論文 95-12-12-30

조건 문맥 변수를 고려한 제어 흐름도 생성 기법

正會員 吳 昞 浩*, 禹 星 喜**, 李 相 鎬***

Generation Method of Control Flow Graph Accounting Conditional Context Variable in Protocol Conformance Testing

Byeong Ho Oh*, Sung Hee Woo**, Sang Ho Lee*** Regular Members

要 約

제어 흐름만을 고려한 프로토콜의 적합성 시험은 많은 문제점을 갖는다. 본 논문에서는 Estelle로 명세화된 프로토콜의 제어 흐름에 조건 문맥 변수를 통합하는 흐름도 생성 방법을 제안한다. 생성 과정은 명세서를 파서를 사용하여 필요한 테이블로 분해하고 조건 문맥 변수는 특성에 따라 입력과 상태에 통합한다. 기호 테이블과 동작 테이블에 의해 상태 전이에 따른 도달 가능 나무를 만들며 이를 토대로 조건 문맥 변수가 고려된 통합 흐름도를 생성한다. 통합 흐름도에 의해 생성된 테스트 스위트는 제어 흐름만을 고려하는 경우에 발생하는 비결정성 및 실행 불가능 경로를 해결할 수 있다.

ABSTRACT

Protocol conformance tests, which consider only control flow, have many problems. In this paper we suggest a generation procedure of flow graph by integrating conditional context variables into the control flow of a protocol specified in Estelle. The generation procedure is as follows. We use a parser to decompose specification to get necessary tables, and then add context variables to input and state according to their characteristics. Based on a reachable tree of state transition, according to the symbol table and the action table, an integrated flow graph can be generated. The test suite generated from an integrated flow graph can solve nondeterminism and infeasible path even if it considers only control flow.

*충남전문대학교 전자계산과
**청주전문대학 전산정보처리과
***충북대학교 컴퓨터과학과
論文番號 : 95307-0905
接受日字 : 1995年 9月 5日

Ⅰ. 서 론

원거리 통신 서비스는 정보화 사회의 기본 구조를 이룰 뿐만 아니라 생활수준을 좌우하는 중요한 요소이다. 최근 네트워크의 이용 확산으로 보다 신뢰성 있고 효율적이며 정확한 프로토콜의 설계 및 구현 기법이 요구된다.

적합성 시험이란 구현된 프로토콜이 요구 사항을 충족하는지를 명세서에서 생성한 테스트 스위트를 적용하므로써 판단하는 과정이다. 적합성 시험에 관하여는 테스트 스위트 적용 환경[1] 및 시험 방법, 테스트 스위트(test suite)의 생성 및 검증 등에 대하여 연구되고 있다. 테스트 스위트 생성 방법은 제어 흐름을 중심으로 테스트 스위트를 최소화하거나 비결정성(non-determinism)을 제거하는 연구가 주로 이루어져왔다[2, 3]. 그러나 제어 흐름만을 고려한 테스트 스위트는 많은 문제점이 있어 최근에는 자료 흐름을 통합한 연구가 진행되고 있다. 일반 소프트웨어의 자료 흐름에 관하여는 자료 흐름 분석 과정에서 실행 불가능 경로의 존재 및 필요성[4, 5], 시험에 사용할 수 있는 8개 기준의 상호 포함 관계 및 효율성 비교[6, 7] 등이 연구되고 있다.

자료 흐름을 고려한 프로토콜의 테스트 스위트 생성에 관하여는 Ural[8]이 일반 소프트웨어 분석 과정에서 사용된 자료 흐름 분석 기법을 Estelle로 명세화된 프로토콜에 적용하여 변수의 정의, 사용 과정을 'I-O-Define chain'으로 연결하고 자료 흐름상의 변칙성(anomaly)을 파악하였다. 그러나 자료 흐름만을 중시하고 제어 흐름을 구분하였으며 실행성(excutability)을 고려하지 않았다. Chun[9]의 논문에서는 명세서를 EFSM 형태로 모델링한 후 'CLP'를 사용하여 전이 블록의 내부 변수의 흐름을 분석하였다. Estelle 명세의 다중 모듈을 고려한 연구[10]는 명세서의 전이 흐름상 나타나는 비결정성을 제거하고 'well-defined form' 형태로 모델화하여 제어 흐름과 자료 흐름을 고려한 테스트 스위트를 생성하였으며 비결정성의 파악과 제거에 중점을 두었다. Sarikaya[11]는 Estelle로 표현된 프로토콜을 정규화(nomalization)시켜 자료 형태에 따라 3가지 종류의 자료 흐름도(DFG)를 그리고 사용 함수 중심의 블록(block)으로 통합하여 변칙성을 분석하였다. Chan의 논문[12]에서는 기존의 제어 흐름도에 자료 흐름에 따른 경로를 추가하고 중복 부분을 제거하여 테스트 스위트를 생성하였으나 실행성을 고려하지 않았다.

본 논문에서는 조건 문맥 변수(conditional context variable)를 통제 가능 여부에 따라 구분하여 입력 및 상태에 결합시키므로써 자료 흐름과 제어 흐름이 통합된 흐름도를 생성한다. 통합 흐름도에 의해 생성된 테스트 스위트는 제어 흐름만을 고려하는 경우에 발생하는 비결정성 및 실행 불가능 경로를 해결할 수 있으며 전체 테스트 스위트의 길이를 감소시킬 수 있다. 본론에서는 제어 흐름 중심의 테스트 스위트 생성 과정에서 발생하는 문제점을 기술한 후 Estelle로 명세화된 프로토콜에서의 흐름도 생성 과정 및 테스트 스위트의 적합성을 분석하였다.

