SAT 프로시져를 이용한 유한 상태 기계의 동치 검사

엄태호 권기현
경기대학교 소프트웨어 공학 연구실
{ever9,khkwon}@kyonggi.ac.kr

김태훈
한국정보보호진흥원 평가연구팀
taihoon@kisa.or.kr

Equivalence Checking Finite State Machines with SAT-Procedure

Taeho Eom, Gihwon Kwon
Kyonggi University

Taihoon Kim
Korea Information Security Agency

요 약

본 연구에서는 만족성 검사기를 이용하여 두 유한 상태 기계의 행위가 동치인지를 검사한다. 기존의 동치 검사는 대부분 BDD 를 기반으로 했었기 때문에 변수 순서 배열 및 내부 BDD 노드 폭발 문제에 시달렸었다. 하지만 여기서는 BDD 대신 명제 논리를 이용하기 때문에 위와 같은 문제점을 피할 수 있다. 하지만 논리식을 만족성 검사기의 입력 형태인 논리곱 정규형으로 변환하는 과정에서 절의 크기는 식의 크기에 지수적으로 증가 하였다.

1. 서론

유한 상태 기계는 시스템의 행위를 모델하는 모델링 언어로써, 반응형 시스템 및 실시간 시스템의 행위를 기술하는데 널리 사용되고 있다. 특히, 유한 상태 기계의 표현력을 높인 모델링 언어인 Statechart [1]는 산업 현장에서 활발히 사용중이다.

모든 입력에 대해서 두 유한 상태 기계의 출력이 항상 동일하다면 두 기계는 동치이다. 동치 검사는 유한 상태 기계로 표현된 모델의 행위 분석에 필수적인 검사로서 시스템 최적화 및 정확성 확인등에 활용되고 있다. 즉, 두 개의 유한 상태 기계가 동치라고 판정된다면 복잡한 유한 상태 기계는 보다 단순한 기계로 대치될 수 있다. 또한 명세와 구현이 모두 유한 상태 기계로 표현된 경우, 동치 검사를 이용해서 구현이 명세를 만족하는지를 결정할 수 있다.

동치 검사는 도달성 분석으로 수행할 수 있다. 검사할 유한 상태 기계 M_A, M_B로부터 곱 기계 $M = M_A \times M_B$ 을 구축한 후 초기 상태로부터 도달 가능한 상태를 방문하면서 그때마다 모든 입력에 대해서 두 기계의 출력이 같은지를 검사한다. 한편, 모델 검사 도구를 이용하여 동치 검사를 할 수 있다 [2]. 모델 검사 도구는 동치 조건을 기술한 시제 논리식 ϕ 와 곱 기계를 표현한 모델 M을 입력 받아서, M 이 ϕ 를 만족하는지를 검사한다. 만족한다면 두 기계는 동치이고, 만족하지 않는다면, 생성된 반례를 통해서 비 동치의 원인을 파악할 수 있다.

본 논문에서는 기존의 동치 검사와는 달리 만족성 검사기를 이용하여 동치 검사를 수행한다. 먼저 유한 상태 기계를 새로 정의한 자료구조로 표현하고, 이들의 연산을 통해 모델 $M = M_A \times M_B$ 의 논리식과 두 기계의 동치 조건인 속성 ϕ를 논리식으로 표현한다. 그런 후, 기존 만족성 검사 도구인 JSAT(The Java SATisfiability Library, [3])를 이용하여 만족 가능한 해석이 있는지 검사한다. 초기 상태로부터 도달 가능한 모든 천이의 논리식이 동치 조건의 논리식에 대해 만족 가능한 해석이 존재하는지 여부를 통해 동치 또는 비 동치를 결정한다.

본 논문의 구성은 다음과 같다. 2 장에서는 만족성 검사기를 이용한 동치 검사를 설명하고, 3 장에서는 구현 및 본 방법의 효과를 분석한 후, 마지막으로 4 장에서 결론 및 향후 연구를 살펴본다.

