論 文

효율적인 통신프로토콜 시험을 위한 비결정성 제거 알고리즘

正會員 許 基 澤* 正會員 李 東 豪**

A Nondeterminism Removal Algorithm for Efficient Testing of Communication Protocols

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

요 약

프로토콜을 명세할때 결정적 유한상태기계(Deterministic Finite State Machine)가 프로코톨의 제어흐름을 쉽게 나타낼 수 있어서 시험항목 생성시 주로 사용되었으나, 실제의 프로토콜들은 한개의 입력에 의해서 한개 이상의 상태로 천이가 발생하는 비결정성(Nondterminism)문제들을 내포할 수 있으므로 결정적 유한상태기계로는 비결정성 문제를 처리할 수 없다. 따라서 본 논문에서는 프로토콜을 먼저 프로토콜의 비결정성 특성을 잘 나타낼 수 있는 비결정적 유한상태기계(Nondeterministic FSM)로 나타내고, 이를 결정적 유한상태기계를 변환하기 위한 알고리즘을 제시하였다.

ABSTRACT

DFSM(Deterministic Finite State Machine) is used because it easily represents the control flow of a protocol in the protocol specification. Real protocols contain problem of nondeterminisms that have more than one enabled transition in the same state by same input. But DFSM does not process nondeterminism. So, in this paper, we first specify a protocol with NFSM(Nonderministic FSM) that may show the characteristics of nondeterminism, and propose an algorithm which converts NFSM to DFSM.

I. 서 론

복잡한 분산시스템의 동작은 일반적으로 프로토콜이라는 일련의 규칙들로 서술되고 분석될 수 있다. 유한상태기계에 관한 이론들이 광범위하게 연구되었고 프로토콜의 제어흐름을 쉽게 나타낼 수 있어서 시험항목 생성시(Test Case Generation) 결정적 유한상태기계가 주로 사용되었다. 그러나 프로토콜이 상태 천이 시스템 모델로 표현될때 많은 프로토콜들은 한상태에서 한개의 똑같은 입력에 의해서 하나 이상의 상태로 천이(Transition)가 발생할 수 있는 비결정성 문제들을 포함할 수 있는데, 이로 인하여 천이에 대한 예측이 어렵기 때문에 프로토콜의 제어흐름을 나타내기가 어려울 뿐만 아니라 지금까지 잘 알려

*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 : 93-158

졌던 결정적 유한상태기계에 적용되었던 시험항목 생성 방법들을 그대로 적용하여 시험항목을 생성할 수 없다[3,5,8,10].

따라서 천이에 대한 예측이 가능토록 하기 위해서는 먼저 프로토콜을 프로토콜들의 비결정성을 잘 나타낼 수 있는 비결정적 유한 상태기계로 서술하고, 이를 그 특성들을 그대로 유지하는 결정적 유한상태기계로 변환하기 위한 알고리즘이 필요하다. 이를 위해서 시험 항목 생성시 영향을 미치는 비결정성 유형과 특성들을 조사하고 이를 제거하기 위한 알고리즘을 제시하였다.

Ⅱ. 시험에서의 비결정성 유형

기존의 시험항목 생성방법들은 주로 하나의 입력값에 의해서 하나의 enabled 천이만이 존재하는 결정적 유한상태기계만을 고려하였다. 그러나 실제의 많은 프로토콜들은 한개의 외부입력값에 의해서 발생되는 천이의 수가 한개 이상인 경우가 있거나, 혹은 운영체제나 네트워크 경영자에 의해서 거절될 수 있는 천이는 IUT가 제어하지 않아도 상태가 변할 수 있는 천이들이므로 이들을 처리하기 위한 방법들이 필요하다[1,2,3,6,8]. [6]에서는 비결정성 유형 3가지와 Well-Defined규격을 먼저 정의하고서 비결정성으로 부터 Well-Defined 규격의 정의를 이끌어 냈고, [4]에서는 비결정적 유한상태로부터 시험항목을 생성하기 위해서 적응적 절차(Adaptive Procedure)를 제안하였고, [9]은 비결정적 유한상태를 정의하고, 이를 위한 UIO트리와 PUIO트리를 생성하여 시험항목을 더 쉽게 생성할 수 있는 형태로 변환하여 이를 처리하였다. 그러나 이 방법들은 시험항목 생성시 비결정성을 갖도록 확장함으로서 천이에 대한 예측이 어렵기 때문에 행위를 순서적으로 예측할 수 없다. 따라서 이런 문제점들을 해결하기 위해서 먼저 비결정성의 유형을 살펴보면 다음과 같다[6,9].

