 절차를 적합성 시험(conformance test)이라 한다[1].

적합성 시험의 관점에서 볼 때 시험 항목의 목적은 IUT가 프로토콜 규격의 요구사항을 만족하는가를 검사하는 것이다[5]. 이와 같은 목적을 위해 시험 항목을 생성하는 것으로 상당히 많은 시험 항목이 자동화 방법이 아닌 사람의 손으로 작성되고 있는 것이 현실이다. 적합성 시험에서는 생성된 시험 항목을 이용하여 일련의 입력(input events)을 IUT에 적용하고, 그에 대한 반응(response events)을 프로토콜 규격에 따른 동작과 비교한다. 그리하여 IUT의 동작이 프로토콜 규격에 일치하고 시험목적을 만족해하면 시험결과는 Pass이다. 그러나 프로토콜 규격에 일치하지 않는다면 시험결과는 Fail이 된다. 만일 IUT의 동작이 프로토콜 규격에 일치하지만 시험목적을 만족하지 않는다면 시험결과는 Inconclusive가 된다[5].

† 준 회 원 : 충북대학교 대학원 전자계산학과
†† 정 회 원 : 한국전자통신연구원 표준연구센터장 책임연구원
††† 정 회 원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 1997년 12월 29일, 심사완료 : 1998년 10월 10일

수 작업으로 작성된 시험 항목은 때로 오류를 가지고 있다[10]. 따라서, 적합성 시험의 완벽성을 위하여 시험성 시험에 사용할 시험 항목의 성당성을 검증하는 것은 중요한 의미를 가진다. 시험 항목에 대한 검증은 시험 항목의 동작을 프로토콜 규격의 동작과 비교하여 시험 항목에 존재하는 오류를 찾아내는 것이다[10].

프로토콜 검증을 위해 일반적으로 reachability tree analysis, duologue matrix analysis, symbolic execution등의 기법이 이용된다[2,3,4,7]. 이 논문에서는 프로토콜의 설계 과정에서 설계된 프로토콜의 검증에 사용되는 이론인 reachability tree analysis를 기반으로 TTCN(Tree and Tabular Combined Notation)형태로 기술된 시험 항목 FSM(Finite State Machine)과 프로토콜 FSM의 상호 작용 분석에 의해 만들어지는 RT(Reachability Tree)를 생성하여 시험 항목의 오류를 검출하는 도구를 설계하고 구현하기로 한다. 기존의 연구[5]에서는 시험 항목 검증을 프로토콜 FSM과 시험 항목 FSM에 의해 제안되었으나, 이 논문에서는 프로토콜 FSM과 시험 항목의 TTCN을 입력받아 시험 항목을 검증하는 도구를 설계 및 구현함으로써 TTCN 형태의 시험 항목을 FSM 형태로 변경하는 번거로움과 수 작업을 줄인다는 장점이 있다.

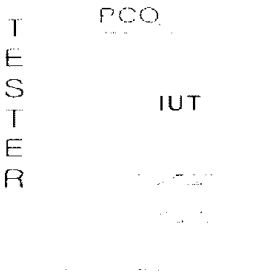
검증도구 구현 환경은 데스크 탑 컴퓨터이며 프로그래밍 언어는 C 언어로 구현한다. 또 기본 OS환경은 윈도우즈환경이다.

이 논문은 2장에서 시험 항목의 검증을 위한 이론을 제안하고, 3장에서는 검증도구의 설계를, 4장에서는 구현한 검증 도구를 이용한 실험 및 평가를 다루고 결론으로 이루어진다.

