Statecharts 명세의 모듈 기반 검증

서선애 ○ 오승욱 조승모 이남희 차성덕 권용래
한국과학기술원 전산학과

Modular Verification of Statecharts Specification

Sun Ae Seo Seung Uk Oh Seung Mo Cho
Nam Hee Lee Sung Deok Cha Yong Rae Kwon
Dept of Computer Science
at Korea Advanced Institute of Science and Technology

요 약

모형 검증을 통한 시스템 명세의 정형적인 검증은 상태 폭발 문제로 인해 많은 어려움을 겪고 있다 여러 개의 병렬 프로세스로 구성된 시스템에서 지수적으로 증가하는 상태의 갯수로 인해 현실적으로 모형 검증을 적용하는 것이 불가능한 경우가 많다 이런 문제점을 해결하기 위해서 시스템을 모듈 단위로 생각하여 정형 검증을 시도하는 많은 연구가 수행되고 있다. 병렬성을 중요한 특성의 하나로 하는 Statecharts 또한 널리 사용되고 있음에도 불구하고 아직 모듈을 바탕으로 검증을 수행하려는 시도가 그리 많지 않다. 본 연구에서는 내장 소프트웨어 시스템에 널리 사용되는 Statecharts명세를 모듈을 바탕으로 검증하는 방법을 제시하고자 한다 먼저 Statecharts에서의 모듈을 정의하고, 그와 같은 정의를 바탕으로 여러 개의 모듈로 구성되어 있는 Statecharts 명세의 모듈 기반 검증 방법을 제안한다 여기서 사용되는 모듈 기반 검증은 환경에 대한 기정이 만족된다면 모듈은 반드시 주어진 성질을 만족한다는 가정-보증 추론(Assume-Guarantee Reasoning)을 이용한다.

1. 서론

시스템 개발의 초기 단계에서 발생하는 오류는 전체 시스템에 미치는 영향이 다른 단계에서 발생한 오류에 비해 훨씬 심각하다는 것은 이미 입증되어 있다 따라서 시스템의 명세가 요구 사항을 제대로 반영하는가를 검증하는 과정은 매우 중요하다 정형적 검증 방법의 하나로 널리 사용되는 것이 모형 검증 방법이다. 모형 검증[1]는 전체 시스템 모형을 상태와 상태 간의 전이로 표현하는 전체 상태 그래프(Global-State Graph)로 나타내고 그래프의 노드를 방문하면서 각 상태에서 주어진 시스템의 성질이 만족되는지를 검사하는 방법이다

일반적으로 시스템 모형이 단순하면 시스템 모형이 만족하는 상태 그래프를 방문하는 시간이 작기 때문에 빠른 시간 내에 시스템의 성질이 만족하는가를 검증할 수 있다. 하지만, 시스템 모형이 많은 수의 병렬 프로세스로 구성되어 있는 경우, 전체 상태 그래프의 크기는 각 프로세스의 상태 크기의 제곱이 된다. 이 경우 전체 상태 그래프의 노드를 방문하면서 많은 수의 상태를 검사해야 하는데, 이 같은 현상을 상태 폭발 문제라고 한다

지금까지 상태 폭발 문제를 해결함으로써 모형 검증의 효율성과 가용성을 높이려는 많은 연구가 진행되었다[2, 3]. 대표적인 연구는 기호화 모형 검증 방법이다[2]. 이 방법에서는 상태의 집합과 상태 전이를 이진 대수식을 사용하여 표현하기 때문에 상태 그래프를 직접적으로 표현하기 위한 데이터 구조가 필요하지 않다. 따라서 보다 많은 상태를 가지는 시스템의 검증에 모형 검증 방법을 적용할 수 있다. 하지만, 병렬 프로세스의 병합으로 인해 발생하는 상태 폭발 문제를 근본적으로 해결할 수 있는 것은 아니다. 이 문제를 해결할 수 있는 가장 자연스러운 방법은 각각의 병렬 프로세스에 대해 검증을 수행한 뒤 그 결과를 조합해서 전체 시스템에 대해 검증을 수행하는 합성적 검증 방법이다[3] 합성적 검증 방법은 기호 모형 검증에 비해 자동적으로 검증을 수행할 수 없다는 한계를 가지고 있지만, 상태 폭발 문제를 유발하는 병렬 프로세스를 직접 다루기 때문에 기호 모형 검증을 적용하기 어려운 시스템의 검증에 사용할 수 있다