①내부 비결정성(Internal Nondeterminism)

어떤 외부의 입력이 없이도 천이가 발생하므로 테스터에 의해서는 제어 될 수 없다. 내부 상황에 관한 상세한 정의는 구현자에 의해서 결정되므로 테스터의 관점에서는 어떤 천이의 선택도 합법적인 것으로 취급되어야만 한다.

②구분가능 비결정성(Distinguishable Nondeterminism)

똑같은 입력에 의해서 enable된 모든 천이가 각각 서로 다른 출력을 생성하는 것으로 테스터는 fire될 천이가 어떤 것 인가를 정확하게 탐지할 수 있다.

③구분 불가능 비결정성(Indistingushable Nondeterminsm)

똑같은 입력에 의해서 enable된 모든 천이가 똑같은 출력을 생성하는 것으로 테스터는 어떤 천이가 fire될 것인가를 탐지할 수 없다.

적합성 시험은 종종 Black Box 시험 형태로 수행되므로[5,7,11] 테스터는 IUT의 내부동작에 관한 정보를 가지고 있지 않으므로 내부 비결정성을 처리할 수 없다. 그리고 구분 가능 비결정성은 똑같은 입력에 의해서 하나 이상의 상태로 천이가 발생하지만 출력값이 서로 다르므로 입출력 순서의 쌍을 동시에 고려하면 서로 다른값이 된다[6,9]. 따라서 구분가능 비결정성도 테스터에 의해서 유일하게 구분될 수 있으므로, 본 논문에서는 구분 불가능 비결정성만을 제

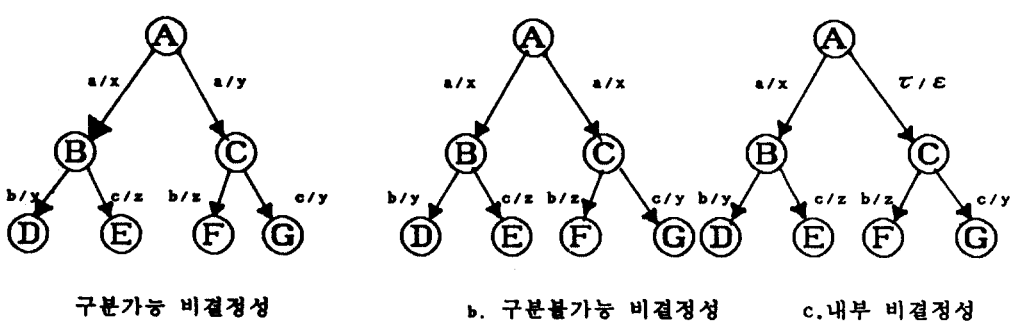


그림 1. 비결정성의 유형
Fig 1. A Type of Nondeterminism

거하는 알고리즘을 제시하였다.

Ⅲ. 비결정성 제거 알고리즘

본절에서는 시험항목 생성시 포함된 비결정성을 제거하기 위해서 먼저 비결정적 유한상태기계모델과 결정적 유한상태기계모델을 정의하고, 비결정성을 제거하기 위한 알고리즘을 제시하였다.

3.1 유한상태기계모델

프로토콜 시험시 포함될 비결정성 문제를 처리하기 위해서 먼저 프로토콜 명세시 프로토콜의 비결정성들을 잘 나타낼 수 있는 비결정적 유한 상태기계(Nondeterministic Finite State Machine : NFSM)를 정의하고, 이를 이용하여 비결정성 유형들을 결정적 유한상태로 변환하기 위한 알고리즘과 결정적 유한상태기계 모델로 변환된 후에 상태수를 최소화 하기 위해서 서로 중복되는 상태들을 제거하기 위한 방법을 제시하였다.

정의3.1 (비결정적 유한상태기계)

비결정적 유한 상태기계는 5개의 쌍인 NFSM = $\langle G, g_0, I, O, T\rangle$로 표현된 천이 시스템이다.

여기서 G : 상태들의 유한집합
g_0 : 초기상태($g_0 \in G$)
I : 외부입력의 유한집합
0 : 외부출력들의 유한집합
T : $G*(I \cup \{\tau\}) \rightarrow G*(0 \cup \{\varepsilon\})$인 천이함수
(단 $\tau \in I$는 내부입력이고, $\varepsilon \in 0$는 내부출력이다.)