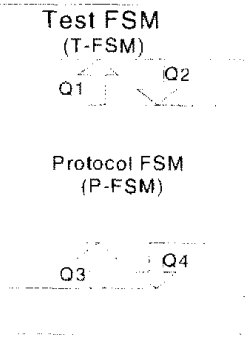
2. 시험 항목의 검증을 위한 이론 제안

2.1 개요

적합성 시험 장치(tester)는 PCO(Point of Control Observation)를 사용하여 IUT를 제어할 수 있어야 하고, 응답을 관측할 수 있어야 한다. 그러나 시험 구조(test architecture)에 따라 PCO의 구성은 여러 가지로 달라질 수 있다[1]. 이 논문에서 사용하는 시험 구조는 local test architecture를 사용하는 것으로 가정한다. Local Test architecture는 (그림 1 a)와 같으며, 검증 대상 시험 항목에 해당하는 Test FSM과 프로토콜 규격에 해당하는 Protocol FSM의 형태로 기술하면 (그림 1 b)의 구조가 된다.



a. Local Test Architecture



b. 두 FSM의 상호작용

(그림 1) Local Test Architecture와 두 FSM의 상호작용

(Fig. 1) Local Test Architecture & Interaction between two FSM

시험 항목 및 프로토콜 규격은 FSM으로 모델링된다. 이들을 각각 T FSM, P FSM 이라부르기로 한다.

여기에서 시험 항목은 TTCN의 형태로 표현되어 있으므로 FSM형태로 변환되었다고 가정한다. 그리고 각각의 PCO는 방향이 서로 다른 두 개의 큐를 사용하여 모델링하며, (그림 1 b)에서의 Q1, Q2, Q3, Q4가 PCO에 사용된 큐이다.

이 논문에 사용된 프로토콜 모델은 식(2.1)과 같이 정의되는 비결정적 FSM이다. 이와 같은 FSM으로부터 다음 상태 함수(next state function) NS는 식(2.2)와 같이 정의되고, 출력 함수(output function) Z는 식(2.3)과 같다.

FSM=(S,I,O), S:상태집합, I:입력 label, O:출력 label (2.1)

$NS : S \times I \rightarrow S$ (2-2)

$Z : S \times PCO.I \rightarrow PCO.O$ (2.3)

FSM은 유향그래프 G = (V,E)이고, 각 간선은 I/O로 레이블링되며, 각 정점은 상태를 표시한다.

식 (2-3)에서와 같이 FSM에서 관측 가능한 외부 행위는 PCO, 방향, 입출력 레이블의 3개 요소를 사용하여 표현된다. 그러나 내부행위는 방향이나 채널과는 관계가 없다.

2.2 Global state의 구조

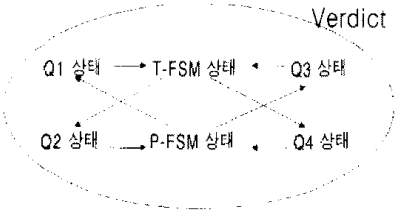
시험 항목 검증기(TCV : Test Case Verifier)는 식 (2-4)와 같이 3개의 튜플로 표현 가능하다.

TCV = { T,P,C } T:T-FSM P:P-FSM C:T와 P를 연결하는 채널 (2-4)

이와 같은 개념을 (그림 1-b)에 나타나 있으며, TCV의 임의의 상태를 나타내는 global state g는 다음과 같이 표현된다.

$$g \in G \subseteq \{state(T) \times state(P) \times state(Q1) \times state(Q2) \times state(Q3) \times state(Q4) \times Verdicts\} \quad (2\text{-}5)$$

TCV에서의 global state를 그림으로 나타내면 (그림 2)와 같다.



