모델 체킹에서 그래프 모양의 반례 생성

이태훈 권기현
경기대학교 정보과학부
{taehoon, khkwon}@kyonggi.ac.kr

Generation of Graph-like Counterexamples in Model Checking

Taehoon Lee, Gihwon Kwon
Department of Computer Science, Kyonggi University

요 약

현재의 모델 체커는 모델이 속성을 만족하지 않을 경우 반례를 사용자에게 보여주어서 디버깅을 돕는다. 모델 체커에서 반례는 중요한 장점 중에 하나이지만 대부분의 모델 체커에서 반례로서 하나의 경로만을 보여주게 된다. 하지만 사용자가 원하는 것은 그 이상의 정보를 원할 수 있다. 따라서 반례에서 좀더 많은 정보를 보여줄 필요가 있다. 이런 종류의 연구로서 트리 형식의 반례 생성과 증명 형식의 반례생성이 있었다. 하지만 이 연구들은 시스템이 가질 수 있는 모든 경로를 알아낼 수는 없고 또한 증명 형식의 반례 생성의 경우 상태공간을 다른 형식으로 변경을 해야 한다. 본 논문에서는 반례로서 도달 가능한 모든 경로를 그래프 형식으로 보여줄 수 있는 그래프 형식의 반례를 정의하고 생성방법에 대해서 알아본다.

1. 서 론

모델 체커는 모델과 속성을 받아들여서 모델이 속성을 만족하는지 만족하지 않는지를 자동으로 검사한다. 모델이 속성을 만족할 경우 모델체커는 참을 돌려주고 모델이 속성을 만족하지 않을 경우 모델체커는 반례를 돌려주어서 모델이 왜 속성을 만족하지 않는지를 알려준다. 하지만 지금까지 나와 있는 모델 체커는 반례로서 하나의 경로만을 돌려준다. 또한 이런 하나의 경로는 속성에 대한 모든 반례를 생성하지 못하게 된다. 따라서 반례로서 좀더 많은 정보가 필요하게 된다.

이러한 연구로서 트리 형식의 반례와 증명 형식의 반례가 연구되었다. 트리 형식의 반례는 반례로서 하나의 경로가 아닌 여러 개의 경로를 트리 형식으로 보여준다. 하지만 전체 CTL 이 아닌 ACTL 에 대해서만 적용이 가능하고 모델이 가질 수 있는 모든 경로를 표현하지는 못한다. 증명형식의 반례 생성은 모델체킹에서 사용하는 상태들의 집합을 증명 형식으로 변경 한다. 그래서 증명을 통하여 시스템에 속성을 만족하지 않는지를 알 수 있게 된다. 하지만 증명에 익숙지 않은 일반 사용자가 사용하기에는 너무 난해하다. 따라서 좀더 쉽고 편하게 사용할 수 있는 반례 형식이 필요하게 된다.

현재 CTL 모델 검사는 모델이 속성을 만족하는지를 검사하기 위해 속성을 만족하는 상태들을 검사를 한 다음에 그 상태 들 속에 초기 상태가 있는 지를 검사를 한다. 이것을 정형적으로 표현하면 다음과 같다.

$$M \models \phi \quad \text{iff} \quad I \subseteq [\phi]$$

만일 초기 상태에서 도달 가능한 모든 상태들의 집합을 검사하고 속성을 만족하는 상태들과 동일한 상태들을 추출하면 오류 상태로 갈수 있는 상태들의 집합이 나온다. 이 상태들의 집합을 그래프 형식으로 재구성 하면 시스템이 가질 수 있는 모든 경로를 가지게 된다.

따라서 본 논문 에서는 시스템이 속성을 만족하지 않을 경우 반례로서 그래프 형식의 반례를 생성하여 속성을 만족하지 않는 모든 경로를 보여주고 사용자에게 편하게 표현을 할 수 있는 방법을 제시한다. 본 논문의 구성은 다음과 같다. 2장에서는 기본적인 CTL 모델 체킹에 대해서 살펴본다. 3장에서는 그래프 형식의 반례에 대해서 정의를 하고 4장에서 결론과 향후 연구 과제를 설명한다.

2. 배경 지식

2.1 기존 반례 생성 기법

기존의 모델 체킹 도구들은 속성이 만족할 경우 참을 사용자에게 돌려주고 속성을 만족하지 못할 경우 초기상태에서 왜 속성을 만족하지 못하는지를 보여주게 된다. 이 때 모델 체킹 도구는 초기상태에서 목표상태까지 가는 하나의 경로를 보여주게 된다. 이 방법은 단지 하나의 경로 만을 보여줄수 있다. AG 에 대한 기본적인 반례 생성 방법은 그림 1과 같다.

