FSM의 행위 일치 알고리즘을 이용한 임베디드 시스템의 합성적 안전성 분석 기법

이 우 진†

요 약

실생활과 밀접한 임베디드 시스템들이 인터넷에 연결되어 점차 복잡해지고 시스템 사용 패턴 또한 다양해짐에 따라 임베디드 시스템의 안전성 문제가 대두되고 있다. 임베디드 시스템의 상호작용에 대한 안전성을 분석하기 위해서는 시스템 모델을 정형적으로 기술하고 이를 이용하는 체계적인 안전성 분석 방법이 필요하다. 이 연구에서는 Labeled Transition Systems (LTS)를 이용하여 시스템 모델과 속성 모델을 기술하고 안전성 속성이 만족되는지 검사하는 방법을 제공한다. 이 연구에서는 기존 합성적 안전성 분석 방법의 문제점을 해결하기 위해 시스템 모델을 속성모델 관점에서 축약하여 생성한 후에 두 모델 간의 행위 일치 관계를 이용하여 안전성 분석을 수행한다.

키워드 : 안전성 분석, 임베디드 시스템, 합성적 분석, 행위일치 알고리즘

Compositional Safety Analysis for Embedded Systems using the FSM Behavioral Equivalence Algorithm

Woo Jin Lee†

ABSTRACT

As the embedded systems closely related with our living become complex by interoperating each other via internet, the safety issue of embedded systems begins to appear. For checking safety properties of the system interactions, it is necessary to describe the system behaviors in formal methods and provide a systematic safety analysis technique. In this research, the behaviors of an embedded system are described by Labeled Transition Systems (LTS) and its safety properties are checked on the system model. For enhancing the existing compositional safety analysis technique, we perform the safety analysis techniques by checking the behavioral equivalence of the reduced model and a property model after reducing the system model in the viewpoint of the property.

Key Words : Safety Analysis, Embedded Systems, Compositional Analysis, Behavioral Equivalence Algorithm

1. 서 론

소프트웨어의 안전성 문제는 의료기기[1], 핵발전소[2], 항공 시스템[3] 등의 안전과 밀접한 실시간 시스템의 소프트웨어 개발에서 많이 연구되어 왔다. 최근 들어, 소프트웨어의 영역이 임베디드 시스템으로 확장되어 실생활과 더욱 밀접해짐에 따라 소프트웨어의 안전성 문제가 다시 대두되고 있다. 예를 들어, 냉장고, 에어컨, 보일러, 가스레인지 등의 기존 생활가전들을 인터넷으로 원격 제어하려는 시도가 많아지고 있다. 기존 생활가전들은 수십 년간 사용되는 과정에서 안전한 사용 패턴이 정립되어 안전성 문제가 발생치 않으나 원격제어 기능이 추가된 시스템에서는 다양한 형태의 상호작용 패턴들이 추가되어 복잡해질 수 있기 때문에 이러한 다양한 상호작용에 대한 안전성 검사가 필요하다.

일반적으로 소프트웨어 안전성 분석 과정은 소프트웨어의 안전성 속성들을 식별하는 단계와 식별된 안전성 속성들이 시스템에서 만족되는지 검사하는 단계로 나뉘어진다. 안정성 속성 식별 단계는 시스템 차원에서 위험을 파악하고 위험을 분석하는 단계로 기능적 요구사항에서 위험 요소를 찾는 FHA(Functional Hazard Assessment) 방법, 위험 요소의 원인을 하향식으로 분석하는 FTA(Fault Tree Analysis) 방법, 오류의 결과를 추적하는 FMEA(Failure Modes and Effects Analysis) 방법 등의 다양한 분석 방법들을 이용하여 안전성 속성들을 찾아낸다[4, 5]. 안전성 속성 검사 단계에서는 발견된 안전성 속성들이 시스템 모델에서 만족되는지 여부를 검사하는 단계로 모델 검사(model checking)[6], 수학적 증명(Theorem Proving)[7], 합성적 안전성 분석 방법[8] 등을 통해 분석한다. 일반적으로 수학적 증명 방법은 시스템

※ 본 연구는 BK 21 사업과 방위사업청과 국방과학연구소의 지원으로 수행되었습니다. (UD060048AD)
† 정 회 원 : 경북대학교 전자전기컴퓨터학부 조교수
논문접수 : 2007년 4월 2일, 심사완료 : 2007년 6월 15일

의 극히 일부분이나 아주 규모가 작은 시스템에 적용가능하며 모델 검사 방법도 시스템 규모가 방대해지면 적용하기 어려운 단점이 있다. 이에 반해 합성적 분석 방법은 계층적이고 점진적인 접근 방법을 취하므로 상대적으로 다른 방법에 비해 큰 규모의 시스템에도 적용이 가능하다. 하지만 합성적 안전성 분석 방법도 안전성 분석에 몇 가지 문제점을 가지고 있다. 기존 합성적 안전성 분석에 대한 상세한 설명과 단점은 2장에서 다룬다.

이 연구에서는 임베디드 시스템들간의 상호작용에 대한 안전성 속성을 분석하는 방법으로 기존 합성적인 분석 방법을 보완 및 수정하여 사용한다. 시스템 모델은 구성요소인 컴포넌트별로 Labeled Transition Systems(LTS) 모델[9]을 기술하고 안전성 속성은 유한상태머신으로 표현한다. 안전성 속성은 항상 만족되어야 하는 불변(invariant) 속성과 절대 일어나지 않아야 하는 제약사항(constraints)으로 구분된다. 불변 속성은 시스템 모델에서 항상 나타나는지 검사하며 제약사항은 부정적으로 표기되므로 대우를 취하여 시스템에 나타나는지를 검사한다. 안전성 속성 분석은 시스템 모델을 속성 모델에 대해 합성적인 방법으로 축약한 다음, 축약된 모델과 속성 모델의 행위 일치 관계를 비교하여 검사한다. 기존 합성적 안전성 분석과의 차이점은 기존 분석에서는 특정 상태에 도달하는지 여부만으로 판단함으로 불변속성을 제대로 분석하지 못하거나 제약사항을 분석하지 못하는 사례가 발생하는 반면, 제안된 합성적 분석에서는 이러한 문제점을 해결하고 있다.