(그림 2) global state의 구조
(Fig. 2) Structure of global state

2.3 판정 분석

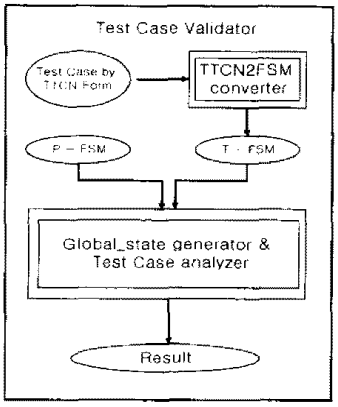
검증 대상인 TTCN으로 표현된 시험 항목의 판정(verdict)은 T-FSM에 표현되며, 궁극적으로 TCV가 생성한 RT(Reachability Tree)의 터미널 경로에 나타나게 된다. 판정에 대한 평가기준은 ov(old verdict), pv(present verdict)을 이용하여 표현하며, ov는 이전 global state의 평가 값을, pv는 생성된 global state의 평가 값을 의미한다. 초기상태의 global state에 ov로써 'none'을 입력한다. 초기상태로부터 천이가 이루어지며 생성되는 global state들 중에서 평가가 존재하지 않는 P-FSM의 입력에 의한 판정평가는 ov를 'none'로 갱신하며, T-FSM의 터미널 경로에 의해 global state가 생성되는 경우에 T-FSM의 평가 값으로 pv값을 입력받으며 이를 ov값과 비교하여 다음과 같이 평가한다.

```
function (ov,pv){
    if(ov == 'none' then return(pv)
    else {
          if(ov == Fail) and (pv == Fail) then return(Fail)
          else if(ov == Pass) and (pv == Inconc.) then
          return(Inconclusive)
          else if(ov == Inconc.) and (pv == Inconc.) then
          return(Inconclusive)
          else if(ov == Pass) and (pv == Pass) then
          return(Pass)
         }
end
```

위의 평가 기준에 의해 생성된 모든 RT의 터미널 global state는 평가 값들을 갖게 된다. 즉, RT의 터미널 global state가 Pass로 평가된 경로는 오류가 없는 error free global state로, Fail로 평가된 global state는 오류가 존재하는 erroneous global state로 검증된다.

3. 시험 항목 검증도구 설계

시험 항목 검증도구는 아래의 (그림 3)과 같은 구조를 가지며, 크게 TTCN2FSM convertor, Global_state generator & Test Case Analyzer로 구성된다. 이 연구에서는 프로토콜 규격인 P-FSM은 자기 순환(self-loop) 경로가 존재하지 않고, 검증 대상 시험 항목은 매크로(macro)를 포함하지 않는다는 가정 하에 검증도구를 설계하였다.



(그림 3) 시험 항목 검증도구의 구조
(Fig. 3) Structure of test case validation tool

시험 항목 검증도구의 개요는 다음과 같다. TTCN2FSM converter에서 TTCN의 형태로 표기된 시험항목이 FSM으로 변환된다. Global_state generator & Test Case Analyzer는 2단계로 이루어진다. 첫 번째 단계에서는, perturbation 알고리즘을 사용하여 모든 도달 가능한 global state를 생성한다. 각각의 global state는 각 FSM의 상태와 채널들의 상태로 구성된다. 두 번째 단계에서는 시험 항목의 오류에 대한 분석이 이루어진다.

3.1 TTCN2FSM convertor

위의 알고리즘은 TTCN표현으로 입력된 시험 항목을 입력받아 FSM형식으로 변환한다. TTCN표현은 한 경로가 끝날 때까지 일정한 간격으로 들여 쓰기가 되어 있으므로 그 경로가 끝나게 되면 들여 쓰기 간격이 달라지게 되어 다른 경로로 바뀌었음을 구별하며, 위 알고리즘에서는 flag를 사용하여 경로가 바뀌었음을 관리한다. 알고리즘의 개요는 다음과 같다. TTCN으로 표현된 내용을 한 라인씩 입력받아 현재상태, 입력, 출력, 다음상태의 형식을 가진 FSM으로 변환한다. "?"는 입력, "!"는 출력을 나타내므로, "?"가 나오면 "?"다음 값을 입력 값으로 저장하고, "!"가 나오면 "!"다음 값을 출력 값으로 저장한다. TTCN의 한 라인을 처리한 후 들여 쓰기 된 다음 라인을 읽어, 공백을 제거하고 위 과정을 되풀이한다. 들여 쓰기 간격이 변하면 한 경로가 끝나고 다음 경로가 계속된다.

