SPIN 과 SMV 가 생성하는 반례의 특성 비교

채여경*, 강혜수, 권령구, 권기현
경기대학교 컴퓨터과학과
{codurud729, ssw2, rkay8496, khkwon}@kyonggi.ac.kr

Comparison of Counter-Examples Generated by Model Checkers SPIN and SMV

Yeogyeong Chae*, Hyesu Kang, Ryoungkwo Kwon, Gihwon Kwon
Department of Computer Science, Kyonggi University

요 약

모델 검증은 시스템이 만족해야 하는 속성을 자동으로 검사하는 정형 검증 기법으로써, 많은 도메인에서 활용되고 있다. 특히 모델 검증 도구들에 따라 상태 공간 탐색 방식이 다르고, 상태 공간 탐색 방식에 따라서 생성되는 반례도 달라진다. 본 논문에서는 모델 검증의 대표적인 도구인 SPIN 과 SMV 에서 생성하는 반례를 상호 비교한다.

1. 서론

모델 검증은 시스템이 만족해야 하는 속성을 정형적으로 검사하는 기법으로써, 모델 검증을 위한 많은 자동화된 도구들이 개발 되어 많은 분야에서 사용되고 있다. 모델 검증의 과정을 단순히 이야기하면, 먼저 검증 대상 시스템을 유한 상태 기계로 모델을 구축하고 만족해야 하는 속성을 CTL(Computational Tree Logic) 또는 LTL(Linear Temporal Logic)과 같은 논리언어를 이용하여 논리식을 작성해야 한다. 그리고 모델 검증 도구는 모델과 속성을 입력으로 받아들여 모델 상에서 속성이 만족되는지를 확인한다. 일반적으로 관심 있는 모델 검증의 결과는 속성이 위반되는 경우로써, 이러한 경우에 모델 검증 도구는 속성이 만족되지 않는 반례(counter example)를 생성 한다. 반례는 모델이 포함하고 상태들의 경로이며, 우리는 이것을 확인하여 모델이 어떠한 경로를 통해 속성이 위반되는지를 확인하여 모델을 수정할 수 있다[1].

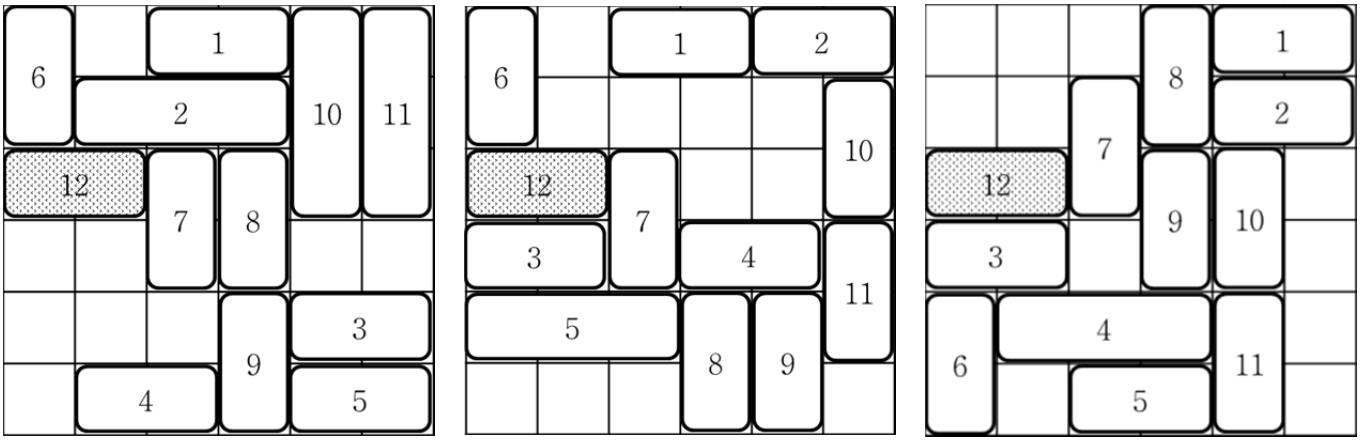
대표적인 모델 검증 도구로써 SPIN[2]과 SMV[3]가 있다. SPIN 은 속성을 LTL 로 작성하는 모델 검증 도구로써, 프로토콜 또는 다중 실행 같이 비동기적이고 병행 실행의 특성을 가진 시스템을 대상으로 하고 있다. SMV 는 CTL 을 사용하는 모델 검증 도구로써, 동기적인 특성을 갖는 시스템을 대상으로 한다. 두 모델 검증 도구의 큰 차이점은 생성하는 반례의 특성이 다르다는 것이다. SPIN 에서 생성한 반례의 길이는 SMV 에서 생성한 반례의 길이보다 상대적으로 길지만, 반례를 얻기 위하여 모델의 상태 공간을 탐색하는 시간은 매우 짧다고 할 수 있다. 반면 SMV 는 최단 경로의 반례를 생성 한다. 이것에 대한 자세한 설명은 이 후에 보일 것이다. 이와 같이 모델 검증 도구가 생성한 반례의 특성에 따라 시스템을 검증할 때 적합한 도구를 사용할 필요가 있다.