이 논문의 구성은 다음과 같다. 제2장에서는 유한상태머신과 LTS에 대해 정의하고 기존 합성적 안전성 분석 방법의 문제점에 대해서 다루며 제3장에서는 예제시스템인 원격제어 가스오븐 시스템을 LTS로 모델링하고 안전성 속성을 유한상태머신으로 기술하는 방법을 제공한다. 제4장에서는 유한상태머신 간의 행위일치를 검사하는 알고리즘을 제공하며 제5장에서는 점진적으로 시스템 모델을 축약하고 행위 일치 알고리즘을 이용한 합성적 안전성 분석 방법을 설명한다. 제6장에서는 합성적 안전성 분석 방법의 상태 공간과 수행시간을 분석하여 합성적 안전성 분석 방법의 효율성을 보인다. 그리고 제7장에서는 결론과 향후 연구방향을 기술한다.

2. 연구 배경

2.1 유한상태머신(Finite State Machines: FSM)

유한상태머신은 다양한 분야의 시스템 모델링 및 분석에 많이 활용되어 오고 있으며 또한 Statecharts[13], LTS 등의 다른 정형적 기법의 기반 정형 언어로 사용되고 있다. 유한상태머신은 아래와 같이 정의된다. 시스템의 행위는 시스템 상태와 상태간의 정의로 표현되고 시스템에서 발생하는 이벤트 등은 입력 알파벳으로 상태전이와 연계하여 표현한다. 그리고 FSM에서는 초기 상태와 종료 상태집합을 가진다. FSM은 개념이 간단하고 구조가 단순하여 소규모 시스템의 모델링 및 분석에 많이 이용되지만 시스템의 규모가 커지고 병행적인 특성이 포함되면 시스템의 상태의 수가 급증하는 문제점이 있다.

정의 1. 유한상태머신

FSM = $(\Sigma, S, \delta, s_0, F)$ 모델은 아래와 같이 5개의 핵심요소로 정의된다.

- Σ는 입력 알파벳을 나타낸다.
- S는 시스템 상태의 집합이다.
- $\delta \subseteq S \times \Sigma \times S$는 상태전이를 나타낸다.
- s_0는 초기상태를 나타낸다.
- F는 종료 상태의 집합을 나타낸다.

2.2 Labeled Transition Systems (LTS)

시스템이 여러 서브시스템으로 구성된 경우, FSM에서는 서브시스템의 상태들을 모두 조합하여 하나의 시스템 상태로 표현하여야 한다. 이 때, 서브시스템 내에 병행적인 특성이 있으면 생성되는 상태 조합이 엄청나게 많아져서 FSM으로는 효율적인 표현이 불가능하다. Labeled Transition Systems (LTS)[9, 10]은 이러한 FSM의 단점을 보완하기 위해 제안된 방법이다. LTS 모델에서는 각 서브시스템을 독립적으로 모델링하고 서브시스템 모델간의 연관관계는 레이블을 이용하여 나타낸다.

정의 2. LTS (Labeled Transition System)

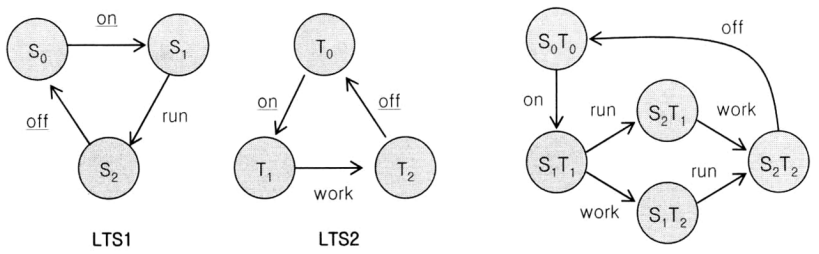
LTS = (Σ, S, δ, s_0) 모델은 아래와 같이 4개의 핵심요소로 정의된다.

- Σ는 LTS에 나타나는 이벤트의 레이블의 집합이다. 이벤트 레이블은 내부 이벤트와 다른 LTS 모델과 통신을 위해 사용되는 공유 이벤트로 구분된다.
- S는 시스템 상태의 집합이다.
- $\delta \subseteq S \times \Sigma \times S$는 상태전이를 나타낸다.
- s_0는 초기상태를 나타낸다.

LTS에서는 FSM과 달리, 종료상태집합을 표현하고 있지 않다. LTS 모델에서는 모든 상태가 종료 상태가 될 수 있다고 간주한다. 여러 개의 서브시스템으로 구성된 LTS 모델을 분석하기 위해서는 일반적으로 전체 시스템 관점의 FSM 모델로 변환한다. LTS 모델에서 트랜지션은 공유 트랜지션과 지역 트랜지션으로 나뉘어진다. 공유 트랜지션은 여러 LTS 모델에 레이블이 공유되는 경우로 병행적 합성(parallel composition)[9]에 의해 수행되며 해당 레이블을 포함하는 각 LTS 모델의 트랜지션이 동시에 수행되어야 한다. 반면, 지역 트랜지션은 하나의 LTS 모델에만 나타나는 트랜지션이므로 독립적으로 수행될 수 있다. (그림 1)은 2개의 LTS 모델과 이를 전체시스템 관점에서 합성된 FSM 모델을 보여준다. 그림에서는 공유 트랜지션을 지역 트랜지션과 구분하기 위해 밑줄로 강조하여 표시하고 있다. 공유 트랜지션인 on은 LTS1과 LTS2 모델이 각각 S_0, T_0일 때만 수행되며, LTS1의 지역 트랜지션인 run은 LTS2의 상태와 무관하게 S_1일 때에 수행된다.

