제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

Statechart Diagram의 정확성 검증을 위한 SMV코드 자동생성

김종한* 박기창** 이상준*** 김병기**
*전남대학교 소프트웨어공학협동과정
**전남대학교 전산학과
***서남대학교 컴퓨터 정보통신학과
chkim@superse.chonnam.ac.kr

Automatic SMVcode Generation for the Correctness Verification of Statechart Diagrams

Chong-Han Kim*, Ki-Chang Park**,Sang-Jun Lee*** ,Byung-Ki Kim**
*Dept. of Interdisciplinary Program of Software, Chonnam National University
***Dept. of Computer Information Communication, Seonam University
**Dept. of Computer Science, Chonnam National University

요 약

잘못된 명세로 인한 자원의 손실을 막기 위해서는 반드시 명세에 대한 검증이 필요하다. 객체지향 모델링 언어의 표준인 UML은 각각의 다이어그램이 사용자의 요구사항을 정확히 반영하고 있는지를 검증하기가 매우 어렵다. 본 논문은 UML의 여러 다이어그램 중 상태 다이어그램(Statechart Diagram)의 명세에 발생해서는 안되는 상태 또는 발생할 수 없는 상태와 같은 오류의 존재여부 등의 정확성을 검증하기 위해 CTL을 이용한 정형검증도구인 SMV를 이용한다. 이를 위해 UML의 상태 다이어그램에서 상태 정보와 상태 천이 정보를 추출하여 SMV로 변환하는 규칙을 찾아내어 UML의 상태 다이어그램의 정확성 검증을 수행하는 방법을 제안하고 자동으로 변환하는 툴을 설계한다.

1. 서론

오늘날 소프트웨어가 대형화되고 여러 기능이 통합되면서 요구사항이 복잡해지고 통제가 힘들어지게 되자, 현상을 단순화하고 추상화하는 과정인 모델링이 나타났다. 객체지향 모델링언어의 통합 표준인 UML(Unified Modeling Language)[1]은 반정형 명세언어로 요구사항에 따라 작성된 다이어그램이 정확한지 보장되지 않는다.[2] 정형기법[3]은 의미전달이 불확실한 자연어 대신 논리적인 언어로 명세하는 정형명세와 정형적인 언어로 명세된 시스템의 정확성 및 안정성을 검증하는 정형검증이 있다. 정형기법은 자연어가 내포하는 애매모호함 때문에 발생될 수 있는 오류, 즉 사용자의 요구사항 자체의 오류나 설계자가 이를 수용하는 과정에서 잘못된 인지로 인한 오류, 또는 설계자의 실수 인한 오류 등을 검증하고 수정함으로써 개발 시 발생하는 인적, 경제적, 시간적 손실을 줄일 수 있다.

모델 체킹[4][6]은 정형검증방법중 하나로서 유한 상태 시스템에서 그 모델이 만족해야할 특성을 시제논리(Temporal Logic)로 명세하여 검증하는 방법이다.

본 연구에서는 UML에서 유한상태기계(Finit State System)를 나타내는 상태 다이어그램이 "사용자 요구사항에 맞게 설계되었는가?" "불필요한 상태와 같은 오류는 없는가?" 등과 같은 정확성을 검증하기 위해, UML의 상태 다이어그램과 정형검증도구인 SMV(Symbolic Model Verification)[4][5][8]와의 연관성을 찾아 상태 다이어그램에서 SMV 언어로의 변환을 위한 코드 추출 방법을 제안한다. 제안한 방법은 타당성은 예시를 통해 제시하고 자동으로 코드 생성하는 툴을 설계한다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문과 관련된 연구들을 살펴보고, 3장에서는 UML 상태 다이어그램에서 변환 규칙을 추출하여 SMV로의 변환하는 방법을 제시하고 자동 코드변환기를 설계 한다.. 마지막 4장에는 결론 및 향후 연구방향을 기술한다.

2.관련연구

2.1 UML 상태 다이어그램

상태 다이어그램(Statechart diagram)이란 단일 객체에 대하여 사건에 반응하여 일으키는 상태 변화를 시간을 축으로 표현한 다이어그램이다. 상태 다이어그램은 상태(State)와 객체의 시작을 나타내는 시작상태(Initial State), 종료를 나타내는 종료상태(Final State), 이전상태에서 다음 상태로의 천이를 나타내는 상태 천이선

(State Transition)으로 구성되어 있다(그림 1).

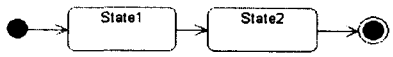

그림 1. 상태 다이어그램의 예

2.2 모델 체킹(Model Checking)

모델 체킹은 정형검증 방법의 하나로 시스템의 동작을 유한상태기계의 형태로 명세하고 그 시스템이 만족해야 할 특성을 트리형식으로 표현하는 CTL(Computer Tree Logic)[5][7][8]이나 LTL(Linear Temporal Logic)과 같은 시제논리를 이용해서 검증한다.

