병행 기계를 이용한 상태도의 동치 검사

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

Equivalence Checking of Statecharts using Concurrent Machines

Tae-Ho Um, Gihwon Kwon
Software Engineering Laboratory, Department of Computer Science, Kyonggi University

요 약

본 연구에서는 상태도를 평탄화 하는 대신에, 여러 기계가 동시에 수행하는 병행 기계로 상태도를 표현한 후 ROBDD를 이용하여 상태도의 동치 여부를 기호적으로 검사하고자 한다. 상태도가 기능적으로 동치라는 것은, 두 상태도는 같은 함수를 구현하고 있다는 것과 같다. 이것은 모든 입력 이벤트에 대하여 두 상태도의 반응이 항상 동일한지를 판정함으로써 가능하다. 즉 상태도의 동치 검사는, 입력 이벤트가 같은 상태들의 집합이 모든 입력에 대해서 출력이 같은 상태들인가라는 문제로 축소된다.

1. 서론

반응형 시스템 또는 내장형 시스템의 행위를 나타내는 Harel의 상태도(Statecharts)는 시각 명세 언어로서 전통적인 유한상태기계에 계층성, 동시성 등을 확장했다[1,2]. 특히 표현력이 높기 때문에, 산업 현장에서 많이 사용되고 있다. 본 논문에서는 두 개의 상태도가 동치인지를 검사하고자 한다. 이러한 동치 검사는 구현이 명세를 만족하는지 또는 최적화등에 사용된다.

상태도를 동치 검사하는 가장 쉬운 방법으로는, 상태도의 계층성을 붕괴하여 커다란 평면 구조의 유한상태기계를 만들어 비교하는 것이다[3]. 그러나 평탄화를 하게 되면, 상태들의 수와 상태들을 연결하는 천이의 수가 크게 증가하는 문제점이 발생한다. 본 논문에서는 평탄화를 거치지 않는 대신, 병행 기계[4,5]를 통해서 상태도의 동치 검사를 수행한다. 평탄화 대신 병행 기계를 이용하면, 전체를 고려하기 보다는 지역만을 고려하기 때문에 동치 검사를 위한 전처리 과정을 수월하게 할 수 있다.

최근 상태도에 대한 동치 검사가 많이 연구되고 있다[3,6]. 행위적인 관점에서, 만약 두 상태도가 모든 외부 입력에 대해서 항상 동일하게 반응한다면 두 상태도는 동치라고 할 수 있다. 즉, 두 상태도의 입력 이벤트가 같은 상태들의 집합에서 모든 입력에 대하여 출력이 같은 상태인지를 검사한다. 이러한 과정은 도달성 검사, 특히 전진 도달성 검사를 통하여 자동으로 수행된다. 2장에서는 곱 기계를 만들기 위해 병행 기계의 구문 확장과 곱 기계의 기호적 표현에 대해서 살펴보고, 3장에서는 동치검사, 4장에서는 실행 예를 통해 동치검사를 설명한다. 마지막으로 5장에서는 결론 및 향후 연구 방향을 제시한다.

2. 병행 기계

병행 기계에서는 'OR 상태'들이 병행적으로 결합되어 있다[4,5]. 계층성과 동시성이 유지됨으로써, 평탄화로 인해서 발생되는 문제점을 방지할 수 있다. 또한 현재 활성중인 상태들의 집합과 이벤트의 변화를 전역적 변화가 아닌, 각 기계별 변화로 지역화할 수 있는 장점을 가지고 있다.

2.1. 구문

병행 기계의 구문은 M = (S, E, O, T, Sub, type, def, Out) 이다. 여기서 S는 상태 집합, E는 입력 이벤트 집합, O는 출력 집합, $Sub : S \rightarrow 2^S$ 는 상태를 그들 자식 상태들로 매핑하는 함수로서 상태들간의 계층성을 나타낸다. $type : S \rightarrow \{pr, se, sh, pa\}$는 상태의 타입을 리턴하는 함수로서 pr, se, sh, pa는 상태도의 BASIC, OR, HISTORY, AND 상태를 나타낸다. $def : S \rightarrow S$ 는 se와 sh 상태의 디폴트 상태를 나타내는 부분 함수이며, $Out : S \times E \rightarrow O$는 이벤트와 상태의 출력 함수를 나타낸다.

