論文 97-22-9-2

비결정성을 갖는 프로토콜을 위한 시험 스위트 생성방법

正會員 金 柄 植*, 全 遇 稷*

Test Suite Generation Technique for Protocols with Nondeterminism

Byeongsik Kim*, Woojik Chun* *Regular Members*

요 약

본 논문에서는 기존의 UIO 순열을 이용하는 방법을 개선하여 비결정적 유한상태기계에서 시험 경우를 생성하는 방법에 대해서 소개하였으며, 시험 경우의 자동생성을 위해서 필수적인 사항인 적합성 관계의 형식적인 정의를 새롭게 정의하였다. 또한 프로토콜의 비결정적인 성질 때문에 시험기는 한 입력을 가했을 때 어떤 출력을 낼지 미리 시험기의 입장에서 알 수 없으므로 적응력있는 시험을 위해서 트리형태의 시험 경우를 생성하는 방법을 소개하였고 시험 경우를 입력과 출력을 분리해서 고려함으로써 시험 경우 기술 언어인 TTCN으로의 변환을 쉽게 하였다.

ABSTRACT

This paper proposes a new test case generation technique for a nondeterministic finite state machine by improving the existing UIO sequence generation method. First, a new conformance relation is defined, which is one of prerequisites for automatic test case generation. Because of the nondeterministic property of protocols, the output of the systems under test is not known deterministically to the tester. Therefore, tree-like test case generation method is introduced for adaptive testing, in which the next input is selected after observing the previous output. Since the test cases are generated with regarding the inputs and outputs as separate events and are represented in tree notation, the test cases are easily converted into TTCN, the international standard test suite specification language.

Ⅰ. 서 론

최근 컴퓨터통신의 급격한 발달로 인하여 다양한 통신 어플리케이션 및 복잡한 프로토콜들이 등장하게 되었다. 이러한 어플리케이션 및 프로토콜 개발시 각 프로토콜 표준안에 따라 개발된 프로토콜들의 상호연동을 위하여 적합성 여부가 중요하게 인식되고 있다. 또한 적합성 시험을 위해서 적합성 시험 경우의 생성방법 및 자동화가 중요한 문제로 대두되었다. 이러한 연구분야에서 대부분의 연구는 단일한(single) 그리고 순수한(pure) 결정적 유한상태기계(DFSM: Deterministic Finite State Machine)에서 시험경우를 생성하는 방법에 기반을 두고 있다[1][2][3][4][5][6][7].

*충남대학교 컴퓨터공학과
論文番號:96346-1105
接受日字:1996年 11月 5日

그러나, 대부분의 실제 프로토콜들은 너무 복잡해서 DFSM만으로 명세화 되기는 불가능하다.

본 논문에서는 비결정적 유한상태기계(NFSM:Nondeterministic Finite State Machine)에서 시험경우를 생성하는 방법을 고려한다. 또한 비결정성은 2.2에서 정의되는것처럼 구별가능한 비결정성으로 제한해서 고려한다. 본 논문에서 제안하는 기법은 잘 알려진 UIO 순열[2]을 이용하는 방법과 [8]에서 제안한 부분 UIO(PUIO) 순열을 사용하는 기법을 확장한 기법에 해당한다.

DFSM에서 시험경우는 입력과 출력의 순열 형태로 표현된다. 그러나 NFSM에서는 시험기가 시험대상 시스템에 대해서 완전한 제어능력을 가지지 못하기 때문에 시험기가 한 입력을 가했을 때 나타나는 출력이 비결정적으로 발생하므로 시험경우들은 입력과 출력 행동들에대한 트리 형태로 표현되어야한다. NFSM의 한 상태에 대한 UIOTree는 특정한 한 상태를 유일하게 식별해 낼 수 있다는 점에서 UIO 순열의 개념과 유사하다고 볼 수 있다. 그러나 본 논문에서 제안하는 UIOTree와 UIO 순열사이에는 두가지의 차이점이 존재하는데 그중 하나는 UIOTree가 비결정성을 해결하기 위해서 UIO 순열들로 구성된 트리의 형태로 표현된다는 점이며, 다른 하나는 TTCN[9]과 유사한 형태로 시험경우들을 쉽게 변환하기 위해서 입력과 출력이 분리된 트리의 형태로 표현된다는 점이다.

[10]에서는 본 연구에서 제안하는 PUIOTree와 유사한 AIO(Adaptive Input/Output) 트리를 이용해서 NFSM으로부터 시험경우를 생성하는 방법을 제안하고 있지만 대부분의 NFSM에서는 AIO 트리가 존재하지 않는 경우가 대부분임에도 불구하고 이런 경우에 대한 해결방법은 제시하지 못하고 있다. 따라서 본 연구에서는 부분 UIOTree(PUIOTree)들을 사용해서 AIO 트리를 갖지 못하는 NFSM에 대해서 시험경우를 생성하는 방법을 제안한다. PUIOTree와 UIOTree의 관계는 PUIO 순열과 UIO 순열의 관계와 대응된다고 볼 수 있다. 즉, UIOTree는 NFSM내의 한 상태를 다른 모든 상태과 구별할 수 있으며, PUIOTree는 NFSM내의 한 상태를 단지 다른 상태들의 일부에 대해서만 구별할 수가 있다.

본 논문은 다음과 같이 구성되어 있다. 제2절에서는 NFSM 모델과 비결정성을 소개한다. 제3절에서는 행동트리와 적합성 관계의 개념을 소개한다. 제4절에서는 UIOTree의 개념과 UIOTree를 갖지 못하는 경우에 PUIOTree를 이용해서 각 상태를 구별할 수 있는 시그니처를 생성하는 방법에 대해서 기술하고 제5절에서는 시험 경우 및 시험 스위트를 생성하는 방법에 대해서 기술한다. 끝으로 6절에서 결론 및 향후 연구과제에 대해서 기술한다.

Ⅱ. 비결정적 유한상태기계(NFSM)

본 논문에서는 단일하고 순수한(즉, 내부확장을 고려하지 않는) 그리고 구별가능한 NFSM에서 시험경우를 생성하는 방법에 대해서 알아보기로 한다. NFSM 모델의 형식적인 정의와 구별가능한 비결정성에 대한 정의에 대해서 다음에 언급하였다.

2.1 NFSM 모델

대부분의 실제 프로토콜들은 한 상태에서 여러개의 전이가 가능하다. 만일 어떤 전이 시스템이 한 상태에서 여러개의 전이가 가능하거나 혹은 자발적인 전이를 일으킬 수 있는 경우에 이런 시스템을 비결정성을 갖는 시스템이라 하고, 이런 시스템은 NFSM으로 모델링이 가능하다.

정의 (NFSM 모델) NFSM 모델 M은 5-튜플 $\langle S, s_0, I, O, T\rangle$에 의해서 표현되는 전이 시스템이라고 정의한다. 여기서,

S : 상태들의 유한집합,
s_0 : 초기 상태, $s_0 \in S$,
I : 입력들의 유한집합,
O : 출력들의 유한집합,
T : 전이 매핑들의 유한 집합 : $S \times I \mapsto S \times O$.

NFSM의 상태전이함수 T는 $s \xrightarrow{i/o} s'$로 표현되며, 이는 입력 i를 받았을때 상태 s에서 상태 s'로 전이가 발생하고 출력 o를 발생시키는 것을 의미한다. 본 논문의 전반에 걸쳐 사용된 표기법을 정리하면 다음과 같다.