■ CTL(Computer Tree Logic)
Path Quantifier(A, E) +Temporal Operator(X, F, G, U)
A:모든 경로대해서
E:최소한 하나의 경로에 대해서

Xp : 다음시점에 p가 참이다. (Next)
Fp : 언젠가 p가 참이다. (Future)
Gp : 언제나 p가 참이다. (Globally)
pUq: q가 참인 시점까지 p가 참이다.(Until)

2.3 SMV

SMV는 CTL이라는 시제논리와 BDD(Binary Decision Diagram)을 이용하여 주어진 논리의 참과 거짓을 구별하는 모델 검증 도구이다.

그림 2은 SMV의 구조를 나타낸다. Logic Model은 상태변수(VAR), 할당(ASSIGN), 정의(DEFINE) 등을 통하여 시스템 표현하는 부분이다. Specifications는 모델의 정확성에 관한 질문(Spec)을 입력하는 부분이다. SMV 모델체커는 질의에 대해 표현된 시스템의 상태를 모두 검사하여 참 또는 거짓을 판별한다.

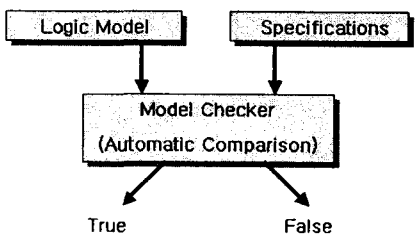

그림 2. SMV Structure

VAR부분은 상태의 type과 같은 상태정보를 표현하며 ASSIGN부분은 상태의 천이조건이나 천이의 전후 상태 등과 같은 상태 천이 정보를 나타낸다. DEFINE부분은 각 상태가 참 상태를 나타내며 마지막으로 SPEC 부분은 검증하고자하는 상태에 대해 직접 입력하는 부분이다. SMV는 이렇게 명세된 것을 자동으로 검증하여 질의에 대한 답으로 True 또는 False를 출력한다.

3. 상태 다이어그램의 SMV 변환 방법

3.1 Rational Rose로 명세된 상태 다이어그램의 상태 및 상태천이 코드 추출

SMV 코드로의 변환을 위해서는 우선 Rational Rose로 작성된 상태 다이어그램 코드에서 상태와 상태 천이 부분을 표현하는 부분을 추출한다.

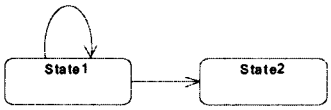

그림 3. 상태와 상태 천이선

그림 4에서 object state 부분에서 state1과 state2의 상태정보와 Transition의 supplier부분에서 천이정보를 찾을 수 있다.

```
statemachine    (object State_Machine "State/Activity Model"
    quid        "3F2FB3DB0261"
    states      (list States
        (object State "State1"
            quid        "3F2FB817080A"
            transitions         (list transition_list
                (object State_Transition
                    quid        "3F2FB8200242"
                    supplier    "State2"
                    quidu       "3F2FB81B0290"
                    sendEvent   (object sendEvent
                        quid            "3F2FB8200253"))
                (object State_Transition
                    quid        "3F2FB8250290"
                    supplier    "State1"
                    quidu       "3F2FB817080A"
                    sendEvent   (object sendEvent
                        quid            "3F2FB8250293")))
            type        "Normal")
        (object State "State2"
            quid        "3F2FB81B0290"
            type        "Normal"))
    partitions  (list Partitions)
    objects     (list Objects)
```

그림 4. State와 State Transition code의 추출

3.2 SMV Code로의 변환

Rose로 작성된 상태 정보와 상태 천이 정보를 기반으로 SMV 코드로의 변환을 수행한다. state1과 state2로 표기된 state 변수부분은 SMV의 변수부분 표1과 같이 VAR로 변환할 수 있다.

표 1. Rose상태 부분의 변환

ROSE	state (object state "state1") state (object state "state2")
SMV	state : {state1, state2};

ASSIGN부분은 state1이 자기 자신과 천이 될 수 있으므로 표2와 같이 변환할 수 있다.

표 2. Rose상태천이 부분의 변환

ROSE	object State "state1"(object Stats_Transition (supplier "state2") object States_Transition (supplier "state1"))
SMV	next(state) := case state = state1: {state1,state2}; 1 : state; esac;

DEFINE과 SPEC부분은 검증하고자하는 조건과 상태를 직접 입력하여 검증한다. DEFINE에서 a는 state1이 참인상태, b는 state2가 참인 상태를 나타낸다.

```
DEFINE
        a := state = state1;
        b := state = state2;
```

SPEC은 검증하고자하는 부분에 조건을 주어 참, 거짓을 밝힌다.

