NuSMV에서 생성된 반례길이 비교

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

Minimal Solution trace generation for game

Taehoon Lee° Gihwon Kwon
Information Science Dept. Kyonggi University

요 약

모델검사는 시스템과 그 시스템이 만족해야할 속성을 입력받아서 시스템이 속성을 만족하는지를 자동으로 보여주는 기술이다. 시스템이 속성을 만족한다면 참을, 만족하지 않는다면 왜 만족하지 않는지에 대한 반례를 보여준다. 모델검사에서 반례는 시스템의 오류를 발견하는데 중요하게 사용된다. 또한 모델검사를 이용해서 게임시스템을 모델링하고 반례를 이용해서 게임에 대한 풀이를 알 수 있게 되었다. 하지만 이런 반례가 최적의 풀이인지는 알 수 없었다. 이 논문은 모델검사에서 나온 게임 풀이가 최단 풀이 경로인지를 살펴본다. 그리고 최단 풀이경로를 출력하도록 NuSMV를 수정하여 원래 NuSMV에서 생성된 게임 풀이와 길이를 비교해 본다.

1. 서 론

모델 검사는 시스템을 유한상태모델로 표현하고 속성을 시제 논리식으로 표현한 후, 시스템이 속성을 만족하는지를 검사한다. 모델이 가질 수 있는 모든 가능한 상태 공간을 철저히 탐색하기 때문에 테스팅이나 시뮬레이션 기법으로 찾을 수 없는 오류를 찾을 수 있다. 그리고 주어진 모델과 속성을 입력하면 검사 과정이 자동으로 진행된다. 또한 모델이 논리식을 만족하지 않는 경우 왜 시스템이 속성을 만족하지 않는지에 대한 반례를 생성하기 때문에 시스템의 디버그 작업을 도와줄 수 있다. 이런 모델검사는 다양한 분야에 사용된다. 그 응용 분야 중 한 분야가 게임 풀이이다. 이 논문에서는 푸쉬푸쉬 게임에 대해 정의한다. 또한 이런 게임 풀이에서모델검사를 적용할 때 풀이의 길이가 최대한 작은 길이를 찾을 수 있어야 한다. 하지만 이전에 구현된 NuSMV 의 경우 최단 경로를 찾을 수가 없었다. 따라서 최단경로를 출력할 수 있는 반례생성기법에 대해 알아보고 또한 NuSMV를 새로 구현하여 최단 경로를 가지는 NuSMV를 새롭게 구현을 하였다. 본 논문의 구성은 다음과 같다. 2장에서는 CTL 모델검사 기법에 대해서 설명을 한다. 3장에서는 푸쉬푸쉬 게임에 대해 정의를 한다.4장에서는 최단길이 풀이에 대해서 설명을 하고 5장에서는 결론을 맺는다.

2. CTL 모델 검사 기술

모델 검사에 사용되는 모델의 핵심 요소는 상태와 상태 간의 천이이며, 이들을 사용하여 시스템의 행위를 모델한다. 특히 CTL(Computation Tree Logic) 모델 검사에서는 크립키 구조라 불리는 모델 $M = (S, I, R, AP, L)$을 사용한다. 여기서,

- S 는 상태들의 집합이다.
- $I \subseteq S$ 는 초기 상태들의 집합이다.
- $R \subseteq S \times S$ 은 상태들 간의 천이를 나타내는 관계이다.
- AP 는 단순 명제들의 집합이다.
- $L: S \rightarrow 2^{AP}$은 각 상태에서 참이 되는 단순 명제들을 해당 상태에 배정하는 함수이다.

모델에 관한 속성은 분기 시제 논리 언어인 CTL로 표현한다. 모델의 속성을 정형적으로 기술하기 위해서 CTL은 두 개의 경로 한정자 A(All), E(Exists)와 네 개의 시제 연산자 X(neXt), F(Future), G(Globally), U(Until)를 갖는다. 경로 한정자와 시제 연산자를 조합하면 8개의 CTL연산자 AX, EX, AF, EF, AG, EG, AU, EU를 얻는다. CTL 식 ϕ, Ψ의 값이 모든 모델과 모든 상태에서 동일하다면 두 식을 동치라고 부르며 $\phi \equiv \Psi$로 표시한다. 동치 관계에 있는 식은 다음과 같다:

$\top \equiv ?\bot$

$\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$

$\phi_1 \Rightarrow \phi_2 \equiv \neg(\phi_1 \wedge \neg\phi_2)$

$\phi_1 \Leftrightarrow \phi_2 \equiv \neg(\phi_1 \wedge \neg\phi_2) \wedge \neg(\phi_2 \wedge \neg\phi_1)$

$AX\phi \equiv \neg EX\neg\phi$

$AF\phi \equiv \neg EG\neg\phi$

$EF\phi \equiv E(\neg\bot U\phi)$

$AG\phi \equiv \neg EF\neg\phi \equiv \neg E(\neg\bot U\neg\phi)$

$A[\phi 1 U \phi 2] \equiv \neg E[\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2)] \wedge \neg EG\neg\phi_2$

두 식이 동치이기 때문에 좌변의 식은 우변의 식으로 대치 가능하다. 따라서 우변에 나오는 연산자의 모임이 CTL 식을 정의하는데 요구되는 최소한의 연산자 집합이다. 위의 경우는 {⊥,¬,∧,EX,EG,EU}이다. 이들을 이용해서 CTL 구문을 정의하면 다음과 같다.

