추상화를 통한 모델의 축소 : 네모라이즈 게임 사례 연구

이 정 림[†] · 권 기 현[††]

요 약

주어진 상태가 도달 가능한지를 판정하는 작업은 모델 체킹과 같은 유한 상태 모델의 분석에서 필수적이다. 만일 모델의 크기가 작다면 상태 공간을 전부 조사해서 판정할 수 있지만, 크기가 큰 경우에는 전체 상태 공간을 조사하는 것이 어렵거나 불가능할 수 있다. 이런 경우, 도달성 여부를 조사하기 전에 추상화를 통해서 모델을 축소해야 하며 이때 사용된 추상화는 false positive 오류를 허용하지 않는다. 본 논문에서는 이러한 추상화를 고안하여 네모라이즈 게임 풀이에 적용하였다. 그 결과, 상태 폭발 문제로 풀 수 없었던 큰 규모의 문제를 추상화를 통해서 해결할 수 있었다.

키워드 : 유한 상태 모델, 도달성 분석, 추상화, 상태 공간 탐색

Model Reduction with Abstraction : Case Study with Nemorize Game

Junglim Lee[†] · Gihwon Kwon[††]

ABSTRACT

Given a state, it is essential to for the finite state model analysis (such as model checking) to decide whether or not the state is reachable. If a size of the model is small, the whole state space is to be explored exhaustively. However, it is very difficult or even impossible if a size of the model is large. In this case, the model can be reduced into a smaller one via abstraction which does not allow the false positive error. In this paper, we devise such an abstraction and apply it to the Nemorize game solving. As a result, unsolved game due to the state explosion problem is solved with the proposed abstraction.

Key Words : Finite State Model, Reachability Analysis, Abstraction, State Space Traversal

1. 서 론

유한 상태 모델은 상태와 전이로 구성된다. 전이는 상태 간의 움직임을 나타내며, 초기 상태에서 '시작하여 0번 이상의 전이를 통해서 도달될 수 있는 상태를 도달 가능 상태라고 부르며 그때까지의 경로를 도달 경로라고 한다. 주어진 상태가 도달 가능인지 또는 도달 불능인지를 판정하는 작업은 모델 체킹(model checking)과 같은 유한 상태 모델의 분석에서 필수적이다[1]. 예를 들어 *unsafe*는 에러를 나타내는 상태라고 가정하자. 만약 *unsafe* 상태가 도달 불능이라면, 해당 모델은 안전하다고 말할 수 있다. 왜냐하면 시스템이 어떤 행위를 취해도 에러 상태에 결코 도달할 수 없고, 바꾸어 말해서 안전한 상태에 항상 머물기 때문이다.

상태의 도달성 여부를 판정하기 위해서는 모델의 상태 공간을 전부 조사하면 된다. 만일 모델의 크기가 작다면 상태 공간을 전부 조사하는 것이 가능하지만, 크기가 큰 경우에는 전체 상태 공간을 조사하는 것이 어렵거나 불가능할 수 있다. 이런 경우, 도달성 여부를 조사하기 전에 추상화를 통해서 모델을 축소해야 한다[2]. 그러나 원본 모델이 아닌 축소된 추상 모델을 조사하기 때문에 다음과 같은 오류가 발생할 가능성이 있다:

- false positive 오류 : 추상 모델에서는 도달 가능이지만, 원본에서는 도달 불능인 경우
- false negative 오류 : 추상 모델에서는 도달 불능이지만, 원본에서는 도달 가능인 경우

이상적으로는 두 가지 오류가 모두 발생하지 않도록 모델을 추상화하는 것이다. 그러나 이러한 경우 바이시뮬레이션(bisimulation)처럼 모델의 큰 축소를 기대하기는 어렵다[3]. 모델의 크기를 크게 축소하려면, 사용된 추상화는 일부 오류를 허용해야 한다. 어떤 오류를 허용하고 또는 불허할 것인지를 결정하는 것은 문제에 따라 다르다. 만일 상태의 도달 가능에 관심이 있다면 false negative는 허용하지만 false positive는 불허하는 추상화를 사용해야 한다. 만일 상태의

