프로토콜 검증 모델 및 검증기 개발

민 재홍*

*한국전자통신연구원

A study on the development of protocol modal and verificator

Jae-hong Min*

*ETRI(Electronics and Telecommunications Research Institute),

E-mail : jhmin@pec.etri.re.kr

요 약

인터넷의 급속한 보급과 더불어,인터넷의 응용분야가 다양화됨에 따라 정보통신관련 S/W들의 규모와 성능면에서 복잡하고 고도화된 시스템 및 소프트웨어를 요구하고 있다. 따라서 사용자의 요구사항을 충족하는 신뢰성 있는 통신 프로토콜 소프트웨어를 개발하는 방법에 대한 중요성이 커지고 있다. 본 고에서는 이용자의 요구사항에 대한 설계단계의 검증을 위하여, SDL을 위한 정형기법 지원도구 중 명세 상에서 행위 부분에 대한 동적 특성을 검정하는 검증기 개발과 SDL Editer 기능과 관련된 도구들과 통합하여 사용자들이 쉽고 편하게 쓸 수 있도록 통합환경 구현을 기술한다.

ABSTRACT

Advanced systems and softwares are required for information communications protocol in information sharing,electronic commerce and intranet applications with the increase of internet usage. Therefore ,the development methodology of telecommunications protocol to meet user's requirement and increase reliability becomes more important. This study aims to develop the verification model and tool for ATM protocol using SDL formal description technique and will becomes basic technique for the implementation of integrated protocol development tool.

I. 서 론

최근 정보통신관련 S/W들은 인터넷의 보급에 힘입어 정보의 공유, 전자상거래, 인트라네트 형성 등 그 규모나 성능 면에서 기능의 복잡성 및 다양화에 따른 고도화된 시스템 및 소프트웨어를 요구하고 있다. 그러나 이용 환경 및 요구 변화가 어떤 형태로 달라지던지 관계없이 사용자와 개발자, 그 사이에 존재하는 전통적인 소프트웨어 개발과정에서 생겨나는 문제들, 예를 들어 사용자의 요구조건을 충분히 이해하고 표현하지 못함으로써 신뢰성 있는 통신 프로토콜 소프트웨어 개발에 대한 구체적인 방안을 제시하지 못하는 문제가 있다.

이러한 문제들은 형식기술방법(Formal Description technique)에 의한 규격명세 기술에 의하여 비정형적인 사용자 요구사항으로부터 정형적인 사용자 요구사항을 유도하여 이를 통한 명세의 무결성 및 완전성을 검정하여 신뢰성 있는 통신프로토콜 소프트웨어를 개발하는 방법으로, 나아가서는 이를 통해 목적코드의 상당 부분을 자동 생성시킴으로써 코딩노력을 절감시킬 뿐 아니라 신뢰성 있는 소프트웨어을 생산할 수 있는 기반을 제공한다.

정보통신망을 위한 프로토콜들이 상호 적절한 기능을 하기 위해서는 요구사항에 대한 정형기법으로의 명세화 작업 및 명세 내에서 상태간의 deadlock, 비정상적인 도달 등과 같은 잠재적 설계에러가 없어야 하며, 사용자 요구사항과 일치하는지 검증하고, 다른 프로세서와 원활한 통

신이 이루어지는지 검사해야 한다.

통신 프로토콜의 검정의 기능은 요구사항과 시스템 명세와의 일치를 검사하고, 명세의 일반적 오류 유무를 조사하는 일련의 행위로써 정보통신 프로토콜 시스템 개발비용을 줄이고 신뢰성을 확보하는 기술로 정확성(Correctness), 안정성(Safety)과 일관성(Consistency) 및 필연성(Liveness) 검증을 주 대상으로 한다.

당 센터에서는 ATM프로토콜 구현에 있어서 핵심적인 기술인 ATM프로토콜 시스템 요구사항에 대한 모델링을 통해 정형기법으로 명세화하고, 명세화된 요구사항 정형모델의 검증을 위한 검증기 개발과 검정을 거친 구현에 대해 기 개발된 적합성 시험생성기를 통합하여 프로토콜 개발도구 통합환경을 구현하였다.

본 고에서는 우선 2장에서는 검정 사항 및 대상에 대해 기술하고, 3장에서는 autumata 기반의 검정알고리즘으로써 안전상과 필연성 검정을 위한 modal mu-calculus 적용 알고리즘을 설명하고, 4장에서는 UNIX/Motif 환경에서 automata 모델의 안정성과 필연성 검정을 수행하는 검증기를 C++언어를 사용하여 구현한 내역을 소개한다. 마지막으로 5장에서는 프로토콜 검정기의 향후 기대되는 활용방안을 제시한다.