<알고리즘 1> TTCN을 FSM으로 변환하는 알고리즘

<Algorithm 1> An Algorithm of altering TTCN into FSM

```
TTCN_to_FSM()
{
        fgets(tt[i],MAX2-1, ttcn );  // ttcn 파일을 한 라인씩 읽어서 tt[i]에 저장
        ttcnsize = i 1;
        if (flag  0){
                for(i=0 ; i<ttcnsize ; i++)          {
                        if (strncmp(tt[i] , c, 2)   !=  0){
                                state[i][0]= 1;          // state[i][0]은 현재 경로 번호 저장
                                state[i][3]= ++strnum; //state[i][3]은 다음 경로번호 저장
                        }  // tt[i]가 공백이 아니면 FSM 변환을 시작한다.
                        else{
                        ...
                        }//tt[i]에 공백이 있을 때 공백을 제거하고 FSM으로 변환
                if( tt[i][2*j] == '!')  // ! 는 출력임을 표시{
                        state[i][1] = 0 ;                          // state[i][1]에는 입력을 저장
                        state[i][2] = atoi(tt[i]+2*j+1) ;}     // state[i][2]에는 출력을 저장
                else if( tt[i][2*j] == '?')  // ?는 입력임을 표시{
                        state[i][1] = atoi(tt[i]+2*j+1) ;     // ?이므로 입력값 저장
                        state[i][2] - 0 ;}                     // 출력이 없으므로 출력에는 0을 저장
                if (state[i][3]   state[i][0] > 1){
                        flag=1;
                        break;
                }// 다음 경로와 현재 경로의 차이가 1을 초과하면 이 경로는 끝이므로
                //   다음 경로를 찾는다.
        }  // for i
} // if
        if(flag==1)  // 다음 경로 찾기
        ...
        }  // if
} // main
```

3.2 Global_State Generator & Test Case Analyzer

위의 알고리즘에서 Excutable transition( ) 함수부분에서는 입력된 두 개의 T-FSM과 P-FSM으로부터 생성될 수 있는 모든 global state를 생성한다. 여기에서 각각의 FSM은 하나의 입력이나 출력레이블만을 갖는다. 우선 초기 상태에서 입력된 두 개의 T-FSM과 P-FSM에서 천이 가능한 경우를 찾는다. 이러한 천이 가능한 경우들이 각각 global state를 생성하게 된다. 생성된 global state는 Insert()함수에서 중복인지 아닌지와 중복이라면 어떤 중복인지를 체크한다. 판정함수는 old verdict와 Insert()함수에서 입력을 받은 present verdict로 구성한다. 이전에 판정된 값과 입력되면서 판정된 값에 의한 판정함수로 시험 항목의 오류 여부에 대한 판정이 이루어진다.

<알고리즘 2> Global_State Generator & Test Case Analyzer 알고리즘

<Algorithm 2> Global-state Generator & Test case Analyzer algorithm

```
Excutable transition( )
T_GS - 임의의 자료구조;{
         char state_a
         char state_b
         char channel_a
         char channel_b}  // 자료 구조
for (End of Fsm_table)          {
                  T_GS = S[i];  //i=0에서부터 증가
                  if (Fsm_table_1의 이전상태 == S[i]의 이전상태)
                       && (Fsm_table_1의 Edge상태 == Send)){
                            T_GS.state_a = Fsm_table_1의 나중상태;
                            for(j=0; j<채널의 최대크기; j ++ ){
                               if((T_GS.channel_b[j] == '0') {
                                      T_GS.channel_b[j] = Fsm_01[t].Edge_value_01[0];
                                      break; }
                            }
                            T_GS.Proc = '2';  //T_GS의 상태가 천이 전인지 후인지 표시
                            Insert();  // 찾아낸 global_state를 입력하고 파악하는 함수
                  }
                  if (Fsm_table_1의 이전상태 == S[i]의 이전상태)
                       && (Fsm_table_1의 Edge상태 == Receive)
                     && (Fsm_table_1의 Edge값 == GS의 채널값)) {
                            T_GS.state_a = Fsm_table_1의 나중상태;
                            T_GS.channel_a의 값 제거;
                            T_GS.Proc = '2';   //T_GS의 상태가 천이 전인지 후인지 표시
                            Insert(); // 찾아낸 global_state를 입력하고 파악하는 함수
                  }
                  if (Fsm_table_1의 이전상태 == S[i]의 이전상태)//Edge의 값이 없을 때
                       && (Fsm_table_1의 Edge상태 == Blank))  {
                            T_GS.state_a = Fsm_table_1의 나중상태;
                            T_GS.Proc = '2';  //T_GS의 상태가 천이 전인지 후인지 표시
                            Insert(); // 찾아낸 global_state를 입력하고 파악하는 함수
                  }
         }
Insert( )
{
        임의의 global_state를 자료에 입력;
        자료의 상태 파악;
}
```