모델 검증 기법을 다양한 분야에 적용할 수 있는데 그 중에서 게임 풀이에 관련된 연구들이 많이 있었다[4][5]. 이러한 연구들에서는 게임의 구성요소들과 규칙들을 모델로 구축하고 게임을 풀기 위한 목표를 속성으로 작성 한다. 하지만 게임을 풀기 위해서는 반례가 필요하므로 속성을 부정을 한 형태이다. 다시 말해, 모델 검증 도구는 부정된 속성을 만족하지 않는 반례를 생성하게 되고 부정하지 않은 속성, 즉 게임의 목표를 달성할 수 있는 상태들의 경로를 포함한다. 하지만 어떠한 모델 검증 도구를 사용할 것이냐는 게임의 특성과 어떠한 형태의 반례를 원하는지에 따라 다를 것이다. 마찬가지로 게임 외 다양한 도메인에 모델 검증 도구를 사용하려 할 때 도메인의 특성을 무시한 채 모델 검증 도구를 사용한다면, 적합한 결과를 얻지 못할 수도 있다.

본 논문에서는 SPIN 과 SMV 에서 반례를 생성하는 개념을 소개하고 GetOut[6]이라는 게임을 각 모델 검증 도구로 반례를 생성하여 비교 한다. 또한, SMV 가 찾은 반례를 참고하여 SPIN 의 상호 작용 시뮬레이션으로 동일한 결과를 얻음을 보인다. 궁극적으로는 각 모델 검증 도구가 생성하는 반례의 관점에서 각 도구의 특성을 기술 함으로써, 도구선택을 돕고자 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 SPIN 과 SMV 의 상태 공간 탐색 방법을 소개하고 그에 따라 생성되는 반례를 비교 한다. 3 장에서는 SPIN 의 상호 작용 시뮬레이션을 이용하여 SPIN 이 SMV 와 동일한 반례를 확인할 수 있음을 보임으로써 SPIN 과 SMV 의 특성을 기술 한다. 마지막 4 장에서 결론 및 요약을 통해 본 논문을 마무리 한다.

본 연구는 지식 경제부의 정보통신 산업 진흥원사업의 일환으로 수행하였음.[2012-0043, 컴퓨터과학 전문프로그램]



(a) 1501 (b) 1750 (c) 2000

(그림 1) 실험 대상 스테이지들의 게임 구성

2. SPIN 과 SMV 의 반례 생성

모델 검증 도구는 대표적으로 SPIN 과 SMV 가 있다. 이러한 모델검증 도구는 검증 대상 시스템을 묘사하는 모델과 속성을 입력으로 하여 속성이 모델 상에서 만족하는 지를 검사 한다. 만약 속성이 만족되지 않는다면, 모델이 포함하는 상태들의 시퀀스인 반례를 자동으로 생성 한다. 하지만, 두 가지 도구들 사이에는 상태 공간을 탐색하는 방식에 차이가 존재하기 때문에 반례의 특성이 달라진다.