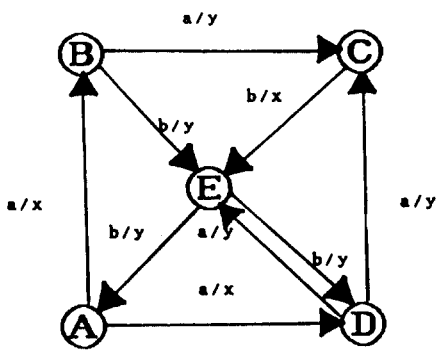


그림 2. 비결정성의 유한상태기계
Fig 2. Nondeterministic Finite State Machine

상태천이함수 T는 $g*I \rightarrow g'*0$ 형태로 표현되는데, 이것은 상태 g에서 I가 입력되면 0를 출력하고 상태 g'로 천이가 이루워지는 것으로 이를 간력히 $g \xrightarrow{i/o} g'$ 형태로 나타낸다.

예를들면 그림2에서 상태A는 한개의 똑같은 입출력 순서 a/x에 의해서 두개의 상태인 B와 D로 천이가 발생하므로 비결정성을 내포하고 있다. 따라서 비결정적 유한상태기계인 N = $\langle G, g_0, I, O, T\rangle$로 표현될 수 있고, 여기서 G = {A, B, C, D, E}, g_0 = A, I = {a, b}, 0 = {x, y}, T는 $\{A \xrightarrow{a/x} B, A \xrightarrow{a/x} D, B \xrightarrow{a/y} C, B \xrightarrow{b/y} E, C \xrightarrow{b/x} E, D \xrightarrow{a/y} C, D \xrightarrow{a/y} E, E \xrightarrow{b/y} A, E \xrightarrow{b/y} D\}$이다.

정의3.2 (결정적 유한상태기계)

결정적 유한 상태기계는 5개 쌍인 DFSM = $\langle G', g_0, I, O, T'\rangle$로 표현된 천이 시스템이다.

여기서 G' : 상태들의 유한 집합
g_0' : 초기상태
I : 유한입력 집합
0 : 유한출력 집합
T' : $G'*I \rightarrow G'*0$인 천이함수

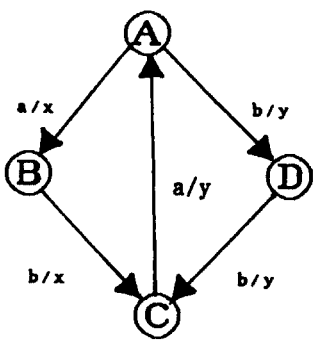


그림 3. 결정적 유한상태기계
Fig 3. Deterministic Finite State Machine

그림3에서 각 상태는 모든 입력값들에 대해서 오직 한개의 출력값만을 가지므로 비결정성을 내포하고 있지 않다. 따라서 결정적 유한상태기계인 D = $\langle G', g_0, I, 0, T'\rangle$로 표현될 수 있고, G' = {A, B, C, D}, g_0' = A, I = {a, b}, 0 = {x, y}, T = $\{A \xrightarrow{a/x} B, A \xrightarrow{b/y} D, B \xrightarrow{b/x} C, C \xrightarrow{a/y} A, D \xrightarrow{b/y} C\}$이다

3.2 비결정성 제거 알고리즘

테스팅 목적을 위해서는 한 상태에서 발생되는 하나의 천이에 의해서 통합여부가 결정되므로 먼저 이들을 정의하면 다음과 같다.

정의3.3 (구분가능 상태)

한 상태 g에서 하나의 천이에 의해서 오직 한 상태로만 천이가 발생하면 이를 구분가능 상태라 한다.

정의3.4 (구분 불가능 상태)

한 상태 g에서 하나의 천이에 의해서 하나 이상의 상태로 천이가 발생하면 이를 구분 불가능 상태라 한다.

정의3.5 (통합가능상태)

서로 다른 상태가 구분 불가능 상태이면 이들은 통합 가능 상태가 된다.