- $s, s', s_1, s_2, \cdots$ NFSM의 상태들

- $i, i', i_1, i_2, \cdots$ I의 범위내에있는 외부입력들
- $o, o', o_1, o_2, \cdots$ O의 범위내에 있는 NFSM으로부터 발생되는 외부 출력들
- $\sigma, \sigma', \sigma_1, \sigma_2, \cdots$ $(I/O)^*$ 범위내의 외부 입력/출력 쌍들의 순열
- $s \xrightarrow{\sigma} s'$ $\exists s_{k(1\le k<n)} \in S$에 대해서, $s \xrightarrow{i_1/o_1} s_1 \xrightarrow{i_2/o_2} s_2 \cdots s_{n-1} \xrightarrow{i_n/o_n} s'$. 여기서, $\sigma = i_1/o_1; i_2/o_2; \cdots; i_n/o_n$.
- $out(s, i)$ NFSM의 한 상태 s에서 입력 i를 가해서 발생하는 출력들의 집합

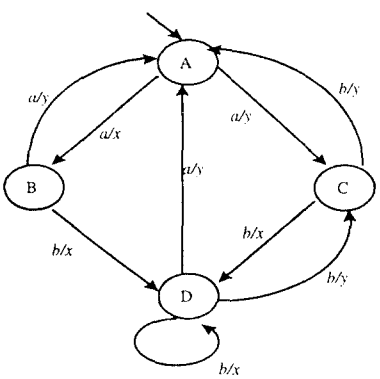


그림 1. NFSM

그림 1의 NFSM은 본 논문 전체에서 예로 사용되며, 상태들의 집합 S는 $\{A, B, C, D\}$, 초기 상태는 A, 입력들의 집합 I는 $\{a, b\}$, 출력들의 집합 O는 $\{x, y\}$이고 전이들의 집합 T는 $\{A \xrightarrow{a/x} B, A \xrightarrow{a/y} C, B \xrightarrow{a/y} A, B \xrightarrow{b/x} D, C \xrightarrow{b/x} D, D \xrightarrow{a/y} A, D \xrightarrow{b/y} C, D \xrightarrow{b/x} D, C \xrightarrow{b/y} A\}$로 표현된다.

2.2 비결정성

전이 시스템을 시험하는 동안에 다음과 같은 세가지 이유로해서 비결정성이 발생하게 된다. 첫째, 하나의 외부 입력에 의해서 여러개의 전이가 가능한 경우 둘째, 내부 입력을 갖고 있는 전이는 외부 시험기에 의해서 제어가 불가능하므로 NFSM의 현재 상태가 외부 입력이 없이 변경되는 경우 셋째, 다른 전이의 지연때문에 현재의 전이가 지연되는 경우이며 이때 비결정성은 지연된 전이들 사이에서도 발생할 수도 있고, 이전에 지연된 전이와 현재의 전이사이에도 발생하는 경우등이 있다.

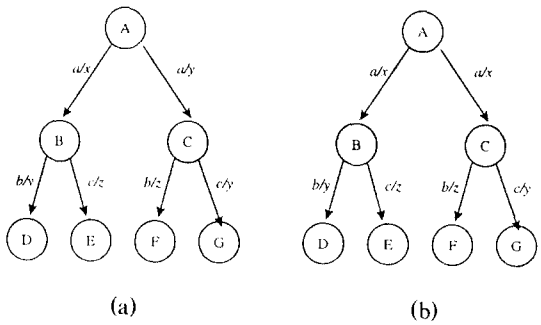


그림 2. 두가지 비결정성

본 논문에서는 전이의 선택에 의해서 발생하는 비결정성에 대해서만 고려하기로 한다. 프로토콜의 적합성 시험에서 시험대상시스템(SUT:System Under Test) 내에서의 비결정성은 외부 시험기와는 독립적이다. 즉, SUT가 발생시키는 출력은 시험기에 의해서 제어가 불가능함을 의미한다. 이런 비결정적인 선택은 한 입력을 가했을때 나타나는 출력에 의해서 어떤 전이가 선택되었는지를 알아낼 수 있는 경우와 그렇지 못한 경우로 나누어 생각할 수 있다. 시험기의 관점에서 이러한 성격에 의해 비결정성을 다음과 같은 두가지 경우로 분류가 가능하며 이를 그림 2에 나타내었다.

- 구별가능한 비결정성:동일한 입력에 의해서 가능한 각각의 전이가 서로 다른 출력을 낼때 시험기는 어떤 전이가 선택되었는지를 알아낼 수 있는 경우(그림 2의 (a))
- 구별불가능한 비결정성:동일한 입력에 의한 전이들이 동일한 출력을 내는 경우 시험기는 어떤 전이가 선택되었는지를 알아낼 수 없는 경우(그림 2의 (b))

구별불가능한 비결정성은 적절한 알고리즘[3]에 의해서 구별가능한 비결정성으로 변환이 가능하므로 본 논문에서는 구별가능한 비결정성에 대해서만 고려하기로 한다.

Ⅲ. 행동트리(Action Tree)와 적합성관계(Conformance Relation)

3.1 행동트리

NFSM은 하나의 입력에 의한 여러개의 출력들중의 한 출력을 생성해내기 때문에 실제로 시험을 행하기 전에 시험 순열을 미리 생성해낼 수 없다. 즉, 시험기가 한 입력을 가했을때 나타나는 출력에 따라서 다음의 입력이 결정되어야 하기 때문이다. 즉, NFSM의 시험에서 다음 입력은 이전의 입출력쌍들의 순열에 의해서 결정된다. 이러한 시험방법을 위해서 본 논문에서는 NFSM으로 표현된 SUT의 행동을 행동트리로 표현한다. 이때 행동트리는 한 입력에 의해서 가능한 모든 출력들을 표현할 수 있다.

정의 (행동트리) NFSM $M=\langle S, s_0, I, O, T\rangle$에 대한 행동트리는 $v_0 \in V$를 루트로 하는 트리 $\langle V, L, v_0\rangle$로 정의된다. 즉,

1. V는 정점(vertex)들의 집합에 해당하며, 각각의 점 $v_0 \in V$는 상태 $s \in S$에 해당하거나 일시적인 정점(transient vertex : v_{tran})에 해당된다. ($state(v)=s$, 그리고 $state(v_{tran}) \neq s$)
2. $\{?, !\} \times L$은 입력행동($?a$) 혹은 출력행동($!a$)을 의미한다.
3. L은 링크들의 집합이며, 각각의 링크 $l \in L$은 $v_s \xrightarrow{i/o} v_d \in T$이면 링크 $v_s \xrightarrow{?i} v_{tran}$이나 링크 $v_{tran} \xrightarrow{!o} v_d$에 해당한다.
4. (tree axiom) 각 정점 $v \in (V-\{v_0\})$에 대해서 $v' \xrightarrow{?i} v$나 $v' \xrightarrow{!o} v$ (여기서, $v' \in V$등의 링크는 반드시 하나만 존재한다.
5. (output completeness axiom) 각 정점 $v_s \in V$에 대해서, 링크 $v_s \xrightarrow{?i} v_{tran} \xrightarrow{!o} v_d$가 존재하면, 각각의 출력 $o' \in out(state(v), i)$에 대해서 링크 $v_s \xrightarrow{?i} v_{tran} \xrightarrow{!o'} v_d$가 반드시 존재해야 한다. 여기서, $o \in out(state(v), i)$는 정점 v에 해당하는 상태에서 입력 i에 의해서 가능한 모든 출력들의 집합을 의미한다.
6. (error output axiom) 각 정점 $v_s \in V$에 대해서 링크 $v_s \xrightarrow{?i} v_{tran} \xrightarrow{!error} v_d$가 존재하면, $v_s \xrightarrow{?i} v_{tran} \xrightarrow{?o} v_d$ 여기서 ($o \neq error$)에 해당하는 링크가 반드시 하나 존재해야 한다.