모델의 상태 공간을 트리 형태로 표현하였을 때, 각 도구의 가장 큰 차이점은 상태 공간의 탐색 방식이며, 깊이 우선 탐색과 너비 우선 탐색 방식에 따라 나뉘어 진다.

SPIN 은 깊이 우선 탐색 방식(depth first search)을 이용 한다. 따라서, 탐색 도중 속성을 위반하는 첫 번째 경우를 발견하게 되면 해당하는 상태들의 시퀀스인 반례를 생성하게 된다. 그렇기 때문에, 반례를 발견하기까지의 탐색 시간은 너비 우선 방식에 상대적으로 짧을 수 있다. 하지만, 모델 상에 속성을 위반할 수 있는 반례가 하나 이상 존재한다고 가정 하였을 때, 현재 찾은 반례의 길이 보다 짧은 반례가 존재 할 가능성 있다.

SMV 는 너비 우선 탐색 방식(breadth first search)을 이용 한다. 따라서, SPIN 과는 달리 첫 번째 반례를 발견하기 위한 탐색 시간은 상대적으로 길 수 있다. 하지만, SMV 는 해당 모델 상에서 가장 짧은 길이의 반례를 생성할 수 있다.

각 모델 검증 도구에서 상태 공간 탐색 방식의 차이로 인하여 반례를 생성하는 것에 차이가 존재하는데, 모델 검증 도구에서 생성하는 반례를 게임 풀이에 활용할 때 이 차이는 명확해 진다. 게임을 모델링 하고 게임의 풀기 위해 달성해야 하는 최종 목적을 속성으로 작성 한다. 단, 속성을 논리적으로 부정하여 입력을 하는데, 부정한 속성이 모델상에서 위반한다는 것은 게임을 풀 수 있는 반례(게임을 시작하여 최종 목적까지 도달할 수 있는 상태들이 경로)를 얻을 수 있다는 것을 의미 한다. 경우에 따라서는 게임을 풀 수 있음이 중요할 수 있지만, 소코반이나 푸시 푸시 게임과 같은 게임을 풀기 위해 몇 번의 스텝이 필요했는지가 아주 중요할 수 있다. 따라서, SPIN 에서는 짧은 시간에 반례를 얻을 수 있는 반면, 그것이 최적의 반례를 보장하지 못할 수 있다. 반대로 SMV 에서는 최적의 반례를 생성할 수 있지만 모델의 상태 공간에 따라 계산 시간은 많이 소모 될 수 있다.

GetOut 게임을 간단히 설명 하면 6 행 6 열의 사각형 공간에 둘 이상의 가로 또는 세로 블록들을 포함하고 있다. 그리고 특정 블록을 정해진 위치까지 이동을 하는 것이 게임의 목표이다. 이 게임은 4 가지 난이도의 총 2000 개의 스테이지를 포함하며 본 논문에서는 최고 난이도의 첫 번째 스테이지 1501, 중간 1750, 마지막 2000 스테이지를 모델 검증 도구로 게임을 풀 수 있는 반례를 생성 하였다. 단, 각 모델 검증 도구에서 게임을 모델링한 방법과 생성된 반례의 자세한 정보는 본 논문의 범위를 벗어나므로 생략 한다. 아래 그림 1 은 실험 대상 스테이지들의 실제 게임 구성을 보여주고 있다. 각 블록은 가로 또는 세로 방향일 수 있고, 블록의 길이는 2 또는 3 일 수 있다. 게임의 목표는 12 번 블록이 3 행 6 열까지 이동하는 것이다.