2. 만족성 검사기를 이용한 동치 검사

2.1. 모델 M의 논리식 표현

모델 M 이 논리식으로 표현되는 방법에 대해서 살펴보자.

정의 1: 6-튜플 $(I, O, S, d, \delta, \lambda)$ 로 표현된 유한 상태 기계는 7-튜플 $(In, Out, S\_C, S\_N, Init, \chi, \gamma)$ 의 논리식으로 표현한다.

- $In = \wedge ap_m{}^I$ 는 기계의 입력에 대한 명제 논리식 표현이다.
- $Out = \wedge ap_m{}^O$는 기계의 출력에 대한 명제 논리식 표현이다.
- $S\_C = \wedge ap_m{}^S$ 는 기계의 상태에 대한 현재 표현 논리식이다.
- $S\_N = \wedge ap_m{}^S$ 는 기계의 상태에 대한 다음 표현 논리식이다.
- $\chi = In \wedge S\_C \wedge S\_N$ 는 천이 관계로 δ에 대응되며 입력, 현재 상태, 다음 상태의 논리식을 논리 곱한다.
- $\gamma = In \wedge S\_C \wedge Out$ 는 출력 함수로 λ에 대응되며 입력, 현재 상태, 출력의 논리식을 논리 곱한다.

여기서 $ap_1, \ldots, ap_m$는 더 이상 나눌 수 없는 원소 명제이다. 유한 상태 기계에서 입력 I의 수가 n 일 때 AP 수는 $m = \lceil \log_2 n \rceil$ 이고, $ap_m{}^I$는 입력 I에 대한 원소 명제들을 의미한다. 원소 명제들은 유일하므로, 중복되어 사용될 수 없다. 출력 O와 상태 S에 대한 원소 명제들도 같은 방법으로 만들어 진다.

유한 상태 기계를 나타내는 논리식에서 모델 M, 속성 ϕ의 논리식을 보다 쉽게 유도하기 위해 새로운 자료 구조를 정의한다.

정의 2: 자료 구조 DT(Decision Tree)는 (V, E)로 구성된다.

- $V \in \{ V_{root}, V_{middle}, V_{leaf} \}$ 즉, 노드 V는 최상위 노드 V_{root} 와 중간 노드 V_{middle},단말 노드 V_{leaf} 로 구성 된다.
- 노드의 값은 $V.value$ 로 표기하며 In, Out, S_C, S_N 의 논리식을 나타낸다.
- E 는 관계(천이 관계, 출력 함수)를 나타낸다.

정의 3: 천이 관계 χ 를 DT 로 나타내면 다음과 같다.

- $V_{root} ::= In$ 즉, χ 에서 입력 논리식을 최상위 노드에 할당한다.
- $V_{middle} ::= S\_C$ 이고, $V_{leaf} ::= S\_N$ 이다. 즉, χ 에서 현재 상태 논리식을 중간 노드에 할당하고, 다음 상태 논리식을 단말 노드에 할당한다.

- $\chi \in E$ 이다. 천이 관계 χ로 노드들의 관계가 만들어 진다.

정의 4: DT로 표현된 천이 관계 χ는 다음의 연산으로 모델 M에 대한 논리식을 유도한다.
- $\forall \chi_A, \chi_B \in E \bullet \{ (\chi_A \times \chi_B) \mid (\gamma_A.V_{root}.value = \gamma_B.V_{root}.value) \}$
- $V_{middle} ::= \chi_A.V_{middle}.value \wedge \chi_B.V_{middle}.value$
- $V_{leaf} ::= \chi_A.V_{leaf}.value \wedge \chi_B.V_{leaf}.value$
- $M = \vee(V_{middle}.value \wedge V_{leaf}.value)$

DT를 통한 모델 M의 논리식은 $V_{middle}.value$과 $V_{leaf}.value$의 논리곱, 각 노드의 쌍들은 논리 합하여 만들어 진다.