위의 정의에서 tree axiom은 루트 정점을 제외한 모든 정점이 정확히 하나의 들어오는(incoming) 링크를 가져야 하며 루트 정점은 들어오는 링크를 갖지않아야 한다는 일반적인 트리의 요구사항을 표현하고 있다. output completeness axiom은 한 입력에 의한 모든 가능한 출력들이 하나의 트리내에 모두 함께 포함되어 있어야 함을 의미한다. 이 이유는 여러개의 가능한 출력들중에서 하나의 출력만이 외부 시험기에 의해서 비결정적으로 관찰되기 때문이다. error output axiom은 핵심(core) 전이만을 유발하는 입력만을 허용해서 error상태로 전이를 일으키는 입력은 허용하지 않게 하는 것을 의미한다. 또한 이 axiom은 error상태로의 전이는 필요한 경우에만 시험되도록 하는 약한(weak) 적합성 시험을 허용한다.

그림 3의 루트 정점 v_0에서 시험기는 두개의 입력 $?a$와 $?b$중에서 하나를 적용할 수 있다. 그리고 시험기는 일시적인 정점으로 전이한다. 이때 점선의 원으로 표시된 일시적인 정점들 v_a와 v_b는 NFSM의 전이가 입력과 출력의 쌍으로 구성되어 있지만(즉, $s \xrightarrow{i/o} s'$) 입력과 출력을 분리해서 고려하는 행동트리로 표현함으로써 나타나는 성질때문에 도입한 일시적인 정

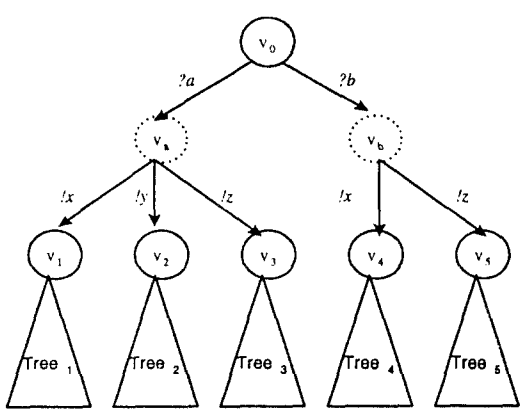


그림 3. 행동트리

점에 해당한다. 시험기가 일시적인 정점에서 IUT의 응답을 기다리는 동안 IUT가 여러개의 가능한 출력들중에서 하나를 선택해서 응답하게 된다.

예를들어, 그림 3의 정점 v_0에서 시험기는 두개의 입력 $?a$와 $?b$중에서 하나의 입력을 가하게 된다. 만일 입력 $?a$를 가한다고 가정하면 일시적인 정점 v_a로 이동하고 IUT의 응답을 기다린다. 이때 IUT가 응답할 수 있는 세개의 출력 $!x$, $!y$, $!z$ 중에서 $!y$를 응답한다면 시험기는 정점 v_2로 이동하게 된다. 이러한 과정이 행동트리의 단말 정점에 이를때까지 반복되게 된다.

행동트리 $AT=\langle V, L, v_0\rangle$를 이용해서 시험 경우를 생성하는 방법을 설명하는데 사용하는 표기법은 다음과 같다.

- $\sigma=?i_1;!o_1;?i_2;!o_2;\cdots;?i_n;!o_n$. 즉, 입력행동($?i_k$)과 출력행동($!o_k$)들로 구성된 순열
- $R(AT, \sigma)=\{v \mid v_0 \xrightarrow{\sigma} v\}$. 행동트리 AT의 루트 정점 v_0로 부터 순열 σ에 의해서 도달 가능한 모든 정점들의 집합
- $In(R(AT, \sigma))=\{?a \mid \forall v\in R(AT, \sigma),\ v \xrightarrow{?a}\}$. 행동트리 AT의 루트 정점 v_0로 부터 순열 σ에 의해서 도달한 모든 상태에서 받아들일 수 있는 입력들의 집합
- $Out(R(AT, \sigma))=\{!a \mid \forall v\in R(AT, \sigma),\ v \xrightarrow{!a}\}$. 행동트리 AT의 루트 정점 v_0로 부터 순열 σ에 의해서 도달한 모든 상태에서 관찰가능한 출력들의 집합

실제 프로토콜에서는 IUT가 한 입력을 받고 출력을 내지 않은 상태에서 여러개의 입력을 계속 받아들이는 경우도 있다. 그러나 본 논문에서는 루트 정점 v_0에서 시작해서 입력과 출력의 순으로 계속 반복해서 나타나는 경우로만 한정하기로 한다.

3.2 적합성 관계

그림 4는 두개의 프로토콜 엔터티 1(PE 1)과 프로토콜 엔터티 2(PE 2)사이의 인터페이스에서 보았을때, 프로토콜이란 표준에 정의되어 있는 양 프로토콜 엔터티들의 행동에 해당함을 나타내고 있다. PE 1의 관점에서 A는 출력에 해당하고 B는 입력에 해당한다. 역으로, PE 2의 관점에서는 A는 입력에 해당하고 B

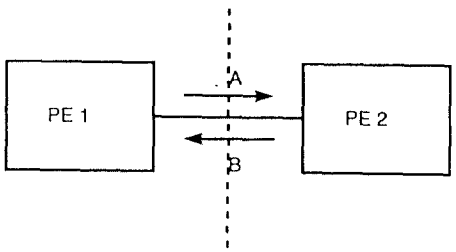


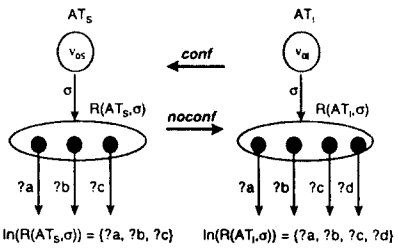
그림 4. 프로토콜

는 출력에 해당한다. 이때 PE 1을 SUT라하고 PE 2를 시험기라고 가정하면, PE 1은 표준에 정의되어 있는 모든 입력을 받아들일 수 있어야 한다. 왜냐하면 PE 2의 관점에서 PE 1은 표준에 따라 구현되어 있다고 가정하고 표준에 정의된 모든 출력을 생성할 수 있기 때문이다. 반면 모든 입력들에 의한 출력들은 표준에 정의되어 있는 출력들중의 일부이어야 한다. 왜냐하면 PE 2는 PE 1이 표준에 정의된 출력만을 생성할 것으로 예상하고 있기 때문이다. 다시말하면 구현된 프로토콜들은 표준에 정의되어 있는 입력들은 모두 받아들일 수 있으면서 그 이외의 입력을 받아들여도 무방하다. 또한 표준에 정의되어 있는 입력들에 의한 출력들이 다시 표준상의 가능한 출력들중의 일부이면 프로토콜 표준에 적합하다고 할 수 있다. 이러한 개념에 의하여 적합성 관계를 다음과 같이 정의할 수 있다.