본 논문에서는 반응 시스템(Reactive System)의 명세에 널리 사용되고 있는 정형적 명세 언어인 Statecharts로 작성된 요구 사항 명세를 모듈을 기반으로 하여 검증하는 방법을 제시하고자 한다. 본 연구에서 제시하는 모듈 기반 검증 방법은 합성적 검증 방법의 하나인 가정-보증 추론(assume-Guarantee Reasoning)[5]에 기반을 두고 있다. 이같은 모듈 기반 검증 방법을 이용하여 여러 개의 컴포넌트로 구성된 Statecharts 명세의 검증에서 발생할 수 있는 상태 폭발 문제를 해결할 수 있다. Statecharts 모듈 정의는 변수들의 사용 범위를 제한하고 그 행위를 정의함으로써 이루어진다. 이 모듈 정의를 바탕으로 모듈 기반 검증을 수행하는 과정을 제시하고 예제를 통해 본 논문의 접근 방법을 적용하였다.

본 논문의 구성은 다음과 같다 2장에서는 Statecharts 및 합성적 검증 방법의 하나인 가정-보증 추론 방법을 소개한다 3장에서는 본 연구의 접근 방법인 Statecharts에서의 모듈을 정의하고, 정의를 바탕으로 합성적 검증 방법을 제안한다. 마지막으로 4장에서는 결론과 향후 연구 방향에 대해 기술한다.

2. 관련 연구

2.1 Statecharts

Statecharts는 대규모의 반응 시스템의 명세 작성을 위하여 개발된 정형적 명세 언어로서, 기존의 유한 상태 기계와 상태 전이 다이어그램이 확장된 형태의 상태 기반 정형 언어이다.

Statecharts에서는 복잡한 상태를 표현하기 위해 상태의 계층 구조와 병렬성을 AND 상태와 OR 상태를 통해 표현할 수 있도록 하였다 표현된다 AND 상태에 속하는 하부 상태들은 병렬적으로 존재하는 상태로서 동시성(Concurrency)을 표현하는 수단으로 사용된다. 시스템이 어떤 AND

상태에 있다는 것은 해당 AND 상태에 속하는 모든 하부 상태에 있다는 것을 의미한다 반면 OR 상태에 있다는 것은 그 상태의 하부 상태 중 특정한 하나의 상태에 있다는 것을 의미한다. 상태의 전이는 상태 전이를 유발하는 조건이 충족될 때 이루어진다 이런 상태 전이에서 내부 이벤트를 발생시킬 수 있으며, 이렇게 발생된 이벤트는 시스템의 다른 모든 Statecharts 로 전해진다. 상태 전이에는 상태 전이를 일으킬 조건과 상태 전이가 일어났을 때 발생되는 출력 이벤트의 쌍으로 표현되는 레이블을 부가할 수 있으며, 형식은 다음과 같다.

Trigger event [trigger condition]/output events

또한 Statecharts에서는 동시성 가설(Synchrony Hypothesis)을 가정하고 있어, 이벤트에 의한 상태 전이가 일어나는 시간은 무시할 수 있을 정도로 짧다고 보고 있다. 따라서 한 이벤트에 의해 상태 전이가 일어나면, 그 상태 전이에 의한 내부 이벤트가 또 다른 상태 전이를 일으켜 연쇄 반응(Chain Reaction)이 가능하게 된다