Ⅱ. Protocol의 제어 흐름

1. Estelle로 명세화된 프로토콜

통신 프로토콜은 서로 다른 시스템간에 시스템 환경과의 입·출력 상호 작용을 지속하는 특수한 속성을 갖는다. 프로토콜 구조를 모델링 하기 위해서는 FSM, EFSM 형태의 모델링 도구가 주로 사용되며 ISO, CCITT에서는 프로토콜의 형식적 명세화 언어인 Estelle, LOTOS, SDL등과 테스트 스위트의 표현 기법인 TTCN을 개발하였다. Estelle는 EFSM 형태로 모델화된 프로토콜을 Pascal 형태의 문맥 구조로 표현하는 프로토콜 명세 언어로 모듈은 계층 구조를 이루며, 채널이라는 인터페이스 구조를 통하여 PDU형태의 메시지와 서비스를 주고받는다[11].

본 논문에서는 Estelle로 명세화된 프로토콜을 대상으로 실험하였다. 그러나 단일 모듈의 Estelle만을 고려하므로써 모듈간의 통신은 고려하지 않았으며, 전이 블록(transition block) 중심으로 단순화하여 사용하였다.

2. EFSM

Estelle는 EFSM 형태의 모델링 기법을 사용하며 자료 흐름을 고려한 Estelle는 다음의 정의 1)과 같은 구조로 모델링 될 수 있다.

정의 1) T=(S, I, O, t, So, V)

S : 상태들의 유한 집합

I : 입력의 유한 집합: I∈V

O : 출력의 유한 집합: O∈ V

t : 전이함수
So: So ∈ S 인 초기 상태
V : 변수의 유한 집합

이때 전이 함수 t는 정의 2)와 같은 튜플(tuple)의 집합으로 정의할 수 있다.

정의 2) t = 〈s, i, p, d, o, a〉
s : 현 상태 , s∈S
i : 입력 변수 i∈V
p : 전이 조건 p∈$f(v, c, op)$, v∈V,
c : 상수, op=논리연산자
d : 다음 상태, d∈S
o : 출력 변수, o∈V
a : 동작(Action)

Estelle로 명세화된 전이 블록은 〈When, From, Provided, To, ActBlock〉과 같이 5개의 튜플로 구성되며, 동작 블록(ActBlock)은 대입문(assignment)과 출력으로 구성되므로 튜플의 각 요소는 각각 정의 2)의 s, i, p, d, (o, a)와 대응된다.

조건 문맥 변수를 통제 가능 여부에 따라 입력과 상태(state)에 통합하여 정의 3)의 모델을 정의한다. 전이 조건 중 입력 변수와 관계된 조건은 입력과 통합하여 통제된 입력(predicated input)을 구성하였고 통제가 불가능한 내부 변수는 상태와 통합하여 현 상태와 다음 상태에 적용하였다. 따라서 정의 2)의 전이 함수는 정의 3)과 같이 FSM 형태로 재구성할 수 있다.

정의 3) t = 〈s′, pi, d′, o, a〉
s′: 현 상태, s′∈ s×uc,
uc: 통제 불능 문맥 변수
(uncontrollable Context Variable)
pi: 통제된 입력(Predicated Input),
pi∈ i×cc
cc: 통제 가능 문맥 변수
(Controllable Context Variable)
d′: 다음상태, d′∈ s×uc
o: 출력, o∈V
a: 동작

3. 제어 흐름 지향 테스트 스위트 생성

적합성 시험에 필요한 테스트 스위트에 관한 대부분의 연구는 FSM을 기본으로 명세서상의 제어 흐름만을 분석하거나 자료 흐름을 독립적으로 고려하였다[9]. 이러한 제어 흐름 위주의 테스트 스위트의 생성은 전체적인 자료 흐름을 분석하는 경우 발생되는 시간적, 공간적 제약성 때문으로 판단되나[12, 13] 다음과 같은 몇 가지 문제점을 갖는다.

1) 테스트 스위트에 대하여 입,출력 항목만을 비교하고 출력 내용(값)에 대한 적합성을 표시할 수 없으므로 완전한 시험 의견(Pass/Fail)의 표현이 불가능하다.

2) 생성된 테스트 스위트에 부분적으로 실행 불가능한 경우가 포함될 수 있으므로[4] 전체 테스트 스위트의 신뢰성이 저하된다.

3) 비결정성을 내포하는 프로토콜의 경우 제어 흐름 분석만으로는 근본적인 테스트가 불가능하다.

프로토콜의 규모가 방대하고 복잡해지므로써 모든 자료 흐름을 프로토콜 시험에 적용하는일은 현실적으로 매우 어려운 작업이다. 따라서 일반 소프트웨어 및 프로토콜에 대한 연구를 기본으로 문맥 변수를 제어 흐름에 통합하므로써 위에 제시한 문제점을 해결하는 테스트 스위트의 생성이 가능하다.