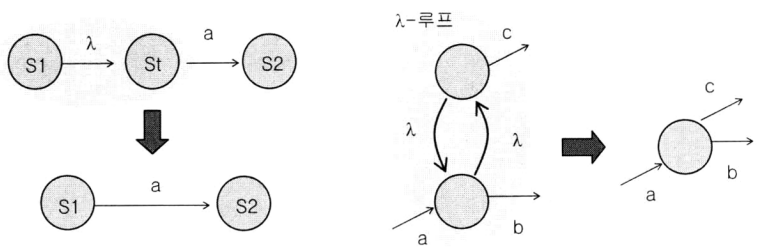
2.3 유한상태머신의 축약방법

유한상태머신으로 기술된 시스템 모델의 축약 방법[11]은 우선 관심이 없는 이벤트들을 λ 트랜지션으로 변환한 후에



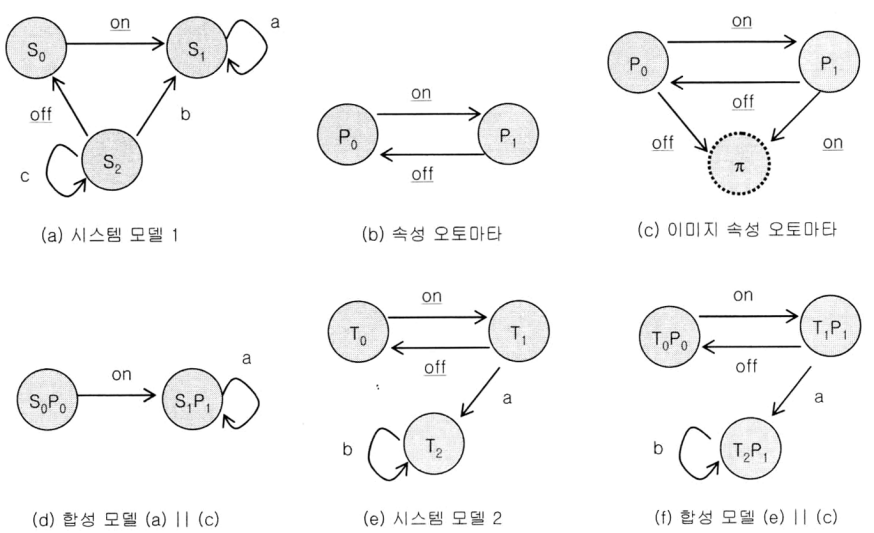
(a) LTS 모델 (b) 합성된 FSM 모델

(그림 1) LTS 모델과 합성된 FSM 모델의 예



(a) λ-트랜지션 축약 (b) λ-루프 축약

(그림 2) 시스템 모델 축약과정의 예



(a) 시스템 모델 1 (b) 속성 오토마타 (c) 이미지 속성 오토마타

(d) 합성 모델 (a) || (c) (e) 시스템 모델 2 (f) 합성 모델 (e) || (c)

(그림 3) 합성적 안전성 분석 방법의 예제

λ 트랜지션을 축약하는 방법을 사용한다. λ 축약 방법은 λ 루프 제거와 독립적인 λ 트랜지션 축약 규칙이 있다. (그림 2(a))는 λ 트랜지션 축약 방법으로 독립적으로 수행되는 내부 λ 트랜지션은 외부적으로 상태 변화가 드러나지 않으므로 축약할 수 있다는 것을 보여준다. 이때, 내부상태인 ST에는 λ 트랜지션이외에 다른 입력 트랜지션이 존재하지 않아야 한다. (그림 2 (b))는 λ 트랜지션으로 구성된 λ 루프는 외부적으로 상태변화가 드러나지 않으므로 축약할 수 있다는 것을 보여준다. λ 축약 규칙은 적용순서는 먼저 λ 루프를 제거한 다음, 독립적인 λ 트랜지션에 대해 λ 트랜지션 축약을 수행한다.

2.4 기존 합성적 안전성 분석의 문제점

합성적 안전성 분석 기법은 S. C. Cheung[8]에 의해 소개된 방법으로 시스템 구성요소들과 시스템 속성을 나타내는 LTS 모델들을 계층적으로 합성하고 내부 상태전이의 축약을 통해 상태폭발의 가능성을 최소화한 효율적인 안전성 분석 방법이다. (그림 3)은 합성적 안전성 분석 단계를 설명하고 있다. (그림 3 (a))에서 시스템의 행위 모델은 시스템의 구성요소별로 여러 LTS 모델로 나타낸다. 그리고 분석하고자 하는 시스템 속성을 속성 오토마타라 불리는 동일한 LTS 모델로 기술한다. 안전성 분석은 시스템 모델과 속성 오토마타를 곧바로 합성하지 않고 효율적인 분석을 위해 속성 오토

마타를 미정의상태(π)를 포함하는 이미지 속성 오토마타로 변환하여 합성한다. 미정의상태는 시스템의 오류상태를 의미하며 분석과정에서 오류상태에 도달하면 분석하는 안전성 속성이 만족되지 않음을 의미한다. (그림 3(c))는 (그림 3(b))의 속성 오토마타의 이미지 속성 오토마타를 보여준다. (그림 3(c))에서 P_0 상태에서는 off 이벤트가 일어나면 의도하지 않는 이벤트가 일어난 것이므로 오류상태로 상태전이가 발생한다.

안전성 분석 검사는 모든 LTS 모델들과 이미지 속성 오토마타 모델을 계층적으로 병행적 합성을 수행하여 이 과정에서 미정의상태(π)가 나타나는지 여부를 검사한다. (그림 3(a))의 예제에 사용된 시스템 모델에서는 on 이벤트 다음에 off 이벤트가 발생하지 못한다. (그림 3(b))의 속성 오토마타는 on 이벤트가 발생하면 반드시 off 이벤트가 발생하여야 하는 불변 속성을 나타내고 있다. 하지만 (그림 3(a))의 시스템 모델과 (그림 3(c))의 이미지 속성 오토마타를 병행적으로 합성하면 (그림 3(d))의 최종 합성 모델이 생성되는데 여기에서는 미정의상태(π)가 나타나지 않으므로 안전성 속성이 만족되는 것으로 분석된다. (그림 3(e))의 또다른 시스템 모델에서는 on 이벤트 다음에 off 이벤트가 발생하는 경우와 a 이벤트가 발생하면 off 이벤트가 발생하지 않는 경우가 있다. (그림 3(f))의 합성모델에 미정의상태가 나타나지 않으므로 안전성 분석 검사 방법에서는 속성이 만족되는 것으로 분석된다.

기존 합성적 안전성 분석의 문제점은 아래와 같은 세가지 유형으로 정리할 수 있다.