정의3.4에 구분 불가능 상태는 하나의 천이에 의해서 하나 이상의 상태로 천이가 발생한다. 그런데 적합성 시험은 Black Box 형태로 이루어지므로 한 상태에서 똑같은 천이에 의해서 하나 이상의 상태로 천이가 발생하면 테스터 관점에서는 이들을 구분할 수 있는 방법이 없다. 따라서 이들을 똑같은 상태로 볼 수 있으므로 결국은 통합가능상태가 된다. 즉 하나의 상태 g_i에서 하나의 천이에 의해서 하나 이상의 상태인 $\{p_0,\ldots,p_j\}$로 천이가 발생하면 $p_0,\ldots,p_j$를 똑같은 한 개의 상태인 P_{new}로 처리하고 $p_0,\ldots,p_j$에 의해서 발생되었던 천이는 새로운 상태인 P_{new}에 의해서 발생된 천이로 처리하면 구분 불가능 상태가 통합가능 상태가 된다. 따라서 통합가능상태들을 통합하여 하나의 상태를 구성하면 구분 불가능 비결정성이 제거될 수 있으므로 이를 위한 알고리즘을 다음에 제시하였다.

```
Algorithm( Convert NFSM to DFSM)
/* Construct DFSM D = ( G', g0', I , O, T') from given NFSM N= ( G , g0 ,I , O ,T ).
   Let 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.
   The State of G' is represented by the form [p1. .... pj]. */

(1)   g0' = [ g0 ]
(2)   Put a Vertex Vs = < gs ,Ps > into a queue.
(3)   WHILE ( queue is not empty ) DO
(4)         remove a vertex Vt = < gt , Pt > from queue.
(5)         IF  two more states are contained in Pt
(6)             THEN {
(7)                   Combine them  into one state [Pt] ∈ T'.
(8)                   FOR each state gi ∈ Pt DO
                                                        d
(9)                       IF ∃ transition of the form gi ---> gd
(10)                         THEN {
                                                          d
(11)                                 replace it with [Pt] ---> gd.
(12)                                 insert Vertex < [Pt] ,gd > into a queue.
(13)                              }
(14)                    DONE FOR
(15)                 }
(16)             ELSE
                                                    t
(17)                   replace it with [gt] ---> [Pt].
(18)    DONE WHILE
```

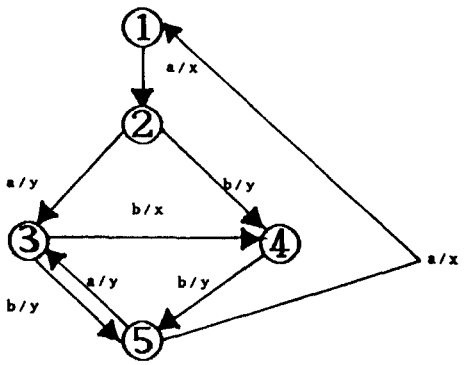
Ⅳ. 적용및 분석

알고리즘이 수행되는 과정을 그림2에 적용시킨 결과를 상태천이표로 나타낸 것이 표1에 있다. 초기상태 A에서 똑같은 입출력 순서 a/x에 의해서 상태 B와 D로 천이가 발생하므로 이를 통합하여 새로운 상태 [B, D]상태에서는 입출력 순서 a/y에 의해서 상태 C와 E로 천이가 발생하므로 이를 통합하여 새로운 상태 [C, E]를 구성하고, b/y에 의해서는 상태 E로 천이가 발생하므로 새로운 상태 [E]를 구성한다. 상태 [C, E]에서는 입출력 순서 b/x에 의해서 E로 b/y에 의해서는 [A, D]로 천이가 발생하므로 이를 처리하고, 통합된 상태 [A, D]에서는 입출력 순서 a/x에 의해서 A로, a/y에 의해서는 [C, E]로 천이가 발생하므로 이를 처리한다. 마지막으로 새로운 상태 [E]에서는 입출력 순서 b/y에 의해서 [A, D]로 천이가 발생하므로 이들을 처리하여 나타낸 결과가 그림3과 같다. 그림3은 표1에서 1=[A], 2=[B, D], 3=[C, E], 4=[E], 5=[A, D]로 표시하여 나타낸 결정적 유한상태기계모델이다.

표 1. 상태천이표
Table 1. State Transition Table

I/O Sequence / state	a/x	a/y	b/x	b/y
[A]	[B, D]			
[B, D]		[C, E]		[E]
[C, E]			[E]	[A, D]
[E]				[A, D]
[A, D]	[A]	[C, E]		

그림5는 트랜스포트 프로토콜의 동작을 LOTOS로 나타낸 것으로 단순성을 위해서 트랜스포트 프로토콜중 자료부분을 제외한 제어부분만을 LOTOS로 명세한 것이다. 이를 비결정적 유한상태기계모델로 나타낸 것이 그림6에 있다[13]. 그림6에서 상태3은 똑같은 출력값 L?CC에 의해서 두개의 상태인 14와 19로 천이가 발생하고, 상태10에서도 내부천이를 나타내는 i에 의해서 상태11과 상태18로 천이가 발생하므로 비결정성을 포함하고 있다. 따라서 비결정성을 제거하기 위해서 제안된 알고리즘을 적용시킨 후의 결과가 그림7에 있다.