2.2. 기호적 표현

병행 기계의 출력 함수와 곱 기계[7], 그리고 동치검사를 기호적으로 표현하는 방법에 대해서 살펴보자. 입력, 상태, 출력을 ROBDD(Reduced Ordered Binary Decision Diagrams)로 유일하게 표현하기 위해 필요한 변수의 개수는 $|m| = \lceil \log_2 n \rceil$ 이다[8]. 현재 상태 s 의 ROBDD 표현을

규칙 1. i번째 기계의 출력 함수의 기호적 표현은 다음과 같다.

- $\overline{Out_i} = \bigvee_{(s,e,o)\in Out_i} (\bar{s}_i \wedge \bar{e} \wedge \bar{o})$

규칙 2. 병행 기계에 대한 출력 함수의 기호적 표현은 다음과 같다.

- $\overline{Out} = \bigvee_{i=1}^{n} \overline{Out_i}$

규칙 3. 곱 기계의 초기상태는 다음과 같다.

- $\overline{Init_{PM}} = \bar{\sigma}_0^1 \wedge \bar{\sigma}_0^2$

여기서 σ_0는 두 병행 기계의 초기 상태들이다.

규칙 4. 곱 기계의 천이 관계는 다음과 같다.

- $\overline{T_{PM}(\sigma,\sigma')} = \exists \bar{e} \bullet (\overline{T(e,\sigma,\sigma')}^1 \wedge \overline{T(e,\sigma,\sigma')}^2)$

여기서 σ와 σ′는 현재 구성(configuration), 다음 구성(configuration), e는 입력 이벤트를 나타내는데, $\overline{T_{PM}(\sigma,\sigma')}$는 두 병행 기계에 대해서 입력 값이 같은 운영상의 천이 관계로 곱 기계의 현재 상태와 다음 상태의 천이 관계를 만든다.

규칙 5. 곱 기계의 출력 함수는 다음과 같다.

- $\overline{Out_{PM}(e,\sigma)} = \Pi_k(\overline{Out_k(e,\sigma)}^1 = \overline{Out_k(e,\sigma)}^2)$

k 는 $\bar{o}$를 ROBDD 로 표현하기 위한 변수의 개수이다. $\overline{Out_{PM}(e,\sigma)}$ 은 $\bar{o}$ 의 각 부울 벡터에서 출력 값이 같은 두 병행 기계의 출력 함수로 곱 기계의 출력 함수를 만든다.

규칙 6. 모든 입력에 대해서 출력이 같은 곱 기계의 상태들을 만든다.

- $\overline{EQUI(\sigma_{PM})} = \forall \bar{e} \bullet (\overline{Out_{PM}(e,\sigma_{PM})})$

규칙 7. 곱 기계 상에서 도달한 모든 상태들이 $\overline{EQUI(\sigma_{PM})}$ 를 ⇒ 하면 두 상태도는 동치이다.

- $\forall \bar{\sigma}_{PM} \bullet (\overline{NEW_k} \Rightarrow \overline{EQUI(\sigma_{PM})}) = True$

여기서 $\overline{NEW_k}$ 는 곱 기계상에서 새로 도달한 상태들을 말한다.

3. 동치검사

곱 기계 상에서 동치 검사를 수행하는 알고리즘은 다음과 같다.

```
EQUIVALENCE_CHECK =
k := 0
NewK:= InitPM
R0 := NewK
Do
    k:=k+1
    NewK = Image(Rk-1, TPM) – Rk-1
    If ((NewK ⇒ EQUI(σPM))=1)
        Rk:=Rk-1∪ NewK
    Else
        return("Not_Equivalence")
While Rk-1 ≠ Rk
return("Equivalence")
```

$\bar{s}$ 로 표시하자. 즉 현재 상태 s 는 m 개의 부울 벡터 v_1, v_2,..., v_m로 표현된다. 이와 같은 방법으로 입력, 다음 상태, 출력을 ROBDD로 표현할 수 있다. 병행 기계의 출력 함수와 곱 기계는 다음과 같이 ROBDD로 표현된다.