※ 본 연구는 한국과학재단 특정기초연구(R01-2005-000-11120-0) 지원으로 수행되었음.
† 정 회 원 : 경기대학교 전자계산학과 석사과정
†† 종신회원 : 경기대학교 정보과학부 교수
논문접수 : 2005년 7월 7일, 심사완료 : 2005년 12월 13일

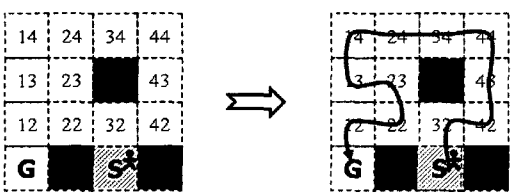
도달 불능이 관심이라면 false positive는 허용하면서 false negative는 불허하는 추상화를 사용해야 한다.

본 논문에서는 추상화를 통해서 모델의 크기를 축소하는 것과 추상 모델을 이용해서 상태의 도달 가능을 판정하는데 관심이 많다. 이러한 경우, 추상 모델에서 얻은 도달 가능 결과는 원본에서도 보존되어야 하기 때문에 사용된 추상화는 false positive를 불허해야 한다. 만약 그렇지 않으면 추상 모델에서 얻은 결과가 원본 모델에서는 유효하지 않는 오류가 발생된다. 이와 같은 유형의 추상화는 궁극성 속성의 검증[4], 안전성 속성이 위반된 경우에서 반례 생성[5], 게임에서 풀이 경로 찾기[6] 등 다양한 분야에 활용될 수 있다. 본 연구에서는 이러한 추상화 기법을 연구하기 위한 실험 대상으로 네모라이즈 게임을 선택했다. 네모라이즈에서의 풀이 경로(게임의 목표 상태로의 도달 경로)를 찾기 위해서는 게임의 상태 공간을 전부 조사해야 한다. 게임의 크기가 작은 경우, 전체 상태 공간을 조사해서 목표 상태로의 도달 경로(즉, 풀이 경로)를 찾아냈다. 그러나 크기가 큰 게임의 경우 상태 폭발 문제(state explosion problem) 때문에 전체 상태 공간을 탐색할 수 없었다[7]. 그래서 위에서 언급한 유형의 추상화를 고안하여 게임을 축소했다. 그 결과 원본 모델에서는 찾을 수 없었던 풀이 경로를 추상 모델에서 찾아내었다.

논문의 구성은 다음과 같다. 2장에서는 실험 대상으로 선정한 네모라이즈 게임의 설명과 풀이 경로를 구하는 과정을 설명한다. 게임의 크기가 큰 경우 상태 폭발 문제가 발생되는데, 이를 해결하기 위한 추상화 기법을 3장에서 제시한다. 4장에서는 추상화를 이용해서 크기가 큰 게임을 풀었던 경험과 결과를 소개하고, 5장에서 결론 및 향후 연구 과제를 기술한다.

2. 문 제

본 논문에서는 추상화 기법을 연구하기 위한 실험 대상으로 네모라이즈 게임을 선정했다. 이 게임은 핸드폰에 탑재되어 이용되고 있으며, 아래 그림에서 보듯이 그래프의 한붓 그리기 게임처럼 플레이어가 시작 지점(S로 표시된 지점)을 출발해서 이동가능한 모든 지점을 한번만 거쳐서 목적 지점(G로 표시된 지점)에 도달되는 경로를 찾는 일종의 도달성 게임이다. 게임이 갖는 상태 공간이 적지 않기 때문에 이러한 게임을 대상으로 추상화 기법을 연구하는 것과 추상 모델을 이용하여 상태의 도달 가능을 연구하는 것이 흥미롭다. 또한 여기에서 연구된 추상화 기술은 소프트웨어 검증에도 사용될 수 있을 것이다.