Ⅲ. 조건 문맥 변수를 고려한 테스트 스위트 생성 방식

1. 문맥 변수의 구분

프로토콜에서 상태의 전이는 자료의 입력과 전이의 조건에 의해 발생되며 전이 조건은 변수 및 상수가 논리연산자로 결합된 상태로 사용된다. 문맥 변수는 프로토콜의 제어 흐름을 결정하는 제어 변수를 총칭하며 조건 문맥 변수는 전이 조건의 기술에 사용되는 좁은 의미의 제어 변수이다. IUT를 Black Box로 볼 때, 문맥 변수가 입력 변수인 경우 테스터(tester)는 PDU의 입력 값을 조정하므로써 제어 흐름을 통제할 수 있다. 그러나 내부 변수가 문맥 변수로 사용된 경우에는 흐름의 외부 통제가 불가능하므로 문맥 변수는 통제 가능 여부로 구분할 수 있다.

본 논문에서는 전이 블록을 기준으로 통제 가능 여부를 구분하였다. 전이 조건에 사용된 변수 중 해당 전이 블록의 입력 변수는 전이 조건과 변수의 입력을 결합시켜 정의 3)의 통제된 입력(pi)으로 변환하였다. 통제된 입력은 조건이 첨부된 입력으로 입력 과정에서 입력 값에 대한 제한을 필요로 하므로 동일한 입력에 의하여 서

로 다른 상태나 출력을 수반하는 비결정성 문제를 해결할 수 있다.

각 전이 블록의 측면에서 통제가 불가능한 내부 변수의 경우 이를 상태와 통합하여 상태 수를 증가시켰다. 따라서 프로토콜은 상태, 내부 문맥 변수의 수, 변수 범위 값의 카르테시안 곱(cartesian product)의 상태 공간을 가진다. 이때 생성되는 상태 공간은 지나치게 크며, 적용 과정에서는 전체 공간 중 일부만이 사용되므로 도달 가능 나무를 이용하여 필요한 상태만을 추출하였다.

2. 테스트 스위트의 생성 과정

문맥 변수를 고려한 테스트 스위트 생성은 그림 1과 같이 여러 단계의 과정을 거쳐 순차적으로 처리된다. 먼저 Estelle로 기술된 프로토콜을 정의 2) 형태로 단순화시켜 정규화된 Estelle 텍스트를 생성한다. Estelle 텍스트는 파싱 과정에서 기능별로 분해되어 테이블을 형성하며, 분해 과정에서 전이 조건에 기술된 문맥 변수 중 통제 불능 변수를 상태에 통합하여 확장된 상태 공간을 생성한다. 확장된 상태 공간과 분해된 테이블 자료를 생성 알고리즘에 적용시켜 도달 가능 나무를 생성한다.

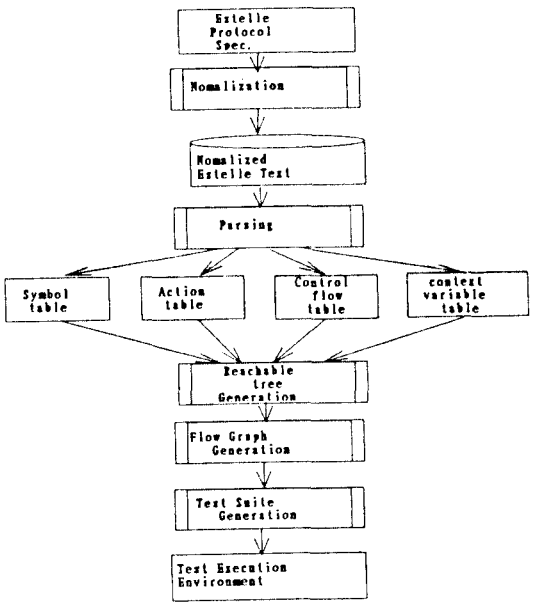

그림 1. 테스트 스위트의 생성 과정
Fig. 1. Generation procedure of test suite

이때 생성되는 도달 가능 나무는 초기 상태에서 출발하여 각 상태의 가능한 입력을 고려하여 생성하므로써 상태 공간을 축소하고 실행성이 고려된 흐름도를 만들 수 있다. 생성된 흐름도에서의 테스트 스위트 생성은 기존의 제어 흐름 기반의 연구 내용을 이용하였으며, 생성 과정은 복잡성을 고려하여 단계적으로 기술하였다.

명세서가 변환되어 테스트 스위트를 생성하고 테스트 실행 환경에 적용하는 주요 과정에서의 세부적인 처리 내용은 다음과 같다.

2.1 정규화(Normalization)

정규화는 Estelle로 명세화된 프로토콜을 전이 흐름 중심으로 단순화하는 과정으로 선언부와 with구문은 확장하고, 복합 조건절과 반복문은 동일한 기능의 단순한 형태로 변형시키므로써 전이 블록 단위로 단순화한다[10]. 정규화된 프로토콜 명세서는 정의 2)와 같이 〈When, From, Provided, To, ActBloc〉 형태의 구문 구조를 갖는다.

2.2 파싱(Parsing)

파싱 과정에서는 정규화된 Estelle 텍스트를 분해하여 필요한 테이블을 생성한다. 처리에 필요한 알고리즘은 다음과 같다.

1. 명세서에 기술된 변수를 분석하여 기호 테이블을 생성한다.