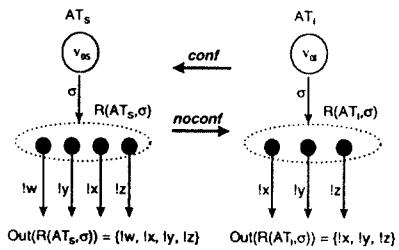
정의 (적합성 관계) 프로토콜 명세와 구현된 시험대상 프로토콜을 두개의 행동트리 $AT_S=\langle V_S, L_S, v_{0S}\rangle$와 $AT_I=\langle V_I, L_I, v_{0I}\rangle$로 표현했을때, 적합성 관계 $AT_I \xrightarrow{conf} AT_S$는 다음과 같이 정의된다.

1) 순열 σ를 적용한후 안정된 정점에서 IUT가 한 입력을 기다리고 있는 경우
$In(R(AT_S, \sigma)) \subseteq In(R(AT_I, \sigma))$
2) 순열 σ를 적용한후 일시적인 정점에서 IUT가 여러개의 가능한 출력들중에서 한 출력을 내는 경우
$Out(R(AT_S, \sigma)) \supseteq Out(R(AT_I, \sigma))$

그림 5.(a)에서는 AT_I 내의 순열 σ에 의해서 도달가능한 모든 정점들의 집합 $R(AT_I, \sigma)$가 $In(R(AT_S, \sigma))$에 정의되어 있는 모든 입력들을 받아들일 수 있으므로



(a)



(b)

그림 5. 적합성 관계

$In(R(AT_I, \sigma)) \supseteq In(R(AT_S, \sigma))$가 되므로 $AT_I \xrightarrow{conf} AT_S$가 성립한다. 또한 그림 5.(b)에서는 $Out(R(AT_I, \sigma)) \subseteq Out(R(AT_S, \sigma))$이므로 $AT_I \xrightarrow{conf} AT_S$가 성립한다. 그러나, 그림 5.(a)에서는 $In(R(AT_S, \sigma)) \not\supseteq In(R(AT_S, \sigma))$이고 그림 5.(b)에서는 $Out(R(AT_S, \sigma)) \not\subseteq Out(R(AT_I, \sigma))$이므로 $AT_S \xrightarrow{noconf} AT_I$가 성립한다. 단, 본 논문에서의 행동트리는 입력/출력의 쌍들에 의해서 전이가 발생하는 NFSM으로부터 생성되므로 시험순열 σ는 입력과 출력이 순서대로 나타나는 형태로 구성된다.

Ⅳ. 상태 시그니쳐 생성

이 절에서는 시험하고자하는 전이의 도달 상태를 검증하기 위해서 입출력 순열로 구성되는 상태 시그니쳐를 생성하는 방법에 대해서 기술한다. 또한 상태 시그니쳐로 UIOTree를 기본적으로 사용하지만 NFSM에서 상태들은 UIOTree를 갖지 못하는 경우가 대부분이므로 이런 상태들을 위해서 PUIOTree를 상태 시그니쳐로 사용하는 방법에 대해서 기술한다.

4.1 NFSM을 위한 UIO 순열 - UIOTree

구별가능한 비결정성을 처리하기 위해서는 IUT에 한 입력을 가했을때 IUT가 내는 출력이 비결정적으로 선택되어서 나타나기 때문에 한 입력에 의해서 나타날 수 있는 모든 출력이 일시적인 정점에 모두 나타나야 하는 UIOTree를 이용해야한다. 또한 이 트리는 어떤 패스를 비결정적으로 선택해서 따라가더라도 모든 패스는 UIO 순열이 되는 형태로 구성되어야 한다.

정의 (UIOTree) NFSM의 한 상태 s에 대한 UIOTree는 $UIOTree(s)$로 표현되며, 행동트리 $\langle V, L, v_0 \rangle$에 해당한다. 즉, $state(v_0) = s$이며 각각의 단말 정점 v에 대해서 v_0부터 v까지의 라벨들의 순열이 모두 UIO 순열이 된다. 즉, s 이외의 다른 상태에서는 이런 순열을 생성할 수 없다.

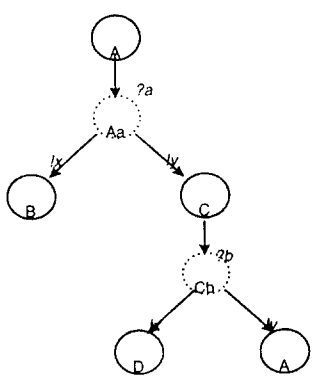


그림 6. UIOTree의 예

그림 6은 그림 1의 NFSM에서 상태 A에 대한 UIOTree를 나타내고 있다. UIOTree의 루트 정점(상태 A)에서 세개의 단말 정점(상태 B, D, A) 까지의 세개의 순열들 [?a;!x], [?a;!y;?b;!x], [?a;!y;?b;!y]은 모두 상태 A에 대한 UIO 순열이다. 또한 한 입력에 의해서 가능한 모든 출력들이 트리내에 모두 포함되어 있다. 다른 상태들과 상태 A를 구별하기 위해서 시험기는 이 UIOTree를 적용시킨다. 만일 시험기가 입력 a를 가했을때 명세에 적합하게 구현된 IUT는 x 혹은 y를 출력으로 낸다. 이때 만일 IUT가 x를 출력으로 낸다면 상태 A는 즉시 구별이 되지만 UIOTree 내에 존재하지 않는 x나 y이외의 출력(ε를 포함해서)

을 낸다면 이 IUT는 명세에 대해서 적합하지 않음을 의미한다. 만일 구현된 프로토콜이 y를 출력으로 내는 경우에는 입력 b가 다시 가해지고 x나 y가 출력으로 나타난 후에 상태 A가 구별된다.

본논문에서 UIOTree를 생성하는 방법은 UIO 순열을 생성하는 방법[8]을 개선한 것으로 차이점이라면 모든 가능한 UIO 순열들을 모두 포함하는 행동트리를 생성한 후 이 행동트리에서 UIOTree를 추출해내는 두 단계로 이루어진다는 점이며, 또한 이 알고리즘은 최종적인 시험 스위트를 TTCN으로의 변환을 쉽게 하기 위해서 입력과 출력이 분리된 UIOTree를 생성한다는 점이다.

4.2 부분(Partially) UIOTree - PUIOTree

NFSM의 대부분의 상태들은 UIOTree를 가지지 못하는 경우가 대부분이다. 예를들어 그림 1에서 상태 B, C, D의 경우가 이에 해당한다. 이런 상태들에 대해서는 UIOTree를 이용할 수 없기 때문에 PUIOTree를 이용해서 시험경우를 생성한다. PUIOTree와 UIOTree의 관계는 PUIO 순열과 UIO 순열의 관계와 대응된다고 할 수 있다. 즉, UIOTree는 다른 모든 상태들과 특정한 한 상태를 구별할 수 있지만 PUIOTree는 단지 일부의 상태들과 특정한 한 상태를 구별할 수 있다. 이때 전이의 도달 상태에 대한 검증을 위해서 여러개의 PUIOTree들을 조합해서 사용하는 방법을 이용할 수 있다.

