論 文

깊이트리를 이용한 효율적인 프로토콜 시험항목 생성

正會員 許 基 澤* 正會員 李 東 豪**

A Effective Generation of Protocol Test Case Using The Depth-Tree

Gi Taek Hur*, Dong Ho Lee** *Regular Members*

요 약

프로토콜의 적합성시험은 컴퓨터 통신에서 상호운용성과 비용의 효율성을 높이기 위해서 매우 중요하다. 적합성 시험은 구현된 내용이 프로토콜 규격에 적합하게 구현되었는지를 검사하는 것으로, 그것의 효율성과 오류검출능력은 시험 항목의 생성방법에 의해서 결정된다. 프로토콜이 유한상태기계로 표현될때 한상태에 여러개의 UIO(Unique Input Output)순서들이 존재할 수 있으므로 이들중 가장 적합한 순서를 선정함으로서 시험 길이를 최소화 할 수 있다. 따라서 본 논문은 시험 길이를 최소화하기 위해서 여러개의 UIO순서들간에 존재하는 최대중복성을 찾기 위한 알고리즘을 깊이트리를 이용하여 구성하였고, 이 알고리즘을 이용하여 최소길이의 시험 순서를 생성하는 예를 보여 주었다.

Abstract

Protocol conformance is crucial to inter-operability and cost effective computer communication. Given a protocol specification, the task of checking whether an inplementation conforms to the specification is called conformance testing. The efficiency and fault coverage of conformance testing are largely dependent on how test cases are chosen. Some states may have more one UIO sequence when the protocol is represented by FSM (Finite State Machine). The length of test sequence can be minimized if the optimal test sequences are chosen. In this paper, we construct the depth-tree to find the maximum overlapping among the test sequence. By using the resulting depth-tree, we generate the minimum-length test sequence. We show the example of the minimum-length test sequence obtained by using the resulting depth-tree.

I. 서 론

복잡한 분산시스템의 동작은 일반적으로 프로토콜이라는 일련의 규칙들로 서술되고 분석될 수 있다[5, 11]. 프로토콜은 확장된 유한상태기계(EFSM : Extended Finite State Machine)로 모델화 될 수 있으며, 프로토콜의 제어부분은 유한상태기계로 서술되고 자료부분은 프로그램 세그멘트로 서술된다[1,3,4,10, 11,12]. 프로토콜의 적합성시험은 분산시스템과 통신시스템을 검증하고 확인하는데 중요한 역할을 수행한

* 東新大學校 電子計算學科
Dept. of Computer Science Dongshin University
** 光云大學校 電子計算學科
Dept. of Computer Science Kwangwoon University
論文番號 : 93-141

다. 프로토콜을 시험하는 방법중에서 가장 잘 알려진 것이 순회(Touring)방법이다[12]. 이는 유한상태기계의 모든 천이들을 적절한 시험 보조순서를 이용하여 에지들을 순회하면서 프로토콜의 전체동작을 확인하는것으로, 상태확인과정(State Identification)이라는 것을 사용하여 시험 보조순서(Test Subsequence)를 생성하고, 이를 이용하여 주어진 상태의 특성들을 유일하게 구별한다. 이를 위해서 유한상태기계를 기초로한 시험 생성방법은 두단계로 구성된다. 첫번째 단계는 유한상태기계에 있는 각각의 상태를 다른 상태들과 구별하기 위해서 필요한 상태확인과정이라는 어떤 특수한 입출력을 유도하는 단계로서 Distinguishing Sequence(DS), Characterizing Set, UIO 순서등의 방법이있다. 두번째단계는 시험 보조순서들을 접속시킴으로서 시험 순서를 형성하는 단계로서 각 시험 보조순서는 천이의 끝상태를 위한 상태확인과정을 천이 바로 다음에다 접속시킴으로서 형성된다[1,3,4,8,10,12,13].

따라서 효율적인 적합성 시험이 이루워지기 위해서는 모든 천이들을 순회하는 시험순서의 길이가 짧아야한다[1,3,4,9,12,13]. 본 논문은 이를 위해서 가장짧은 시험순서를 생성하기 위해서 한 상태에 존재하는 여러개의 UIO순서중 가장 짧은 시험순서를 선정하는것은 시험 보조순서간의 최대중복성을 찾는것과 동일하므로 이를 위한 깊이트리 구성 알고리즘을 제안하였다. 깊이트리에 포함된 패스중 가장 깊이가 깊은 패스가 천이간에 최대 중복성을 갖는 것이므로, 이를 이용하여 시험 순서의 길이를 최소화 할 수 있는 방법을 제시하였다.