3.3 Rose로 작성된 상품주문 상태 다이어그램의 SMV코드로의 변환 예

그림.5와 같은 형태의 상품주문시스템에 관한 간략한 상태 다이어그램을 가정한다.

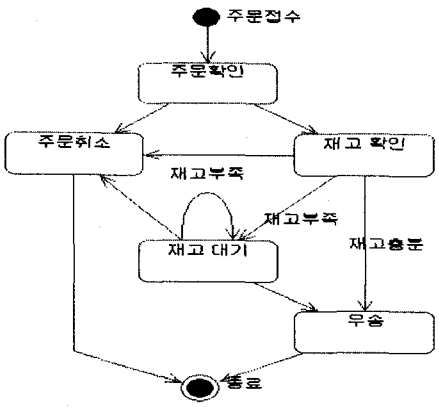

그림 5. 상품주문의 상태 다이어그램 예

그림 6은 그림 5의 다이어그램에 대한 Rose의 소스코드이다. object State "주문접수"는 type 이 "startState" 이므로 초기상태를 뜻한다. "주문접수"상태에서 천이된 object State "주문확인"상태는 supplier "주문취소"와 supplier"재고확인"의 두 상태로 천이된다. object State에 의한 상태와 supplier에 의한 상태천이를 추출할 수 있으므로 다음과 같이 Rose 코드를 SMV로 변환할 수 있다.

```
statemachine    (object State_Machine "State/Activity Model"
    quid        "3F310620009C"
    states      (list States
        (object State "주문접수"
            transitions (list transition_list
                (object State_Transition
                    supplier    "주문확인" )
            type            "StartState")
        (object State "주문확인"
            transitions (list transition_list
                (object State_Transition
                    supplier    "주문취소" )
                (object State_Transition
                    supplier    "재고 확인")
            type            "Normal")
        (object State "주문취소"
            transitions (list transition_list
                (object State_Transition
                    supplier    "종료"      )
            type            "Normal")
        (object State "재고 확인"
            transitions (list transition_list
                (object State_Transition
                    supplier    "재고 대기"
                    Event       (object Event "재고부족")
                (object State_Transition
                    supplier    "우송"
                    Event       (object Event "재고충분")
                (object State_Transition
                    supplier    "주문취소"              )
            type            "Normal")
        (object State "재고 대기"
            transitions (list transition_list
                (object State_Transition
                    supplier    "재고 대기"     )
                (object State_Transition
                    supplier    "우송"          )
                (object State_Transition
                    supplier    "주문취소"      )
            type            "Normal")
        (object State "우송"
            transitions         (list transition_list
                (object State_Transition
                    supplier    "종료"      )
            type            "Normal")
        (object State "종료"
            type            "EndState"))
    partitions  (list Partitions)
    objects     (list Objects)
```

그림 6. Rose로 명세 된 상품주문 다이어그램의 코드

각 state name을 표3과 같이 재정의 한다.

표3. State name의 재 정의

s0	주문접수	s4	재고대기
s1	주문확인	s5	우송
s2	재고확인	s6	종료
s3	주문취소		

재 정의된 상태 변수 명을 이용하여 다음과 같이 SMV 코드로 변환할 수 있다.

```
MODULE main
VAR
        state : {s0,s1,s2,s3,s4,s5,s6};
ASSIGN
        init(state) := s0;
        next(state) := case
        state = s0 : s1;
        state = s1 : {s2, s3};
        state = s2 : {s3, s4, s5};
        state = s3 : s6;
        state = s4 : {s3, s4, s5};
        state = s5 : s6;
        1 : state;
esac;
```

이렇게 명세된 시스템을 검증하기 위한 예로 다음과 같은 몇 가지 경우를 예측할 수 있다.

①주문이 확인되면 우송이 되는 경우가 생길 수도 있다.

②주문이 확인되면 항상 우송되는 것은 아니다.

③주문취소가 일어나면 어떤 경우라도 우송이 되어서는 안 된다.

주문확인이 일어난 상태, 즉 "s1이 참인 상태"를 a, 주문취소가 일어난 상태, 즉 "s4가 참인 상태" b라고 하고, 우송하는 상태, 즉 "s5가 참인 상태"를 c라고 놓고

각 상황을 명세하면 다음과 같이 작성될 수 있다.
SEPC부분은 위에서 가정한 ①,②,③의 경우를 명세한다.

```
DEFINE
        a := state = s1;
        b := state = s3;
        c := state = s5;

SPEC AG(a -> EF c)
SPEC AG(a -> AF c)
SPEC AG(b -> EF c)
```

이렇게 변환된 코드를 SMV를 이용하여 검증한 결과가 그림 7에 제시되어 있다.