정의 5: 모델 M의 초기 상태 $Init$는 두 기계의 초기 상태를 논리곱 함으로써 형성된다.
- $Init = Init_A \wedge Init_B$

2.2. 속성 ϕ의 논리식 표현

지금 까지는 모델 M을 논리식으로 만드는 부분을 살펴 보았다. 동치 조건인 속성 ϕ를 논리식으로 표현하는 부분에 대해서 살펴보자.

정의 6: 출력 함수 γ를 DT로 나타내면 다음과 같다.
- $V_{root} ::= S\_C$
- $V_{middle} ::= In$, $V_{leaf} ::= Out$
- $\gamma \in E$

정의 7: DT로 표현된 출력 함수 γ는 다음의 연산으로 속성 ϕ에 대한 논리식을 유도한다.
- $\sigma = \forall \gamma_A, \gamma_B \in E \bullet \{ (\gamma_A.V_{root}.value \wedge \gamma_B.V_{root}.value) \mid (\gamma_A.V_{middle}.value = \gamma_B.V_{middle}.value) \wedge (\gamma_A.V_{leaf}.value = \gamma_B.V_{leaf}.value) \}$
- $\phi = \vee \sigma$

속성 ϕ에 대한 논리식은 각 σ을 논리합 하여 만들어 진다.

2.3. 동치 검사 알고리즘

지금부터는 모델이 속성을 만족하는지($M \models \phi$)를 통해 동치 검사를 수행하는 방법에 대해서 살펴보자.

정의 8: 상태에 대한 현재 표현과 다음 표현은 다음처럼 변경이 가능하다.
- $\Delta(S\_N) = \exists S\_N \bullet (\Omega \wedge S\_N)$
- $\nabla(S\_C) = \exists S\_C \bullet (\Omega \wedge S\_C)$

여기서 Ω는 각 상태의 현재 상태와 다음 상태를 논리곱 한 $\Omega = \vee(S\_C \wedge S\_N)$ 이며, $\Delta(S_N)$는 다음 상태로 표현된 논리식을 현재 상태 논리식으로 변경하고, $\nabla(S_C)$는 반대를 나타낸다.

정의 9: 만족성 검사기를 사용하여 동치 검사를 수행하기 위해서 초기 상태 $Init$와 속성 ϕ의 현재 상태로 표현 논리식을 다음 상태로 표현된 논리식으로 변경한다.
- $Init' = \nabla(Init)$
- $\phi' = \nabla(\phi)$

이것은 천이의 만족성을 검사함으로써 동치 검사를 수행하기 위한 것이다.

정의 10: 동치 검사는 다음처럼 수행 된다.
- 변환된 초기 상태 $Init'$의 논리식은 변환된 속성 ϕ'의 논리식에 대해서 만족 가능한 해석이 존재해야 한다. 그렇지 않으면, 비 동치이다.
- 초기 상태로부터 고정점에 도달 할 때 까지 상태의 모든 천이 논리식들이 변환된 속성 ϕ'의 논리식에 대해 만족 가능한 해석이 존재하면, 두 기계는 "동치"이다.
- 그렇지 않고, 고정점에 도달 할 때 까지 하나의 천이라도 변환된 속성 ϕ'의 논리식에 만족 가능한 해석이 존재하지 않는다면, 두 기계는 "비 동치"이다.

2.4. 만족성 검사기의 입력 형태로 변환

검사할 논리식은 만족성 검사기의 입력 형태인 논리곱 정규형으로 표현한다. 논리곱 정규형은 논리식에 분배 법칙을 반복 적용함으로써 얻어진다. 검사할 식을 논리곱 정규형으로 변환하는 분배 법칙은 다음과 같다.
- $\omega \vee (\psi_{11} \wedge \psi_{12}) \equiv (\omega \vee \psi_{11}) \wedge (\omega \vee \psi_{12})$
- $(\omega_{11} \wedge \omega_{12}) \vee \psi \equiv (\omega_{11} \vee \psi) \wedge (\omega_{12} \vee \psi)$

기존 만족성 검사기 JSAT은 DIMACS 형식으로 입력 받아서, 만족성 또는 불 만족성을 결정한다. 따라서, 만족성 여부를 검사할 논리곱 정규형은 DIMACS 형식으로 변환한다.