2. 각 전이 블록내의 동작 블록을 분해하여 동작 테이블을 생성한다.

동작 블록은 대입문, 출력, 기능 호출(function call)로 구성된다. 기능 호출은 확장할 경우 동일한 기능을 갖는 대입문으로 대체할 수 있으므로, 동작 블록은 대입문 블록과 출력으로 나누고 대입문 블록으로 블록별 동작 테이블을 구축한다.

3. 각 전이 블록을 정의 3) 형태로 단순화하여 제어 흐름 테이블을 생성한다.

4. 내부 변수가 전이 조건에 사용된 경우 문맥 변수 테이블을 생성한다.

파싱 과정에서 생성되는 각 테이블의 내용은 다음과 같다.

1) 기호 테이블(ST, Symbol Table)

기호 테이블은 전이에 의해 변화하는 변수들의 값을 참조하기 위해 생성되며 변수의 흐름을 정의(D,

Define), 계산적 사용(CU, Computational Use), 문맥적 사용(PU, Predicated Use), 정의 및 계산적 사용(DCU), 정의 및 문맥적 사용(DPU)으로 구분하여 활용 형태를 기술한다.

2) 동작 테이블(AT, Action Table)

정규화 과정을 거친 명세서는 전이 블록별로 독립된 형태로 출력되며 기능 호출은 대입문으로 변환되어 테이블 형태의 수식이 구축된다. 동작 테이블은 전이 블록별로 구축하며 Prefix 형태로 연산자를 먼저 표현하고 변수 또는 상수 형태의 연산 대상(operand)을 기술한다. 생성된 테스트 스위트는 순차적인 입력 PDU의 집합이므로 입력 및 전이 조건에 의해 정해진 전이 수행 과정에서 동작 테이블을 호출하므로써 기호 테이블을 갱신한다.

3) 제어 흐름 테이블(CFT, Control Flow Table)

처리 과정을 자동화하기 위하여 제어 흐름도를 테이블 형태로 구성한다. 제어 흐름 테이블은 정규화 후의 동작 블록을 동작과 출력으로 구분하여 입, 출력, 현 상태 및 다음 상태, 전이 조건과 동작을 기술하므로써 정의 3)의 형태로 구성한다. 이때 사용되는 입력 변수는 전이 조건과 결합시켜 입력 값에 대한 제한을 필요로 하는 통제된 입력('정의 3'의 pi)으로 사용한다.

4) 문맥 변수 테이블(CVT, Context Variable Table)

제어 흐름을 결정하는 조건 문맥 변수 중 입력 변수는 IUT 외부에서 제어 흐름을 통제할 수 있으므로 문맥 변수 테이블에서 제외한다. 따라서 문맥 변수 테이블에는 입력 변수를 제외한 내부 변수가 각 변수의 상태 조합으로 등록되며 제어 흐름 테이블과 연결된다. 등록된 전이 조건은 도달 가능 나무 구성 과정에서 상태와 통합되어 상태의 확장에 사용된다.

구성되는 각 테이블은 제어 흐름 테이블을 중심으로 연계되어 상호 참조가 이루어지며 참조 과정은 그림 2와 같다.

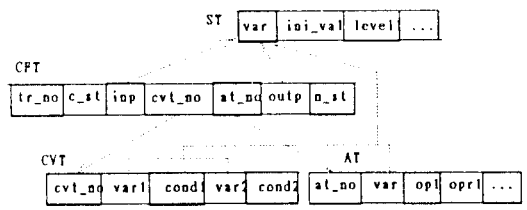

그림 2. 테이블간의 참조도
Fig. 2. Reference map between tables

2.3 도달 가능 나무 생성(Reachable Tree Generation)

도달 가능 나무는 파싱 과정에서 생성되는 흐름도에서의 실행성을 보장하고 확장된 상태 공간 중 미사용 상태를 제거함으로써 상태 수를 축소시킨다. 생성 방법은 파싱의 결과로 얻어진 테이블에 생성 규칙을 적용한다. 생성 과정에서는 정의 1)의 기본 상태에 문맥 변수 테이블의 내부 변수 및 입력 값을 카르테시안 곱으로 연산한 확장된 상태 공간이 사용되며 전이 조건을 참조한다. 도달 가능 나무의 생성 규칙은 다음과 같다.

1) 초기 상태에 문맥 변수의 초기 값을 결합하여 근노드로 설정하고 기호 테이블을 초기화한다. 근노드를 현 상태(정의 3의 s',)로 지정한다(level=0).

2) 제어 흐름 테이블을 순차 검색하여 현 상태를 시작 상태로 하는 전이 블록을 선택한다. 검색 대상이 되는 현 상태는 조건 문맥 변수가 결합되어있어 전이 조건이 고려되므로 선택된 전이 블록은 실행성이 보장된다.

3) 선택된 상태가 조상 노드에 존재하면 단계 7로 분기한다.

4) 현 상태에서 선택된 블록의 제어 흐름 테이블을 호출하여 다음 상태를 생성한다.

5) 동작 테이블을 호출하여 기호 테이블을 갱신한다.

6) 기호 테이블의 문맥 변수 값과 다음 상태를 결합하여 목적 상태를 생성하고, 목적 상태를 자노드로 추가한다.