2.2 가정-보증 추론

상태 폭발을 해결하기 위한 가장 자연스러운 접근 방법은 모듈에 기반하여 시스템을 분해(decomposition)하는 것이다. 분해된 시스템 모듈에 대해 모듈의 성질이 전체 시스템에서 만족되는 것을 보인다면, 모듈의 성질을 전체 시스템의 성질로 간주할 수 있고 더 나아가 이런 모듈의 성질을 시스템의 다른 성질을 이끌어 내기 위해 사용할 수 있다 어떤 모듈의 성질이 전체 시스템에서 만족되는 것을 보장하려면, 모듈 검증시 시스템에서 모듈을 제외한 환경(environment)에 대한 가정이 필요하게 된다. 이런 접근 방법이 바로 가정-보증 paradigm이다[5]. 즉, 컴포넌트의 환경(environment)이 가정(assumption)을 만족한다면 컴포넌트의 어떤 성질들이 반드시 만족된다는 것을 보장할 수 있다는 것을 의미한다

[5]에 의하면, 어떤 시스템이 Moore Machine 으로 정의된 M과 M'의 두 컴포넌트의 조합으로 구성되어 있을 때 다음이 만족된다

$$\frac{M||T\models\psi \quad M' \preceq T}{M||M'\models\psi}$$

T는 M의 환경에 대한 가정인 ϕ를 모형으로 나타낸 것이고, $\preceq$는 두 모형 사이에 프리오더(preorder)*가 존재함을 의미한다. 본 연구에서는 이같은 가정-보증 추론을 바탕으로 Statecharts의 모듈 기반 검증을 제시하고자 한다.

3. Statecharts 의 합성적 검증

이 장에서는 기존의 연구를 바탕으로 Statecharts 모듈을 재정의 하고 그와 같은 정의를 기반으로 모듈 기반 검증을 위한 가정-보증 추론 방법을 제안한다.

3.1 Statecharts 모듈 정의

Statecharts에서는 이벤트들의 순서 관계 및 반응성에 대해서는 동시성 가설을 가정하고 있다. Statecharts 명세에서 사용하는 상태들이나 이벤트들은 서로 밀접한 관계를 갖고 명시된 인과 관계에 의해서 동작한다 따라서, Statecharts 명세에서는 컴포넌트 간의 의존성을 최소화하고 컴포넌트 간의 통신을 인터페이스만으로 한정함으로써 응집도를 높이려는 모듈성이 부족하다 Statecharts에서 발생하는 상태 폭발 문제를 해결하기 위해서는 우선 기존의 Statecharts에 대해 모듈성에 대한 정의를 도입할 필요가 있다 Statecharts 에서 모듈에 대한 정의가 이루어진다면, Statecharts 에서 부족한 modularity를 보강함으로써 합성적 분석 방법을 적용할 수 있고, Statecharts 명세를 재사용 가능하게 함으로써 시스템 개발의 초기 단계에 이루어지는 요구 사항 명세의 품질을 높이고 효율적인 검증을 할 수 있게 된다

효율적인 검증 및 재사용의 단위로서 Statecharts 명세는 내부에서 사용하고 있는 변수[1] 들의 사용 가능한 범위를 한정하고 명세의 행위를 정의함으로써 이루어진다. 추상화된 모듈 행위를 이용하여 모듈 기반 검증을 수행하기 위해서는 변수의 사용 범위에 따라 모듈 행위를 추상화할 필요가 있다. 이처럼 변수의 성질을 제한하는 것은 Statecharts 의 특성 중 하나인 브로드캐스팅을 위반하는 것이지만 모듈 정의에 반드시 포함되어야 한다. Statecharts 의 행위는 기본적으로 주어진 Statecharts 명세로서 정의된다. 하지만 모듈을 이용한 효율적인 검증을 수행하기 위해서는 주어진 Statecharts의 명세 중에서 외부에 영향을 미칠 수 있는 행위들을 찾아내야 한다.

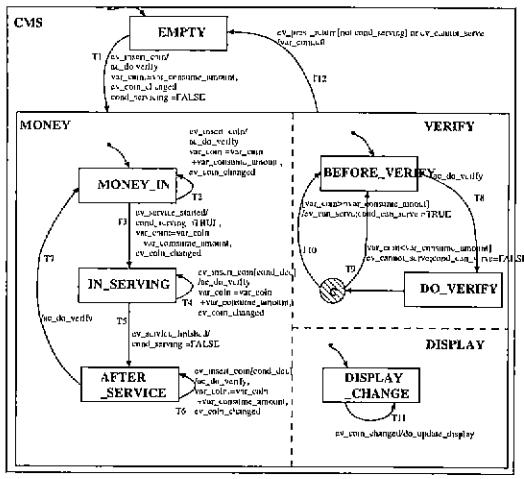

그림 1. Coin Management Subsystem(CMS)