II. 검정 사항 및 대상

프로토콜은 deadlock이나 잘못된 행위가 존재하지 않는 안전성과 정의된 행위를 반드시 해야 하는 필연성 등과 같은 특성들을 지니고 있고, 프로토콜 검정은 이러한 특성을 지닌 프로토콜 명세에 대한 deadlock, livelock, state reachability, 3 action reachability, determinist 검사로써 검정을 수행한다.

o deadlock: 한 상태에서 다음 어떤 상태로의 천이가 존재하지 않기 때문에 다음 행위를 할 수 없는 경우, 즉 그 상태에서 나가는 천이가 존재하지 않는 경우로써 그래프 재표기 시스템을 사용하여 automata의 모든 상태가 강한 연결인지를 검사한다.

o livelock: 프로토콜(명세) 상태들의 부분 집합 내에서 그 상태들만을 무한히 반복적으로 천이하는 경우로써 그 부분 집합 이외의 다른 상태로의 천이가 존재하지 않는다. 따라서 비생산적인 주기, 즉 같은 경로를 계속 반복하는 경로를 가지고 같은 메시지 교환이 유한의 개수 만큼 또는 무한으로 행하여지는 상태에 들어가는 것을 의미하며 deadlock과 동일하게 그래프 재표기 시스템의 강한 연결 검출 기법을 사용한다.

o reachability: 프로토콜(명세)이 작동되기 시작할 때, 프로토콜(명세)의 시작에서 정의되어지는 특별한 상태로써 초기 상태가 존재하는데, 이 초기상태로부터 프로토콜(명세)은 정의된 천이순서에 의해 일부 또는 모든 다른 상태에 도달하게 된다. 프로토콜(명세)이 정의된 천이 순서에 의해 정의된 상태로 도달한다면 그 천이와 상태에 대해 이 명세는 올바른 프로토콜(명세)이다. 그리고 reachability는 상태에 대한 것과 행위에 대한 것으로 구분된다.

- State reachability

두개의 행위와 한개의 상태를 입력하여, 첫번째 행위에서 시작해서 두 번째 행위가 특정 상태에서 발생하는지 검사한다.

Modal mu-calculus 논리식은

$$Z.(Y.A(\langle action1\rangle tt[action2]Y)) \wedge [-]Z, A = \{state\}$$

형태이며 automata에 존재하는 모든 첫번째 행위에 대해 어떤 두번째 행위가 특정 상태에서 발생하는지 검사한다. 만약 두번째 행위가 특정 상태에서 발생한다면 논리식은 참이다. 그러나 두번째 행위가 발생함에도 불구하고 특정상태에서 발생하지 않는다면 논리식은 거짓이 된다.

- 3 action reachability

세 개의 행위를 입력하여, 이 행위가 순서대로 발생하는지 검사한다.

Mcdal mu-calculus 논리식은

Z.[action1](Y.<action3>tt[action2]Y) ∧[-]Z

이며 모든 첫번째 행위와 두번째 행위에 대해 어떤 세 번째 행위가 발생하는지 검사하고, 이들이 순서대로 일어나는지 검사한다. 만일 행위가 순서대로 일어나지 않는다면, 논리식은 거짓이 된다.

o determinist: 어떤 상태의 특정 행위에 의해 다음 상태가 두 가지 이상 존재하는 경우로써 각 상태에서 동일한 행위에 대해 서로 다른 다음 상태가 존재하는지를 검사한다.

III. Automata 검정 알고리즘

1. Automata

본 절에서는 프로토콜을 검증하는 정형 명세 기법 중 프로토콜 정형 명세를 위한 의미모델로써 많이 사용되는 automata를 정의하면 아래와 같다.

$A=<S, So, \Sigma, Tr, F>$

S : 상태의 집합

So : 시작 상태의 집합, $So \subseteq S$

Σ : 심볼의 집합

Tr : 천이(Transition)의 집합,

$r \subseteq S \times \Sigma \times S, tr \subseteq Tr, tr=<s,a,s'>$

F : 최종 상태의 집합, $F \subseteq S$

오토마타를 A라 하면 S는 시스템이 가질 수 있는 상태들의 집합을 나타내며, So는 오토마타가 시작할 수 있는 상태들의 집합이다. 그러나 통신프로토콜에 있어서는 오토마타는 결정적이어야 하므로 시작 상태는 단지 유일하게 하나만 존재하여야 한다. 심볼은 한 상태에서 다른 상태로 천이를 유발시킬 수 있는 사건들의 집합을 말하는데, 통신 프로토콜에 있어서는 특정 사건이라기보다는 입력에 따른 출력의 쌍으로써 표현하기 적합하기 때문에 입력과 출력이 하나의 사건을 구성하는 I/O FSM을 많이 사용한다.