- 불변속성을 제대로 검사 못하는 경우: 불변속성이 만족되지 않는데도 검출을 못하는 사례로 (그림 3(a))와 같이 시스템 모델에 on 이벤트 다음에 off 이벤트가 일어나지 않는데도 기존 합성적 안전성 분석방법에서는 만족하는 것으로 분석된다.
- 속성 모델에 나타나지 않는 행위를 제대로 검사 못하는 경우: (그림 3(e))의 시스템 모델에서는 on 이벤트 다음에 off 이벤트가 나타나는 경우도 있지만 on 이벤트 다음에 a 이벤트가 발생하여 속성 이외의 행위가 발생한다. 이러한 경우도 기존 합성적 안전성 분석방법에서는 분석이 되지 않는다.
- 제약사항으로 표현된 안전성 속성의 분석이 불가능: 기존 합성적 분석 방법에서는 불변속성만을 분석할 수 있으며 반례로 분석을 수행하는 제약사항은 분석을 수행할 수 없다.

이 연구에서는 이러한 문제들을 해결하기 위해 근본적인 접근방법으로 시스템 모델과 속성 모델간의 행위 일치 관계를 비교하여 시스템 안전성 만족 여부를 검사하고자 한다.

3. 시스템 행위 모델링 및 안전성 특성 모델링

이 장에서는 예제 시스템인 가상의 원격제어 가스오븐 시스템에 대해서 간략히 설명하고 LTS를 이용하여 예제 시스템을 모델링하고 안전성 속성을 기술하는 방법을 설명한다.

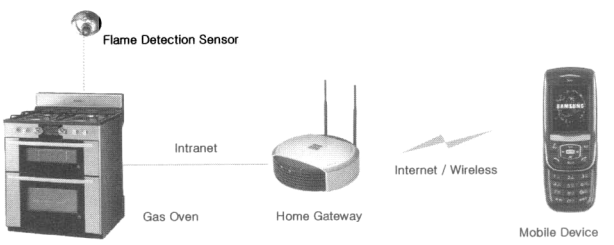
3.1 예제 시스템

원격제어 기능이 보편화되면서 기존의 전자제품에 원격제어 기능을 추가한 가상의 시스템을 예제시스템으로 사용한다. 원격제어 가스오븐 시스템은 기존 시스템에 원격제어 기능을 추가한 것으로 휴대 모바일 단말기에서 시스템을 원격으로 조작할 수 있는 기능을 가진다. (그림 4)는 원격제어 가스오븐 시스템의 구성도를 보여준다. 예제시스템에는 가스오븐의 불꽃을 확인할 수 있도록 하기 위해 불꽃감지 센서를 추가한다.

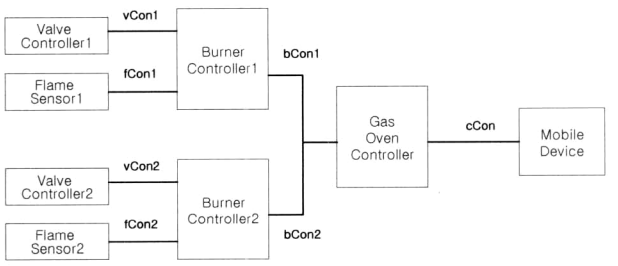
3.2 LTS를 이용한 시스템 모델링

이 절에서는 LTS를 이용한 시스템 행위 모델링 방법과 유한상태머신으로 안전성 속성을 기술하는 방법을 다룬다. (그림 5)는 버너가 2개인 원격제어 가스오븐 시스템의 블록 다이어그램을 보여준다. 버너가 더 추가될 경우는 버너 제어기 부분이 추가되며 가스 오븐 제어기 부분이 좀더 복잡한 형태가 된다.

시스템의 행위 모델은 시스템의 컴포넌트들을 각각 LTS로 모델링한다. (그림 6)은 예제 시스템의 일부 컴포넌트를 각각 LTS로 모델링한 예를 보여준다. 컴포넌트간의 의존성은 LTS 모델의 공유 레이블을 이용하여 나타내며 공유 레이블은 그림에서 밑줄로 나타내고 있다. 예를들어, 모바일 단말기의 con1, coff1 레이블은 가스 오븐 제어기와의 의존성을 표현하고 있다. 모바일 단말기에서 가스오븐을 켜면, con1이라는 이벤트가 발생한다. 이벤트 con1은 가스오븐제어기의 con1과 동시에 수행된 다음, 이벤트 bon1이 일어나서 버너 제어기의 bon1과 동시에 수행된다. 버너 제어기는 bon1 이후에 von1 이벤트가 생성되어 밸브 제어기의 von1

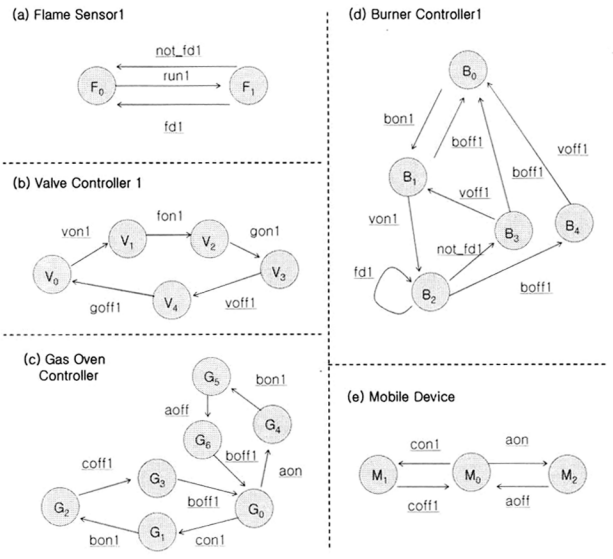


(그림 4) 원격제어 가스 오븐 시스템의 구성도

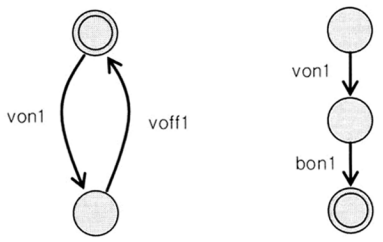


vCon1={von1, voff1}, **fCon1**={fd1, not_fd1}, **bCon1**={bon1, boff1}, **vCon2**={von2, voff2}, **fCon2**={fd2, not_fd2}, **bCon2**={bon2, boff2}, **cCon**={aon, aoff, con1, con2, coff1, coff2}

(그림 5) 원격제어 가스오븐 시스템의 블록 다이어그램



(그림 6) 예제 시스템의 주요 LTS 모델



(a) SP1 : von1→voff1 (b) SP2 : !(von1→bon1)

(그림 7) FSM을 이용한 시스템 속성 기술 예제

이 동시에 수행되면, 밸브제어기는 불꽃을 일으키고(fon1) 가스밸브를 열어(gon1) 점화시킨다. 버너 제어기에서는 von1 이벤트 후에 불꽃 센서에서 불꽃을 감지하지 못하면(not_fd1), 밸브 제어기에서 밸브를 닫도록 voff1 이벤트를 전달하고 점화과정을 다시 수행한다. 모바일 단말기에서 가스오븐을 끄는 과정도 유사한 절차로 진행된다.