변수의 사용 범위는 크게 세 가지로 나눈다 외부 환경으로부터 제공되고 모듈 내부에서 변경이 불가능한 외부 변수(EV), 모듈 내부에서 변경되며 외부에서 볼 수는 있지만 변경이 불가능한 공유 변수(IV), 그리고 모듈 내부에서만 볼 수 있고 변경할 수 있는 내부 변수(PV)가 그것이다. 실생활에서 많이 사용하는 공중 전화 및 커피 자판기와 같은 시스템에서 중요한 부분으로 사용되는 동전 처리부(CMS)를 모델링한 Statecharts(그림 1)에서 변수 정의는 표 1과 같다

Statecharts 모듈 CMS 의 행위 정의는 Statecharts 원래의 Statecharts 전체 모습인 그림 1과 같다. 하지만, 이 모듈을 기반으로 검증을 하기 위해서는 모듈 CMS 의 추상화된 행위를 정의하기 위한 방법이 필요하다 이 연구에서는 추상화된 행위를 변수의 초기값, 이벤트의 초기 조건, 이벤트/반응의 수행 쌍 그리고 이벤트 간의 순서 관계로 정의하였다 실제로

*검증에 대한 relation이 reflexive하고 transitive한 수학적 구조를 의미한다 여기서 두 모형 간에 preorder가 존재한다는 것은 한 모형이 성질을 만족할 때 preorder 의 관계에 있어 그것보다 작은 모형도 역시 성질을 만족한다는 것을 의미한다

[1] 이 변수는 Statecharts에서 사용하는 이벤트, 변수 들을 통칭하는 의미로 사용한다

표 1. CMS의 변수 정의

EV	var_consume_amount,cond_dec ev_insert_coin, ev_press_return ev_service_started, ev_service_finished
IV	ev_can_serve ev_cannot_serve cond_can_serve
PV	var_coin, ev_coin_changed, ac_do_verify ..

CMS 의 추상화된 행위를 이용하여 모형 검증을 하기 위해서 본 연구에서는 정의한 CMS 의 추상화된 행위는 표 2와 같다.

표 2: CMS의 추상화된 행위 정의

Initialization	cond_dec = TRUE/FALSE var_consume_amount := VALUE
Event Pre-condition	[TRUE] ev_insert_coin [TRUE] ev_press_return [cond_can_serve=TRUE] ev_service_started
Event/Action pair	ev_insert_coin / ev_can_serve ev_insert_coin / cond_can_serve = TRUE ev_insert_coin / ev_cannot_serve ev_insert_coin / cond_con_serve := FALSE
Event Ordering	ev_service_started , ev_service_finished

3.2 모듈 기반 검증 방법

본 연구에서 사용하는 모듈 기반 검증 방법은 [5]의 연구를 따른다. 즉, 외부 환경의 가정하에 모듈 M이 성질 ϕ를 만족하는 것을 보이고 외부 환경인 모듈 M'으로부터 M의 가정이 올바름을 보임으로써 두 모듈 M, M'로 구성된 시스템이 성질 ϕ를 만족함을 보인다

이것을 이용하여 Statecharts 모듈을 검증하기 위해서는 다음과 같은 검증 작업을 수행해 주어야 한다.

- 모듈 M' . 모듈 M의 환경에 해당하는 M'은 M의 외부 변수 중에서 M'의 공유 변수에 속하는 것들에 대해서 추상화 행위를 도출한다($M' \preceq T$). 이와같은 추상화된 행위는 3장 1절에서 언급한 것과 같이 변수의 초기화, 이벤트의 선행 조건, 이벤트/반응 쌍, 이벤트의 순서 등과 같은 성질로 나타난다.
- 모듈 M : 모듈 M'에서 제공하는 추상화된 행위를 가정으로 성질을 검증한다 ($M||T \models \psi$) 다른 모듈에서 제공하는 가정을 바탕으로 모형 검증을 수행하는 과정은 SMV[2]를 이용한다