아래의 표 1 은 GetOut 게임을 SPIN 과 SMV 를 이용하여 반례를 생성하고 그것의 결과를 보여주고 있다.

<표 1> SPIN, SMV 로 생성한 GetOut 게임풀이 결과

	SPIN		SMV	
스테이지	시간	길이	시간	길이
1501	0.05”	9336 스텝	12”	58 스텝
1750	0.02”	3792 스텝	20”	58 스텝
2000	0.01”	2006 스텝	3’ 55”	66 스텝

앞서 설명한 것처럼 SPIN 과 SMV 의 상태 공간 탐색에 따라 반례의 특성이 다르다고 하였다. 표 1 에서 보면 SPIN 에서는 각 스테이지의 반례를 생성하기 위해 필요한 계산 시간은 1 초도 안될 만큼 빨랐으나 반례의 길이는 SMV 의 그것보다 너무나 길

었다. 실제 SPIN 이 생성한 반례대로 게임을 진행하여 게임을 풀 수 있었으나 1501 스테이지의 경우 반례를 통해 게임을 푸는 것은 너무 많은 시간이 필요하였다. 또한, 동일한 두 위치를 블록이 반복해서 움직이는 것과 같이 게임을 푸는데 있어서 불필요한 움직임들이 상당수 포함되어 있었다. 반대로 SMV 가 생성한 반례는 모델 상에서 가장 짧은 경로의 반례를 생성하여 SPIN 에서 그것의 길이보다 현격히 짧았으며, 해당 반례는 게임을 해결하기에 유용하였다. 궁극적으로 이러한 특성은 모델 검증 도구의 상태 공간 탐색 방식에 기인한다.

3. 상호 작용 시뮬레이션

SPIN 과 SMV 는 상태 공간을 탐색하는 방식에 차이가 존재하기 때문에, 반례의 특성이 달랐다. 반례 길이의 관점에서 SPIN 은 상대적으로 허용할 수 없을 만큼 길었는데, 그렇다면 SPIN 의 입력 모델의 상태 공간에는 SMV 와 같은 최적의 반례가 존재하는지에 대해 확인할 필요가 있었다.

SPIN 에서 반례를 찾기 위해 무작위(random) 시뮬레이션, 상호 작용(interactive) 시뮬레이션, 가이디드(guided) 시뮬레이션, 검증(verify)과 같이 총 4 가지 실행 방법을 제공 한다. 이 중 검증은 모델과 속성을 입력 받아 자동으로 반례를 생성하는 것으로써, 앞서 표 1 과 같은 실험을 수행할 때 사용한 방법이다. SPIN 은 상태 공간에서 특정 상태에서 전이 가능한 상태들이 둘 이상 존재할 때, 비 결정적으로 다음 상태를 선택하여 전이 한다. 이러한 점은 SPIN 이 다수의 프로세스들이 존재하는 병행 실행의 특성을 가지는 시스템이 주요 대상을 이기 때문이다.

상호 작용 시뮬레이션은 검증과 달리 둘 이상의 전이 가능한 상태 중 하나를 사용자가 직접 선택할 수 있다. 내가 원하는 상태로 전이 가능하다는 것은 GetOut 게임에서 내가 원하는 블록을 매번 움직이게 할 수 있다는 의미이다. 단, 이것을 통해 확인할 수 있는 것은 반례와 다소 다른 개념이지만, 반례 그 자체도 상태들의 경로이기 때문에 최적의 게임 풀이를 위해서는 의미적으로 동일하다고 할 수 있겠다.

우리는 SMV 에서 생성한 최적의 반례를 따라 SPIN 의 상호 작용 시뮬레이션에서 동일한 반례를 얻을 수 있는지 확인 하였다. 방법은 다음과 같았다. SMV 에서는 각 블록들은 숫자 값을 가지는 변수들이지만, SPIN 에서는 각 블록은 하나의 프로세스로 모델링 되었다. 따라서, SMV 의 반례에서 특정 블록이 움직였다는 것을 대응하는 프로세스를 실행하였다는 것으로 고려하여 선택하였다. 스테이지 1750 에서 상호 작용 시뮬레이션의 첫 번째 스텝에서 전이 가능한 상태들은 다음과 같다.