3.3 FSM을 이용한 안전성 특성 기술

시스템의 안전성은 시스템 모델이 만족하여야 하는 특성으로 연관된 이벤트로 나타낼 수 있다. 합성적 안전성 분석 방법[8]에서는 시스템 속성을 속성 오토마타라는 LTS로 나타내고 있다. 하지만 LTS는 모든 상태가 수용 상태이므로 기술된 속성이 제대로 만족되는지 확인하기 어렵다. 이 연구에서는 시스템에 나타나는 이벤트를 이용하여 안전성 속성을 FSM으로 표현하고 속성 모델의 수용상태를 명확히 나타낸다. 시스템 속성은 크게 항상 만족되어야 하는 불변 속성과 하나의 사례도 나타나지 않아야 하는 제약사항으로 나뉠 수 있다. 시스템 속성 모델은 항상 긍정적인 형태로 표현되어야 하므로 제약사항은 대우명제로 변환하여 나타낸다. "가스밸브 제어기는 밸브가 열린 후에는 안전을 위해서 반드시 밸브를 잠궈야 한다."와 같은 불변 속성은 (그림 7(a))와 같이 나타낸다. 즉, von1 이벤트가 발생하면 반드시 voff1 이벤트가 일어나야 한다. 만약 von1 이벤트가 발생하였는데, voff1이 발생하지 않으면 시스템 속성이 만족되지 않은 것으로 간주한다. 시스템 속성 모델에서는 반드시 이중원으로 표현된 수용 상태에서 종료되어야 한다. "버너 제어기는 작동되기 전에 가스밸브 조작이 일어나서는 안 된다"와 같은 제약사항은 (그림 7(b))와 같이 대우명제인 "von1 이벤트 후에 bon1 이벤트가 수행될 수 있다"와 같은 속성을 나타낸다.

4. FSM 모델의 행위 일치성 검사

FSM의 일치성 검사 알고리즘은 1950년에 Huffman [11, 12]에 의해 제안되었으며 두 FSM을 하나로 합쳐 동일한 그룹의 상태들을 점진적으로 식별하는 분할 방법(partitioning)으로 일치성을 검사한다. 다른 유사한 방법으로 합성 모델을 만들어 두 모델의 행위일치 관계를 검사하는 bi-simulation 기법[9]이 있다. 하지만 이러한 두 방법들에서는 대상 시스템 모델 이외에 추가적으로 합성 모델의 저장 공간이 필요하다는 단점이 있다. 이러한 단점은 비교 대상 시스템이 방대해질 때 더욱 부각된다. 또한 시스템 모델과 속성 모델처럼 추상화 레벨이 서로 다른 경우에 적용하기 위해서는 기존 알고리즘을 추가적으로 확장하여 사용하여야 한다. 이 연구에서는 추상화 레벨이 서로 다른 두 유한상태머신의 행위 일치성 검사 알고리즘을 제안하며 이 알고리즘에서는 대상 모델 이외에 분석에 필요한 상태 공간을 최소화한다.

FSM의 행위 일치성 검사 알고리즘에서는 두 FSM의 상태간의 매핑 관계를 설정하고 매핑된 두 상태 그룹간에 입출력 트랜지션의 정보가 동일함을 보인다. 먼저 두 FSM의 초기상태를 서로 매핑한다. 그리고 추상화 레벨이 낮은 시스템 모델에서 초기상태에서 도달가능한 모든 상태들을 검색하면서 두 FSM 상태간의 매핑정보를 도출한다. 속성 모델은 모두 공유 트랜지션으로 되어 있지만 추상화 레벨이 낮은 시스템 모델 내에는 공유 트랜지션과 내부 트랜지션이 있다. 내부 트랜지션이 수행될 때는 수행 전후 상태의 상태 매핑 정보는 동일하게 설정하고 공유 트랜지션은 속성 모델의 상태변화의 값을 매핑정보로 설정한다. 두 FSM 모델의 상태정보 간의 일대일 매핑이 성립되기 위해서는 우선 양쪽 모두에 매핑되지 않는 상태가 존재하기 않아야 하며 매핑에 의해 동일한 상태그룹으로 묶여져 있는 그룹간에는 서로소(disjoint) 관계가 성립되어야 한다. (그림 8)은 시스템 모델의 상태를 검색하면서 속성 모델과의 매핑정보를 설정하는 재귀적인 함수인 mapping_traverse()의 의사코드를 보여준다.

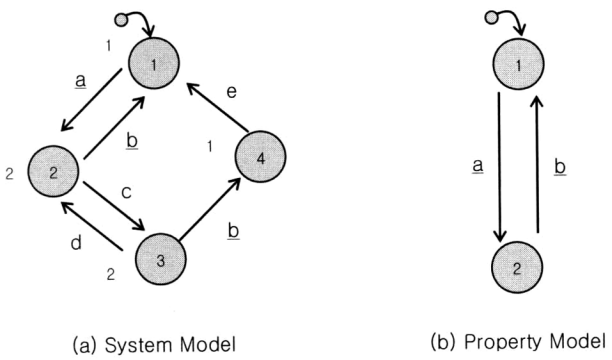
먼저, 행위일치 알고리즘의 첫번째 단계로 두 FSM 모델의 상태정보를 매핑하는 과정을 예로 살펴보자. (그림 9)는 두 FSM 모델을 연관된 상태그룹으로 매핑한 결과를 보여주고 있다. 두 FSM 모델 모두 초기상태는 상태1이다.