(그림 1) 네모라이즈 게임

네모라이즈와 같은 도달성 게임은 대부분 유한 상태 모델 $M=(S, A, R, init, goal)$로 표현된다. 여기서 S는 상태들의 집합, $A=\{left, right, up, down\}$는 플레이어가 매 단계에서 선택할 수 있는 움직임의 집합, $R: S\times A\rightarrow S$은 현재 상태와 플레이어가 선택한 움직임을 받아서 다음 상태를 돌려주는 상태 전이 함수이다. 그리고 $init\in S$과 $goal\in S$은 각각 게임의 초기 상태와 목표 상태를 나타낸다. 게임을 풀기 위해서는 주어진 목표 상태가 도달 가능한지 그리고 도달 가능하다면 도달 경로를 구해야 한다.

목표 상태의 도달성 여부를 결정하기 위해서 인공 지능 또는 모델 체킹 기법 등이 사용될 수 있다. 본 연구에서는 탐색 기법 보다는 모델의 추상화에 관심이 더 많기 때문에 후자인 모델 체킹을 사용한다. 모델 체킹으로 목표 상태의 도달성을 결정하기 위해서 목표 상태로의 도달성을 시제 논리식으로 표현하면 다음과 같다(시제 논리식의 구문과 의미는 [8]에 상세히 설명되어 있다. 여기서는 논문 전개에 필요한 부분만 언급할 것이며 나머지 사항은 해당 문헌을 참조하기 바람).

$$EF\,goal$$

여기서 E는 "경로가 존재한다(Exist)"를 나타내며, F는 "언젠가 또는 미래(Future)"를 나타내는 시제 연산자이다. 그러므로 위의 식의 의미는 "언젠가는 $goal$로 도달되는 경로가 존재한다"이다. 주어진 게임 모델에 대해서 위의 논리식을 검사함으로서 목표 상태의 도달 가능여부를 판정할 수 있다. 모델 체킹이 $EF\,goal$을 검사한 결과로 참을 출력한다면, $goal$ 상태는 도달 가능이다. 왜냐하면 $goal$로의 도달 경로가 존재하기 때문이다. 이러한 도달 경로를 얻기 위해서 다음과 같이 $EF\,goal$의 쌍대(dual)를 사용한다.

$$AG\neg goal$$

여기서 $\neg$는 부정 기호이며, A는 "모든(All) 경로"를 나타낸다. 그래서 $AG\neg goal$ 식의 의미는 "어떠한 시도를 해도 목표 상태에 도달할 수 없음"이다. 만약 $goal$에 도달되는 경로가 하나라도 있다면 $AG\neg goal$의 모델 체킹 결과는 거짓이며, 이 경우 모델 체킹은 $goal$로의 도달 경로를 반례로 출력한다. 이때 생성된 반례가 곧 게임의 풀이 경로이다.

예로서, 모델 체킹 도구인 SMV(Symbolic Model Verifier, [9])을 사용하여 (그림 1)의 네모라이즈 게임을 위에서 제안한 방식대로 풀이한 결과가 아래에 있다. 반례로 생성된 경로 *up, right, up, up, left, left, left, down, right, down, left, down*는 게임의 풀이 경로로서, 플레이어는 시작 지점에서 출발해서 12개 지점을 한 번씩만 통과한 후에 목표 지점에 도달한다. 모델 체킹으로 게임을 풀기 위해서, 유한 상태로 기술된 게임 모델을 SMV 입력 언어로 변환했으며 반례를 얻기 위한 목적으로 시제 논리식은 $AG\neg goal$로 명세했다.