- 12 RedBlock 80
- 11 VerticalBlock 103
- 7 VerticalBlock 103
- 1 HorizontalBlock 61

SMV 의 반례를 보면 첫 스텝에서 그림 1 의 스테이지 1750 에서 1 번 블록이 왼쪽으로 한 칸 움직였다. 따라서, 위의 4 가지 전이 가능한 상태 중 1 번 프로세스가 왼쪽으로 움직이게 하는 4 번째 상태로 전이하도록 선택할 수 있다. 이러한 방식으로 SMV 의 반례와 동일하게 블록을 움직이게 하였다.

우리는 세 개의 스테이지를 상호 작용 시뮬레이션으로 SMV 와 동일한 반례를 얻을 수 있었다. 이러한 결과는 SPIN 의 모델에서도 게임을 풀기 위해 SMV 와 동일한 상태들의 경로를 포함하고 있음을 확인하는 동시에, SPIN 과 SMV 의 상태 공간 탐색의 방식 차이를 명확히 보여주는 것이다.

4. 결론

본 논문에서는 대표적인 모델 검증 도구인 SPIN 과 SMV 가 생성하는 반례를 통해 각 도구가 어떠한 특성을 가지고 있는지를 설명 하였다. 각 도구가 생성하는 반례의 차이는 도구가 상태 공간을 어떻게 탐색하는지에 달려 있다. SPIN 은 깊이 우선 탐색 방식, SMV 는 너비 우선 탐색 방식을 이용하고 있다. 그러한 방식들의 근본적인 특성에 따라 SPIN 은 반례를 짧은 시간에 발견할 수 있었지만, 반례의 길이 관점에서 최적이라고 할 수 없었다. 반대로 SMV 는 반례를 발견하는 시간은 상대적으로 길었지만, SPIN 보다 훨씬 최적의 반례를 얻어낼 수 있었다.

또한 GetOut 이라는 게임을 통하여 이 특성을 더욱 명확히 밝혔다. 동일한 스테이지라도 SPIN 은 게임 풀이를 빠르게 해내었지만, SMV 보다 반례의 길이도 길고 불필요한 블록의 움직임을 가지는 것을 생성하였기 때문에 특정 스테이지에서는 해당 반례로 게임을 직접 풀기 위해 비효율적 이었다. 그에 반해 SMV 의 반례는 계산 시간이 상대적으로 오래 걸렸지만 최적의 게임 풀이를 얻을 수 있었다.

게임을 풀이하는데 있어 풀 수 있는지가 중요하다면 SPIN 을 이용하여 빠른 시간에 원하는 답을 얻을 수 있고, 최적의 게임 풀이를 얻는 것이 중요하다면 SMV 를 이용할 수 있을 것이다. 즉, 이러한 모델 검증 도구의 특성을 비교함에 따라 특정 도메인에 모델 검증 도구를 사용 할 때 적합한 도구를 선택하는데 도움이 될 것이다.

참고문헌

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

[2] G. H. Holzmann, “The model checker SPIN”, IEEE Transcactions on Software Engineering, vol.23, no.5, pp.279-295, 1997.

[3] K. Mcmillan, “Symbolic Model Checking”, Kluwer Academic Publishers, 1993.

[4] P. Madhusudan, W. Nam, R. Alur, “Symbolic Computational Techniques for Solving Games”, Workshop on Bounded Model Checking, 2003.

[5] S. Park and G. Kwon, “Bounded model checking for LTS”, In the Proceedings of KSEJW, vol. 5, no. 1, 2007.

[6] GetOut, https://itunes.apple.com/kr/app/naganeungilchajgi-get-out/id441476515?mt=8