7) 현 상태를 시작 상태로 하는 모든 블록에 대해 2)~6)의 과정을 반복하고 끝나면 단계 8로 전이한다.

8) level을 1 증가시킨다.

9) 전이 가능한 노드가 존재할 때까지 2)~8)의 과정을 반복한다.

2.4 흐름도 생성(Flow Graph Generation)

흐름도를 생성하기 위해서 먼저 도달 가능 나무에서 사용된 상태를 검색하여 상태 집합을 생성한다. 이때 파악된 상태 집합은 도달 가능 나무 생성 단계에서 고려한, 확장된 상태 공간의 부분 집합이 되며, 사례 자료로 실험 결과 확장된 상태 공간 중 일부만이 상태 집합으로 사용된다. 흐름도는 상태 집합을 노드로 설정하고 도달 가능 나무의 근노드에서 잎노드에 이르는 모든 경로를 arrow로 연결하며 입출력을 레이블로 기술한다. 도달 가능 나무는 전이 조건이 고려되었으므로 이를 기준으로 하여 생성되는 흐름도는 모든 경로가 실행성이 보장되는

실행 가능 흐름도가 된다.

2.5 테스트 스위트 생성(Test Suite Generation)

실행 가능 흐름도는 FSM 형태로 생성되므로 이미 연구된 제어 흐름 중심의 테스트 스위트 생성 방법을 모두 사용할 수 있다. 생성된 흐름도는 제어 흐름만을 고려한 경우보다 상태수가 확장되었고, 입력 변수가 문맥 변수의 개념을 적용하여 특성화되었으므로 상태별 UIO가 짧고 더 많은 종류를 갖는다. 따라서 이때 생성되는 테스트 스위트의 길이는 제어 흐름 중심의 방법보다 짧아지므로 테스트 비용을 줄일 수 있다.

자료 흐름을 고려한 테스트 스위트는 표현 과정에서 단순한 자료의 입.출력 시퀀스 외에 입.출력되는 자료의 값(value)에 대한 기술이 필요하다. 모든 변수에 대한 자료 흐름을 고려할 경우 각 PDU에 포함된 변수 값의 범위가 커서 전체 범위에 대한 테스트는 불가능하므로, 입력되는 변수의 특정 값을 선정하는 방법이 연구되어야 한다.

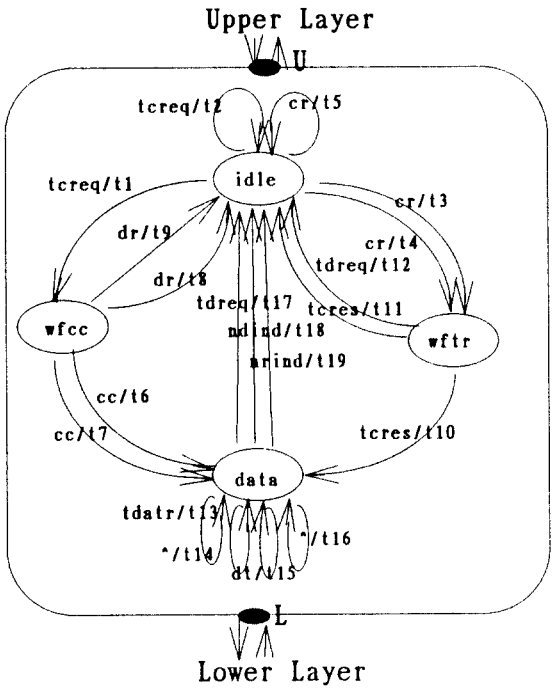

그림 3. TP0의 개략적인 흐름도
Fig. 3. General flow graph for TP0

Ⅳ. 사례 연구

1. Estelle로 명세화된 TP0

3장에서 기술한 생성 방식을 검증하기 위하여 사례 자료를 대상으로 흐름도와 테스트 스위트를 생성하였다. 사례 자료는 Estelle로 명세화된 TP0를 선정하였으며 정규화된 프로토콜[8,11]을 사용하였다. CCITT에 의해 개발된 TP0 프로토콜은 그림 3과 같이 내부적으로 4개의 상태와 U, L의 2개의 인터페이스를 가지며, 상위 및 동급 계층간에 자료를 주고받는 5개의 하위 경로(subtour)로 구성되어있다[11]. 정규화된 19개의 전이 블록에 대하여 제어 흐름만으로 테스트 스위트를 구성하는 경우, 발생되는 문제점은 다음과 같다.

1) 생성된 테스트 스위트의 입력 PDU내의 각 변수 값이 상태의 전이 과정에서 출력 PDU에 정확히 반영되었는지에 대한 결과를 알 수 없다.

2) t13, t15 블록은 'data' 상태에서 buffer에 자료를 전송하고, t14, t16 블록에서 buffer의 내용을 출력한다. 이때 t14, t16 블록은 'buffer〈〉not empty'의 전이 조건을 가지므로 각각 t13, t15가 선행되지 않을 경우 실행 불가능 경로를 형성한다.

3) 전이 조건을 고려하지 않을 경우 제어 흐름상에 다음과 같은 비결정성이 존재한다.