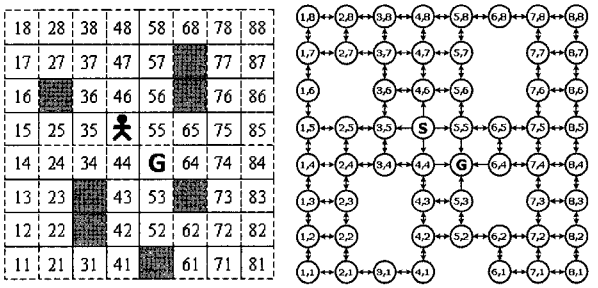
MoveToUp	1	0	1	1	0	0	0	0	0	0	0	0	-	
move	up	right	up	up	left	left	left	down	right	down	left	down	-	
s	31	32	42	43	44	34	24	14	13	23	22	12	11	

(그림 2) 모델 체킹 도구 SMV로 구한 게임 풀이 경로

〈표 1〉 여러 모델 체킹 도구로 네모라이즈 1판을 풀이한 결과

	모델 체킹 도구	SMV	SPIN		Esterel
	탐색 방법	너비우선탐색	깊이우선탐색	너비우선탐색	너비우선탐색
설명	탐색 방향	역방향	정방향	정방향	역방향
	상태 공간 표현	묵시적	정방향	정방향	묵시적
결과	시간(Sec)	00:01<	00:01<	00:01<	00:01<
	메모리(MB)	1.1	2.6	2.3	11.2

본 논문에서는 게임을 풀기 위해서 SMV 뿐만 아니라 SPIN[10], Esterel[11] 등 다양한 모델 체킹 도구를 사용했다. 사용된 모델 체킹 도구의 간단한 설명과 함께 (그림 1)의 네모라이즈 1판을 풀이하는데 사용된 메모리 사용량과 소요된 시간이 <표 1>에 있다. 다행히도 게임이 복잡하지 않아서 모델을 축소하지 않고서도 세 가지 모델 체킹 도구 모두 1초 이내에 풀이 경로를 찾아냈다.



(그림 3) 추상화없이 풀 수 없었던 네모라이즈 10판과 플레이어의 이동을 나타내는 그래프

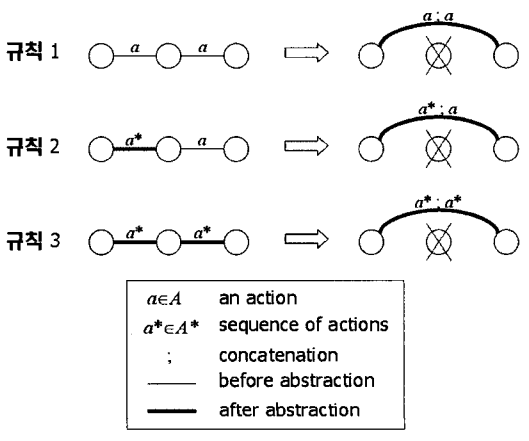
3. 상태 공간 축소를 위한 추상화

전 장에서 살펴본바와 같이 게임의 크기가 작은 경우에는 추상화를 적용하지 않고서도 풀이 경로를 쉽게 찾아냈다. 그러나 크기가 큰 게임의 경우에는 상태 폭발 문제가 발생하기 때문에 풀이 경로를 찾는데 소요되는 시간이 크게 증가하거나 또는 불가능 할 수 있다. 이러한 경우 모델의 크기를 축소하기 위해 추상화를 적용해야 한다. 예를 들어 (그림 3)에 있는 네모라이즈 10판은 2^{65}의 상태 공간을 갖기 때문에 추상화를 통해서 상태 공간을 축소하지 않고서는 풀이 경로를 찾을 수 없었다.

1장에서 언급한바와 같이, 목표 상태의 도달 가능을 판정하는데 사용되는 추상화는 false positive 오류를 허용하지 않는다. 따라서 축소된 추상 모델에서 목표 상태에 도달 가능하면, 원본 모델에서도 당연히 도달 가능하다. 만약 false positive 오류를 허용한다면, 추상 모델에서 목표 상태에 도달 가능해도 원본 모델에서는 도달 불능인 경우가 존재하기 때문에 추상 모델에서 작업하는 것이 무의미하다.