3. 구현 및 분석

2장의 정의들을 토대로 유한 상태 기계로부터 동치 검사를 자동으로 수행하는 시스템의 개요도는 다음과 같다.

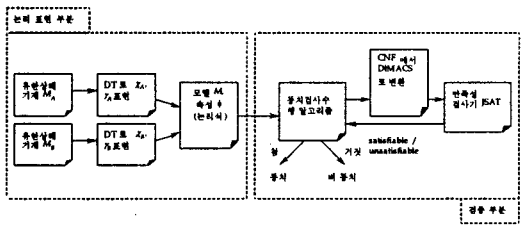


그림 1. 시스템의 개요도

예제를 통해 만족성 검사기를 통한 동치 검사의 수행 결과를 살펴보자.

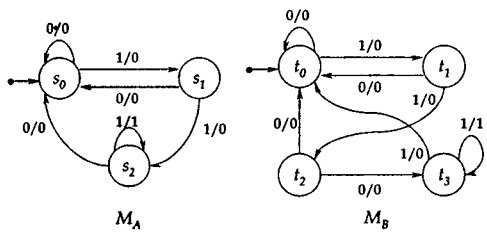
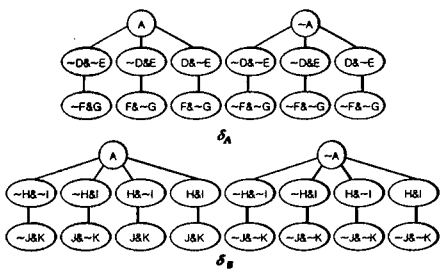


그림 2. 유한 상태 기계 M_A와 M_B

그림 3은 그림 2에 대해서 정의 3과 6을 적용하여 만들어진 DT를 표현한다.



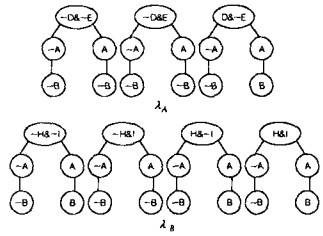


그림 3. DT 로 표현한 M_A 와 M_B의 천이 관계 χ 와 출력 함수 γ

그림 4 는 DT 의 연산으로 만들어진 M의 논리식을 도식적으로 표현하고 있으며, 속성 ϕ의 논리식을 나타낸다. M에 대한 도달성 분석의 도식적 표현과 변경된 속성 ϕ'의 논리식은 그림 5 에서 나타내고 있다. 그림 6 은 만족성 검사기를 이용하여 동치 검사를 수행한 결과이다.