(1) 'idle' 상태에서 'U. tcreq'가 입력되는 경우: t1, t2
(2) 'idle' 상태에서 'L. cr'이 입력되는 경우: t3, t4, t5
(3) 'wfcc' 상태에서 'L. cc'가 입력되는 경우: t6, t7
(4) 'wfcc' 상태에서 'L. dr'이 입력되는 경우: t8, t9
(5) 'wftr' 상태에서 'U. cres'가 입력되는 경우: t10, t11

2. TP0에서의 테스트 스위트의 생성

테스트 스위트의 생성을 위하여 TP0에 3장에서 제시한 과정을 순차적으로 적용하였다. 사용된 프로토콜이 이미 정의2 형태의 구문을 갖는 정규화된 구조이므로, 먼저 필요한 테이블을 생성하는 파싱 과정이 적용된다.

파싱 과정에서는 기호 테이블을 생성하고 기능 호출을 대입문으로 대치하여 동작 테이블을 생성한다. 다음에

문맥 변수 중 내부 변수는 통제 불능 변수로 분류하여 문맥 변수 테이블을 생성하며, 통제 가능 변수는 통제된 입력의 형태로 결합시켜 정의 3)의 구조를 갖는 제어 흐름 테이블을 생성한다. 파싱의 결과 제어 흐름 테이블에서 분석된 상태 수는 4이며, 문맥 변수 테이블에는 문맥 변수 중 통제 불가능 변수인 'in_buffer', 'out_buffer'가 등록된다. 제어 흐름 테이블의 전이 조건은 문맥 변수 테이블과 'cvt_no'로 연결된다. 제어 흐름 테이블과 문맥 변수 테이블은 각각 표 1, 표 2와 같이 생성되었다.

도달 가능 나무는 전체 상태 공간과 초기 상태를 구하고 생성 규칙을 적용하므로써 생성된다. 제어 흐름 테이블의 4개의 상태에 2개의 문맥 변수는 각각 2개의 상태('empty(^)', 'not_empty(f)')를 가지므로 가능한 총 상태 수는 '4*2*2'로 16개이며, 입, 출력 버퍼의 용량은 1로 간주하였다. 초기 상태 'idle'을 문맥 변수의 초기 값인 'empty(^)'와 통합하면 근노드는 'idle(^, ^)'이며, 생성 규칙에 의하여 확장한 결과 도달 가능 나무는 그림 4와 같이 생성된다.

실행 가능 흐름도를 작성하기 위해서 그림 4의 도달 가능 나무를 분석하면 전체 16개의 상태 공간 중 7개만이 사용되었다. 7개의 상태를 그리고, 도달 가능 나무의 근노드에서 잎노드에 이르는 경로를 arrow로 연결하여 그림 5와 같이 실행 가능 흐름도를 작성한다. 그림 5는 FSM 형태로 기술되었으며 제어 흐름만을 고려한 흐름도와 비교할 때 다음과 같은 특성을 갖는다.

표 1. 제어 흐름 테이블
Table 1. Control Flow Table

Tr_No	현재상태 (c_st)	입력(inp)	전이조건번호 (cvt_no)	동작명 (at_no)	출력 (outp)	다음상태 (n_st)
t1	U.idle	tcreq.p1	c1	a1	L.cr	wfcc
t2	U.idle	tcreq.p2	c1	a2	U.tdind	idle
t3	L.idle	cr.p3	c1	a3	U.tcind	wftr
t4	L.idle	cr.p4	c1	a4	U.tcind	wftr
t5	L.idle	cr.p5	c1	a5	L.dr	idle
t6	L.wfcc	cc.p6	c1	a6	U.tccon	data
t7	L.wfcc	cc.p7	c1	a7	U.tccon	data
t8	L.wfcc	dr.p8	c1	a8	L.ndreq, U.tdind	data
t9	L.wfcc	dr.p9	c1	a9	L.ndreq, U.tdind	idle
t10	U.wftr	tcres.p10	c1	a10	L.cc	data
t11	U.wftr	tcres.p11	c1	a11	L.dr, U.tdind	idle
t12	wftr	U.tdreq	c1	a12	L.dr	tdreq
t13	data	U.tdatr	c1	a13	^	data
t14	data	^	*c2*	a14	L.dt	data
t15	data	L.dt	c1	a15	^	data
t16	data	^	*c3*	a16	U.tdati	data
t17	data	U.tdreq	c1	a17	L.ndreq	idle
t18	data	L.ndind	c1	a18	U.tdind	idle
t19	data	L.nrind	c1	a19	U.tdind	idle

표 2. 문맥 변수 테이블
Table 2. Context Variable Table

cvt_no	변수명 (var1)	전이조건1 (cond1)	변수명 (var2)	전이조건(cond2)
c1	in_buffer	empty(^)	out_buffer	empty(^)
c2	in_buffer	empty(^)	out_buffer	not_empty(^)
c3	in_buffer	not_empty(^)	out_buffer	empty(^)
c4	in_buffer	not_empty(^)	out_buffer	not_empty(f)