게임 풀이에서 요구되는 위와 같은 추상화를 얻기 위해서, 네모라이즈 10판을 분석해보자. 게임에서 플레이어의 이동을 (그림 3)과 같이 양방향 그래프로 나타내보자. 이 게임은 한 붓 그리기와 같이 지나온 지점을 재방문할 수 없기 때문에, 차수가 2인 지점은 다른 곳으로 빠져나갈 수 없이 한 방향으로만 외길을 지나야 한다. 예를 들어 차수가 2인 지점 (1, 6)을 살펴보자. 지점 (1, 5)에서 위로 올라가면 다른 곳으로 빠져나갈 수 없을 뿐만 아니라 지나온 지점을 재방문할 수 없기 때문에 반드시 (1, 6)을 거쳐서 (1, 7)에 도달할 것이다. 반대 방향도 마찬가지이다. 그래서 중간 경유 지점인 (1, 6)을 제거한 후에 (1, 5)와 (1, 7)을 직접 연결할 수 있다. 또한 (1, 8)의 경우도 마찬가지이기 때문에 (1, 8)을 제거한 후에 (1, 7)과 (2, 8)을 직접 연결하면 된다. 이러한 관찰을 통해서 차수가 2인 지점은 삭제 가능하며 이것을 일반화한 것이 (그림 4)의 규칙 1이다. 그림에서 가는 실선은 추상화 이전의 움직임을 나타내며 단일 움직임($a \in A$)이 실선위에 표시되어 있다. 반면에 굵은 실선은 추상화 이후의 움직임을 나타내며 움직임의 시퀀스($a^* \in A$)가 실선위에 표시되어 있다.

다른 추상화 규칙을 위해서 (1, 7)과 (2, 8)의 연결을 살펴보자. 이미 (1, 7)과 (2, 8)은 추상화되었기 때문에 (1, 7)에서 위로 올라가면 반드시 (2, 7)에 도달한다. 따라서 (2, 8)을 축소할 수 있다. 이것을 일반화한 것이 규칙 2이다.



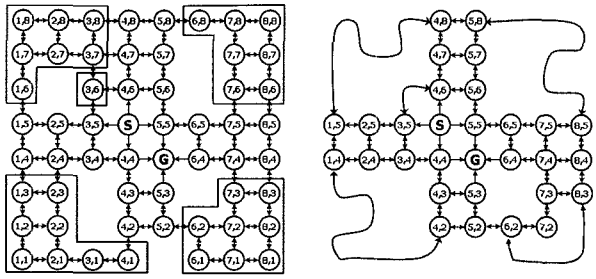
(그림 4) 추상화 규칙과 범례

뿐만 아니라 (1, 5)와 (1, 7)이 추상화되었고 (1, 7)과 (2, 7)이 추상화되었기 때문에 (1, 5)에서 위로 올라가면 (3, 7)에 도달한다. 따라서 중간 경유 지점인 (1, 7)을 제거할 수 있다. 이것을 일반화한 것이 규칙 3이다.

위의 규칙을 규칙 1, 규칙 3, 그리고 규칙 2의 순서대로 점진적으로 반복 적용하면 (그림 5)와 같이 축소된 플레이어의 추상 그래프를 얻는다. 그 결과 플레이어가 이동할 지점이 총 57개에서 30개로 거의 절반 정도로 줄어들었다. 네모라이즈 게임을 모델링할 때 각각의 지점은 방문 또는 미방문의 두 가지 상태를 갖기 때문에 부울 변수로 표현된다. 따라서 각각의 지점을 표현하기 위해서 30개의 부울 변수가 사용된다. 또한 플레이어가 이동할 수 있는 지점 집합은 열거형으로 표시되는데 30개의 지점을 표현하기 위해서 5개의 부울 변수가 필요하다. 뿐만 아니라 매 단계에서 플레이어가 선택할 수 있는 움직임 수가 4이기 때문에 2개의 부울 변수가 필요하다. 따라서 축소된 추상 그래프를 활용한다면, 모델 체킹에 사용되는 모델의 크기를

$$2^{30} \times 2^5 \times 2^2 = 2^{37}$$

로 축소할 수 있다.