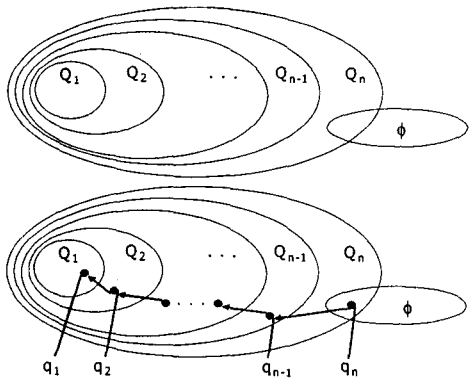

그림 1 AG 에 대한 반례 생성

2.2 트리 모양의 반례 생성 기법

기존 반례 생성 기법이 주는 정보가 부족했기 때문에 좀 더 많은 정보를 주는 연구가 진행이 되었다. 그중에 하나가 CMU에서 제안한 트리 모양의 반례 이다. 트리 형태의 반례를 ACTL 에서만 생성이 가능하기 때문에 ACTL 속성에 대해서 트리 형태로서 반례를 되돌려 준다. 반례 생성 기법은 CMU에서 제안한 자동 추상화 기법에 사용되고 있다. 하지만 이 기법은 모든 CTL 식을 지원하는 것이 아니고 또한 모든 가능한 경로를 보여줄 수가 없다.

2.3 증명 모양의 반례 생성 기법

트리 모양의 반례는 반례가 나왔다고 하더라도 왜 만족하게 되는지 이해가 쉽게 되지가 않는다. 또한 왜 ACTL 이 만족하는지와 왜 ECTL 이 만족하지 않는지에 대한 정보를 제공할 수가 없다. 따라서 이런 속성에 대해서도 모델 체킹 도구에서 증명형태로 반례를 제공하는 방법을 토론토 대학에서 개발했다. 이 방법은 CTL 식을 만족하는 상태들을 증명 모양으로 보여준다. 이 방법의 장점은 기존의 방법이 ACTL 이 왜 만족하는지와 ECTL 이 왜 만족하지 않는지도 보여줄수가 있다. 하지만 이 기법은 증명과 정리증명기에 익숙하지 않다면 사용하기 어려운 단점이 있다.

3. 그래프 형식의 반례 정의

기존의 반례는 하나의 유한 또는 무한의 경로를 되돌려 줬다. 이 경로는 초기 상태에서 속성을 만족하지 않는 상태까지 가는 경로이다. 하지만 이런 종류의 반례는 단 하나의 경로만을 되돌려준다. 하지만 실제 시스템에서는 여러 가지 종류의 속성을 위반하는 상태로 가는 경로가 존재 한다. 따라서 좀더 많은 종류의 정보를 반례를 이용해서 얻어야할 필요가 있다. 또한 기존의 반례는 어떤 한 가지 경로만을 보여줄 수 있기 때문에 ECTL 이 만족하지 않는 것을 보여주는 것이나 ACTL 이 만족하는 것을 보여줄 수가 없었다. 이런 문제점을 해결하기 위해 그래프 모양의 반례를 정의한다.

CTL 식 ϕ 와 모델 M 이 주어 졌을 때 ϕ를 만족하는 상태들의 집합 $\|\phi\|$ 과 초기 상태에서 도달 가능한 상태들의 집합을 구하는 함수 $Reachable_state(I)$가 있다고 할 때 $\phi = Reachable\_state(I) \cap \|\phi\|$ 는 초기상태에서 목표 상태까지 도달 가능한 상태들의 집합이 된다.

그래프 G 는 다음과 같이 정의 한다. $G=(V,E)$ 여기서 V 는 버텍스의 집합을 나타내고 E는 에지의 집합 $E \subseteq V \times V$이다. 버텍스 V 는 다음과 같은 튜플로 구성 되어 있다. $v=(S_v, B, P, S_c)$ 여기서 S_v 는 상태들의 집합, B 는 상태들이 속성을 만족하는지를 나타내는 불린 변수. P 는 이전 버텍스의 집합, S_c는 다음 버텍스의 집합 이다.

최상위 버텍스의 S_v는 모델의 초기 상태와 동일하다 그리고 B 는 모델이 속성을 만족 하면 참이 되고, 만족하지 않으면 거짓이 된다. P는 ε이 된다. 현재 버텍스의 다음 버텍스는 현재 버텍스에 있는 상태의 다음 상태로서 구할 수 있다. 만일 다음 상태가 속성을 만족하는 상태라면 그 상태를 포함하고 있는 버텍스는 B에 true를 가지고 있고 다음 상태가 속성을 만족하지 않는 상태라면 false를 가지게 된다. 이런 방식으로 계속 다름 그래프를 생성하면 전체 도달 가능한 상태에 대해서 현재 상태들의 집합이 속성을 만족하는지를 나타내는 그래프를 생성할 수 있다. 그림 2에서는 예제 모형과 그림 3에서는 이것에 대한 그래프 모양의 반례 예제를 보여준다.