정의 (PUIOTree) NFSM의 한 상태 s에 대한 PUIOTree는 $PUIOTree(s)$로 표현하며 이 트리는 행동트리 $\langle V, L, v_0\rangle$에 해당한다. 이때 $state(v_0)=s$이며 각 단말 정점 v에 대해서 v_0 부터 v까지의 라벨들의 순열은 배타집합(Exclusion Set) E_v를 갖는 PUIO 순열에 해당한다.

각 단말 정점 v가 배타집합 E_v를 갖고 있는 PUIOTree가 상태 s의 시그니처로 사용될 때 E_v내에 있는 상태들과는 상태 s가 구별될 수 없음을 의미한다. 즉, $E_v=\phi$이면 상태 s는 $(S-\{s\})$ 내에 있는 모든 상태들과 구별될 수 있지만 $E_v\neq\phi$인 경우에는 상태 s는 $(S-\{s\}-E_v)$ 내의 상태들하고는 구별이 가능하지만 E_v 내의 상태들하고는 구별이 불가능하다.

그림 7은 그림 1의 NFSM에 대한 상태 C와 상태 D

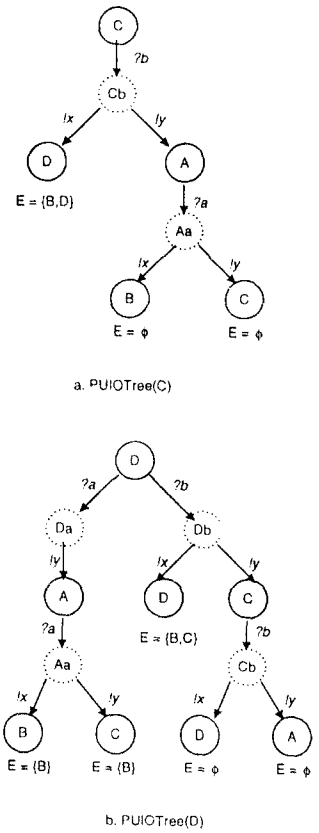


그림 7. 상태 C와 D에 대한 PUIOTree

의 PUIOTree를 보여 주고 있다. 그림 7.a의 PUIOTree(C)에서 PUIOTree의 루트 노드(상태 C)에서 단말 노드(상태 B, 상태 C, 상태 D)까지의 순열들은 각각 배타집합 ϕ, ϕ, {B, D}를 가지고 있는 PUIO 순열에 해당한다. 이런 경우에 시험기가 상태 A, 상태 B, 상태 D로부터 상태를 구별하기 위해서 이 PUIOTree를 적용시킬때 올바르게 구현된 프로토콜이라면 입력 b에 대해서 x나 y를 출력으로 내어야 한다. 만일 구현된 프로토콜이 y를 출력으로 내는 경우에 시험기는 입력 a를 한번 더 적용시켜봄으로써 다른 모든 상태들 A, B, D와 상태 C를 구별해 낼수가 있다. 그러나 만일 이 PUIOTree에 있지 않은 출력을 내게 되면 이 구현된 프로토콜은 표준에 적합하지 않음을 의미한다. 그림 7.b의 PUIOTree(D)에서도 마찬가지로 입력 a를 두 번 계속해서 적용시키면 상태 B 이외의 모든 상태들과 상태 D를 구별할 수가 있으며, 입력 b를 적용시

키는 경우에는 구현된 프로토콜이 출력으로 x나 y 중에 어느것을 내느냐에 따라서 달라지는데 x를 출력으로 내는 경우에는 상태 B 와 상태 C를 제외한 상태 A와 상태 D를 구별할 수가 있으며, 출력으로 y를 내는 경우에는 입력으로 b를 한번 더 적용함으로써 다른 모든 상태들과 상태 D를 구별할 수 있는 시험 순열을 생성해 낼 수가 있다.

시험 수행(test run)이 배타집합이 공집합이 아닌 단말 정점에서 끝나게 된다면 배타집합내의 상태들과 시험하고자하는 상태를 구별하기 위해서 부가적인 시험이 필요하게 된다. 이때 [2]에서 제안한 transfer sequence와 distinguishing sequence를 사용하는 방법을 이용한다면 시험기는 상태 s를 각각의 상태 $s_i \in E_v$와 구별하기 위해서 부가적인 transfer sequence와 distinguishing sequence를 사용해야 한다.

그러나 이러한 방법은 DFSM의 경우에는 사용해도 특별한 문제를 발생시키지는 않지만 NFSM에 적용시킬때는 여러가지 문제점을 발생시킨다. 시스템내의 비결정적인 성질때문에 시험기는 transfer sequence를 사용해서 시스템을 원하는 상태로 이동시킬 수가 없으며, distinguishing sequence도transfer sequence와 동일한 문제점을 가지게 된다. 즉, 이런 비결정적인 성질때문에 시스템은 한 상태를 다른 상태들과 구별하기위해서 원하는 경로를 선택하지 않을수도 있다. 따라서 이러한 순열들만을 가지고는 NFSM의 행동을 표현하기에는 부적절하다. 그러므로 트리를 이용하는 표기법을 사용해야 한다[8][10][11][12].

[10]에서 제안하는 AIO(Adaptive Input/Output) 트리를 이용하는 방법은 본 연구와 유사하게 UIO 순열의 개념을 기반으로 하는 AIO 트리를 사용하지만 이 연구의 한계는 AIO 트리가 존재하는 경우에만 사용할수 있으며, 실제로 대부분의 상태들이 AIO 트리를 갖고 있지 않는 경우가 많이 있는데도 불구하고 AIO 트리가 존재하지 않는 경우에 대한 해결책은 제시하지 못하고 있다. 그러나 본 연구는 AIO 트리가 존재하지 않는 상태들에 대해서도 그대로 적용이 가능하다는데 장점이 있다. DFSM에서 한 상태에 대한 UIO 순열이 존재하지 않는 경우에 여러개의 PUIO 순열을 조합해서 상태 시그니처로 사용하는 방법과 유사하게 NFSM의 한 상태가 UIOTree를 가지고 있지 않은 경우에 여러개의 PUIOTree를 조합해서 상태 시그니처로 사용할 수 있다. 어떤 상태 s가 UIOTree는 가지고 있지 않지만 PT_1과 PT_2 두개의 PUIOTree를 가지고 있다고 가정해보자. 이때 PUIOTree는 행동트리에 해당하며, 행동트리의 output completeness axiom에 의하여 동일한 입력의 모든 전이는 하나의 트리내에 모두 포함되어야 한다. PT_1과 PT_2가 각각 입력 i_1과 $i_2(i_1 \neq i_2)$의 모든 전이들을 포함하고 있다고 가정해보자. 이때 PT_1을 적용하는 시험 수행이 배타집합 E_1을 갖는 단말 정점에서 끝나고, PT_2를 적용하는 다른 시험 수행은 배타 집합 E_2를 갖는 단말 정점에서 끝난다면 상태 s는 E_1과 E_2 모두에 있지 않는 모든 상태들과는 구별이 가능하다. 그러므로, $E_1 \cap E_2 = \phi$이면 이 두개의 시험 수행을 통해서 다른 모든 상태들과 상태 s를 구별할 수 있게 된다.