Ⅱ. 프로토콜 시험방법

프로토콜은 결정적 유한상태기계(Deterministic Finite State Machine)인 5개 쌍인 (S, I, O, NS, Z)로 명세될 수 있는데 여기서 $S=\{S_1,\ldots,Sn\}$은 유한상태의 집합을, $I=\{I_1,\ldots,I_m\}$은 유한입력집합을 그리고 $0=\{0_1,\ldots,0_N\}$은 유한출력집합을 나타낸다. 유한상태기계는 두개의 함수인 다음상태 $NS: S\times I\rightarrow S$와 출력 $Z: S\times I\rightarrow 0$에 의해서 동작한다. 유한상태기계는 방향성그래프인 $G=(V, E)$로 표현되고, 집합 $V=\{V_1,\ldots,V_N\}$은 유한상태 기계의 명세된 상태의 집합 S를 나타내고, 레벨 $L=I_k/0_l$로 표현된 방향성 에지 (vi, vj)는 ((vi, vj : L)로 표현됨) 상태 S_i에서 상태 S_j로의 천이를 나타낸다[2,7,8,10,12].

유한상태기계의 구현은 일반적으로 순회방법을 사용하여 시험 될 수 있다. 이 방법에서 상태확인과정은 유한상태기계의 모든 상태를 구별하기 위해서 사용되는 입출력 순서이다. 입출력 순서는 $(i_1/o_1)(i_2/o_1)\ldots.(i_n/o_n)$ 형태로 표현되고, 개개의 입출력순서에 대해서 유한상태기계의 구현이 올바른가의 검증은 입력 i_n에 대해서 출력이 o_n인가를 확인하는 과정으로 볼 수 있다. DS는 유한상태기계의 모든 초기상태를 위해서 서로 다른 출력순서를 생성하는 방법이다. 다른 순회방법으로 Chacterizing Set이 있다. 이것은 입력스트링들의 집합으로서 유한상태기계의 모든상태에 이 입력스트링을 적용한 경우에 마지막 출력값들이 서로 다르게 생성된다. 이 방법에서 상태확인과정은 입출력순서로 구성되고, 각 순서는 Chacterizing set의 한 원소를 검사한다. 이 방법의 Fault Coverage가 순회방법중에서 가장 낮은 것으로 판명되었다[8]. 새로운 방법이 [8]에서 제안되었다. 이것은 UIO순서의 사용을 기초로 하는데, 상태 S_i의 UIO순서는 상태 S_i를 다른 모든 상태로부터 유일하게 구분할 수 있는 것을 말한다.

유한상태기계의 모든 천이를 시험할때 최소비용의 시험 순서를 찾는 문제는 Rural Chinese Postman Tour(RCPT)와 같다[1]. UIO순서는 거의 모든 유한상태기계에서 발견할 수 있지만, DS는 그렇지 못하다. DS는 UIO순서의 특수한 경우이다는 것이 증명되었고, UIO순서의 길이가 DS의 것보다 훨씬 더 짧다는 것이 증명되었다[1,3,8]. 따라서 본 논문도 현재 가장 많이 사용되고 있는 UIO순서를 사용하여 시험 보조순서를 생성하였다.

Aho et al.[1]은 두단계로 구성된 시험 생성방법중 첫번째 단계에서 각 상태를 위한 최소길이 UIO순서를 사용하고, 두번째 단계에서는 reset or selp loop를 갖는 유한상태기계를 위해서 RCPT를 찾는 알고리즘을 사용함으로서 시험순서 길이를 줄였다. Shen et al's의 방법은 Aho et al.방법을 다중최소길이 UIO순서를 갖는 방법을 사용하여 시험 순서의 길이를 4%에서 37%까지 줄였다[13]. 그러나 이 방법들은 시험 보조순서를 중복시키지 않고도 시험순서의 길이를 그만큼 줄인것이다. Bo Yang과 Hasan Ural은 다중 UIO순서중 중복성을 갖는 시험 보조순서를 찾기 위해서 경험적(Heuristic)방법을 사용하였고[3,4], Choi는 적응적(Adaptive)프로시듀어를 사용하여 시험 순서의 길이를 줄였다[9].