여기서 Image는 현재 상태에서 다음 상태들의 집합을 구하며, 곱기계 상에서 새로 도달한 값 NewK가 EQUI(σ_{PM})를 함축(implication)하는지를 검사하는 것이다. 고정점에 도달 할 때까지 모두 참이면 두 상태도는 동치이다.

4. 실행 예

간단한 상태도를 통해 동치 검사를 어떻게 수행하는지 살펴보자.

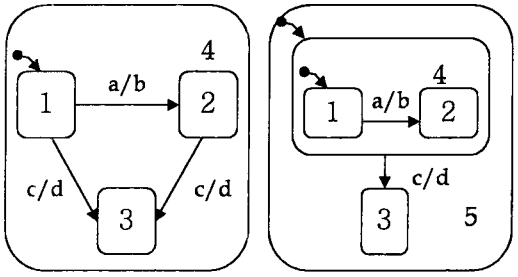

그림 1. 상태도 1과 상태도 2([6]에서 인용)

상태도 1과 2에 대한 병행 기계는 아래 그림2와 같다.

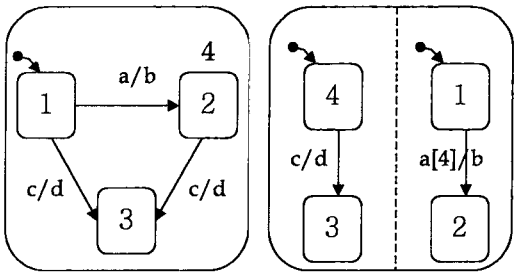

그림 2. 상태도 1과 상태도 2에 대한 병행 기계

병행 기계 HM_1 과 HM_2를 ROBDD 로 표현한 그림은 아래와 같다.

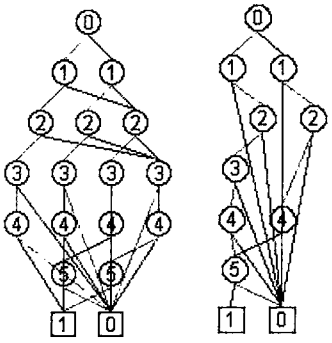

그림 3. 병행 기계 HM_1에 대한 $\overline{T}$ 와 $\overline{Out}$

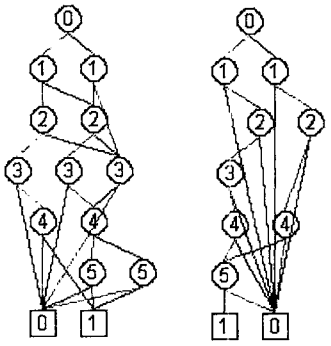

그림 4. 병행 기계 HM_2에 대한 $\overline{T}$와 $\overline{Out}$

병행 기계 HM_1과 HM_2에 대한 곱 기계를 ROBDD로 표현한 그림은 다음과 같다.

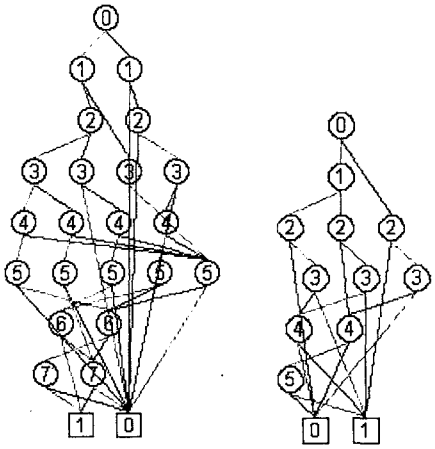

그림 5. 곱 기계 $\overline{T_{PM}(\sigma,\sigma')}$와 곱 기계의 $\overline{EQUI(\sigma_{PM})}$