여러개의 PUIOTree들을 하나의 PUIOTree로 결합시킴으로써 전체 배타집합의 크기를 줄일 수 있으므로 동일한 PUIOTree를 적용하는 시험수행의 횟수가 줄어들게 된다. 그러나 PUIOTree를 적용시키는데 있어서의 복잡성은 증가하게 된다. 따라서 최적의 PUIOTree 즉, 최소의 배타집합을 갖는 트리를 생성하는 것이 최적화를 위한 중요한 문제에 해당한다.

PUIOTree의 배타집합들의 집합을 정의하기 위해서 그림 8의 트리를 가정해 보자. 이 트리의 루트 정점 v는 입력 $?i_1, i_2, \cdots ?i_n$에 대한 링크들을 가지고 있다. 이때 한 입력 $?i_k$에 대한 각각의 링크를 $v \xrightarrow{?i_k} v_{i_k}$, $1 \leq k \leq n$라 하고 $!o_{k,j}$가 입력 $?i_k$에 대한 출력이라 한다면 각 정점 v_{i_k}는 일시적인 정점에 해당하고 출력 $!o_{k,j}$, $(v_{i_k} \xrightarrow{!o_{k,j}} v_{k,j})$에 해당하는 링크들을 갖게 된다. 이때 각 정점 $v_{k,j}$는 단말 정점이 되거나 다른 서브트리의 루트 정점이 된다.

이런 경우 정점 $v_{k,j}$에 대한 배타집합들의 집합을 $Eset(v_{k,j})$라 하고 $v_{k,j}$가 배타집합을 갖는 단말 정점이라 한다면 $Eset(v_{k,j})$은 $\{E_{k,j}\}$이거나 혹은 자식 정점들의 $Eset$들을 계산하기 위한 함수에 사용된다.

외부 시험기의 관점에서 시험 수행이 정점 v에 도달하게 될 때, 시험기는 $?i_1, ?i_2, \cdots, ?i_n$중의 어떤 입력이라도 시험기의 의도에따라 적용시킬 수 있다. 만일 시험기가 올바르게 구현된 프로토콜에 입력 $?i_k$를 적용시키면 일시적인 정점 v_{i_k}로 전이한 후 이 일시적인 정점

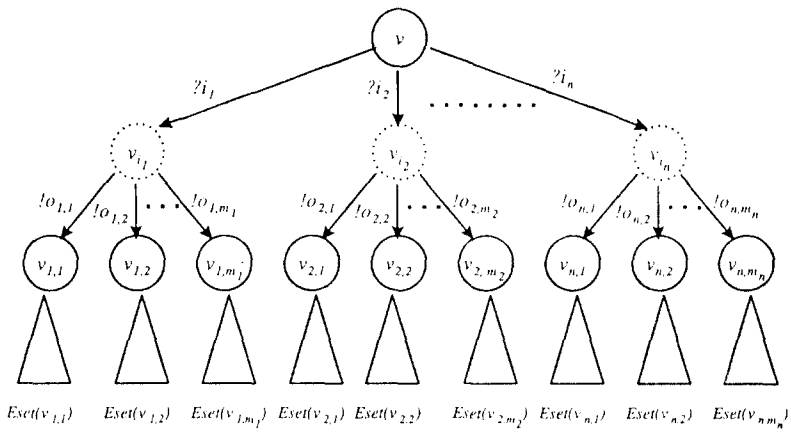


그림 8. PUIOTree내의 서브트리에 대한 배타집합들의 계산

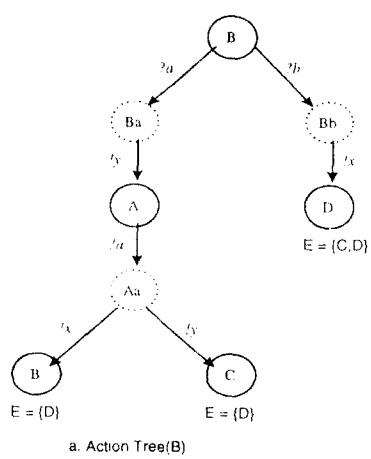
에서 $v_{i_k} \xrightarrow{!o_{k,1}} v_{k,1}$, $v_{i_k} \xrightarrow{!o_{k,2}} v_{k,2}$, …, $v_{i_k} \xrightarrow{!o_{k,m_k}} v_{k,m_k}$에 해당하는 전이들중의 하나가 비결정적으로 발생된다. 즉, 출력의 경우에 시험기의 의도와는 무관하게 IUT가 출력을 생성한다. 이런 경우, PUIOTree는 어떤 전이가 발생되는지에 따라서 $Eset(v_{k,1})$, $Eset(v_{k,2})$, …, $Eset(v_{k,m_k})$내의 배타집합들로부터 상태 s를 구별할 수 없는 경우도 있다. 따라서, 입력 $?i_k$에 의한 정점 v를 루트로 하는 서브 트리에 대한 배타집합들의 집합은 다음과 같이 표현될 수 있다.

$1 \le k \le n$에 대해서, $Eset(v)/?i_k = \{Eset(v_{k,j}), 1 \le j \le m_k\}$

위의 식은 한 입력에 의해서 나타날 수 있는 정점 v를 루트로 하는 PUIOTree의 배타집합을 계산하는 식으로써 각각의 가능한 출력들의 배타집합들에 대한 합집합을 계산해서 worst-case의 배타집합을 계산해 내는 식으로 이 집합의 한 원소에 해당하는 집합 하나가 비결정적으로 선택된다는 의미이며 이중 어느 하나가 출력으로 선택되어도 3.2절에서 정의한 적합성 관계에 의해서 적합하다는 결론을 내릴 수 있다. 또한, 입력의 경우에 시험기의 의도에 따라 각각의 입력을 적용할 수 있으므로 적합성 관계의 정의에 의해서 입력 $?i_1$, $?i_2$, …, $?i_n$를 포함한 모든 입력의 적용이 가능해 진다. 정점 v에서 각각의 입력 $?i_1$, $?i_2$, …, $?i_n$이 최소 한번은 적용될 수 있도록 서브트리가 충분히 여러번 적용될 수 있다면 각 입력 $?i_k$에 대해서 배타집합 $E_k \in Eset(v)/?i_k$를 얻을수 있다. 모든 입력들 $?i_1$, $?i_2$, …, $?i_n$에 대한 전체 서브 트리의 배타집합의 집합은 $Eset(v)/\{?i_1, ?i_2, \cdots, ?i_n\}$ 혹은 $Eset(v)$로 표현될 수 있다고 할때 이 집합의 원소들은 배타집합들의 n-wise intersection에 해당한다. 이 내용을 수식화해서 나타내면 다음과 같다.

$$Eset(v) = \{E \mid E = \bigcap_{k=1}^{n} E_k, \quad E_k \in Eset(v)/?i_k\}$$

위식에서 E_k는 각각의 입력에 의한 배타집합 $Eset(v)/?i_k$들 중에서의 한 배타집합에 해당한다.



a. Action Tree(B)