천이는 특정 천이 함수를 사용함으로써 상태를 바꾸는 요인이 되는데 위의 정의식에도 보듯이 천이 함수는 상태와 심볼 그리고 다음 상태의 쌍으로 구성된다. 마지막으로 최종 상태는 오토마타가 인식(accepted) 가능한 상태의 집합을 의미한다. 즉 심볼들의 순서열인 언어(Language)가 오토마타에 의해 인식되어지는 상태를 나타낸다. 통신프로토콜에 있어서는 초기 상태와 최종 상태가 동일하다.

다음 절에서는 automata 모델을 기반으로 그래프 재표기 시스템을 사용한 안전성 특성 검정 알고리즘을 소개한다.

2. 그래프 재표기 시스템(Graph Rewriting System)

그래프 재표기 시스템은 automata(혹은 방향화 된 그래프)의 어떤 상태(또는 그래프 노드)에서도 어떠한 다른 상태(또는 그래프의 노드)로도 경로(path)가 존재할 때 강한 연결이 되는 특성을 이용한 것으로 이는 주어진 프로토콜 명세(automata 혹은 그래프)가 deadlock과 livelock이 없음을 나타내므로 이의 적용을 통해서 안전성을 검정한다.

우선 automata $A=<S,So,\Sigma,tr>$을 방향화 된 그래프(oriented graph) $G=(V,E)$로 표기할 수 있는데, 여기서

V = S의 상태에 해당하는 노드의 집합

E = 천이 함수 tr에 해당하는 방향화 된 아크의 집합

으로 두고, 이 기본 방향화 된 그래프의 노드에 레이블 첨가하여 그래프 재표기 시스템을 정의한다.

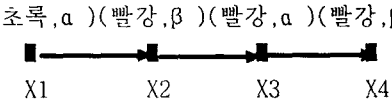
먼저 초기 상태 So에 해당하는 노드에 색깔 레이블 짝(빨강, 파랑)을 부여하고, 나머지 노드에는 색깔 레이블 짝(초록, 초록)을 부여하면, 색깔로 레이블화 된 그래프가 생성된다. 노드 색깔 집합은 C(V)={초록, 빨강}× {초록, 파랑}이다. 이와 같이 얻어진 색깔로 레이블화 된 그래프에 다음의 그래프 재표기 법칙 R1과

R2를 적용한다 :

o 재표기 법칙 R1

α ,β {초록, 파랑, 빨강}이라 하고, 노드 X1, 즉 색깔 (초록,α)인 것이 아크 (X1,X2)에 의해 노드 X2, 즉 색깔 (빨강, β)에 연결되어 있으면 X1은 새로운 색깔 레이블 (빨강,α)을 부여한다.

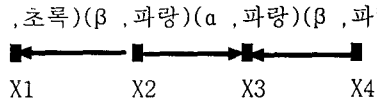
이 법칙은 다음의 그래프 재표기 법칙으로 도식화된다.



o 재표기 법칙 R2

만약 노드 X1, 즉 색깔 (α ,초록)이 아크 (X2,X1)에 의해 노드 X2, 즉 색깔 (β ,파랑)에 연결되어 있으면, X1은 새로운 색깔 레이블 (α ,파랑)을 부여한다.

이 법칙은 다음의 그래프 재표기 법칙으로 도식화된다.



위에서 제시된 그래프 재표기 법칙 R1과 R2를 적용하면서 automata의 그래프 재표기 시스템을 이용한 안전성 검정 과정은 다음의 5단계로 이루어진다.

[안전성 검정 단계]

① Automata의 초기 상태에는 (빨강, 파랑) 색깔 짝을 부여하고, 나머지 상태들은 (초록, 초록) 색깔 짝을 부여하여 레이블화 된 그래프를 생성한 후 ②를 수행한다.

②레이블화 된 automata 상에서 (β ,파랑)→(α ,초록) 형태로 된 천이가 존재하면 ③을 수행하고,(초록,α)→(빨강,β) 형태로 된 천이가 존재하면 ④를 수행한다. 만약 위에 언급된 형태의 천이가 존재하지 않으면 ⑤를 수행한다.