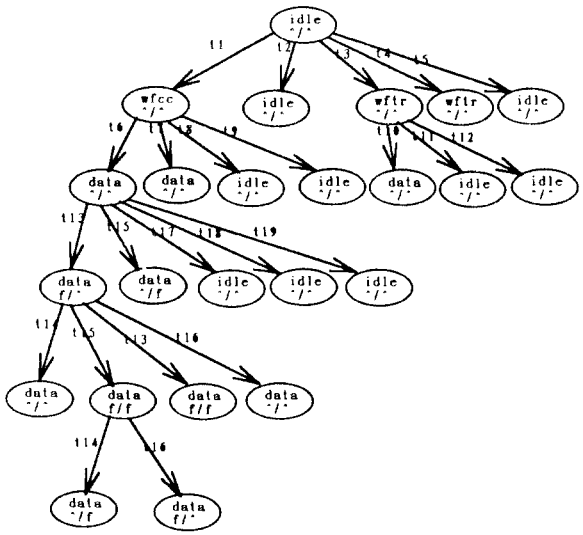

그림 4. TP0의 도달 가능 나무
Fig. 4. Reachable tree of TP0

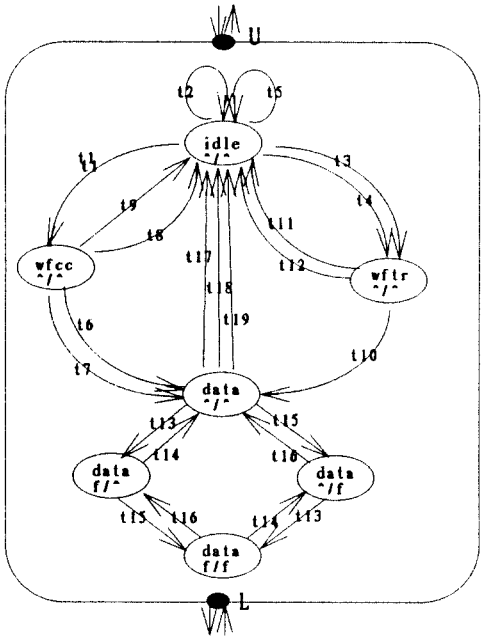

그림 5. TP0의 실행 가능 흐름도
Fig. 5. Feasible flow graph of TP0

1) 기존의 4개 상태가 문맥 변수가 반영된 형태로 통합되어 전체 가능 상태 16개중 7개만이 사용되었다.

2) 전이의 입력에 문맥 변수가 고려되어 통제된 입력이 사용되었으며 상태 증가에 따라 전이의 수가 19개에서 23개로 증가하였다.

실행 가능 흐름도에서의 테스트 스위트의 생성은 기존의 FSM 지향 흐름도에서의 연구 내용[2]을 활용할 수 있다. 표3에서는 테스트 스위트의 생성 사례로 't10' 전이에 대한 테스트 케이스를 제어 흐름만을 고려한 기법과 비교하였다. 't10' 전이의 경우, 초기 상태(reset)에서 'wftr' 상태로 전이한 후(cr), 't10' 전이 블록이 테스트되고(tcres), 도달 상태(tail state)를 확인하기 위하여 'data(^/^)' 상태의 UIO가 체크된다 (reset -> cr -> tcres -> UIO). 비교 결과 제안된 방법에서는 제어 흐름만을 고려하는 경우 존재하던 비결정성이 제거되고 실행성이 보장됨을 알 수 있다.

3. 결과 분석

제어 흐름을 고려한 프로토콜의 경우 4.1절에서 기술

표 3. t10 transition의 테스트 케이스
Table 3. Test case for t10 transition

비교항목	제어 흐름 지향 기법		문맥 변수 고려 기법(본 논문)	
	case	문제점	case	특성
reset	reset			reset
transition sequence	cr/tcind	t1~t3간의 비결정성 존재	cr.p3/tcind 비결정성 해결	입력 조건(p3) 추가로
tested arrow	tcres/cc	t10~t11간의 비결정성 존재	tcres.p10/cc	입력 조건(p10) 추가로 비결정성 해결
UIO	t13, t14, t14, t16	t15, t16의 경우 실행 불능	t13, t15	t14, t16 제거로 실행성 향상

한 것과 같이 여러 가지 문제점을 갖는다. 본 논문에서는 통제 가능 문맥 변수를 입력에 통합하여 입력 변수에 제어 조건을 추가함으로써, 테스터의 입장에서 입력에 대한 제어가 복잡해지고, 값을 제어할 수 있는 테스터가 필요하다는 제한 사항을 갖는다. 또한 상태 수가 증가함에 따라 흐름도의 복잡성이 증대되었다.

그러나 생성된 테스트 스위트는 기존의 제어 흐름 지향의 생성 기법에 비하여 다음과 같은 장점을 갖는다.

1) 실행성을 고려하여 흐름도가 생성되었으므로 4.1절에서 검토된 실행 불가능 경로가 제거되었다.

2) 제어 흐름만을 고려한 경우에 발생되는 4.1절의 5가지의 비결정성 경로는 입력 자료에 문맥 변수를 통합하므로써 근본적으로 해결되었다.

3) 입력에 제어 조건을 추가하여 입력을 차별화하므로써 UIO의 길이가 짧아진다. 따라서 동일 경로에 대한 테스트 스위트의 길이가 제어 흐름만을 고려한 경우보다 짧아진다.

4) 각 상태에 여러 종류의 UIO가 존재하므로써 동일한 길이를 갖는 여러 종류의 테스트 스위트를 생성할 수 있다. 따라서 사용율이나 오류 가능성이 높은 특정 경로에 대한 부분적인 테스트가 필요한 경우[13]에 효율적으로 사용할 수 있다.