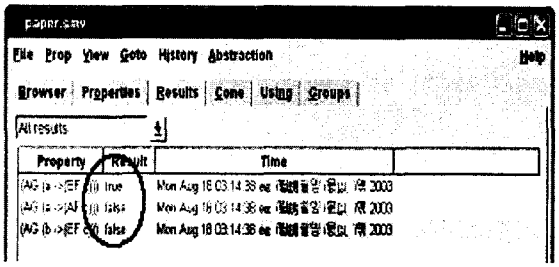

그림 7. SMV로 작성된 코드와 검증 결과

그림 7에서 검증된 부분을 살펴보면 ①의 상태는 하나의 Path만 만족하면 되므로 사건이 발생할 수 있는 경우가 있기 때문에 예측대로 True가 출력되었고, ②상태는 발생할 수 없는 경우가 생길 수 있어 항상 만족하는 결과가 나오지는 않으므로 False가 출력되었다. ③과 같은 경우는 존재하지 않으므로 False가 출력되었다. 이와 같이 원하는 상태의 정확성을 검증할 수 있다.

3.4 SMV코드변환기의 설계

그림 8은 상태다이어그램에서 상태정보와 상태 천이 정보를 파싱한 후 SMV코드로 변환을 위한 class Diagram이다.

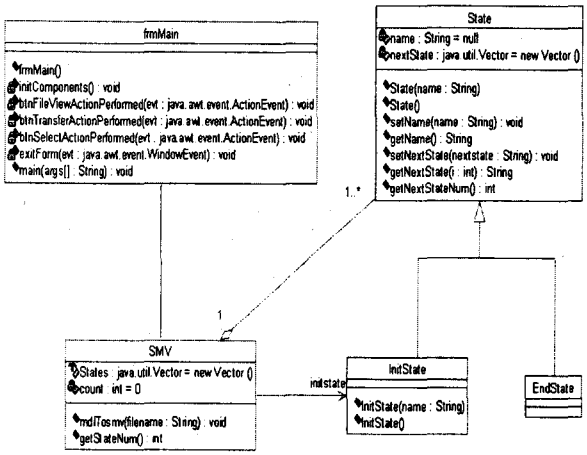

그림8. SMV코드 변환기의 class diagram

State class는 상태 다이어그램에서 상태정보와 상태 천이정보를 추출하여 변환하고 state class에서 상속된 InitState class는 초기상태부분을 SMV코드로 변환한다. SMV class에서 상태 다이어그램의 포맷인 mdl 파일을 smv 파일로 변환한다.

4. 결론 및 향후 연구방향

시스템의 설계 시 검증은 매우 중요한 일이다. 본 논문은 시각 명세언어로 널리 쓰이고 있는 UML 상태 다이어그램을 Rational Rose를 이용해 명세하였고, 명세된 다이어그램에서 정형검증 도구인 SMV의 언어와의 연관성을 찾았다. 이를 기반으로 SMV언어로 변환할 수 있는 규칙을 찾을 수 있고 SMV를 통해서 검증한 예를 제시함으로써 타당성을 입증하였고 자동으로 변환하는 툴을 설계하였다. 향후 연구과제로 상태 다이어그램과 SMV코드와의 연관규칙을 정형화하여 윈도우용 툴로 구현하고 SMV코드를 확장하여 천이조건에 따라 천이하는 부분까지 고려하겠다.

참고문헌

[1] Roger S. Pressman "Software Engineering A Practltiners' Approach" 3rd Ed. McGraw Hill

[2] UML포럼 http://www.webmania.co.kr/forum

[3] Andrew Harry, *Formal Methods Fact File*, John Wiley & Sons, 1999.

[4] William Chan, Richard J. Anderson, Paul Beame, Steve Burns, Francesmary Modugno, David notkin, Jon D. Reese, "Model Checking Large Software Specifications," IEEE Transactions on Software Engineering, Vol.24, NO.7, July 1998.

[5] E. M. Clarke, O. Grumberg, and D. E. Long, Checking and Abstraction," In Proceedings of the Nineteenth Annual ACM Symposium on Principles of Programming languages, January 1992.

[6] 방기석, 이주용, 최진영, "SPIN을 이용한 CTL 모델체킹 방법론 연구," 한국 정보처리학회 춘계 학술발표논문집 제8권 제1호, 2001년

[7] 박사천, 권기현, "계층형 크립키 구조를 위한 CTL모형 검사 알고리즘," 정보처리학회 춘계 학술 발표대회 VOL.09 NO.01 pp.407-410, April 2002.

[8] 권기현 "모델체킹의 이해", 소프트웨어 교육 세미나자료집, May 2003.

[9] 이승훈, 김진경, 문일, "SMV를 이용한 보일러 공정 운전 절차의 안전성 검색," 한국 화학공학회지, Vol. 37 No.5 October 1999.