이 과정을 통해서 두 개의 AND 컴포넌트로 구성된 Statecharts 시스템 전체에서도 이 성질이 만족됨을 알 수 있다 ($M||M' \models \psi$). 이 검증 방법을 이용하여 앞에서 정의한 동전 처리기에 대해서 검증을 수행하였다 모듈 M'의 추상화된 행위는 표 2로 정의하였다. 또 M'을 사용하는 다른 모듈 M을 실생활에서 많이 사용하는 공중 전화기와 커피 자판기로 명세하여 모형 검증을 수행하였다. 모형 검증을 수행할 때 공중 전화기나 커피 자판기 명세는 동전 처리부를 원래의 형태로 사용하지 않고 CMS 모듈에서 제공하는 추상화된 행위(표 2)를 가정하여 모형 검증을 수행한다 공중 전화기에 그림 1의 동전 처리부를 그대로 적용하여 검증을 하여 본 결과 사용된 BDD 노드 수가 13670개였으나, 추상화된 동전 처리부의 행위를 사용하는 모듈 기반 검증을 수행하였을 때는 2725개의 BDD 노드가 생성되어 큰 차이를 보임을 알 수 있었다

4. 결론 및 향후 연구 방향

본 논문에서는 반응 시스템의 명세에 많이 사용되는 Statecharts 명세를 모듈을 기반으로 검증하는 방법을 제안하였다 우선 모듈 기반 검증을 위해서 Statecharts에서 모듈의 정의를 제안하고 모듈을 바탕으로 가정-보증 추론 방법을 이용하여 모듈 기반 검증을 하는 방법을 제시하였다 그리고, 동전 처리부 Statecharts 명세를 이용하여 CMS 모듈을 정의하였고, 제안된 검증 방법을 바탕으로 동전 처리부를 모듈로 사용하는 동전식 공중 전화 시스템과 커피 자판기 시스템을 검증하였다. 제안된 방법론을 예제 시스템에 적용하여 본 결과 모형 검증을 수행할 때가 이전 방법에 비해 효율적인 결과를 보였다

본 논문은 기존의 합성적 검증에 관한 연구를 Statecharts 명세에 대해 적용하고 그것을 기반으로 더욱 효율적인 Statecharts 명세의 정형적 검증 기법을 제안한 데 의의를 둔다. 하지만, 예제에서 적용된 것과 같은 모듈의 추상화된 행위 정의는 전체 시스템의 검증에 충분한 정보를 제공하는지를 보장할 수 없다. 따라서, 모듈의 정의로부터 사용자가 원하는 모듈의 추상화된 행위를 추출할 수 있는 방법론에 대한 연구가 필요하다

참고 문헌

[1] E M. Clarke, E. A. Emerson and A P Sistal, "Automatic verification of finite-state concurrent systems using temporal logic specifications", *ACM Transactions on Programming Languages and Systems*, 8(2).244-263, 1986

[2] K L McMillan, *Symbolic Model Checking: An Approach to the State Explosion Problem*, PhD thesis, Carnegie Mellon University, 1992

[3] E M Clarke D E. Long and K. L McMillan, "Compositional Model Checking", *Proceedings of the Fourth IEEE Symposium on Logic in Computer Science*, June 4-8, 1989, Asilomar, CA.

[4] R. J. Anderson, P. Beame, S. burns, W. Chan, F. Modugno, D. Notkin and J. D. Reese, "Model checking large sofware specifications", In *Proceedings of the 4th ACM SIGSOFT Symposium on the Foundation of Software Engineering*, pages 156-166, 1996.

[5] O Grumberg and D. E. Long, "Model Checking and Modular Verification", *Transactions on Programming Language and Systems*, Vol. 16, No. 2, 1994

[6] D Harel and A Naamad, "The STATEMATE semantics of Statecharts", *ACM Transactions on Software Engineering and Methodology*,5(4):293-333, 1996.

[7] C. Huizing. *Semantics of Reactive Systems. Comparison and Full Abstaction*, PhD thesis. Technische Universiteit of Eindoven, 1991.