4. 실험 및 평가

이 논문에서는 제안 시험 항목 검증도구의 타당성을 보이기 위하여TP-2(transport protocol class 2)를 이용한다. 실험1에서는 오류가 없는 시험 항목을 입력하여 실험하고, 실험2에서는 실험 1에서 사용한 시험 항목의 판정부분에 오류를 추가하여 검사한다.

4.1 실험1

실험1에서는 오류가 없는 TTCN표현의 시험 항목을 이용하여 검증한 과정을 기술하기로 한다.

4.1.1 Test Case의 TTCN표기

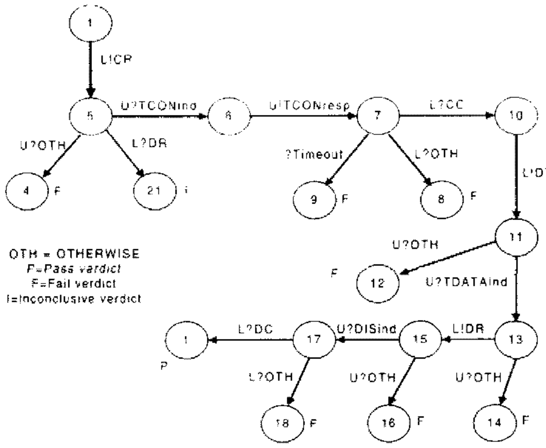
(그림 4)는 검증대상 시험 항목에 대한 TTCN 표현이다.

```
L!CR
 Start(A)
   U?TCONind
     U?TCONresp
        L?CC
          Cancel(A)
            L!DT
              U?TDATAind
                L?AK
                  L!DR
                    U?TDISind
                      L?DC                Pass
                      L?OTHERWISE         Fail
                    U?OTHERWISE           Fail
                L?OTHERWISE               Fail
              U?OTHERWISE                 Fail
        ?Timeout                          Fail
        L?OTHERWISE                       Fail
   L?CR                                   Inconclusive
   U?OTHERWISE                            Fail
```

(그림 4) Basic Interconnection Test Case
(Fig. 4) Basic interconnection Test case

4.1.2 T-FSM

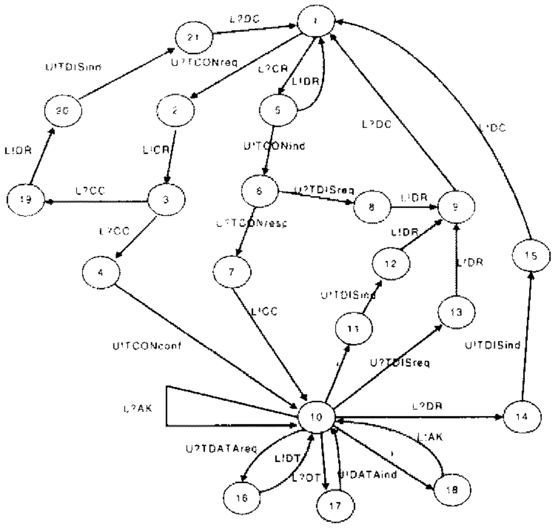
(그림 4)의 TTCN표기는 TTCN2FSM converter에 의해 (그림 5)과 같이 T-FSM으로 변환된다.



(그림 5) 그림 4에 의한 T-FSM
(Fig. 5) T-FSM of Fig. 4

4.1.3 P-FSM

(그림 6)은 TP-2의 P-FSM이다. 여기서는 각 상태에서 다음상태로의 천이에서 하나의 입력과 출력만을 고려하였다[5].