5) 실행 가능 흐름도는 기존의 제어 흐름 그래프와 동일한 형식과 표현 기법을 갖는다. 따라서 그 동안의 제어 흐름 지향의 테스트 기법을 모두 활용할 수 있다.

Ⅴ. 결 론

적합성 시험에 사용되는 테스트 스위트의 타당성은 시험 환경 및 적용 방법과 함께 중요한 연구과제 중의 하나이다. 기존의 테스트 스위트 생성에 관한 대부분의 연구는 FSM을 모델로 하여 명세서상의 자료 흐름은 무시하고 제어 흐름만을 분석하였다. 그러나 자료 흐름을 고려하지 않을 경우 완전한 시험 의견의 표현이 불가능하고 생성된 테스트 스위트에 부분적으로 실행 불가능한 경우가 포함되는 등 문제점을 갖게 된다.

본 논문에서는 이 같은 문제점을 해결하기 위하여 전이 조건의 기술에 사용된 문맥 변수를 제어 흐름에 통합하므로써 부분적으로 자료 흐름이 고려된 실행 가능 흐름도와 테스트 스위트를 생성하였다. 생성 과정에서 조건 문맥 변수를 통제 가능 여부에 따라 구분하였고 전이 조건을 제어 흐름에 고려하므로써 비결정성을 근본적으로 해결하였으며 도달 가능 나무를 활용하여 실행 불가능 경로를 제거하였다.

제안된 과정을 검증하기 위하여 처리과정을 Estelle로 명세화된 TP0 프로토콜에 적용하였다. 분석 결과 통합 흐름도에 의해 생성된 테스트 스위트는 제어 흐름만을 고려하는 경우에 발생하는 비결정성 및 실행 불가능 경로 문제를 제거하였으며, 짧은 UIO를 사용하므로써 전체 테스트 스위트의 길이를 감소시킬 수 있었다. 또한 최종적으로 생성되는 흐름도가 FSM의 표현 형식을 가지므로 그 동안의 테스트 스위트 생성에 관한 대다수 연구 내용을 수정 없이 적용할 수 있다.

다만 통제된 입력을 사용하므로써 이를 제어할 기능을 갖는 보다 정교한 테스터 및 표현 형식이 필요하며 내부 변수 전체의 자료 흐름 및 모듈간의 통신을 고려하는 후속 연구가 필요하다.

참고문헌

1. 이상호, S. T. Vuong, "프로토콜 적합성 시험을 위한 통합 환경," 정보과학회논문지, 제21권 제5호, pp.944-960, 1994.
2. D.P.Sidhu, T.K.Leung, "Formal Method for Protocol Testing: A Detailed Study", Tran. on SE, Vol. 15, NO. 4, pp.413-426, 1989.
3. 우성희, 오병호, 이상호, "Estelle로 표현된 프로토콜의 테스트 스위트 생성에 관한 연구," 정보과학회 춘계학술발표논문집, 1994.
4. M. Chellappa, "Nontraversible Paths in a Program," IEEE Tran. on SE, Vol. SE-13, NO. 6, pp.751-756, 1987.
5. P. G. Frankl, E. J. Weyuker, "An Applicable Family of Data Flow Testing Criteria," IEEE Tran. on SE, Vol. 14, NO. 10, pp.1483-1498, 1980.
6. P. G. Frankl, S. N. Weiss, "An Experimental Comparison of the Effectiveness of Branch Testing and Data Flow Testing, " IEEE Tran. on SE, Vol. 19, NO. 8, pp.774-787, 1993.
7. S. Rapps, E. J. Weyuker, "Selecting Software Test Data Using Data Flow Information," IEEE Tran. on SE, vol. SE-11, NO. 4, pp.367-375, 1985.
8. H. Ural, B. Yang, "A Test Sequence Selection Method for Protocol Testing," IEEE Tran. on Comm. , Vol. 39, NO. 4, pp.514-523, 1991.
9. W. Chun, P. D. Amer, "Test Case Generation for Protocols Specified in Estelle," FORTE '90, pp.197-210, 1990.
10. 이도영, "상호 통신하는 확장된 FSM모델에 근거한 통신프로토콜의 형식적 적합성 시험에 관한 연구," 포항공과대학 대학원 박사학위논문, 1991.
11. B. Sarikaya, G. V. Bochmann, E. Cerny, "A Test Design Methodology for Protocol Testing," IEEE Tran. on SE, Vol. SE-13, NO. 5, pp.518-531, 1987.
12. W. Y. L. Chan, S. T. Vuong, M. R. Ito, "On Test Sequence Generation for protocol," Symposium on Protocol Specification, Testing and Verification, pp.119-130, 1990.
13. D. P. Sidhu, A. Chung, C. S. Chang, "Probabilistic Testing of OSI Protocols," IEEE Tran. on Comm, Vol 42, NO. 7, pp.2432-2440, 1994.

吳 昞 浩(Byeong Ho Oh) 정회원

한국통신학회 논문지 제20권 제6호
현재 : 충남전문대학 전자계산과 교수

禹 星 壽(Sung Hee Woo) 정회원

한국통신학회 논문지 제20권 제6호
1995년 9월~현재 : 청주전문대학 전산정보처리과 교수

李 相 鎬(Sang Ho Lee) 정회원

한국통신학회 논문지 제20권 제6호
현재 : 충북대학교 컴퓨터과학과 교수