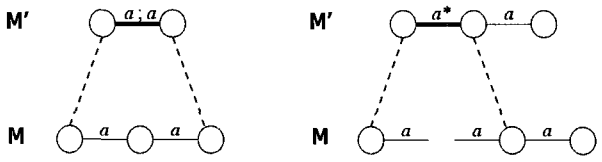
(그림 5) 규칙을 반복 적용해서 얻어낸 추상화된 이동 그래프

축소된 추상 모델을 모델 체킹해서 얻은 결과는 원본 모델에서도 보존되어야 한다. 다시 말해서 추상 모델에서 게임을 풀 수 있는 도달 경로를 찾았다면, 그 경로가 원본 모델에도 반드시 존재해야 한다. 이러한 조건이 성립되어야만, 원래 모델 대신 추상 모델을 사용한 것이 의미가 있기 때문이다. 따라서 추상화가 false positive를 보이지 않음을 입증해야 한다.

이것은 아래와 같이 귀납법으로 증명할 수 있다. 즉 추상 모델에서 구한 도달 경로가 원본 모델에서도 그대로 모방될 수 있음을 보여야 한다. 지금까지 살펴본 바와 같이 추상 모델에서는 지점이 제거되면서 플레이어의 움직임이 결합되었다. 따라서 추상 모델에서 상태 전이 함수 R'를 다음과 같이

$$R' : S \times A* \rightarrow S$$

정의할 수 있다. 즉 상태 전이를 일으키는 것이 단일 움직임이 아닌 움직임의 시퀀스이다. 따라서 추상 모델의 시작 지점에서 여러 번의 움직임을 통해서 다음 지점으로 이동했



(그림 6) 귀납법에 의한 증명 : 기본 경우(좌), 귀납 경우(우)

다면, 원본 모델에서도 같은 지점에서 시작해서 동일한 움직임에 의해서 이동된 다음 지점이 같음을 보여야 한다. 추상 모델에서 움직임이 유한 길이의 시퀀스이기 때문에 (그림 6)과 같이 귀납법으로 증명할 수 있다. 그림에서 M과 M'은 각각 원본 모델과 추상 모델을 나타낸다.

먼저 시퀀스의 길이가 2인 기본 경우(basis case)를 살펴보자. 축소 모델에서 차수 2인 지점이 제거되면서 두 개의 움직임이 결합된 시퀀스를 원본 모델에서 그대로 따라 할 수 있음을 보이기 위해서는 제거된 지점이 존재함을 다음과 같이 보여야 한다.

$$\exists \langle a, b \rangle \in A^*, \exists s, t \in S \cdot R'(s, \langle a, b \rangle) = t$$
$$\Rightarrow \exists a, b \in A, \exists u \in S, \exists s, t \in S \cdot R(s, a) = u \wedge R(u, b) = t$$

길이가 2 이상인 귀납 경우(inductive case)는 (그림 6)과 같이 길이 n까지 결과가 같다고 가정한 후에 $n+1$번째가 같음을 보이면 된다. 이와 같은 귀납법 증명을 통해서 추상 모델의 도달 경로가 원본 모델에서도 그대로 유효함을 입증할 수 있다.

4. 실 험

게임의 풀이 경로를 찾기 위해서 세 개의 모델 체커(SMV, SPIN 그리고 Esterel)를 사용했다. <표 2>에서 보듯이 (그림 3)의 네모라이즈 10판을 풀이한 결과, 상태 폭발 문제로 인해서 세 개의 모델 체커 모두 실패했다. 표에서 기호 ∞은 3시간 이상 경과해도 결과를 얻을 수 없었던 경우를 나타낸다. 그래서 전 장에서 제안한 추상화 규칙을 적용하여 모델의 크기를 축소한 후에는 세 개의 모델 체킹 모두 풀이 경로를 찾는데 성공했다. 실험에 사용된 컴퓨팅 환경은 다음과 같다 : CPU 종류는 인텔 펜티엄 4, CPU 속도는 2.6Ghz, 그리고 메인 메모리는 1GB.