③(β ,파랑)→(α ,초록) 형태의 레이블화 된 천이의 해당 상태들에 (β ,파랑)→(α ,파랑) 형태로 레이블을 변형하고 ②를 수행한다.

④(초록,α)→(빨강,β)형태의 레이블화 된 천이의 해당 상태들에 (빨강,α)→(빨강,β)형태로 레이블을 변형하고 ②를 수행한다.

⑤레이블화 된 결과 automata의 모든 상태들이 (빨강, 파랑) 레이블 형태인 색깔 짝으로 구성되어 있으면 automata는 강한 연결 형태가 되어 안전성 특성을 만족하고 그렇지 않으면 강한 연결 형태가 아니므로 안전성 특성을 만족하지 못한다.

3. GRS를 이용한 안전성 검정 적용의 예

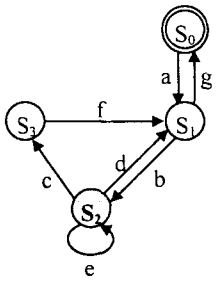
그래프 재표기 시스템을 automata 모델에 적용하여 검정한다.④에서 최종적으로 모든 상태가 (R/B)가 됨으로 이 automata 모델은 강한 연결 상태이므로 deadlock이나 livelock이 없다.

4. GRS를 이용한 안전성 검정의 장점

그래프 재표기 시스템에 의한 강하게 연결된 component 찾는 알고리즘은 고전적인 그래프 이론에 의거한 강하게 연결된 component 찾는 알고리즘에 비해 더욱 효과적이다. 왜냐하면, 고전적인 strongly connected component find 계산 알고리즘은 O(Max(n, e))(여기서 n은 노드의 수이고, e는 아크의 수)의 time complexity를 요구하는 반면, 위에 언급된 알고리즘은 O(2(n-1))를 요구한다. 일반적으로 통신 프로토콜은 automata(혹은 그래프)로 많이 명세화되어 사용되고 구현되어진다. 프로토콜을 명세한 automata(혹은 그래프)는 프로토콜의 행위 특성에 따라 e의 수가 n의 수보다 훨씬(일반적으로 수배 이상) 크기 때문에 그래프 재표기 시스템을 사용한 알고리즘을 적용하면 O(2(n-1))의 time complexity로 간단하게 구현할 수 있어 더욱 효과적이다.

5. Automata 행위 도달성 검정

Automata 모델에서 요구되는 행위가 궁극적으로 구현되어 있는지 확인하는 과정으로, 3개의 행위를 적용하였을 경우 참조된 automata 모델이 해당 행위순서를 보여주는지를 검사하는 필연성 검정과정에 해당한다. LTS 모델에 기반을 둔 행위 도달성 검정 알고리즘은 LTS와 automata 모델의 행위 특성 표현 차이점인 자기루프와 반복 행위순서 문제로 인해 역추적 기법으로 행위들이 만족하는지를 완전하게 검사할 수는 없다. LTS 모델에 기반을 둔 행위 도달성 검정 알고리즘의 문제점을 살펴보자. (그림 1)는 automata의 행위 특성인 자기루프와 반복행위순서를 포함하는 automata 모델이다.



(그림 1) 자기루프와 반복행위순서가 있는 automata

(그림 2)는 (그림 1)를 LTS 행위 역추적 기법을 통하여 b→ e→ d 행위순서를 LTS 기반의 검정알고리즘에 적용한 결과이다.

X	X1	X2	X3	X4	X5	X6	X7
So	0	0	0	1	1	1	1
S1	0	0	0	1	1	0	1
S2	1	1	1	1	1	0	1
S3	0	0	0	1	1	1	1

M[1]=< > M[2]=< >

(그림 2) bit-vector, 배열 M[i]

위 결과에서 보듯이 역추적 기법으로 행위순서가 automata를 만족하는지를 검사하면 X3가 참이 되는(행위 d를 발생하는) 상태 S2를 검출하고는 행위 e가 발생하는 상태 S2를 검출하지 못한다. 즉, S2에서 livelock이 발생하므로 LTS 행위 도달성의 전제 조건을 만족하지 못하므로 LTS 검정 알고리즘에서는 발생 불가능한 행위순서로 판정한다. 하지만 automata 관점으로 볼 때는 이 행위가 언젠가는 발생해야 하므로 결과는 참이 되어야 한다. 따라서 LTS 행위 역추적 기법이 아닌 automata모델의 행위 특성(자기루프,반복행위순서)을 고려한 새로운 검정 알고리즘이 제안되어야 한다.

automata의 행위 특성을 고려한 행위 도달성 알고리즘은 다음과 같다..