(그림 6) TP-2에 대한 P-FSM
(Fig. 6) P-FSM of TP-2

4.1.4 실험결과

(그림 5)의 검증 대상 시험 항목과 (그림 6)의 프로토콜 FSM자료를 이용하여 생성한 RT는 터미널 경로로서 9개의 global state가 생성되었으며, 이중 error-free 경로는 2개 erroneous 경로는 7개이었다. TCV에 의하여 생성된 RT의 판정 분석에 의한 검증 결과가 <표 1>에 있다.

검증결과에서 error-free경로는 E_F로 erroneous는 E_N으로 표기한다.

〈표 1〉 시험 항목 검증도구에 의한 검증 결과

〈Table 1〉 Validation result for test case validation tool

경 로	시험 항목의 판정 값	TCV의 검증 결과
1->5->6->7->10->11->13->15->17->1	Pass	E_F
1->5->6->7->10->11->13->15->17->18	Fail	E_N
1->5->6->7->10->11->13->15->16	Fail	E_N
1->5->6->7->10->11->13->14	Fail	E_N
1->5->6->7->10->11->12	Fail	E_N
1->5->6->7->8	Fail	E_N
1->5->4	Fail	E_N
1->5->21	Fail (Inconclusive)	E_N

생성된 RT의 터미널 global state를 상세히 분석한 내용은 다음과 같다. TCV의 검증 결과인 RT는 두 개의 Pass평가가 나타난 터미널 global state(두 경로는 동일한 상태 값을 가지고 있어 하나의 경로로 표현되어 있음)와 7개의 Fail평가가 나타난 터미널 global state로 구성되어 있다. 여기에서 판정이 Pass인 global state는 오류가 존재하지 않는 error free global state이고, 판정이 Fail인 global state는 오류가 존재하는 erroneous global state이며, Fail의 종류로는 queue overflow, Inconclusive, deadlock이었다. 본 실험에서는 대부분의 Fail은 deadlock이며 하나의 Inconclusive를 얻었다. 그러므로 이 실험 결과는 검증대상 시험 항목이 올바로 작성된 것임을 나타낸 것으로 이는 실험에 사용한 시험 항목이 오류가 없는 정확한 것이기 때문이다.

```
L!CR
 Start(A)
   U?TCONind
     U?TCONresp
         L?CC
           Cancel(A)
             L!DT
               U?TDATAind
                 L?AK
                   L!DR
                     U?TDISind
                       L?DC              Pass
                       L?OTHERWISE       Fail
                     U?OTHERWISE         Pass    --①
                 L?OTHERWISE             Fail
               U?OTHERWISE               Pass    --②
         ?Timeout                        Fail
         L?OTHERWISE                     Fail
     L?CR                                Pass    --③
     U?OTHERWISE                         Fail
```

(그림 7) 오류를 포함한 시험 항목

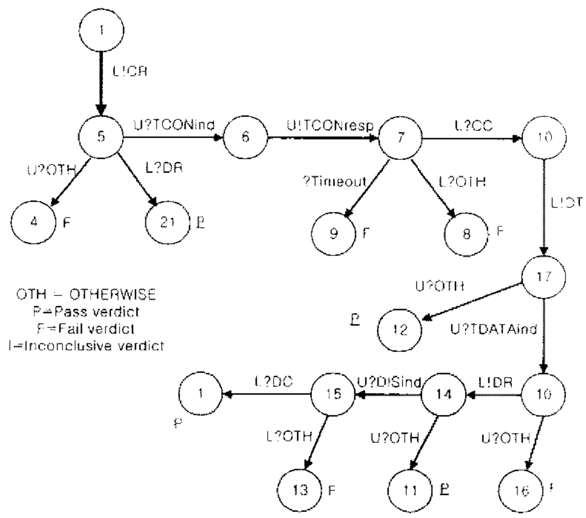
(Fig. 7) Test case including error

4.2 실험 2

(그림 5)의 시험 항목에 다음 (그림 7)에서 밑줄친 부분과 같이 세 부분의 오류가 있는 TTCN표기의 시험 항목을 만들어 이를 이용하여 실험하기로 한다.