<표 2>에서 보듯이 SPIN을 사용했을 때 소요된 시간은 1초 이내로 가장 적었다. 사실 실험을 하기 이전에 우리는 SMV의 성능이 제일 우수할 것이라고 미리 예측했었다. 그러나 결과는 예상과는 다르게 SPIN(DFS, 깊이 우선 탐색

〈표 2〉 네모라이즈 10판의 추상화 이전과 이후의 풀이 결과

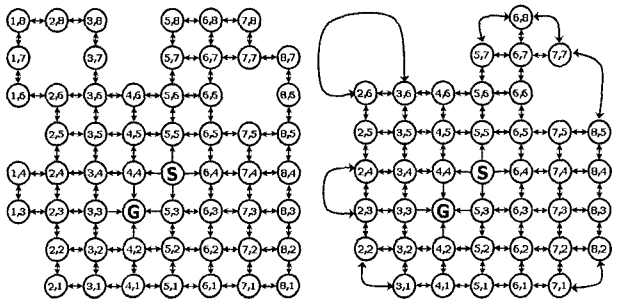
모델 체커	추상화 이전		추상화 이후	
	시간(Sec)	메모리(MB)	시간(Sec)	메모리(MB)
Esterel	∞	∞	04:59	4.0
SMV	∞	∞	00:38	133.1
SPIN(DFS)	∞	∞	00:01<	4.9

옵션을 사용한 경우)이 제일 우수했다. <표 3>에서도 보듯이 모든 경우에 있어서 SPIN(DFS)을 이용했을 때 가장 적은 시간이 소요되었다. 한편, 8판은 세 가지 도구 모두 풀지 못했다. (그림 7)에서 보듯이 바깥 경계 부분보다 안쪽에 모여 있기 때문에 추상화 규칙을 적용해도 지점 수가 크게 축소되지 않았다. 10판의 경우 추상화를 통해서 지점의 수가 30개로 줄어든 반면에 8판의 경우에는 추상화 이후에 남아있는 지점의 수가 42개로서, 10판보다 8판을 푸는 것이 더 복잡하다. 따라서 게임을 푸는데 요구되는 시간은 판수의 높고 낮음보다는 지점의 수가 얼마나 많이 축소되느냐에 달려있다.

우리는 선행 연구를 통해서 네모라이즈와 비슷한 푸쉬 푸쉬 게임을 모델 체킹으로 풀었었다[12]. 선행 연구와 본 연구의 큰 차이점중의 하나는 모델 체킹 도구의 활용에 있다. 선행 연구에서는 SMV만을 사용하였다. 그러나 본 연구에서는 SMV 뿐만 아니라 SPIN, Esterel 까지 사용했다. 세 가지 모델 체커들 모두 산업계에서 널리 사용되기 때문에 도구에 대한 경험을 얻을 수 있을 뿐만 아니라 여러 모델 체킹 도구를 비교해 볼 수 있었다. 이미 언급한바와 같이 네모라이즈 풀이에서 기대와는 달리 SPIN의 기록이 가장 우수했다. 한편, 선행 연구인 푸쉬 푸쉬 게임 풀이에서 SPIN이 SMV보다 우수하지 못했었다. 따라서 게임을 풀 때 어느 모델 체킹 도구가 우수한지를 밝히는 것은 매우 좋은 후속 연구 주제가 될 것이라고 생각한다.

〈표 3〉 추상화를 통한 네모라이즈 10판까지의 풀이 결과

네모라이즈 레벨	Esterel	SMV	SPIN(BFS)	SPIN(DFS)
1판	00:01<	00:01<	00:01<	00:01<
2판	00:01<	00:01<	00:01<	00:01<
3판	00:46	00:04	00:01<	00:01<
4판	00:27	00:01	00:01<	00:01<
5판	06:05	00:25	00:04	00:01<
6판	07:46	00:18	00:03	00:03
7판	15:43	00:30	00:04	00:03
8판	∞	∞	∞	∞
9판	39:08	00:26	00:11	00:02
10판	04:59	00:38	00:03	00:01<