①Automata를 model checking 알고리즘을 적용하여 bit 배열을 생성한다.

②생성된 bit 배열에서 X6가 거짓(0)이고, X2가 참인 상태를 검사한다. 만약 만족하는 상태가 존재하면 ③을 수행하고 존재하지 않으면 종료한다.
(action 1이 도달되고 action 2가 발생하는 상태를 검출)

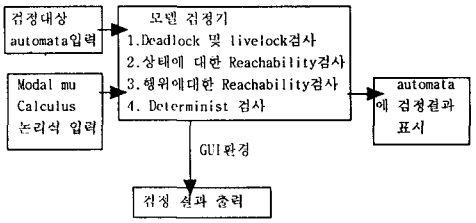
③생성된 bit 배열에서 X2가 참(1)이고, X3가 참(1) 인 상태를 검사한다. 만약 만족하는 상태가 존재하면 발생 가능한 행위순서이고 존재하지 않으면 발생 불가능한 행위순서이다.
(action 2가 도달되고 action 3이 발생하는 상태를 검출)

위의 automata 행위 도달성 검정알고리즘은 행위순서를 순차적으로 검색하게 된다. (그림 2)의 bit vector를 예로 들면 행위 b를 받고 행위 e를 발생시키는 S2를 검출한 후 행위 e를 받고 행위 d를 발생시키는 S2를 검출함으로써 행위순서는 automata를 만족한다고 판정한다.

IV. 검정기 구현

당 센터에서 개발한 검정기는 SDL 명세의 행위 명세 부분을 중간 모델로 변환한 automata에서 교착상태 유무 조사와 주어진 천이 시스템이 임의의 상태에 대하여 초기 상태에서 도달 가능한지 검사하는

기능을 C++을 이용하여 UNIX 호환기종에서 동작하도록 구현하였다. 이러한 기능을 구현하기 위해 사용자에 의해 입력된 논리식을 어휘 분석과 구문 분석과정을 거친 후 B.cleaveland와 B.steffen에 의해 개발된 model checking 알고리즘인 Solve를 적용하여 교착상태 유무, 임의의 상태에 대하여 초기 상태에서 도달 가능한지를 판단하였다. 또한 사용자환경에서 쉽게 다룰 수 있도록 UNIX/Motif 환경 하에서 GUI기능에 의해 검증결과를 Window에 나타내었다. 구현된 검증기에 대한 전체 구성도는 (그림3)와 같다.



(그림3) 검정기 전체 구성도

1. 검정기 구성

본 프로토콜 검정기 시스템은 요구사항 분석단계에서 정의한 다음과 같은 기능을 제공하도록 설계한다.

o MAIN(종합 메뉴)
- 이용자가 원하는 작업을 선택할 수 있는 안내기능을 User-friendly한 형태로 제공한다
- 선택할 수 있는 작업은 크게 프로토콜 검정과 적합성 시험생성으로 나눌 수 있고, 프로토콜 검정은 Automata Generation, Safety검사, Determinist 검사, State Reachability, Action Reachability로 세분된다.
- 기타 기능으로 이용자 Password관리 검정기 시스템 사용 절차 및 방법에 대한 안내기능을 제공한다.

o Automata Generation
SDL 형식 기술기법으로 표현된 프로토콜로부터 중간모델, 즉 I/O FSM (Input/Output Finite State Machine)으로 표현된 명세를 Object-Geode의 Simulation 기능을 사용하여 생성된 Automata file을 검정기에서 사용할 Data Structure형태로 변환한다.

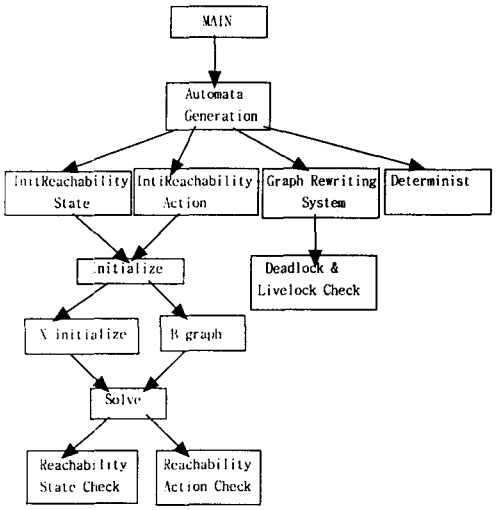
o Safety 검사
그래프 재표기 시스템(Graph Rewring System, 이하 GRS라함)을 사용하여, Automata모델이 안전성을 만족하는지 여부를 검사하여 화면에 결과를 출력한다.