(그림 9(a))의 시스템 모델의 초기상태에서 매핑값을 1로 설정한다. 두 모델에서 공유 트랜지션은 a와 b이며 나머지 c, d, e는 시스템 모델의 내부 트랜지션이다. 두 모델 모두

```
bool mapping_traverse(FSM *property, int state_id, int mapping_id)
{
    if ( StateList[state_id].done )
            return;

    StateList[state_id].done = true;
    for ( each tran_i in StateList[state_id] ) {
        if ( shared (tran_i.id) ) {// shared transitions
            StateList[tran_i.to].mapping_id = property ->get_next(mapping_id, tran_i);
            if (StateList[tran_i.to].mapping_id < 0 )
                return false;
        }
        else {                              // internal transitions
            if (StateList[tran_i.to].mapping_id >= 0 &&
                StateList[tran_i.to].mapping_id != mapping_id )
                return false;
            StateList[tran_i.to].mapping_id = mapping_id;
        }
        mapping_traverse(property, tran_i.to, StateList[tran_i.to].mapping_id);
```

(그림 8) 두 FSM 모델의 상태매핑정보 설정하는 알고리즘



(a) System Model (b) Property Model

(그림 9) 두 FSM 모델의 상태 매핑 설정의 예

트랜지션 a가 수행되어 상태 2로 전이되므로 상태 2의 매핑값은 속성모델의 현재 상태인 2로 설정된다. 상태 2에서 트랜지션 b가 수행되면 속성모델도 상태1로 전이되므로 상태 1의 매핑값이 다시 1로 설정되는데, 이 값이 기존의 매핑값과 동일하므로 문제가 되지 않는다. 상태 2에서 내부 트랜지션 c가 수행되면 내부 트랜지션이므로 상태 3의 매핑값을 2로 설정한다. 이제 상태 3으로 가서 내부 트랜지션 d가 수행되므로 상태 2의 기존의 매핑값과 새로운 매핑값이 모두 2이므로 문제가 되지 않는다. 상태 3에서 트랜지션 b가 수행되면 속성 모델이 1로 바뀌므로 상태 4의 매핑값은 1로 설정된다. 상태 4에서 내부 트랜지션 e가 수행되면 동일한 매핑값 1이 설정되므로 상관없다. 결과적으로 두 FSM 간의 동일한 행위 상태를 매핑하면 { {1, 2}, { 3, 4} }와 { {1}, {2} }의 형태로 상태그룹이 형성되고 이들간에는 일대일 매핑관계가 성립됨을 알 수 있다.

이제 행위일치 알고리즘의 두번째 단계로, 두 FSM간의 연관된 상태그룹 간에 입출력 트랜지션 정보가 동일한지를 검사한다. 두 FSM 모델간의 트랜지션의 입출력 정보를 비교해보면, { 1, 2 }와 { 1 }의 입출력 트랜지션은 각각 a와 b로 동일하며 {3, 4}와 {2}의 입출력 트랜지션은 각각 b와 a로 동일하다. 그러므로 두 모델은 공유 트랜지션 a와 b에 대해 동일한 행위를 가짐을 알 수 있다.

이상과 같이, 두 추상화 레벨이 다른 FSM 모델의 행위 일치성을 검사하기 위해 두 모델간의 매핑 정보를 검사하여 설정하고 일대일로 매핑되는 두 상태그룹간의 입출력 트랜지션 정보를 비교하였다. 이러한 검사과정에서는 매핑정보를 저장하는 추가적인 공간만 소요되며 각 단계에서 한번씩 시스템 모델을 검색하면 되므로 효율적인 검사가 가능하다.

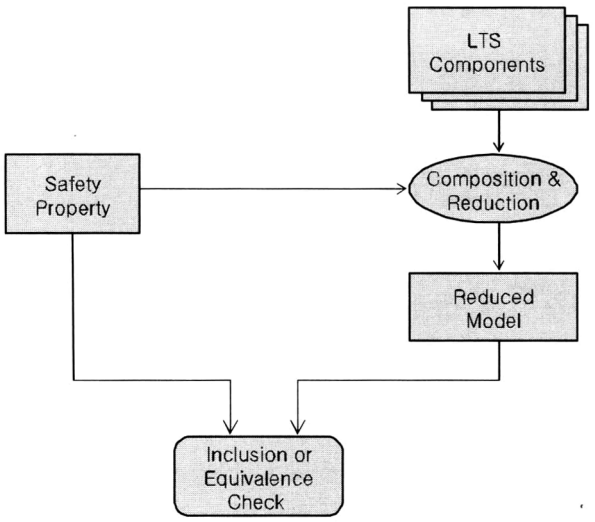
5. 안전성의 합성적 분석 방법

2장에서 살펴보았듯이, 기존 합성적 안전성 분석 방법에서는 불변속성과 제약사항으로 표현되는 안전성 속성들을 제대로 분석하지 못하는 사례들이 발생한다. 이러한 문제점들을 해결하기 위해 시스템 속성 관점에서 시스템 모델의 행위를 추상화하여 동일한 행위를 가지는지를 검사하고자 한다. 즉, 시스템 속성과 시스템 모델의 행위 일치 관점에서 안전성 속성이 만족되는지를 검사한다. 두 모델의 행위를 비교하기 위해 4장에서 제안한 행위 일치 검사 알고리즘을 사용할 수 있지만 시스템의 규모가 방대할 경우 시스템 모델에 대한 FSM을 생성하는 데에 어려움이 따른다. 합성적 안전성 분석 방법은 시스템 모델을 안전성 속성에 대해 축약하여 생성한 다음, 시스템 축약 모델과 속성 모델의 행위 일치 관계를 검사한다. (그림 10)은 합성적으로 시스템 모델을 축약하여 축약된 시스템 모델과 속성 모델간의 행위 일치 관계를 검사하는 과정을 보여준다. 시스템의 불변속성과 제약사항은 서로 다른 특성을 지니므로 아래와 같이 구분하여 분석한다.