천이 t1:(1,2;a/x)
t2:(2,3;a/y)
t3:(2,4;b/y)
t4:(3,4;b/x)
t5:(3,5;b/y)
t6:(4,5;b/y)
t7:(5,1;a/x)
t8:(5,3;a/y)

그림 4. 그림2에 대한 결정적 유한상태기계
Fig 4. Deterministic Finite State Machine of Figure 2

지금까지 비결정성 문제를 처리하기 위해서 먼저 비결정성을 잘 나타낼 수 있는 비결정적 유한상태기계를 정의하고 이를 결정적 유한상태기계로 변환하기 위한 알고리즘을 제시하였다. 그런데 변환 알고리즘을 적용하여 나타낸 결정적 유한상태기계 모델이 변환되기 전에 인식되었던 모든 천이들을 인식할 수 있는가의 여부를 검증하기 위한 과정이 필요하다. 이를 위해서 Tr(S)를 S가 받아들일 수 있는 입출력 순서들의 집합이라고 하면 [1,9,11,12], 다음과 같은 정리4.1이 성립될 수 있다.

정리4.1 (적합성 동치관계)
적합성 시험시 두개의 유한상태기계 N과 D에 대해서 적합성 동치(Conformance Equivalence)가 성립하기 위한 필요충분조건은 Tr(N)=Tr(D)이다.
(증명)
정의 3.1에서 정의된 비결정적 유한상태기계 N에서 인식할 수 있는 모든 입출력 순서가 정의 3.2에서 정의된 결정적 유한상태기계 D에 의해서도 인식되어야하고 역으로 D에 의해서 인식되었던 모든 입출력 순서도 역시 N에 의해서 인식되어야 두개의 유한상

```
specification T_Protocol[U,L]no exit
    behavior TP_Simple[U,L]
    where

      process TP_Simple[U,L]: noexit :=
        U?TCONreq;L!CR (L?CC;L!DR;U!TDISind;L?DC;TP_simple[ts,ns]
                        []
                        L?CC;U!TCONconf;Open[ts,ns]
                       )
        []
        L?CR(L!DR;TP_simple[U,L]
             []
             U!TCONind( Uclose[U,L]
                        []
                        U?TCONresp;L!CC;Open[U,L]
                       )
             )
      endproc

      process Open[U,L]: noexit :=
        (U?TDATAreq;L!DT;Open[U,L]
         [] L?DT;U!DT;Open[U,L]
         [] L?AK;Open[U,L]
         [] Uclose[U,L]
         [] Rclose[U,L]
         [] i;U!TDISind;L!DR;L?DC;Open[U,L]
         [] i;L!AK;Open[U,L]
        )
      endproc

      process Uclose[U,L]: noexit :=
        U?TDISreq;L!DR;L?DC;Open[U,L]
      endproc

      process Rclose[U,L]: noexit :=
        L?DR;U!TDISind;L!DC
      endproc

endspec
```

그림 5. LOTOS로 명세된 트랜스포트 프로토콜
Fig 5. A transport protocol specification in basic LOTOS

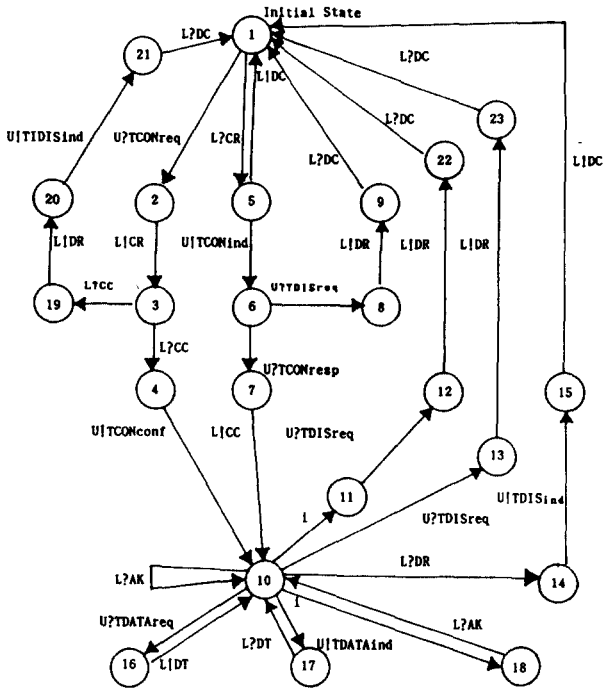


그림 6. 그림5에 있는 트랜스포트 프로토콜에 대한 비결정적 유한상태기계
Fig 6. NFSM for the transport protocol specification in Fig.5