(그림 7) 8판의 추상화 이전(좌)과 추상화 이후(우)

5. 결 론

핸드폰에 탑재된 네모라이즈는 게임 규칙을 준수하면서 초기 상태에서 목적 상태로 가는 경로를 찾는 게임이다. 게임의 크기가 작은 경우는 상태 공간 전체를 조사해서 최단 풀이 경로를 찾아내는 것이 보통이다. 그러나 게임의 크기가 큰 경우 상태 폭발 문제로 인해서 상태 공간 전체를 조사할 수 없었다. 그래서 본 논문에서는 false positive 오류를 인정하지 않는 추상화 방법을 사용하여 모델의 크기를 축소했다. 그 결과 축소된 추상 모델에서 얻은 도달 경로는 원본 모델에서도 존재하는 도달 경로였다. 그래서 전체 상태 공간을 탐색하지 않고도 마치 전체를 탐색한 것과 같은 결과를 얻을 수 있었다.

그러나 제안된 추상화 규칙만으로는 해결할 수 없는 단계들이 아직 많이 남아있기 때문에 이보다 더 정교한 추상화 규칙이 요구되었다. 여기서는 차수가 2인 지점에 대해서 집중적으로 다루었는데, 향후에는 2보다 더 큰 차수를 갖는 지점에 대해서 추상화 규칙을 연구하고자 한다. 또한 게임을 대상으로 얻어낸 추상화 기법을 소프트웨어 검증에 응용하는 것도 앞으로 연구해야 할 과제가 될 것이다.

참 고 문 헌

[1] Z. Yang, "Performance Analysis of Symbolic Reachability Algorithms in Model Checking," Mater Thesis, Rice University, Department of Computer, 1999.

[2] Y. Lu, "Automatic Abstraction in Model Checking," Ph.D. Thesis, Carnegie Mellon University, Department of Electrical and Computer Engineering, 2000.

[3] K. Fisler and M.Y. Vardi, "Bisimulation and Model Checking," In Proceedings of Advanced Research Working Conference on Correct Hardware Design and Verification Methods, Lecture Notes in Computer Science 1703, pp.338-341, 1999.

[4] A. Bouajjani, A. Legay and P. Wolper, "Handling Liveness Properties in (ω-)Regular Model-Checking," In Proceedings of Infinity '04, 2004.

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

[6] G. Kwon, "Applying Model Checking Techniques to Push Push Game Solving," In Proceedings of SERA2003, Lecture Notes in Computer Science 3026, pp.290-303, 2003.

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

[8] E. M. Clarke, O. Grumberg and D. Peled, Model Checking

MIT Press, 1999.
[9] http://www-cad.eecs.berkeley.edu/~kenmcmil/smv/
[10] http://spinroot.com/spin/whatispin.html#A
[11] http://www-sop.inria.fr/esterel.org/Html/About/AboutEsterel.htm
[12] 권기현, "모델 체킹에서 상태 투영을 이용한 모델의 추상화," 정보처리학회논문지D, 한국정보처리학회, 제11-D권 제6호, pp.1295-1300, 2004.

이 정 림
e-mail : jrim@kyonggi.ac.kr
2004년 경기대학교 정보과학부(학사)
2004년~현재 경기대학교 전자계산학과 석사과정
관심분야 : 정형 기법, 임베디드 소프트웨어, 소프트웨어 모델링

권 기 현
e-mail : khkwon@kyonggi.ac.kr
1985년 경기대학교 전자계산학과(학사)
1987년 중앙대학교 전자계산학과(이학석사)
1991년 중앙대학교 전자계산학과(공학박사)
1998년~1999년 독일 드레스덴 대학 전자계산학과 방문교수
1999년~2000년 미국 카네기 멜론 대학 전자계산학과 방문교수
1991년~현재 경기대학교 정보과학부 교수
관심분야 : 소프트웨어 모델링, 소프트웨어 분석, 정형 기법 등