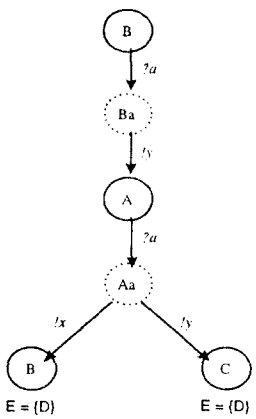


그림 9. 상태 B의 PUIOTree 생성

예를들어 그림 9.a의 트리에서 루트 정점 B (v_0)는 입력 ?a와 입력 ?b에 대한 두개의 링크를 가지고 있다. 이 입력들에 대한 배타집합들은 $Eset(v_0)/?a=\{\{D\}\}$와 $Eset(v_0)/?b=\{\{C, D\}\}$에 해당한다. 모든 가능한 입력 ?$a$와 ?$b$에 대한 v_0의 배타집합들에 대한 집합은 다음과 같다.

$$Eset(v_0)=Eset(v_0)/\{?a, ?b\}=\{\{D\}\}$$

프로토콜 적합성 시험에서 최적의 PUIOTree가 되기 위해서는 최소의 배타집합들에 대한 집합을 가져야 한다. PUIOTree의 모든 정점에서 가능한 모든 입력에 의한 전이들을 포함하므로써 최적의 PUIOTree를 구성할 수 있다. 그러나 한 정점에서 가능한 입력들의 부분집합만을 가지고도 최소의 배타집합들을 계산할 수 있기 때문에 다른 입력들은 필요하지 않은 경우도 있다. PUIOTree의 각 정점에서 입력의 수가 적을수록 외부 시험기가 이 PUIOTree를 적용시킬때 시험수행의 횟수가 적어지게 된다. 이와같이 각 정점에서 입력들의 수가 최소인 PUIOTree가 최적의 트리가 된다. $Iset$내의 모든 입력 ?i가 최소 한번은 적용될 때 PUIOTree내의 정점 v를 루트로 하는 서브트리에 대한 배타집합들의 집합을 $Eset(v)/Iset$이라 할때 이는 다음과 같은 식으로 표현할 수 있다.

$$Eset(v)/Iset=\{E \mid E=\bigcap_{?i \in Iset} E_{?i},\ \text{여기서, } E_{?i}\in Eset(v)/?i\}$$

PUIOTree의 정점 v에서 모든 가능한 입력들의 집합을 $Iposs$라 하자. $Iposs$의 부분집합 $Isub$ (즉, $Isub \subset Iposs$)에 대해서 만일 $Eset(v)/Iposs=Eset(v)/Isub$이라면 입력 i ($i\in(Iposs-Isub)$)에 의해서 도달할 수 있는 정점 v의 서브트리들은 필요가 없다. 따라서 이러한 서브트리들은 시험능력의 저하없이 잘려져도 무방하다. 이때 $Eset(v)/Iposs=Eset(v)/Isub$에 해당하는 입력들의 최소 집합을 $Imin$이라 한다. PUIOTree의 각 정점이 $Imin$ 내의 입력들에 의한 전이만을 포함하게 함으로써 최적의 PUIOTree를 생성할 수 있다. 예를들면 그림 9.a에서 트리의 루트 정점 (이를 v_0라 하자.)은 두개의 가능한 입력 ?a와 ?b를 가지고 있다. 이 트리의 배타집합들의 집합 $\{\{D\}\}$는 입력 ?a만을 가지고도 얻을수 있으므로 정점 v_0에서 $Imin$은 $\{?a\}$가 된다. 그러므로, 최적의 PUIOTree를 구성하기 위해서 입력 ?b에 해당하는 링크는 잘라내도 무방하다.

V. 시험스위트 생성

한 전이를 시험하는데 필요한 시험경우는 다음과 같은 네가지 요소들의 결합으로 이루어진다.

1. NFSM내에서 전이의 시작상태로 시스템을 이동시키는 순열
2. 시험하고자 하는 전이의 입력과 이 입력에 의한 출력
3. 전이의 도달상태를 검증하기 위한 도달 상태의 시그니쳐
4. 시스템을 다시 초기 상태로 이동시키기 위한 특별한 입력인 ri

한 프로토콜의 적합성을 시험하는데 필요한 모든 시험경우의 집합을 시험 스위트라 한다. 이런 시험 스위트를 생성하기 위해서는 시스템의 비결정적인 성질 때문에 SUT를 초기 상태에서 시스템의 모든 상태로 이동 시킬 수 있는 순열들이 $PathTree$ 형태로 표현되어야 한다.

정의 (PathTree) NFSM M의 $PathTree$는 $PathTree(M)$으로 표현되며, 행동트리 $\langle V, L, v_0\rangle$에 해당한다. 즉, $state(v_0)=s_0$이고 각각의 상태 $s\in S$에 대해서 $state(v)=s$인 정점 $v\in V$가 최소 하나는 존재해야 하며

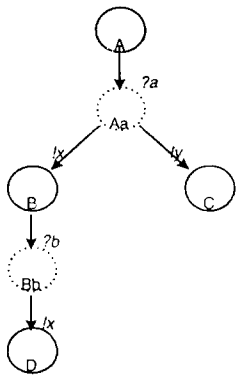


그림 10. PathTree

$state(v_{tran}) \neq s$이다.

여러개의 *PathTree*가 존재하는 경우에는 최소 깊이의 트리를 선택하는 것이 시험 경우를 짧게 만들어 주기 때문에 더욱 효과적이다. 이런 *PathTree*의 예를 그림 10에 나타내었으며 이를 이용해서 각각의 전이 $s_s \xrightarrow{i/o} s_d$에 대한 시험 스위트를 생성하는 절차를 다음에 나타내었다.

1. 일시적인 정점 v_{tran}을 사용해서 전이 $s_s \xrightarrow{i/o} s_d$를 $v_s \xrightarrow{?i} v_{tran} \xrightarrow{?o} v_d$로 변형한다. 이 변형된 전이를 $state(v_s)=s_s$에 해당하는 *PathTree*(*S*)의 정점 v_s에 연결시킨다.
2. *UIOTree*(s_d)나 *PUIOTree*(s_d)를 정점 v_d의 서브 트리로 첨가한다.
3. 시험 수행의 최종결과에 해당하는 결정 부분을 *UIOTree*(s_d)나 *PUIOTree*(s_d)의 각 단말 정점에 덧붙힌다. 이때 *UIOTree*(s_d)가 붙는 모든 단말 정점들은 "성공"이라는 결정 부분을 갖게 된다. *PUIOTree*(s_d)가 붙는 경우에는 배타집합이 ϕ인 단말 정점들은 "성공(Success)"이라는 결정 부분을 갖게 되며, 배타집합이 ϕ가 아닌 단말 정점들은 "비결정(Inconclusive)"이라는 결정 부분을 갖게 된다.

그림 11은 전체 시험 스위트의 일부분을 표시한 예로써, 점선으로 나타나 있는 부분이 *PathTree* 부분이며 굵은 화살표로 표시된 부분은 시험하고자 하는 전이부분에 해당한다. 이 그림에서는 $C \xrightarrow{b/y} A$ (전이 t5)의 전이가 비결정적으로 발생시 상태 A에 대한 상태 시그니처 즉, 상태 A에 첨가된 *UIOTree* 혹은 *PUIOTree*를 시험기에 가해서 최종적으로 이 전이에 대한 성공의 판정을 내릴 수 있는 예에 해당한다.