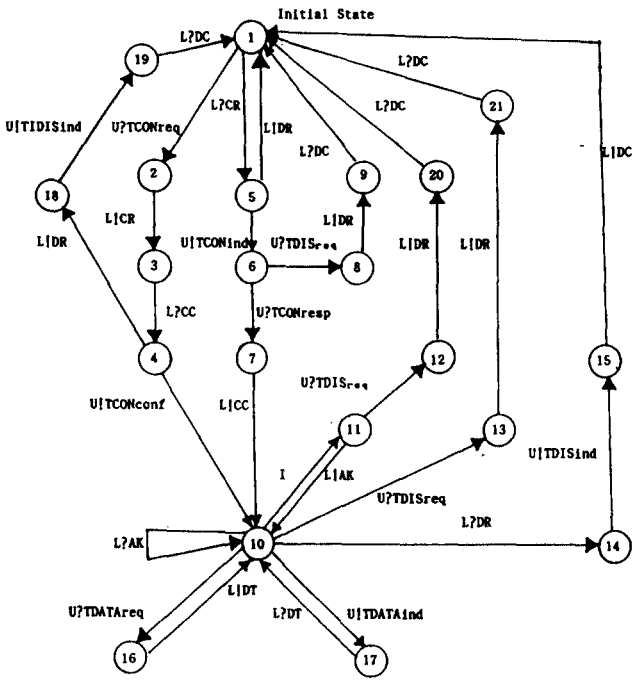


그림 7. 결정적 유한상태기계로 표현된 트랜스포트 프로토콜
Fig 7. The transport protocol specified with DFSM

태기계 N과 D를 동치관계로 볼 수 있다. 따라서 n을 상태수라 하고 귀납법을 사용하여 증명하면 다음과 같다.

i) $n=1$이면 상태수가 하나이므로 $G=g_0$이고 $G'=g_0'=\{g_0\}$이므로 성립된다.

ii) $n=k$일때 상태 g_0에서 g_n까지 발생되는 천이들의 집합으로 구성된 path $\{t_1,...,t_n\}$이 N상에 존재하면 D에도 상태 g_0'에서 g_n'까지의 path $\{t_1,...,t_n\}$이 존재한다고 가정한다.

iii) $n=k+1$ 일때 N상에 상태 g_0에서 새로운 상태 g_{n+1}까지의 path $t_1,...,t_{n+1}$이 존재할때 즉 $g_n \xrightarrow{t_{n+1}} g_{n+1} \in T$ 일때 D상에 $g_n' \xrightarrow{t_{n+1}} g_{n+1}' \in T'$임을 보이면 된다. ii)에서 D상에 상태 g_0'에서 상태 $g_n'(g_n \in g_n')$까지의 path $t_1,...,t_n$이 존재하고 $g_n \xrightarrow{t_{n+1}} g_{n+1} \in T$ 이므로 D에도 상태 g_{n+1}로 천이를 발생시키는 t_{n+1}이 존재해야만 한다. 따라서 g_{n+1}를 포함하는 상태들의 집합을 g_{n+1}'라고 하면 D에도 상태 g_0'에서 상태 g_{n+1}'까지의 path $t_1,...,t_{n+1}$이 존재한다. 역으로, D상에 g_0'에서 g_{n+1}'로 천이를 발생시키는 path $t_1,...,t_{n+1}$이 존재하면 $g_n \in g_n'$에 대해서 N상에도 상태 g_0에서 g_{n+1}까지의 path $t_1,...,t_{n+1}$이 존재해야만 한다. 그런데 D상에 $g_n' \xrightarrow{t_{n+1}} g_{n+1}'$이 존재하고 $g_{n+1}'(g_{n+1}' \in g_{n+1})$은 N에 있는 상태들의 집합이므로 $g_n' \xrightarrow{t_{n+1}} g_{n+1}' \in T$가 성립된다.