그림1에서처럼 TEST(v_1, v_2 ; b/x) = b/x b/y이고

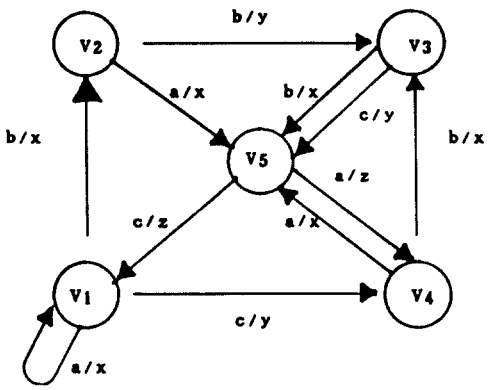


그림 1. 유한상태에 대한 그래프의 표현
Fig 1. A Graph Representation of a FSM

TEST(v_2, v_3 ; b/y) = b/y b/x c/z일때 시험 보조순태가 v_2이고 TEST(v_2, v_3 ; b/y)의 HEAD 상태가 v_2로 같으므로 이를 중복 시킬 경우에는 입력순서가 b/x b/y b/x c/z가되어 시험 순서의 길이를 하나 줄일 수 있다. 따라서 시험 순서의 길이를 최소화 하는것은 시험 보조순서간에 중복성을 극대화 하는 것과 일치한다. 따라서 이를 찾기위해서 시험 보조순서들이 연결되면서 시험 순서를 형성하므로 한 상태에 존재하는 여러개의 UIO순서중에서 최대중복성을 갖는 UIO순서를 선정하여 접속하면 최소길이의 시험순서가 된다. 본 논문에서는 시험 보조순서간의 최대중복성을 찾기 위해서 깊이트리를 구성하였고, 깊이트리 중 깊이가 가장 큰 패스가 최대중복성을 갖는 시험 보조 순서가 되므로 이를 찾기 위한 알고리즘을 제시하였다.

Ⅲ. 깊이트리 구성 알고리즘

[12]에서는 프로토콜의 전체 시험 순서를 구성하기 위해서 필요한 모든 시험 보조순서들간의 접속 방법으로 최적화 방법을 제시하였다. 이 최적화 방법은 상태확인과정과 함께 RCPT알고리즘을 이용하였다. 상태확인과정은 유한상태기계의 어떤 다른 상태로부터 한 상태를 구별할 수 있는 능력을 갖는다. 상태확인과정인 UIO순서가 유한상태기계에 있는 각 에지의 부분 동작을 검증하기 위해서 사용된다. 이때 최소길이의 시험 순서를 생성하기 위해서는 부분 동작을 확인하기 위해서 사용되는 UIO순서가 짧아야 한다. 따라서 본 논문에서 제안한 깊이트리를 이용한 최소길이의 시험 순서를 얻기위한 과정은 다음 3단계로 구성된다.

단계1 : 그래프 G = (V, E)에 있는 모든 상태들에 대한 최소길이의 UIO순서를 구한다. 만약 이때 UIO순서를 갖지 않는 상태가 존재하면 시험 능력을 향상시키기 위해서 PUIO (Partial UIO)를 사용하였다.

단계2 : 단계1에서 구해진 최소길이의 UIO순서들 중 최대중복성을 갖는 UIO순서들을 찾기 위한 길이트리를 구성한다.

단계3 : 단계2에서 구해진 깊이트리를 이용하여 최소길이의 시험 순서를 생성한다.

3.1 최소길의 UIO순서 생성 알고리즘

적합성 시험은 종종 Black Box 시험으로서 수행되므로 테스터는 IUT(Implementation Under Test)의 내부동작에 관한 정보를 가지고 있지 않다. 그래서 IUT의 동작은 IUT에 적용될 다음 단계의 입출력 순서에 의해서 확인될 수 있다[1,7,10].

1. IUT를 일련의 입력 순서를 적용시켜 원하는 원시상태 g_s로 옮긴다.
2. IUT에 가능한 입력중 하나를 적용시키고 출력을 확인한다.
3. 목적상태가 맞는가를 검증한다.

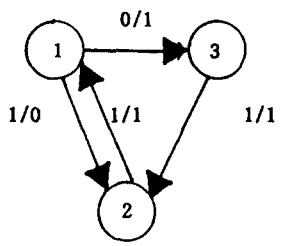
구현된 내용이 기대되었던 상태에 있는가를 확인하기 위해서 상태확인과정이 필요하다. 상태확인과정은 기계의 내부상태를 확인할 수 있는 입출력 순서를 적용시킴으로서 이루워진다. 본 연구에서는 UIO순서를 상태확인과정으로 사용하였다.

한 상태를 위한 UIO순서는 어떤 다른 상태에서는 볼 수 없는 입출력 순서이므로 천이의 목적상태를 다른 상태로부터 구별하기 위해서 시험순서 생성시 사용될 수 있다. 각 천이 $g_s \xrightarrow{i/o} g_d$를 위한 시험 보조 순서는 TEST($g_s$, g_d ; i/o) = Spath ‖ i/o ‖ UIO(g_d)로 정의된다. 여기서 Spath(g_s)는 초기상태에서 상태 g_s까지의 가장 짧은 패스를 나타내고, i/o는 입출력을, UIO(g_d)는 목적상태를 확인하기위한 UIO순서를 나타낸다.