o Determinist 검사
Automata 모델에서 어떤 상태에서 같은 행위에 의한 다른 천이가 존재하는 지를 검사하고,만족하지 않은 상태가 존재하면 화면에 결과를 출력한다.

o State reachability
Automata 모델에서 어떤 상태에 대한 입력과 출력행위가 올바르게 수행되는지를 검사하는 부분으로써, automata 검증 알고리즘을 사용하여 그 결과를 화면에 출력한다.

o Action reachability
automata 모델에서 어떤 3개의 연속적인 행위가 발생하는 지를 검사하는 부분으로써,automata 검증 알고리즘을 사용하여 그 결과를 화면에 출력한다.

Automata 프로토콜 명세에 대한 C++언어로 구현된 모델 검정기 구성은 (그림4)와 같다.



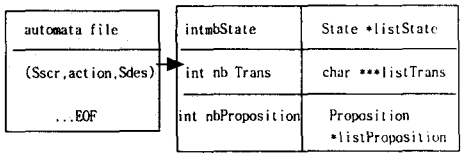
(그림 4) 검증기 모듈 구성도

Main 함수는 automata 요소를 포함하는 파일 이름과 함께 실행시키면, 검사하고자 하는 부분을 선택하게 된다. 즉, 안전성을 검사하고자 할 때 Deadlock & Livelock Check를 선택하게 되면 그래프 재표기 시스템을 수행하여 강한 연결 여부를 검사하게 되고 검출된 상태가 행위가 없는 상태이면 Deadlock, 행위가 있으면 Livelock을 판별한다. 그리고 필연성을 검정하기 위해 State Reachability Check, Action Reachability Check를 선택하게 되면 이미 정의된 Modal mu-calculus 논리식에 따라서 초기화를 하게 되는데, 이때 초기화는 BitArray를 초기화하는 X.initialize, edged-labeled directed graph를 생성하는 B.graph를 호출하게 된다. 이렇게 초기화 시킨후 solve 알고리즘을 적용하고, 초기 선택에 따라 Bit Array를 갱신하고 State Rechability, 3 action Reachability를 검사하게 된다.

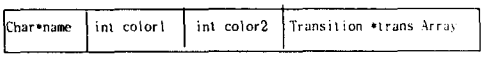
2.모듈별 기능

1) Automata Generation

automata 모델에 저장된 file을 읽어들여서 automata data structure를 생성

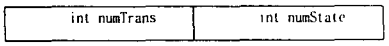


[State]



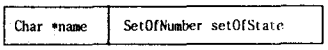
name: 상태명, color1과 color2는 레이블,
transArray 배열은 천이

[Transition]



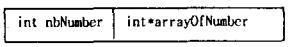
numTrans: 행위번호, numState: 도달상태번호

[Proposition]



name: 명제명(ex: tt, ff 혹은 Atomic proposition)

[SetOfNumber]



nbNumber: arrayOfNumber의 갯수,
*arrayOfNumber: 번호

2) Safety

automata data structure를 입력으로 하여 레이블을 초기화시킨 후, GRS를 수행하여 automata 모델이 안전성 여부를 만족하는지를 검사하여 화면에 출력

o GRS initialize

automata data structure를 입력받아 초기상태는 (Red/Blue)로, 그외의 상태는 (Green/Green)으로 레이블을 설정하고 GRS로 data structure를 넘겨준다.

o GRS

초기화된 automata data structure에서 그래프 재표기법을 사용하여 automata data structure의 상태 레이블을 천이에 따라 갱신한다.

o Safety check

레이블이 갱신된 automata data structure의 상태 중에서 레이블이

(Red/Blue)가 아닌 상태들을 검출하여 화면에 출력한다.

3) Determinist

automata data structure를 입력으로 하여, 어떤 상태에서 같은 행위에 의한 다른 천이가 존재하는지를 검사하여 화면에 출력한다.