(그림 7)의 시험 항목을 변환하여 다음 (그림 8)과 같은 FSM을 얻는다.

검증 대상 시험 항목의 변환된 결과인 (그림 8)의 11,12,21번 경로에서 의도적으로 발생시킨 오류인 Pass라는 결과를 볼 수 있다. (그림 6)과 (그림 7)의 내용을 이용하여 실험한 결과가 <표 2>이다.



(그림 8) 오류를 포함한 시험 항목의 T-FSM

(Fig. 8) T-FSM of test case with error

〈표 2〉 잘못된 시험 항목에 대한 검증 결과

〈Table 2〉 A validation result for fulty result test case

경 로	시험 항목의 판정 값	TCV의 검증 결과
1->5->6->7->10->11->13->15->17->1	Pass	E_F
1->5->6->7->10->11->13->15->17->18	Fail	E_N
1->5->6->7->10->11->13->15->16	Pass	E_N
1->5->6->7->10->11->13->14	Fail	E_N
1->5->6->7->10->11->12	Pass	E_N
1->5->6->7->8	Fail	E_N
1->5->4	Fail	E_N
1->5->21	Pass	E_N

생성된 RT의 터미널 global state를 상세히 분석한 내용은 다음과 같다. 이 실험에서도 RT는 네 개의 Pass평가가 나타난 터미널 global state(두 경로는 동일

한 상태 값을 가지고 있어 하나의 경로로 표현되어 있음)와 네 개의 Fail평가가 나타난 터미널 global state로 구성되어 있다. 검증 대상 시험 항목인 (그림 7)의 밑줄 친 부분의 판정 값은 Pass이나, TCV와 판정 결과는 E_N(erroneous)이므로 주어진 시험 항목 (그림 7)의 밑줄 친 부분에 오류가 포함되었음을 알 수 있다.

만일 (그림 7)의 밑줄 친 부분①에서 U?OTHERWISE 부분이 U?CC이고 평가 값이 Fail이라면 U?CC역시 더 이상의 천이가 이루어 질 수 없으므로 Fail로 평가되는 것은 당연하다. 그러나 이러한 시험 항목에 대한 오류 검사는 TCV에서는 검증되지 않는 한계를 가지고 있다.

5. 결론 및 향후 연구

프로토콜 개발 과정에 있어 프로토콜 구현이 규격과 일치한다는 것을 확신하는 것은 매우 중요하다. 그러나 실제 적합성 시험 과정에서 실제로 오류가 발생하는지, 발생한다면 어디에서 발생하는지 결정하는 것은 매우 어려운 일이다. 또한 수 작업으로 작성된 시험 항목은 때로 오류를 가지고 있을 가능성이 있다. 따라서 프로토콜 적합성 시험을 위한 시험 항목의 정당성을 검증하는 것은 매우 중요한 것으로, 시험 항목에 대한 검증은 시험 항목의 동작을 프로토콜 규격의 동작과 비교하여 시험 항목에 있을 수 있는 오류를 찾아내는 것이다.

이 연구에서 제안한 시험 항목 검증도구는 TTCN으로 표현된 시험 항목을 검증하는 검증도구다. 이러한 검증도구를 이용함으로써 수 작업에 의해 작성된 시험 항목의 오류를 찾아 적합성 시험의 수준을 향상시킬 수 있다. 실험에서와 같이 TCV는 시험 항목의 오류의 위치를 판단할 수 있다. 이러한 기능을 갖는 검증도구의 구현은 첫째 적합성 시험의 완벽성에 기여, 둘째 자동화, 셋째 TTCN으로 표현된 시험 항목의 검증에 활용가능 등의 이점을 제공한다. 하지만 제안 검증도구는 timeout에 관한 처리 기능을 갖고 있지 않으며, 실험2에서 언급한 바와 같이 전달메시지 자체가 잘못된 경우 판정이 Fail이면 메시지에 대한 오류를 발견하지 못한다. 향후 연구에서는 이러한 timeout에 관한 처리부분과 전달메시지 내용의 오류판정에 대해서도 연구하고자 한다.