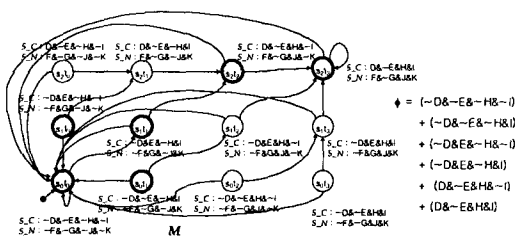


그림 4. M에 대한 논리식의 도식적 표현과 속성 ϕ의 논리식

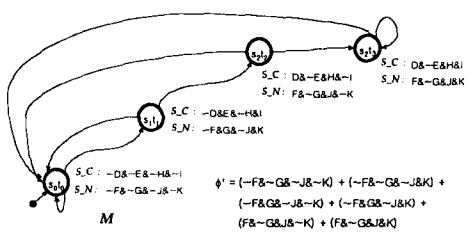


그림 5. M에 대한 도달성 분석의 도식적 표현과 속성 ϕ'의 논리식

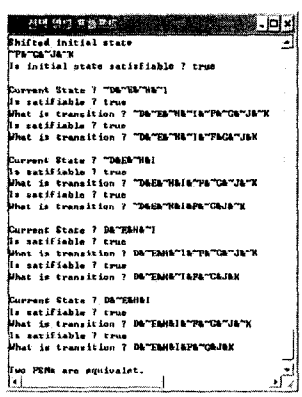


그림 6. 그림 2 에 대한 결과

지금까지 방법으로 여러 모델들에 적용 시켜보았다. 여러 모델들에 적용한 결과 동치인지, 비 동치인지를 정확히 구분하였으나 모델 1 과 모델 4 는 전체 상태 수가 12 개에 불과 하지만, 검사 할 천이에 대한 논리식을 논리곱 정규형으로 변환하는 과정에서 4104 개의 절수를 보이고 있다. 이것은 식에 대해서 지수적인 크기이다.

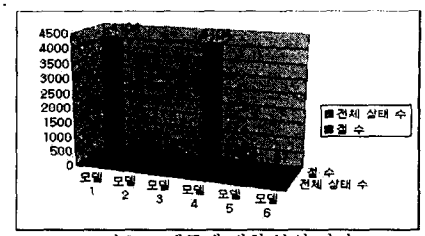


그림 7. 모델들에 대한 분석 결과

4. 결론 및 향후 연구

본 연구에서는 유한 상태 기계의 동치 검사를 만족성 검사기로 해결하는 방법을 제시하고 구현을 통하여 확인하였다. 검사될 두 기계를 DT 를 통해서 표현 하였고, DT 의 연산으로 모델 M 과 동치조건을 나타내는 속성 ϕ의 논리식을 만들었다. 그런 후, 동치 검사를 수행 하기 위해, 도달 가능한 모든 천이가 속성 ϕ에 대해서 만족 가능한 해석이 존재하는지를 만족성 검사기를 통해 수행하였으며, 시스템의 구현을 통해서 자동화하였다. 여러 예제에 시스템을 적용해 본 결과 동치 여부의 결과는 정확하였고, 수행 시간도 효율적이었으나, 검사 할 천이를 논리곱 정규형으로 변환하는 과정에 식의 크기에 지수에 해당하는 절수의 결과가 나타났다. 이것은 커다란 시스템 검증을 방해하는 주요한 요인이다.

본 연구에서는 동치 검사 문제를 [4]등의 기존 연구에 비해서 BDDs 가 아닌 만족성 검사기를 통하여 수행하였다. 본 연구와 관련해서 앞으로 계속 연구해야 할 내용은 다음과 같다. 첫째, 검사할 원식이 만족 가능하면 검사될 식의 크기와는 상관없이 만족성을 유지하면 되므로, 효율적인 논리곱 정규형 변환 기법이 요구된다. 둘째, 기존의 만족성 검사기를 이용한 방법[5, 6]들과의 비교 연구이다. 셋째, 만족성 검사기를 이용하여 유한 상태 기계 보다 표현력이 높은 상태도에 대해서 동치 검사를 수행하는 것이다.

참고문헌

[1] D. Harel, "Statecharts: A visual formalism for complex systems," Science of Computer Programming, Vol.8, 1987.

[2] E.M. Clarke, E.A. Emerson, A.P. Sistla, "Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications," ACM Transactions on Programming Languages and Systems, Vol.8, No.2, pp.244-263, 1986.

[3] http://cafe.newcastle.edu.au/daniel/JSAT/

[4] C.A.J. van Eijk, J.A.G. Jess, "Exploiting Functional Dependencies in Finite State Machine Verification," In Proceedings of the European Design and Test Conference ED&TC, pp.9-14, 1996.

[5] A. Biere, A. Cimatti, E.M. Clarke, and Y. Zhu, "Symbolic Model Checking without BDDs," In Proceedings of Tools and Algorithms for the Analysis and Construction of Systems (TACAS'99), LNCS 1579, 1999.

[6] P.Bjesse. Symbolic model checking with sets of states representation as formulas. Technical Report CS-1999-100, Department of Computer Science, Chalmers technical university, March 1999.