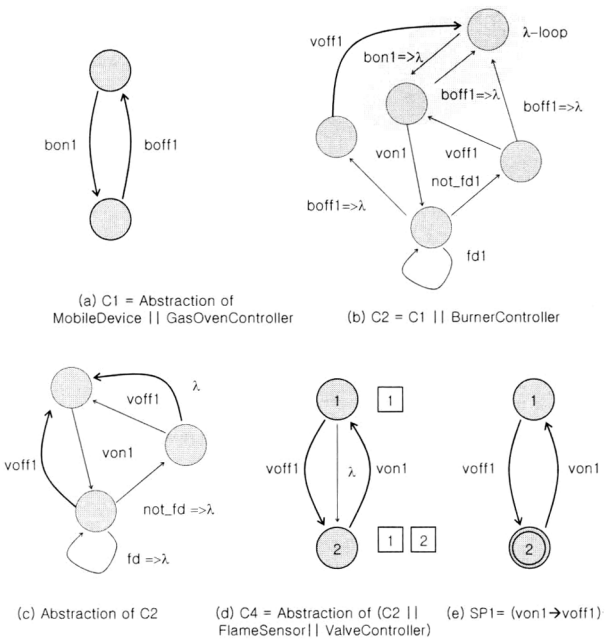
- 제약사항 분석: 제약사항은 시스템 모델에 일어나지 않아야 하는 특성이므로 대우 속성을 만족하는 하나의 사례만 찾아도 제약사항이 만족되지 않는 것이므로 대우 속성 모델의 행위가 시스템 모델에 포함되어 있는지 여부를 검사한다. 포함 관계의 검사는 속성 모델의 모든 트레이스가 시스템 모델에 나타나는지 여부로 이루어진다.
- 불변속성 분석: 불변 속성은 속성 모델의 행위들이 시스템 모델에 모두 반영되어야 하며 또한 속성 모델에 반하는 행위가 시스템 모델에 존재하지 않아야 한다. 이 경우는 두 모델간의 행위 일치 검사를 수행한다.

시스템 모델과 시스템 속성 모델간의 행위일치를 효율적으로 검사하기 위해 합성적 방법을 통해 시스템 모델을 속성 모델에 사용된 이벤트 집합으로 축약한 후에 행위일치 검사를 실시한다.

(그림 11)은 버너가 하나인 원격제어 오븐시스템을 합성적 방법으로 안전성을 분석하는 일부 과정을 보여준다. 버너가 하나인 시스템에서는 모바일 단말기에서 aon, aoff 와 같은 명령어가 필요치 않으므로 모바일 단말기와 가스오븐 제어기 부분이 (그림 4)에 나타난 모델보다 단순한 형태를 띤다. (그림 11(a))는 모바일 단말기와 가스오븐 제어기를 합성하여 축약한 모델인 C1을 나타낸다. (그림 11(b))는 C1



(그림 10) 행위 일치를 이용한 합성적 안전성 분석 과정



(그림 11) 점진적 합성 및 축약과정 수행하는 안전성 속성의 분석의 예

모델과 버너 제어기 모델을 합성한 모델인 C2를 보여준다. 합성된 모델에서 bon1과 boff1이 내부 트랜지션으로 바뀌므로 λ 루프와 λ 트랜지션을 제거할 수 있다. (그림 11(c))는 C2를 축약한 모델을 보여준다. (그림 11 (d))는 나머지 컴포넌트를 합성한 최종 축약 모델을 보여준다. 시스템의 축약 모델과 (그림 11(e))의 안전성 속성 모델의 행위일치성을 검사하기 위해 우선 두 모델의 상태 매핑관계를 설정하면 (그림 11(d))와 같이 나타난다. 그림에서 볼 수 있듯이, 매핑관계가 서로소 관계가 만족되지 않으므로 두 모델은 일치하지 않는다. 즉, SP1의 안전성 속성이 만족되지 않는다. 합성적 모델 생성과정을 역으로 추정하여 (von1→λ)$^+$가 발생하는 경우를 찾아보면 버너제어기 부분에서 B3 상태에서 B0 상태로 voff1 없이 초기상태로 상태전이가 발생하는 오류를 발견할 수 있다.

6. 비교 분석

합성적 안전성 분석에 소요되는 시간과 생성되는 상태공간을 분석하기 위해 Intel 2.4Ghz CPU와 2GB 메모리, Windows XP의 PC 환경에서 MS Visual C++에서 텍스트 기반의 합성적 안전성 분석 알고리즘을 구현하였다. 논문에서 구현한 알고리즘은 LTS 모델을 FSM으로 합성하는 병행적 합성 알고리즘, 내부 트랜지션을 추상화하는 축약 알고리즘, 추상화 레벨이 다른 FSM 모델의 행위일치 알고리즘, 그리고 이들을 활용하는 합성적 안전성 분석 알고리즘이다. <표 1>은 이를 이용하여 버너가 하나인 원격제어 가스오븐 시스템에서 SP1을 합성적으로 분석하는 과정에서 생성된 상태 및 아크의 수를 나타내고 있다. <표 1>의 두번째 칼럼의 값들은 축약을 하지 않은 모델의 크기를 나타내고 세번째 칼럼의 값들은 단계적으로 합성되는 모델의 크기를 나타내며 네번째 칼럼은 단계적으로 합성된 모델을 축약한 모델을 나타낸다. 마지막 칼럼은 합성과정에 단계적으로 나타나는 원래 모델과 축약모델간의 축약율을 보여준다. 표에 나타난 바와 같이 FSM 모델에서는 분석모델의 크기가 급증하는데 반해 중간단계에서 상태 및 아크 축약을 수행하는 합성적 분석 방법에서는 분석 상태 모델을 적정한 크기로 유지됨을 알 수 있다.

<표 2>는 앞서 발견된 오류를 수정한 예제 시스템에서 버너의 수를 증가시키면서 SP1에 대해 합성적 안전성 분석 알고리즘들을 이용하여 분석한 시간들을 비교하여 보여준다. 세번째 칼럼의 첫번째 접근방법은 LTS 모델들을 단순히 계층적으로 합성하여 생성한 최종 FSM 모델에 대해 행

〈표 1〉 합성적 안전성 분석의 생성모델의 크기와 내부정보 축약율

생성된 모델	FSM 생성모델	합성과정 원모델	합성과정 축약모델	축약율(%)
C0 = MobileDevice	2(2)	2(2)	2(2)	0.0(0.0)
C1 = C0 || GasOvenController	4(4)	4(4)	2(2)	50.0(50.0)
C2 = C1 || BurnerController	10(19)	5(9)	3(6)	40.0(33.3)
C3 = C2 || FlameSensor	20(44)	3(6)	2(3)	33.3(50.0)
C4 = C3 || ValveController	76(180)	2(3)	2(3)	0.0(0.0)
Total	112(249)	16(24)	11(16)	31.3(33.3)