Ⅵ. 결론 및 향후 연구 과제

프로토콜의 적합성 시험이란 프로토콜 명세를 기반으로 하여 구현된 프로토콜이 실제로 프로토콜 명세에서 요구하고 있는 모든 사항에 적합한지를 시험하는 과정이다. 그러나, 이러한 과정은 전문가에 의해서 수동적으로 행해지는 경우 막대한 양의 시간과 노력을 필요로 한다. 따라서 프로토콜 적합성 시험의 자

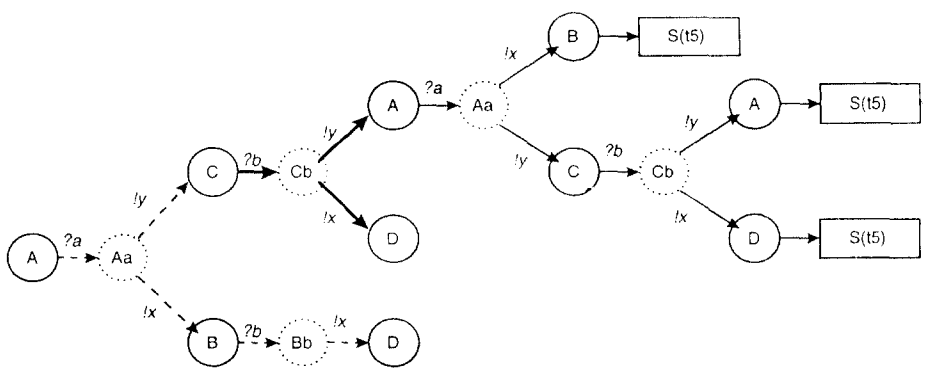


그림 11. 시험 스위트의 일부분

동화가 절실히 요구되며 이를 위해서는 형식 명세 기술 언어로 표준화 되어있는 Estelle[13] 이나 LOTOS [14]등과 같은 형식화된 프로토콜 명세를 기반으로 하여 시험 경우를 자동으로 생성해야할 필요가 있다. 이런 시험 경우의 자동 생성에 대한 많은 연구들이 지금까지 진행되어 왔지만 각 연구가 자동화에 필요한 모든 사항을 모두 만족하는 연구 결과는 제시하지 못하고 있다. 따라서 본 논문에서는 적합성 시험의 자동화가 이루어지기 위해서 필수적인 요소인 시험 경우의 자동 생성 방법에 대해서 제안하였고 형식적인 적합성 관계를 정의하였다.

또한 대부분의 프로토콜 명세는 비결정성을 갖고 있기 때문에 결정적 유한상태기계에서만 시험 경우를 생성할 수 있는 방법은 실제 프로토콜에 적용할때 많은 제한점을 가지고 있다. 이런 연유로 본 논문에서는 비결정적 유한상태기계로 표현되는 프로토콜에서 시험 경우를 생성하기 위해서 행동트리를 이용하는 방법을 제시하였다.

그러나 본 논문에서 제시하는 방법은 프로토콜의 제어의 흐름(control-flow)만을 시험 할 수가 있다. 실제 프로토콜에서는 제어의 흐름만이 아니라 한 상태에서 다른 상태로 전이가 일어날때 발생하는 데이타의 흐름(data-flow)까지도 고려해야한다. 또한 하나의 상태내에서도 내부 변수의 값들이 변경되는 경우가 상당히 많이 존재한다. 따라서 본 연구에서 제안하는 연구 결과가 실제 프로토콜을 시험하기 위한 완벽한 시험기가 되기 위해서는 이런 데이타의 흐름문제와 내부 변수가 존재하는 경우의 시험능력도 포함되어야 한다.

참 고 문 헌

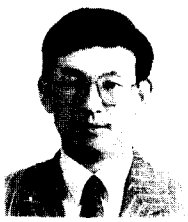
1. S. Naito and M. Tsunoyama. Fault Detection for Sequential Machine by Transition Tours. In *Proc. IEEE Fault Tolerant Computing Conf.*, 1981.
2. K. Sabnani and A. Dahbura. A Protocol Test Generation Procedure. *Computer Networks and ISDN Systems* 15(4), September 1988, 285-297.
3. T. S. Chow. Testing Software Design Modeled by Finite State Machine. *IEEE Transactions on Software Engineering* 4(3), May 1978, 178-187.
4. Z. Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New-York, 1978.
5. A. V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar. An Optimization Technique for Protocol Conformance Test Generation based on UIO Sequences and Rural Chinese Postman Tours. In S. Aggarwal and K. Sabnani (eds), *Protocol Specification, Testing, and Verification VIII*, North-Holland, Amsterdam, 1988, 75-86.
6. W. Y. L. Chan, S. T. Vuong, and M. R. Ito. An Improved Protocol Test Generation Procedure Based on UIO's. *SIGCOMM '89 Symposiun: Communications Architecture and Protocols in Computer Comm. Review* 19(4), September 1989, 283-294.
7. W. Chun and P. D. Amer. Improvements in UIO Sequence Generation and Partial UIO Sequences. Technical Report, Comp. & Info. Sci., University of Delaware, 1992.
8. W. Chun. *Test Case Generation for Protocols Specified in Estelle*. Ph.D. Thesis, Comp. & Info. Sci., University of Delaware, 1992.
9. Information Processing Systems - *Open System Interconnection. ISO International Standard 9646: OSI Conformance Testing Methodology and Framework*.
10. P. Tripathy and K. Naik. Generation of Adaptive Test Cases Nondeterministic Finite State Models. In Proc. 5th International Workshop on Protocol Test Syatems, Montreal, Québec, Canada, September 1992.
11. M. Ghriga and P. Frankl. Adaptive Testing of Non-Deterministic Communication Protocols. In Proc. 6th International Workshop on Protocol Test Syatems, Pau, France, September 1993.
12. A. Petrenko, N. Yectushenko, A. Levedev, and A. Das. Nondeterministic State Machines in Protocol Conformance Testing. In Proc. 6th International Workshop on Protocol Test Syatems, Pau, France, September 1993.
13. Information Processing Systems - Open System Interconnection. *ISO International Standard 9074*:

Estelle-A Formal Description Technique Based on an Extended State Transition Model.

14. Information Processing Systems-Open System Interconnection. *ISO International Standard 8807: LOTOS-A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour.*

金 柄 植(Byeongsik Kim) 정회원

1986년 3월~1993년 2월 : 충남대학교 컴퓨터공학과 (공학사)
1993년 3월~1995년 2월 : 충남대학교 컴퓨터공학과 (공학석사)
1995년 2월~현재 : 충남대학교 컴퓨터공학과 박사과정
1997년 8월~현재 : 미국 National Institute of Standards and Technology, Guest Researcher



全 遇 稷(Woojik Chun) 정회원

1978년~1982년 : 서울대학교 컴퓨터공학과(학사)
1982년~1984년 : 서울대학교 컴퓨터공학과(석사)
1984년~1987년 : 한국전자통신연구소 데이터통신연구실
1987년~1989년 : University of Delaware 전산학과(석사)
1989년~1992년 : University of Delaware 전산학과(박사)
1992년~1993년 : 한국전자통신연구소 정보통신표준센타
1993년 4월~현재 : 충남대학교 컴퓨터공학과 조교수
1997년 8월~현재 : 미국 National Institute of Standards and Technology, Guest Researcher