참 고 문 헌

[1] OSI Conformance Testing Methodology and Framework : DIS 9646.

[2] C.H.West. General Techniques for Communications Protocol Validation. IBM Journal of Research and Developments. Vol.22, No.4, July 1978, pp.393-404.

[3] P.Zafiropulo, et al. Towards Analysing and Synthesizing Protocols. IEEE Tran. on Comm., Vol.COM-28, No.4, pp.651-661, April 1980.

[4] S.T Voung, D. D. Hui, and D. D. Cowan. VALIRA - A Tool for Protocol Validation via Reachability Analysis. Proc. of the IFIP WG6.1 6th. International Workshop on Protocol Specification, Testing, and, Verification, VI Sarikaya and Bochmann (deitors), North Holland, 1987, pp.35-41.

[5] K.Naik and B.Sarikaya. Verification of Protocol Conformance Test Cases Using Reachability Analysis. The Journal of systems and software, pp.41-65, September 1992.

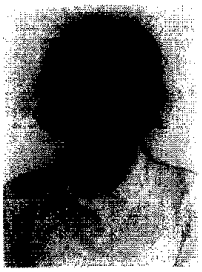
[6] B. Sarikaya. Principles of Protocol Engineering and Conformance Testing, 1993, pp.31-54.

[7] P. Zafiropulo,. Protocol Validation by Duologue-Matrix Analysis, IEEE Trans. on Comm., Vol. COM-26, No.8, pp.1187-1194, Aug., 1978.

[8] S. H. Woo, B. H. Oh, and S.H. Lee. Design of a Test Suite Validator for Protocol Conformance Testing, 한국통신학회논문지, Vol.20, No.6, pp.153-163, June 1995.

[9] K.Sabnani and A.Dahbura. A Protocol Test Generation Procedure, Computer Networks and ISDN System 15, 1988, pp.285-297.

[10] Son. T. Vuong, Sang Ho Lee, and Peter Zhou, Protocol Test Validation : Principles, Tools, and Examples, Invited Paper, Proc. of CFIP'93, Montreal, Canada., Apr., 1993.



이 옥 빈

e-mail : lobin@cbucc.chungbuk.ac.kr

1990년 조선대학교 전자계산학과 졸업(학사)

1993년 조선대학교 대학원 전자계산학과 졸업(석사)

1995년 ~ 현재 충북대학교 대학원 전자계산학과 박사과정

관심분야 : 컴퓨터 네트워크, 프로토콜 공학, 인터넷 응용

이 현 정

e-mail : hyunjlee@trut.chungbuk.ac.kr

1997년 충북대학교 컴퓨터과학과 졸업(학사)

1997년 ~ 현재 충북대학교 대학원 전자계산학과 석사과정

관심분야 : 컴퓨터 네트워크, 프로토콜 공학 인터넷 응용

진 병 문

e-mail : bmchin@pec.etri.re.kr

1976년 서울대학교 공과대학 전기공학과 졸업(학사)

1983년 서울대학교 공과대학원 전자계산기공학과 졸업(공학석사)

1996년 한국과학기술원 전산과 졸업(공학박사)

1980년 ~ 현재 한국전자통신연구원 표준연구센터장(책임연구원)

관심분야 : 프로토콜 공학, 인터넷, ATM, 정보통신 네트워크, 시험 기법

이 상 호

e-mail : shlee@cbucc.chungbuk.ac.kr

1976년 숭실대학교 전자계산 공학사

1981년 숭실대학교 대학원 시뮬레이션 공학석사

1989년 숭실대학교 대학원 컴퓨터네트워크 공학박사

1979년 한국전력 전자계산소 프로그래머

1981년 ~ 현재 충북대학교 컴퓨터과학과 교수

관심분야 : Protocol Engineering, Network Security, Network Management, Network Architecture