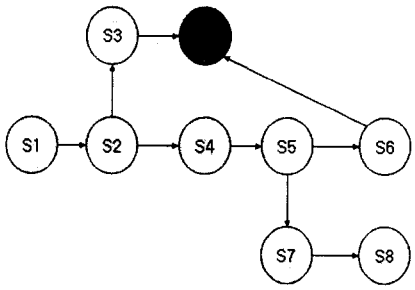

그림 2 예제 모델

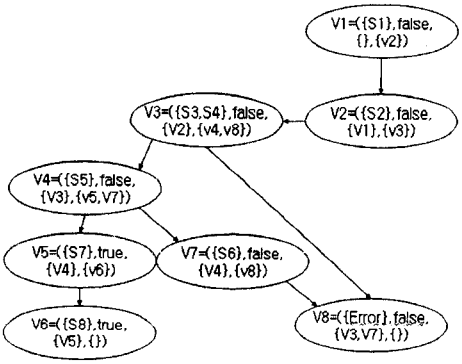

그림 3 예제 모델에 대한 $AG\neg Error$ 의 반례

그림 2 의 간단한 모델을 속성 $AG\neg Error$ 에 대해서 간단한 그래프 형식의 반례를 생성해 보았다. 그림 2에서 보듯이 모델이 가질 수 있는 모든 경로에 대해서 그래프 형식으로 보여주기 때문에 기존의 방법에 비해서 좀더 많은 정보를 보여주게 된다.

모델 체킹에서 AG 에 대한 그래프 모양의 반례를 생성하는 알고리즘은 그림 4와 같다. 그림 4에서 보는 알고리즘을 이용해서 AG 에 대한 그래프 모양의 반례를 생성할 수 있었다. 일반적인 모델 체커는 모델에 대한 모든 행위를 검사한다. 따라서 검사하는 상태들을 재구성하면 모든 행위를 그래프 모양으로 구성을 할 수 있다. 이런 그래프 모양의 반례는 시스템의 모든 경로를 포함하고

있으며 이를 응용하면 많은 분야에 적용을 할 수가 있다.

```
AG(I, φ, V)
φ= Reachable_state(I)∩ [ AGφ ]
α:=Img(S) ∩  [ φ ]
β:=Img(S) ∩  φ
α':=Img(S) ∩  ¬ [ φ ]

repeat
        S=Unchecked_vertex_state()
        if Img(S) ∈ Computed_state then continue
        if α ≠ ε  then add_vertex(α,true)
        if β ≠ ε  then add_vertex(β,true)
        if α' ≠ ε  then add_vertex(α',false)

until All_vertex_checked
```

그림 4. AG 에 대한 반례 생성 알고리즘

4. 결론

기존의 모델 체킹 기술은 반례로서 단 하나의 경로만을 보여주었고, 이 후 개선된 방법에서도 시스템이 가질 수 있는 모든 상태를 검사하지만 시스템이 가질 수 있는 모든 경로를 되돌려 주진 않는다. 그래프 모양의 모델 체킹에서는 시스템의 모든 상태를 검사하고 또한 시스템의 모든 경로를 되돌려 준다. 따라서 시스템이 가질 수 있는 모든 경로를 검사를 할 수 있다. 또한 기존 모델체킹 도구에서 이용하고 있는 상태들의 의미를 그대로 사용하고 있기 때문에 모델 체킹 도구를 이용한 사람이라면 쉽게 접근할 수 있다.

하지만 본 논문에서 제시한 알고리즘은 많은 고정점 계산을 수행하고 따라서 많은 메모리를 필요로 하게 된다. 따라서 좀더 적은 메모리에서 사용가능한 알고리즘을 개발할 필요가 있다. 그리고 아직은 모든 CTL 연산자에서 사용이 가능하지 않다. 따라서 모든 CTL 연산자에서 사용가능하도록 확장할 필요가 있다. 그리고 모든 경로를 되돌려 주기 때문에 사용자가 원하는 경로를 탐색하기가 힘들 수 있다. 따라서 쉽게 경로를 찾을 수 있는 프로그램의 개발이 필요하다.

참고문헌

[1] E.M. Clarke, O. Grumberg, and D. Peled, Model Checking, MIT Press, 1999.

[2] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith., " Counterexample-Guided Abstraction Refinement," in the Proceedings of Computer Aided Verification, pp.154-169, 2000.

[3] P.E. Ammann, P.E. Black, and W. Majurski, " Using Model Checking to Generate Tests from Specifications," in the Proceedings of ICFEM '98, pp.46-54, 1998.

[4] E.M. Clark, Y.Lu, S. Jha, and H. Veith "Tree-Like Counterexamples in Model Checking." In Proceedings of the Seventeenth Annual IEEE Symposium on Logic in Computer Science(LICS'02),page 19-29,

[5] Marsha Chechik, Arie Gufinkel "Proof-Like Counter-Examples",In Proceeding of TACAS'03 LNCS2619, page 160-175.

[6] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, " Progress on the State Explosion Problem in Model Checking," in the Proceedings of 10 Years Dagstuhl, LNCS 2000, pp.154-169, 2000.

[7] E.M. Clarke, O. Grumberg, K.L. McMillan, and X. Zhao, " Efficient Generation of Counterexamples and Witness in Symbolic Model Checking," in the Proceedings of Design Automation Conference, pp.427-432, 1995.

[8] A. Cimatti, E.M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, "NuSMV 2: An OpenSource Tool for Symbolic Model Checking." in the Proceedings of CAV'02, 2002.