한 상태가 여러개의 UIO순서를 가질 수 있으므로

```
Algorithm1 (UIO/PUIO sequence generation for state g)
 /* let a vertex vσ be a tuple ⟨gσ, Pσ , Eσ ⟩ where
  gσ is the state that results from state g after firing sequence σ ,
  Pσ is the set of states that are reachable by σ any state in ( G - {g} - Eσ ),
  Eσ  is the current set of states which prevent σ  from being a UIO sequence.
  let PUIOset be a set of tuples ⟨PUIO sequence, exclusion set⟩;*/
(1)  put a vertex vλ =⟨g,G-{g},∅ ⟩ into a queue OPEN and a list VISITED;
(2)  PUIOset ⟨---∅  ;
(3)  While (OPEN is not empty) do
(4)    remove a vertex vσ  =⟨gσ , Pσ, Eσ ⟩ from the head of OPEN;
                                                  i/o
(5)    for each transition of the form  gσ ----⟩ gd  do
(6)      σnew  ⟨--- σ ¦¦ i/o ;
                                                    i/o
(7)       Pnew ⟨--- {g"¦∃ g'∈  Pσ  such that  g'----⟩ g";
                                                                  σnew
(8)       if (gd∈Pnew ) then Enew ⟨-- Eσ ∪ {g'¦∃ g'∈G - {g} such that g'----⟩gd},
                             Pnew ⟨--- Pnew - {gd};

(9)          else Enew ⟨--- Eσ  ;
(10)      vσnew  ⟨--  ⟨gd, Pnew , Enew ⟩;
(11)      if (∃ ⟨σp, Ep⟩∈  PUIOset such that Ep ⊆  Enew ) then continue ;
(12)         else if (Pnew =∅ ) and (Enew = ∅ )  then return  σnew as UIO(g);
(13)            else if (Pnew = ∅ ) then PUIOset ⟨--- PUIOset ∪ ⟨ σnew , Enew ⟩;
(14)               else if (vσnew ∈ VISITED )then continue;
(15)                  else append vσnew onto the tail of OPEN and VISITED;
(16)   done for
(17)  done while
(18)  return PUIOset as a set of tuples ⟨PUIO sequence, exclusion set⟩;
```



천이 t1: (v_1,v_3; 0/1)
t2: (v_3,v_2; 1/1)
t3: (v_2,v_1; 1/1)
t4: (v_1,v_2; 1/0)

그림 2. 단순형태의 유한상태기계
Fig 2. A Simple FSM

그들중 가장 짧은 것을 사용하는것이 시험 보조순서의 길이를 줄일 수 있으므로 효율적이다. 그리고 UIO순서를 갖지않는 상태를 처리하기 위해서 Chun이 제안한 PUIO(Partial UIO)순서를 사용하여 한 상태를 다른 상태와 부분적으로 구별하였다. 알고리즘 1은 주어진 한 상태의 UIO와 PUIO순서를 생성하기 위한 알고리즘을 기술하였다[10].
그림2에 알고리즘1을 적용시켜 나타난 결과가 표1에 있다[3,4].

표 1. 그림2에 대한 최소길이의 UIO순서
Table 1. Minimum-Length UIO Sequences for Fig.2

state	UIO
1	0/1 1/0
2	1/1 0/1 1/1 1/0
3	1/1 1/1

일단 유한상태기계의 각 상태를 위한 최소길이의 UIO순서가 찾아지면, 유한상태기계의 각 천이를 위해서 시험 보조순서가 형성될 수 있다. 천이(g_s, g_d ; i/o)를 검증하기위한 입/출력순서를 그 천이의 시험 보조순서라하고 TEST(g_s, g_d ; i/o)로 표현한다. TEST (g_s, g_d ; i/o)는 IUT를 원시상태 g_s로 옮기고, 입력 I를 적용시켜 출력이 0인가를 확인하고 UIO(g_d)를 적용시켜서 목적 상태가 옳바른가를 검증한다.