$\phi ::= p|\bot|\neg\phi|\phi 1 \wedge \phi 2|EX\phi|EG\phi|E(\phi_1 U \phi_2)$

3. 푸쉬푸쉬 게임 모델링

푸쉬푸쉬게임은 다음과 같은 5개의 튜플로 정의할 수 있다.

$G = <Q, A, \delta, qi, qf>$

Q 는 상태들의 집합이고

A = {*left*, *right*, *up*, *down*}는 이동 방향의 집합이다.

$\delta : Q \times A \rightarrow Q$ 는 상태 전의 함수

qi 와 qt 는 Q.의 초기와 최종 상태를 나타낸다.

에이전트의 상태 기계와 셀의 상태 기계를 동기식으로 결합하면 게임 전체의 상태 기계 $T_{game} = T_{agent} \otimes T_{cell_1} \otimes \cdots \otimes T_{cell_n}$를 얻는다. 여기서 기호 ⊗는 동기식 결합 연산자이다. 즉, 게임의 행위는 상태 기계 $T_{game} = (S_{game}, I_{game}, D, \delta_{game}, F_{game})$로 모델링 할 수 있다.

- $S_{game} = S_{agnet} \times S_{cell_1} \times \cdots \times S_{cell_n}$는 게임이 갖는 상태 공간이다.
- $I_{game} = (I_{agnet}, I_{cell_1}, \cdots, I_{cell_n})$는 게임의 초기 상태이다.
- $\delta_{game} : S_{game} \times D \rightarrow S_{game}$ $\times S_{cell_n}$는 상태 전이 함수이다. 즉, $\delta_{game}((x,y), c_1, \cdots, c_n, d) = \delta_{agent}((x,y), d), \delta_1(c_1, d), \cdots, \delta_n(c_n, d))$이다.
- F_{game}는 목표 셀들의 집합이다.

지금까지 유한 상태 기계로 게임을 모델링 하였다. 게임에 있는 개체와 그들의 행위를 모델링 하였다. 이와 같은 모델의 의미는 크립키 구조로 $M = (S, I, R, X, L)$ 정의 할 수 있다.

- $S = S_{agnet} \times S_{cell_1} \times \cdots \times S_{cell_n} \times D$는 상태 공간의 집합이다.
- $I = I_{agent} \times I_{cell} \times \cdots \times I_{cell_n}\{d \in D\}$는 초기 상태 집합이다.
- $R((x,y), c_1, \cdots, c_n, d, \delta_{agnet}((x,y), d), \delta_1(c_1, d), \cdots, \delta_n (c_n, d), d \in D)$는 전이 관계이다.
- $X = \{cell_1, \cdots, cell_n\}$는 단순 명제들의 집합이다.
- $L(s) = L((x,y), c_1, \cdots, c_n, d) = \{cell_1 | c_1\} \cup \cdots \cup \{cell_n | c_n\}$는 라벨 함수이다.

전술한 바와 같이, 푸쉬푸쉬 게임 풀이란 초기 상태에서 목표 상태로 공을 모두 이동시키는 경로를 찾는 것이다. 목표 상태는 $F_{game} = \{cell_1, \cdots, cell_n\}$이다. 즉 게임마다 목표 상태에 도달하는 경로가 최소한 하나이상 존재한다. 만약 그러한 경로가 결코 존재하지 않는다고 하면 이는 틀린 것이 되고 모델 검사는 이에 대한 반례로 초기 상태에서 목표 상태로 가는 경로를 생성한다. 이러한 경로가 푸쉬푸쉬 게임을 푸는 답이 되는 것이다. 게임을 풀 수 있는 답을 유도하기 위해서 '목표 상태로 갈 수 있는 경로가 결코 존재하지 않는다' 를 CTL로 표현하면 $AG\neg\phi$이다. 여기서 $\phi \equiv \bigwedge_{i=1}^{|F_{game}|} cell_i \in F_{game}$이다.

4. 최단길이 풀이생성

$[\![q_t]\!] = \{s \mid q_t \in L(s)\}$ 는 M에 있는 도달해야 될 목표상태이다. 경로 $= <s_1, \ldots, s_n>$는 다음과 같은 조건을 만족하는 풀이경로이다.