여기서 0부터 7은 ROBDD 노드의 순서를 의미한다. 따라서, $\overline{T_{PM}(\sigma,\sigma')}$에서 노드 0은 곱 기계에서 2로 해석된다. 그러므로, {0→2, 1→3, 2→4, 3→5, 4→6, 5→7, 6→8, 7→9}로 해석된다. $\overline{EQUI(\sigma_{PM})}$ 도 {0→2, 1→4, 2→6, 3→8, 4→10, 5→11}을 의미한다. 곱 기계의 초기상태 <2:0, 4:0, 6:0, 8:0>로 부터 1 번째 새로 도달 한 값 <2:0, 4:1, 6:0, 8:1><2:1, 4:0, 6:1, 8:0> 이고, 2 번째 새로 도달 한 값 <2:1, 4:0, 6:1, 8:1>이며, 3 번째 새로 도달 한 값은 F 이다. 곱 기계에서 초기 상태로부터 모든 새로 도달한 값 $\overline{NEW_k}$ 에서 곱 기계의 $\overline{EQUI(\sigma_{PM})}$ 를 함축하고 있다. 또한 3번째에서는 새로 도달한 값이 없는 고정점에 도달함으로써 두 상태도는 동치이다.

5. 결론 및 향후 연구

두 상태도의 동치 검사를 수행하기 위해서, 각 상태도를 병행 기계로 나타내고, 병행 기계는 ROBDD 를 이용하여 기호적으로 표현했다. 그런 후, 곱 기계를 통해 동치 검사를 수행하여 두 상태도가 동치인지를 검사하였다. 평탄화는 상태도를 하나의 큰 유한상태기계로 보기 때문에 기호적으로 표현할 때 복잡하며, 명세의 수정 및 삭제가 발생하는 경우 전체를 다시 수정해야 한다. 한편, 병행 기계를 사용할 경우에는 각 기계만 고려하면 되므로, 평탄화에서 발생되는 단점을 줄인다. 또 두 상태도가 기능적으로 같다는 것은 이벤트가 발생했으나 반응을 하지 않는 것도 포함되어야 하는데, 평탄화에서는 천이의 조건이 만족될 경우만 기호적으로 표현한다.

앞으로 계속 연구해야 하는 방향은 동치검사문제는 결국 도달성 검사의 문제로 단순화된다. 따라서, 도달성 검사를 효율적으로 수행하는 방법에 대해서 많은 연구가 이루어져야 한다. 특히, 도달성 검사를 전체 상태공간에 대해서 수행하지 않고, 꼭 필요한 부분만 도달성 검사가 이루어 진다면 매우 효율적일 것이다.

참고문헌

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

[2] D. Harel, A. Naamad, "The STATEMATE semantics of Statecharts," ACM Transactions on Software Engineering and Methodology, Vol.5, No.4, pp.293-333, 1996

[3] 구자철, 권기현, "기호적 기법을 이용한 상태도의 동치 검사", 제 5 회 산.학.연 SW 공학기술 학술대회, pp.214-219, 2001.

[4] J. Lind-Nielsen, et. al., "Verification of large state/event systems using compositionality and dependency Analysis," In Proceedings of TACAS'98, Lecture Notes in Computer Science 1384, pp.201-216, 1998.

[5] G. Behrmann, et. al., "Verification of hierarchical state/event systems using reusability and compositionality," In Proceedings of TACAS'99, Lecture Notes in Computer Science 1579, pp.201-216, 1999.

[6] A. Maggiolo-Schettini, A. Peron and S. Tini. "Equivalence of Statecharts," In Proceedings of CONCUR'96, Lecture Notes in Computer Science 1119, pp.687-702, 1996.

[7] G. Cabodi, et. al., "A new model for improving symbolic product machine traversal," In Proceedings of DAC'29, pp.614-619, 1992

[8] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. on Computer, Vol.35, No.8, pp.677-691, 1986