3.2 깊이트리 구성 알고리즘

[1,4,9]에서 지적된 것처럼, 시험 순서의 길이는 시험 보조순서들을 중복시킴으로서 훨씬 더 줄여질 수 있다. 표1에서 처럼 한 상태에 여러개의 UIO순서가 존재할 수 있으므로 이들 중에서 시험 순서의 길이를 최소화 할 수 있는 UIO를 선택하는 것이 바람직하다. 적합성 시험이 효율적으로 이루워지기 위해서는 유한상태기계로 표현된 그래프의 모든 에지들이 적어도 한번은 방문되어야 하고, 방문된 에지들의 수가 적어야 한다. 그런데 테스터는 적합성 시험중 IUT 내부의 동작에 관한 정보가 없으므로 기계의 내부동작이 정확히 수행되었는지를 확인하기 위해서 목적상태에 도달했는가를 검증할 수 있는 상태확인과정이 필요하다. 예를 들면 그림2에서 천이 t_2을 위한 시험 보조순서인 TEST(v_3, v_2 ; 1/1)은 천이를 기계의 원시상태인 v_3로 가져오는 입력 0/1과 원시상태 v_3에서 v_2로의 천이를 야기시키는 입출력 순서 1/1과 목적상태를 확인하기 위한 UIO(v_2) = 1/1 0/1 혹은 1/1 1/0으로 접속된 0/1 1/1 1/1 0/1 혹은 0/1 1/1 1/1 1/0 이 된다. 유사하게 천이 t_1, t_3, t_4의 시험 보조순서는 각각 0/1 1/1 1/1, 1/0 1/1 0/1 혹은 1/0 1/1 1/0, 1/0 1/1 0/1 혹은 1/0 1/1 1/0이 된다. 적합성 시험을 위해서는 이 모든 천이들이 적어도 한번은 순회되어야 한다. 이때의 시험 보조순서의 길이가 13이 된다. 그런데 상태1과 3은 UIO순서가 두개이지만 이들 중에서 목적 상태를 확인하기 위해서 UIO순서를 적용하므로, UIO순서가 다음 천이의 원시상태의 입력순서가 될 수 있는 입력값을 선택함으로서 중복성을 최대화 할 수 있다.

정의1 : (깊이트리)

유한상태기계에 있는 상태 g를 위한 깊이트리는 (V, E)로 구성된다. 여기서

1. 정점 V는 $\langle g_t, P_t\rangle$로 구성되고 g_t는 현재상태를, P_t는 g_t로부터 입출력 순서 t에 의해서 도달 할 수 있는 상태의 집합으로 표현된다.
2. 에지 E는 입출력 순서 t에 의해서 정점 v_s에서 v_d로 천이가 발생하면 $v_s \xrightarrow{t} v_d$로 표현되고, t가 UIO(v_s)의 구성 성분이 될때만 깊이트리에 삽입된다.
3. 깊이트리의 깊이 k는 에지 E의 레벨로서 루트정점에서 부터 시작해서 깊이트리에 새로운 자식 정점이 삽입될 때마다 1씩 증가된다.

정리1 :

깊이트리에서 깊이가 가장 큰패스가 천이간의 최대중복성을 갖는다.

(증명)

정의1에서 깊이트리가 $\langle V, E\rangle$로 표현되고, 정점 V는 $\langle g_t, P_t\rangle$로 구성된다. 이때 새로운 정점 $\langle g_d, P_d\rangle$가 깊이트리에 삽입되기 위해서는 $\langle g_d, P_d\rangle$를 $\langle g_t, P_t\rangle$에 연결하는 에지 E_t가 UIO(g_t)의 일부가 되어야만 깊이트리에 삽입될 수 있다. 그런데 $\langle g_t, P_t\rangle$와 $\langle g_d, P_d\rangle$의 브랏지 순서집합이 서로 중복되는 UIO순서에 의해서 연결되므로 깊이트리에서 깊이가 깊으면 깊은 패스일수록 중복성이 큰것이다. 따라서 깊이트리에서 깊이가 가장 큰 패스가 시험 보조순서간의 최대중복성을 갖는 패스가 된다.

따라서 본 논문은 정리1에서 살펴본것 처럼 시험 보조순서간의 최대중복성을 찾는것이 가장 짧은 시험 순서를 생성하는 것과 동일하다는 것을 알 수 있으므로 이를 위한 깊이트리를 구성하였다. 깊이트리에 포함된 패스중 깊이가 가장 깊은 것이 천이간의 최대중복성을 갖는 것이므로 깊이트리를 이용하여 시험순서를 최소화할 수 있는 다음과 같은 깊이트리 구성 알고리즘을 제시하였다.