4) State reachability

automata data structrue를 입력받고, 사용자로부터 키보드로 검사상태(S), action 1과 action 2를 입력받아, state reachability formula에 맞는 Block을 생성하고, 이에 따라 bit array, M array 그리고 Edge-labeled-graph를 생성한다. 그리고 나서 bit array를 초기화 시키고 M array에 <상태,변수>짝을 FIFO(First Input First Output) 큐로 저장한다. solve에서는 Edge-labeled-graph를 참조하여 M array 그리고 초기화된 bit array를 가지고, M array가 빌 때까지 <상태, 변수>짝을 추가/삭제하면서 bit array를 갱신한다. 최종적으로 갱신된 bit array에서 action 1 S action 2 가 존재하는지를 검사하고 그 결과를 화면에 출력한다.

o Block initialize(state reachability)
automata data structure와 사용자로부터 키보드로 검사상태 S, action 1 그리고 action 2를 차례로 입력받아 state reachability에 해당하는 논리식 Z.(Y.A(<action2>tt[action1]Y))[-]Z,A = {S}에 따라 max block과 min block으로 나누고 변수를 생성하여 block을 초기화한다.

o Edge-labeled graph
초기화된 블록으로부터 각 변수의 상관관계를 나타내는 그래프를 생성한다. 이 그래프는 solve 알고리즘 수행시 참조된다.

o M array creation
block의 갯수만큼 즉, min block과 max block의 Marray를 생성한다. 이 M array는 FIFO(Fisrt Input First Out)의 형태 추가되고 삭제된다.

o Bit array creation
(automata 모델의 상태갯수 block의 변수 갯수)만큼의 비트 배열을 생성한다.

o M array addition/deletion
bit array 초기화시 추가할 <상태,변수> 짝을 저장하고, solve 수행시 <상태,변수>를 추출하여 bit array를 갱신하고, 추가할 <상태,변수> 짝이 있을 때는 해당 M array에 추가한다.

o Bit array initialize/update
생성된 bit array에 초기화 규칙을 적용하여 초기화시킨후 solve에 넘겨준다. solve 수행시 bit array는 갱신되고, 이 과정은 M array가 빌 때까지 계속된다. 최종적으로 갱신된 bit array는 check state reachability에 참조된다.

o Solve
생성된 Edge-labeled-graph를 참조하고, M array로부터 <상태,변수> 짝을 하나씩 추출하여 갱신 규칙에 따라 bit array를 갱신한다. 이 과정은 각각의 M array가 빌 때까지 계속된다.

o Check State reachability
갱신된 bit array의 변수 X2, X4 그리고 X5가 모두 참(1)인 상태가 존재하는지를 검사한다. 만약 만족하는 상태가 존재한다면, automata 모델은 상태도달성(action 1 S action 2)을 만족한다. 그렇지 않을 경우, 상태도달성을 만족하지 않으므로 이러한 일련의 천이 순서는 절대 발생하지 않음을 나타낸다.

5) action reachability

automata data structrue를 입력받고, 사용자로부터 키보드로 action 1, action 2 그리고 action 3를 입력받아, action state reachability formula에 맞는 Block을 생성하고, 이에 따라 bit array, M array 그리고 Edge-labeled-graph를 생성한다. 그리고 나서 bit array를 초기화 시키고 M array에 <상태,변수>짝을 FIFO(First Inpu First Output) 큐로 저

장한다. solve에서는 Edge-labeled-graph를 참조하여 M array 그리고 초기화된 bit array를 가지고, M array가 빌 때까지 <상태,변수>짝을 추가/삭제하면서 bit array를 갱신한다. 최종적으로 갱신된 bit array에서 연속적인 일련의 행위순서인 action 1 action 2 action 3 가 발생하는 지를 검사하고 그 결과를 화면에 출력한다.

o Block initialize(action reachability)
automata data structure와 사용자로부터 키보드로 action 1, action 2 그리고 action 3를 차례로 입력받아 action reachability에 해당하는 논리식 Z.[action1](Y.(<action3>tt[action2]Y))[-]Z에 따라 max block과 min block으로 나누고 변수를 생성하여 block을 초기화한다.

o Edge-labeled graph, M array creation
Bit array creation,M array addition / deletion, Bit array initialize /update, Solve의 6개 기능은 state reachability의 각각의 기능과 동일함.

o Check action reachability
행위 추적 기법을 사용하여 행위 순서가 automata 모델에서 발생하는지를 검사한다. 먼저 갱신된 bit array의 변수 X2가 참(1)이고 X6가 거짓(0)인 상태가 존재하는지를 검사한다. 만약 상태가 존재하지 않는다면, 이 행위순서는 발생하지 않으므로 검사를 종료하고 결과를 출력한다. 그렇지 않고 존재한다면 X2와 X3가 모두 참(1)인 상태가 존재 여부를 검사한다. 만약 존재하지 않는다면, 검사를 종료하고 결과를 출력하고, 그렇지 않고 존재한다면, 이 automtata 모델은 행위순서(action 1 action 2 action 3)인 행위도달성을 만족한다고 화면에 출력한다.