예를들면 그림1.b에 있는 비결정적 유한상태기계를 변환알고리즘을 적용하여 나타낸 결정적 유한상태기계가 그림8에 나타나 있다. 여기서 그림1.b에서 정의되었던 입출력 순서들인 a/x b/y, a/x c/z, a/x b/z, a/x c/y가 그대로 그림8에서도 정의되므로 두개의 유한상태 기계를 적합성 동치관계로 볼 수 있다. 마찬가지로 그림2에서 정의되었던 모든 입출력 순서의 쌍들인 a/x b/x, a/x b/y, a/x b/y a/y, a/x b/y b/x a/y,...등들도 역시 그림4에서 그대로 정의될 수 있으므로 두개의 유한상태기계도 역시 동치관계가 성립된다.

정리4.1에서 변환 알고리즘을 적용할 경우에 비결정성이 제거될 수 있음을 보였지만, 최대 상태수가 2^G만큼 증가될 수 있으므로 상태폭증현상이 발생할 수 있다. 그러나 2^G개의 상태중에서 초기상태를 제외한 상태들중 입력천이가 없고 출력천이만 있는 상태는 초기상태에서 도달할 수 없는 상태이므로 제거되어야한다. 그리고 변환 알고리즘을 적용시켜 나타난 통합된 상태들중에서 상태동치관계가 성립되면 역시 이 상태들도 통합되어야 한다.

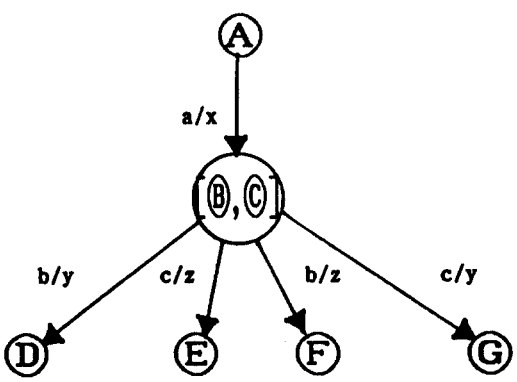
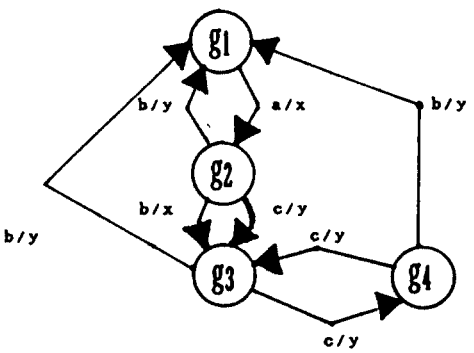
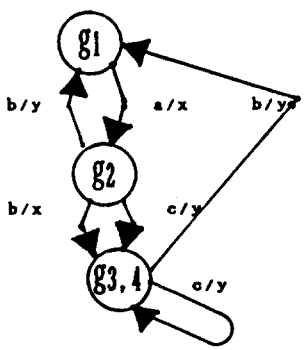


그림 8. 그림1.b를 변환시킨 유한상태기계
Fig 8. The Converted DFSM of Figure1.b



(a)



(b)

그림 9. 상태동치관계
Fig 9. Equivalence Relation of States

정의3.6 (상태동치관계)

서로 다른 상태 g_a와 g_b가 똑같은 한개의 천이에 의해서 같은 상태로 천이가 발생하면 상태 g_a와 g_b를 상태동치관계(Equivalence Relation of States)라 한다.

예를들면 그림9.a에서 상태 g_3와 g_4는 똑같은 한개의 천이 c/y에 의해서 같은 상태인 g_1으로 천이가 발생하므로 상태동치관계가 성립된다. 그러므로 상태 g_3와 g_4는 한개의 상태인 $g_{3,4}$로 통합되고 그 결과가 그림9.b에 있다. 따라서 상태수를 최소화하는 과정은 다음 절차에 의해서 이루워진다.

첫째, 한 상태에 입력천이가 없이 출력천이만 있는 도달불가능 상태를 모두 제거한다.

둘째, 상태동치관계(Equivalence Relation of States)들을 찾아서 초기의 동치류(Equivalence Class)를 구성한다.

셋째, 같은 입출력 순서에 의해서 서로 다른 동치류로 가는 천이가 발생하면 이를 분할하여 새로운 동치류를 구성한다.

마지막으로, 세번째 과정을 더 이상 분할이 일어나지 않을때 까지 반복수행한다.