1. $s_1 \in I$,
2. $s_n \in [\![q_t]\!]$,
3. s_n 은 도달가능하다.

덧붙여서 어떤 경로를 라고 나타내고 만일 다른 모든 가능한 경로를 라고 나타낼 때, | | | |인 경우 를 최단 풀이 경로라고 할 수 있다.

주어진 모델에서 우리는 최단 풀이 경로를 다음과 같이 찾을 수 있다.

$$MC(M, AG\neg q_t) = \begin{cases} \varepsilon & \text{if } M \models AG\neg q_t \\ \pi & \text{if } M \not\models AG\neg q_t \end{cases}$$

이것은 모델 검사기가 생성한 반례 $= <s1,,sn>$ 가 하나의 풀이를 나타낸다고 볼 수 있다. 그리고 는 가능한 풀이가 없다는 것을 나타내게 된다. 우리는 이러한 방법을 이용해서 NuSMV 의 반례가 최단의 경로인지를 조사해 보았다. NuSMV에서 생성된 반례의 길이를 먼저 조사를 한 후에 NuSMV를 수정해서 가능한 모든 반례를 생성할 수 있게 한 후에 최소의 길이를 가지는 반례를 생성하도록 수정한 후 원래 NuSMV에서 생성된 반례와 비교를 해보았다. 표1은 NuSMV에서 생성된 푸쉬푸쉬 1판에서 50판까지 의 반례의 길이이다.

표1 푸쉬푸쉬 풀이 길이 비교

판수	원래 NuSMV	최단 풀이	차이
1	11	10	1
2	90	89	1
3	116	114	2
4	34	33	1
5	51	50	1
6	81	79	2
7	44	44	0
8	170	170	0
9	35	34	1
10	57	57	0
11	29	29	0
12	56	56	0
13	56	55	1
14	74	72	2
15	101	101	0
16	64	64	0
17	98	97	1
18*	129	129	0
19*	61	60	1
20	106	105	1
21	57	57	0
22	82	82	0
23	123	122	1
24	89	89	0
25	85	83	2
26	85	83	2
27	61	61	0
28	162	162	0
29	89	89	0
30	101	101	0
31	118	116	2
32	126	126	0
33	261	261	0
34	x	59	0
35	56	56	0
36	156	154	2
37	121	120	1
38	40	38	2
39	65	64	1
40	x	77	0
41	110	108	2
42	137	137	0
43	35	35	0
44	172	171	1
45	169	169	0
46	108	106	2
47	96	96	0
48	175	173	2
49	x	84	0
50-1	76	74	2
50-2	84	84	0
50-3	85	84	1
50-4	97	96	1

표1 에서 보았듯이 NuSMV 에서는 최단 풀이 경로를 생성하지 못했다. 이 논문에서 찾아낸 최단 풀이 경로와 길이가 같은 경우도 있었지만, 많은 판에서 1,2 의 차이가 있었다.

5장 결론

실제로 NuSMV에서 구현된 반례 생성기법은 최단 경로를 보장해 주지 못한다. 최단 경로를 보여주는 경우도 있었지만, 그렇지 못한 경우도 많이 있었다. 최단 경로를 생성해주는 반례 생성 기법에 대한 알고리즘은 이미 여러 논문들이 있었지만 NuSMV 에서는 최단 경로를 생성하지 못했다. 본 논문에서는 NuSMV를 수정하여 최단 풀이 경로를 찾아주도록 수정을 하였고 원래 NuSMV 에 비해 짧은 경로를 보여주었다. 하지만 원래 NuSMV에서 최단 경로를 생성하지 못한 이유와 본 논문에서 구현한 이론에 대한 정의를 아직 수행하지 못했다. 따라서 새로 구현한 이론에 대한 명확한 정의와 수정전의 NuSMV에서 최단 경로를 생성하지 못한 이유를 찾아야 한다.

참고문헌

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

[2] E.M. Clark ,O Grumberg , S Jha , Y Lu, H Veith, Counterexample-guided Abstraction Refinement , In Proceedings of CAV'00 , 2000

[3] KL McMillan , "Symbolic Model Checking : An Approach to the State Explosion Problem", PhD thesis Carnegie Mellon University , 1992.

[4] 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 Proceeding of International Conference on Computer-Aided Verification 2002

[5] Michael Huth , Mark Ryan , "Logic in Computer Science : Modeling and Reasoning about System" , CAMBRIDGE University Press, 2000.

[6] E.M. Clark ,O Grumberg , S Jha , Y Lu, H Veith, Progress on the State Explosion Problem in Model Checking , In Proceedings of 10 Years Dagstuhl, LNCS 2000, pp154-169,2000