Ⅳ. 적용 및 분석

알고리즘2가 수행되는 과정을 그림2에 적용시킨 결과가 그림3에 나타나 있다. 먼저 그림2의 유한상태기계의 초기상태인 1에서 입출력순서 1/0에 의해서 상태2로의 천이가 발생하고, 0/1에 의해서는 상태3으로 천이가 발생한다. 따라서 이를 위한 초기상태의 정점은 $\langle 1,\{2,3\}\rangle$이 되고 입출력 순서 1/0와 0/1에 의해서 상태 2,3으로 각각 천이가 발생하므로 정점 $\langle 2, 1\rangle$, $\langle 3,2\rangle$이 생성된다. 정점 $\langle 2,1\rangle$이 깊이트리에 삽입

Algorithm 2(Construct The Depth Tree)

```
/* Let a Vertex Vt  be a tuple ⟨ gt,Pt⟩ where
        gt is the current state
        Pt is the state that results from state gt after firing
           Sequence t.
   Let RB[n] be a Reference Bit and if transition i is referenced
        then RB[i] becomes 1.
   Let Depth[i] and NewDepth[i] represent the depth of transition i
        in the depth tree and  Variables k  are used   to represent the
        depth of transitions */

(1) Put a Vertex Vs = ⟨ gs,Ps⟩ into a queue OPEN;
(2) While(The same vertex vt do not exist  in depth tree )do
(3)    remove a vertex vt=⟨gt,Pt⟩ from the head of OPEN;
                                              i/o
(4)    for each transition i of the form  gt---⟩ gd∈Pt do
(5)    /* Check UIOS */
(6)       gnew ⟨--- gd:
                                                  s
(7)       Pnew ⟨--- {g'¦ ∃ g ∈gd such that ( g ---⟩ g' and do not exist
                     in  the same path)};
(8)       tnew ⟨--- i/o ¦¦ s;
(9)       Vtnew ⟨--- ⟨gnew,Pnew⟩;
(10)      if tnew becomes the components of  UIOS(gt) :
(11)      /* Expand the depth Tree */
(12)        Then{
(13)              Append Vtnew to the depth tree;
(14)              Compute k the depth of transition i and store k into NewDepth[i];
(15)              if RB[i]=0 then {
(16)                              RB[i]=1 and Depth[i]=k;
(17)                              Append vtnew onto the tail of OPEN;
(18)                              Continue;
(19)                             }
(20)                        else if Depth[i] ⟨ NewDepth[i];
(21)                             then {
(22)                                    delete all transitions derived from old vt
                                           in the depth tree;
(23)                                    clear the RB[i] and Depth[i] of the deleted
                                           transitions on the depth tree;
(24)                                    append newtransition i onto the tail of OPEN;
(25)                                    Depth[i]=Newdepth[i];
(26)                                  }
(27)                             else Discard the Transition;
(28)             }
(29)          Else Discard the Transition;
(30)     done for
(31) done While
(32) return
```

되기 위해서는 정점 ⟨1,{2,3}⟩와 새롭게 생성된 정점 ⟨2,1⟩를 연결하는 에지의 레벨 1/0이 정점 1의 UIO순서의 일부가 되는가를 확인하고 일부가 되면 천이 $1\xrightarrow{1/0}2$가 깊이트리에 존재하는가를 확인하기 위해서 RB[4]의 값을 조사한다. RB[4]의 값이 0이므로 천이 $1\xrightarrow{1/0}2$을 깊이트리에 삽입한다. 다음은 천이 4를 깊이트리에 삽입하기 위해서 천이 $1\xrightarrow{0/1}3$에 대한 UIO(1) 순서를 조사한다. 그런데 0/1이 UIO(1)의 일부가 되므로 깊이트리에 삽입여부를 결정하기 위해서 RB[1]의 값을 조사한다. 이때 역시 이값도 0이므로 정점 ⟨3,2⟩가 깊이트리에 삽입된다. 다음은 천이 3인 $2\xrightarrow{1/1}1$를 깊이트리에 삽입하는 과정을 살펴보자. 이를