〈표 2〉 FSM의 행위 일치성을 이용한 SP1의 합성적 안전성 분석의 성능 비교

(단위:초)

버너 수	컴포넌트 수	합성적 모델 생성	합성적 모델 생성후 축약과정 적용	축약을 통한 합성적 모델 생성
1	5	1.609	1.625	1.640
2	8	7.094	7.109	2.672
3	11	844.953	851.297	3.859
4	14	277085.760	277968.900	5.735
5	17	—		10.344
6	20	—		30.906
7	23	—	—	106.922
8	26	-		784.985

위일치 알고리즘을 적용하여 안전성을 분석한 것이며, 네번째 칼럼의 두번째 접근 방법은 최종 FSM 모델을 축약 알고리즘으로 축약한 후에 행위일치 알고리즘을 적용한 것이며, 마지막 칼럼의 세번째 접근 방법은 합성과정에서 축약 알고리즘을 적용하여 계층적으로 최종모델을 생성하고 행위일치 알고리즘을 적용한 것을 나타낸다.

<표 2>의 첫번째와 두번째 접근 방법에서는 전체상태 공간을 생성한 후에 분석을 진행하는데, 버너의 수가 5개 이상이 되면 상태의 수가 수백만 이상이 되어 버리므로 분석에 사용한 PC에서는 메모리가 부족하여 더 이상의 분석을 하지 못하였다. 두 가지 경우 모두 버너 수가 증가함에 따라 분석에 소요되는 시간도 기하급수적으로 증가함을 알 수 있다. 또한 두번째 접근 방법에서는 안전성 분석이전에 추가적으로 축약 알고리즘을 적용한 것으로 오히려 축약 알고리즘을 적용하지 않은 첫번째보다 더 많은 시간이 소요됨을 보인다. 이는 λ-루프를 검사하는 축약 알고리즘이 행위일치 알고리즘보다 시스템의 규모에 더 민감하게 반응하기 때문이다. 마지막 접근 방법에서는 합성과정에서 속성 모델을 고려하여 분석 영역을 최소로 유지함을 알 수 있다. 그리고 버너의 수가 증가하더라도 안전성 분석 알고리즘의 수행시간은 상대적으로 완만하게 증가함을 알 수 있다.

7. 결론 및 향후연구

임베디드 시스템이 점차 생활과 밀접하게 사용되고 또한 여러 시스템이 연동되어 복잡해짐에 따라 시스템의 안전성 문제가 대두되고 있다. 이 연구에서는 임베디드 시스템의 상호작용에서 발생할 수 있는 안전성 문제를 체계적이고 효율적으로 분석할 수 있도록 합성적 안전성 분석 방법을 제시하였다. 합성적 안전성 분석 방법은 FSM의 행위 일치 개념으로 수행되며 안전성 속성에 따라 시스템 분석모델을 축약할 수 있으므로 효율적인 분석모델 생성이 가능한 장점이 있다.

현재 텍스트 기반의 분석과정이 지원되긴 하지만 시스템 모델을 텍스트로 입력하여야 하는 불편함이 있다. 효율적인 모델링과 분석을 수행하기 위해서는 시스템 모델과 안전성 속성 모델을 그래픽 편집기로 기술할 수 있어야 하며 분석과정을 시각적으로 나타내는 것이 필요하다. 향후 연구로는 그래픽 모델링 편집기 및 시뮬레이션 기능을 갖춘 합성적 안전성 분석기를 개발할 예정이다.

참 고 문 헌

[1] Edward A. Addy, "Methodology of independent software nuclear safety analysis," Proc. of 5th International Symposium on Software Reliability Engineering, pp.76-83, Nov., 1994.

[2] K. Sayre, J. Kenner, P.L. Jones, "Safety models : an analytical tool for risk analysis of medical device systems," Proc. of 14th IEEE Symposium on Computer-Based Medical Systems, pp.445-451, July 2001.

[3] A.C. Tribble, S.P. Miller, "Software safety analysis of a flight management system vertical navigation function- a status report," Proc. of the 22nd Digital Avionics Systems, Oct. 2003.

[4] A.C. Tribble, S.P. Miller, "Software intensive systems safety analysis," IEEE Aerospace and Electronic System Magazine, Vol.19, No.10, pp.21-26, Oct. 2004.

[5] Nancy G. Leveson, *Safeware : System Safety and Computers*, Addison-Wesley Publishing Company, 1995.

[6] G. Parthasarathy, M.K. Iyer, K.T. Cheng, and L.C. Wang, "Safety property verification using sequential SAT and bounded modeling checking," IEEE Design and Test of Computers, Vol.21, No.2, pp.132-143, Mar-Apr., 2004.

[7] W. Atkinson, J. Cunningham, "Proving properties of a safety-critical system," Software Engineering Journal, Vol.6, No.2, pp.41-50, Mar., 1991.

[8] S. C. Cheung, J. Kramer, "Checking safety properties using compositional reachability analysis," ACM TOSEM, pp. 49 78, 1999

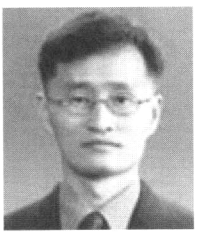
[9] Robin Milner, Communication and Concurrency, Prentice Hall, 1989.

[10] S. Uchitel, et al., "Synthesis of behavioral models from scenarios," IEEE trans. on software engineering, Vol.29, No.2, pp.99-115, 2003.

[11] P. J. Denning, et al., *Machines, Languages, and Computation*, Prentice Hall, 1978.

[12] D. A. Huffman, "The synthesis of sequential switching circuits," Journal of Franklin Institute, 1954.

[13] D. Harel, "Statcharts: A visual formalism for complex systems," Sci. Comput. Prog., Vol.8, pp.231-274, 1987.



이 우 진

e-mail : woojin@knu.ac.kr

1992년 경북대학교 컴퓨터과학과(학사)

1994년 한국과학기술원 전산학과(공학석사)

1999년 한국과학기술원 전산학과(공학박사)

1999년~2002년 한국전자통신연구원 S/W공학연구부 선임연구원

2002년~현 재 경북대학교 전자전기컴퓨터학부 조교수

관심분야 : 실시간 임베디드 시스템 모델링 및 분석, Requirements Engineering, Petri nets, 웹 서비스 기술 등