이 방법을 그림9.a에 적용시키면 초기상태는 두개의 동치류인(g_1, g_2)와 (g_3, g_4)로 분할된다. 먼저 (g_1, g_2)가 분할될 수 있는지의 여부를 결정하기 위해서 모든 입출력 순서의 쌍인 a/x, b/x, b/y, c/y에 대해서 천이가 서로 다른 동치류로 발생되는 가를 조사한다. 그런데 g_1에서는 입출력 순서 a/x에 의해서는 g_2로 천이가 발생하나 g_2에서는 천이가 발생치 않으므로 상태 g_1과 g_2는 서로 다른 동치류로 분할되어야 한다. 그러나 상태 g_3와 g_4는 입출력 순서 c/y에 의해서 같은 동치류인 (g_3, g_4)로 천이가 발생하고 b/y에 의해서는 똑같은 g_1상태로 천이가 발생하므로 g_3과 g_4는 더 이상 분할될 수 없으므로 통합되어야 한다. 따라서 그림9.b에 최소화된 결정적 유한상태기계 모델이 제시되었다.

V. 결 론

본 논문에서는 많은 프로토콜들이 포함하고 있는 비결정성 문제를 처리하기 위한 변환알고리즘을 제안하였다. 이를 위해서 비결정성의 유형중 규격의 행위를 예측하기가 가장 어려운 구분 불가능 비결정성을 처리하기 위해서 먼저 비결정성의 쉽게 나타낼 수 있는 비결정적 유한상태 기계모델을 정의하였고, 이를 결정적 유한상태 기계모델로 변환하기 위한 알고리즘을 제시하였다. 결정적 유한상태 기계모델로 변환된 후에 서로 중복되는 상태를 제거하기 위한 절차가 필요하므로 이를 위한 방법도 역시 제시되었다. 그리고 변환 알고리즘을 적용하여 나타낸 결정적 유한상태 기계모델이 변환되기 전에 인식되었던 천이들을 똑같이 인식할 수 있는가를 조사하기 위해서 유한상태 기계모델들간의 동치관계를 조사하여 검증하였다.

앞으로 더 연구되어야 할 내용은 내부천이를 처리하기 위한 방법과 Fault 모델을 처리할 수 있도록 변환 알고리즘이 더 확장되어야 할 것이고, 프로그램 세그먼트로 서술된 프로토콜의 자료부분을 처리할 수 있는 방법들에 관한 연구가 더 진행되어야 할 것이다.

참 고 문 헌

1. R.D.Nicola, "Extensional Equivalence for Transition Systems," Acta Information 24(2), April, pp.211-237, 1987.
2. Ed Brinksama, "A Theory of the Derivations of Tests," Protocol Specification, Testing, and Verification Ⅶ, North-holland, Amsterdam, 63-74, 1988.
3. B. Sarikaya, V. Koukoulidis, and G.V.Bochman, "Method of Analyzing Extended Finite State Machine Specifications," Computer Communication13(2), March, pp.83-92, 1990.
4. Y.Choi, S.Moon and A.Kershenbaum, "Approaches Utilizing Segment Overlap to Minimize Test Sequence," 10Th International IPIF Symposium on Protocol Testing and Verification, June 1990.
5. D.P. Sidhu, "Protocol Testing : The First Ten Years, The Next Ten Years," 10Th International IPIF Symposium on Protocol Specification, Testing and Verification, June, Invited Paper, 1990.
6. 이도영, "상호통신하는 확장된 FSM모델에 근거한 프로토콜의 형식적 적합성 시험에 관한 연구," 포항공대 박사학위논문, 1991.
7. K.Naik and B. Sarikays, "Testing Communication Protocols," IEEE Software, January, pp.

27-37, 1992.

8. J.P. Favreau, D. Hogrefe and J.Kroon, "Formal Methods in Conformance Testing : Status amd Expectations," Working Draft Reports 1992.
9. W.J. Chun, "Thest Case Generation for Protocols Specified in ESTELLE," Ph.d. Thesis Delaware, May 1992.
10. G.Gotzhein, "On Conformance in the Context of Open Systems," Proceddings of the 12th International Conference On Distributed Computing Systems, June, pp.236-243, 1992.
11. A.Ghedamsi and G.V.Bochman, "Test Result Analysis and Diagnostics For Finite Statate Machines," Proceedings of the 12th International Conference On Distributed Computing Systems, June, pp.244-251, 1992.
12. A. Ghedamsi, G.V. Bochman, and R.Dssoui, "Multiple Fault Diagnostics for Finite State Machines," IEEE INFOCOM'93, Mach, pp. 782-791, 1993.
13. K.Naik and B,Sarikaya, "Testing Communication Protocols," IEEE Software, Jan., pp 27-37, 1992.

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