위해서 먼저 RB[3]의 값을 먼저 조사한다. 이때 RB[3]의 값이 0이므로 역시 새로운 정점 〈1,3〉이 깊이트리에 삽입된다. 다음은 천이 2인 $31\xrightarrow{1/1}2$이 깊이트리에 삽입되기 위해서는 RB[2]의 값을 조사한다. B[2]의 값이 0이므로 새로운 정점 〈2,1〉이 깊이트리에 삽입된다. 다음에 조사되어야할 정점은 〈1,3〉이므로 천이 1인 $1\xrightarrow{0/1}3$이 깊이트리에 삽입될 수 있는가를 확인해야 된다. 먼저 레벨 0/1이 UIO(1)의 일부가 되므로 새로운 정점 〈1,3〉이 깊이트리에 삽입된다. 그런데 RB[1]의 값이 0이 아니므로 천이 $1\xrightarrow{0/1}3$이 이미 깊이트리에 삽입된 것이다. 따라서 그 천이에 대한 기존의 Depth[1]과 새롭게 생성된 NewDepth[1]의 값을 비교한다. 그런데 Depth[1]의 값은 1이고 NewDepth[1]의 값은 3이므로 Depth[1]의 값이 1인 천이 1과 천이 1에 연결된 모든 정점들이 깊이트리에서 제거되고 그 패스에 포함되었던 천이 들의 RB[i]의 값도 0으로 clear된다. 다음은 천이 2인 $3\xrightarrow{1/1}2$의 RB[2]의 값이 0으로 clear되었으므로, 새로운 정점 〈2,1〉이 깊이트리에 삽입된다. 그리고 더 이상 삽입될 천이들이 존재치 않으므로 최종 결과가 그림3에 있다.

천이의 순서는 t4 t3 t1 t2가 되므로 이 순서대로 천이들을 순회하는것이 순회되어야 할 시험순서의 길이를 최소화 하는 것이다. 시험 순서의 길이가 적으면 적을수록 순회되어야 할 길이가 줄어드는 것과 같으므로 시험 순서의 길이가 적으면 적을수록 더 효율적이다. 따라서 그림3의 결과에 의한 시험 순서는 다음과 같다.

(결과)

천이순서 : t4 t3 t1 t2

입력순서

t4 : 1/0 1/1 0/1

t3 : 1/0 1/1 0/1

t1 : 0/1 1/1 1/1

t2 : 0/1 1/1 1/1 0/1

순 서 : 1/0 1/1 0/1 1/1 1/1 0/1

결론적으로 천이순서 t4와 t3는 완전히 중복되고 t3와 t1은 0/1이 중복된다. 그리고 천이 t1과 t2는 0/1 1/1 1/1이 중복되므로 시험 순서의 길이는 13에서 6으로 줄어들게 되어 시험순서의 길이가 약 52%정도 감소되었다.

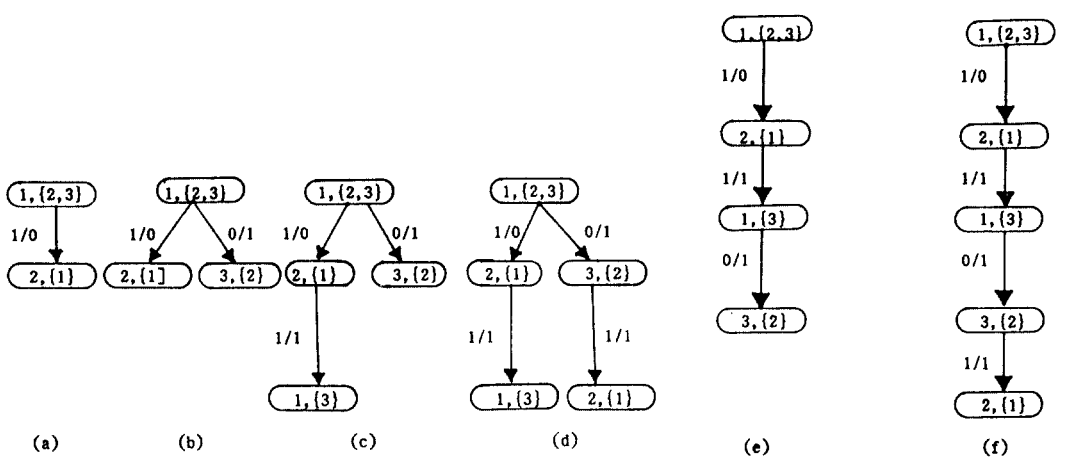


그림 3. 알고리즘 2의 수행과정

Fig 3. The execution progess of algorithm 2

V. 결 론

본 논문에서는 통신프로토콜의 효율적인 적합성 시험을 위해서 최적인 시험 항목을 생성하기 위한 깊이트리 구성 알고리즘을 제안하였다. 적합성 시험시 결정적 유한상태기계로 표현된 프로토콜의 모든 천이들이 적어도 한번은 순회되어야 하므로, 순회되는 시험 순서의 길이가 적으면 적을수록 효율적이다. 순회될 시험 보조순서는 브릿지 순서의 집합에 의해서 접속되므로 접속될 시험 보조순서간의 최대 중복성을 찾기위해서 여러개의 UIO순서중 중복성이 가장 큰 UIO순서를 사용하여 깊이트리를 구성한다. 그러면 깊이트리에서 깊이가 가장 큰 패스상에 존재하는 천이 순서대로 시험 항목을 생성하게 되면 시험 보조순서간의 중복성이 최대가 되므로 최소 길이의 시험 항목이 생성된다. 본 논문에서 제안된 알고리즘을 간단한 예에 적용한 결과 시험 순서의 길이가 약 52% 정도 감소 되었고, 시험 생성 방법도 유한상태기계에 새로운 에지들을 추가하지 않아도 되므로 다른 방법들에 비해서 훨씬 더 간단하다.