V. 결 론

초고속정보통신망 구축을 위한 광대역 종합 정보 통신망(B-ISDN) 구현을 위해 국내외적으로 많은 연구가 이루어지고 있다. ATM 기술 및 서비스 전개에 있어 성공의 핵심 사항은 표준화와 여러 구현 제품 간의 상호운용성을 보장하는 것이라고 할 수 있는데, 실제로 여러 이기종 제품 간의 상호운용성을 보장하기 위해서는 시스템이 지원하는 모든 인터페이스에 대해 적합성 시험 및 개발과정에서의 명세에 대한 체계적인 검정이 수행되어야 한다.

. 본 연구에서는 SDL형식 기술기법으로 표현된 ATM 프로토콜로부터 생성된 중간 모델 인 automata에 Modal-μ -calculus에 의해 검정대상인 deadlock, livelock, reachability 및 liveness에 대한 표현과 해당 알고리즘 적용 및 분석을 수행하는 검정기 개발이다. 모델 검정기에서 사용할 논리는 강력한 표현력을 가진 Modal-μ -calculus를 사용하고 행위 명세서에 대한 검정 기능을 C++언어로 구현하였다. 또한 당 센터에서 개발한 SDL검정기와 적합성시험 생성기를 상용 SDL Editer & Simulator도구들과 통합하여 사용자들이 쉽고 편하게 쓸 수 있도록 환경구현 및 통합 시스템을 개발하였다..

이와 같은 연구 결과로 얻어진 프로토콜 개발 통합환경은 당 센터에서 추진중인 ATM 적합성시험기술연구의 일부로 활용이 가능하며, 향후 HAN/B-ISDN의 개발품에 대한 적합성/상호운용성 시험서비스의 제공에 활용될 기반기술연구로 활용이 기대된다. 그리고 21세기 정보화 사회를 위한 핵심기술인 ATM/B-ISDN, PCS, IN 등 국책과제로 수행되는 각종 프로토콜 개발 과정에서 프로토콜 개발 통합환경으로 활용될 수 있다.

향후 연구 사항으로는 두개 이상의 fixed point를 사용한 여러가지 modal mu-calculus논리식에 대해서 검증할 수 있는 기능을 구현하는 것이다. 또한 시간

개념이 추가된 실시간 시스템에 대한 검증기능 구현도 향후 추진되어야 할 연구 사항이다.

참고 문헌

[1]Kenneth L.McMillan, Symbolic model checking,Kluwer Acacenic Publishers, Model checking, 1996.

[2]P.V.Koppol and K.C.Tai,Conformance Testing of Protocol specification as Labeled Transition system, International Workshop on Protocol Test System,IWPTS95, pp.143 -158, Evry, France, September, 1995.

[3]E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal specifications. ACM Transactions on Programming Languages and Systems, 8(2):224-263, 1986.

[4]G.Bhat and R.Cleaveland, Efficient local model checking for fragments of the modal mu-calculus,In Proceeding of TACAS96, 1996.

[5]R.Cleaveland,M.Dreimuller and B.steffen,Faster Model-checking for the Modal Mu-Calculus.,To appear in Proceedings of the 1992 Workshop on Computer-Aided Verification, Lecture Notes in Computer Science.

[6]Emerson,E.A. and C.-L.Lei, Efficient Model Checking in Fragments of the Propositional Mu-Calculus. ,Proceedings of the First Symposium on Logic in Cimputer Science ,267-278, 1986.

[7]박용범, 김태균, 김성운, LTS 명세 검증을 위한 모델 검증기 개발 한국정보처리학회논문지 5권4 호, 1998.4

[8]A.Arnold and B. Crubille. A Linear Algorithm To Solve Fixed-Point Equations on Transition System .Information Processing Letters 29:57-66, 30 September 1988.

[9] R.Alur and D.L.Dill. A theory of timed automata Theoretical Computer Science, 126:183-235, 1994

[10]김성운외 2명, 적합성시험에서 그래프 재표기 시스템을 활용한 강한 연결 판단방법,한국정보처리학회논문지 4권 5호,1997.5

[11]Aho, A.O., Hopcroft, J.E., and Ullman,J.D.,The Design and Analysis Analysis of Computer Algorithms Addison-Wesley,Reading, Mass. 1974.

[12]진병문, 김성운, 최영한, Automated Test Generation from Specifications based on Formal Description Techniques, ETRI Journal, vol.19 No.4, 1997.12