앞으로 더 연구되어야 할 내용은 비결정적 유한상태기계에서도 사용될 수 있도록 제안된 깊이트리 구성 알고리즘이 확장되어야 할 것이고, 프로그램 세그먼트로 서술된 프로토콜의 자료부분을 처리할 수 있는 방법들에 관한 연구가 더 진행되어야 할 것이다.

참 고 문 헌

1. Alfred V.Aho, Anton T. Dahbura, David Lee and M.Uyar, "An Optimization Technique for Protocol Coformance Test Generation Based on UIO Sequences and Rural Chinese Postman Tours," Protocol Specification, Testing, and Verification VIII, PP75-86, 1988.
2. B.Sarikaya, "Conformance Testing : Architectures and Test Sequence," Computer Networks and ISDN Systems 17(2), July, pp.111-126, 1989.
3. B.Yang and H.Ural, "On FSM-based Optimization Test Sequence Generation," Proc.2nd Internation Conference on Comm. Systems, Nov. pp118-125, 1990.
4. B.Yang and H. Ural, "Protocol Conformance Test Generation Using Multiple UIO Sequences with Overlapping," ACM, 1990.
5. C.A.Vissers, R.L.Tenny, and G.V.Bochman, "Formal Description Techniques," Proceedings of the IEEE Vol.71, December, pp.1356-1364, 1983.
6. J.P.Favreau, D.Hogrefe, and J.Kroon, "Formal Methods in Conformance Testing : Status and Expectations," Working draft Reports, 1992.
7. K.Naik and B.Sarikaya, "Testing Communication Protocols," IEEE Software, January, pp 27-37, 1992.
8. Krisan SABNANI and Anton DAHBURA, "A Protocol Test Generation Procedure," Computer Networks and ISDN Systems 15, pp. 285-297, 1988.
9. M.S.Chen, Y.Choi, and A.Kershenbaum, "Approaches Utilizing Overlap to Minimize Test Sequence," In Proc. 10th International IFIP Symposium on Protocol Specification, Testing, and Verification, Ottawa, Canada, June 1990.
10. W.Chun, "Test Case Generation for Protocols Specificed in ESTELLE," Ph.d Thesis, Comp. & Info.Sci., University of Delware, 1992.
11. W.Y.L.Chan, S.T. Vuong and M.R.Ito, "An Improved Protocol Test Generation Procedure based on UIOs," Proc. ACM SIGCOMM Symp. Comm. Arch. and Protocol Review 19 (4), September, pp.283-294, 1989.
12. X.Sun, Y. -N.Shen, F.Lombadi and D.Sciuto, "Protocol Conformance Testing By Discriminating UIO Sequences," Proc. 11th Workshop on Protocol Specification, Testing and Verification, pp.328-343, 1991.
13. Y.N. Shen, F.Lombardi, and A.T. Dahbura, "Protocol COnformance Testing Using Multiple UIO Sequence," Proceeding 9th IFIP Symposium on Protocol Specification, Testing and Verification, Twente, Holland, June, 1989.

許 基 澤 (Gi Taek Hur) 정회원

1960년 8월 9일생

1984년 : 전남대학교 자연대 계산통계학과 졸업(이학사)
1986년 : 전남대학교 대학원 계산통계학과 졸업(이학석사)
1989년 ~ 현재 : 광운대학교 이과대학 전자계산학과 박사과정 수료
1986년 : 목포대학교 강사
1987년 : 전남대학교 자연대 조교
1988년 : 전남대학교 자연대 전산통계학과 강사
1989년 ~ 현재 : 동신대학교 자연대 전자계산학과 조교수
1990년 ~ 현재 : 동신대학교 전자계산소장

李 東 豪 (Dong Ho Lee) 정회원

1957년 1월 15일생

1979년 2월 : 서울대학교 전자공학과 졸업(공학사)
1983년 2월 : 서울대학교 컴퓨터공학과 졸업(공학석사)
1988년 2월 : 서울대학교 컴퓨터공학과 졸업(공학박사)
1984년 9월 ~ 현재 : 광운대학교 전자계산학과 부교수
※주관심분야 : 프로토콜 공학, 네트워크 성능평가