SyncCharts를 이용한 실시간 시스템의 정형 명세와 검증*

김성재[0], 최진영
고려대학교 컴퓨터학과
{sjkim, choi}@formal.korea.ac.kr

Formal Specification and Verification of Real Time System using SyncCharts

Sung-Jae Kim [0], Jin-Young Choi
Dept of Computer Science & Engineering, Korea University

요 약

실시간 시스템은 신뢰성이 중요하다. 특히 고안전성 시스템 (Safety-Critical System)은 안전과 직결되므로 높은 신뢰도가 요구된다. 본 논문에서는 Reactive System 의 모델링 및 검증을 위해 개발된 정형 검증 언어인 SyncCharts 를 이용한 실시간 시스템의 스케줄가능성 분석 (*Schedulability Analysis*)을 통해 시스템에 대한 요구조건의 만족여부와 태스크들의 수행가능성을 검증하는 방법을 제시한다

1. 서 론*

원자력발전 및 항공 시스템과 같은 실시간 시스템은 대표적인 고안전성 시스템(Safety-Critical System)으로 높은 신뢰도가 요구된다. 때문에 그러한 실시간 시스템의 설계 및 구현에 대한 안전성을 보장하기 위해 다양한 연구가 진행되어 왔다.

본 논문에서는 실시간 시스템의 안전성 검증을 위한 방법으로 정형기법[1]을 제시한다. 정형 기법(Formal methods)은 수학과 논리학에 기반을 둔 방법으로 하드웨어 시스템이나 정형 소프트웨어 시스템을 명세하거나 검증하는 방법론들이다. 수학적 기호를 사용하여 시스템을 명세하고 검증할 특성 또한 논리식을 통해 기술하여 시스템의 사용자가 요구하는 특정 조건에 대한 만족여부를 수학적 성질을 이용하여 검증함으로써 자연어가 내포하는 애매모호함이나 불확실성을 최대한 줄일 수 있다. 즉 복잡한 시스템이 특정 조건을 만족하는지를 검증하여 검증된 시스템에 대한 신뢰성을 가지게 할 수 있는 것이다.

정형 기법은 정형 명세 (formal specification)와 정형 검증(formal verification)의 두 가지로 나눌 수 있는데, 정형 명세는 정형 논리(formal logic) 또는 수리 논리 (mathematical logic) 등을 이용하여 시스템이 동작할 환경 , 시스템이 만족해야 할 요구 사항, 요구사항을 수행할 시스템 설계 등을 기술하는 것이다.

정형 명세는 다시 요구 명세와 설계 명세로 나눌 수 있다. 요구 명세는 시스템이 무엇을 만족해야 하는가를 정의해 놓은 명세이고 설계 명세는 시스템이 어떻게 이루어져 있는가를 나타낸다

정형 검증은 정형 논리 또는 수리 논리등에서 제공하는 증명 방법을 이용, 정형 명세를 분석하여 시스템의 무모순성 및 완전성을 검증하거나, 설계가 주어진 가정에서 요구사항을 만족하는지를 검증하는 기법이다.

본 논문에서는 이러한 정형 검증 연구의 일환으로, Reactive System 의 모델링 및 검증을 위해 개발된 도식적 정형검증 언어인 SyncCharts[2]를 이용한 실시간 시스템의 스케줄가능성 분석을 통해 보다 신뢰성 있는 실시간 시스템을 설계하는 방법을 제시한다

2. 도식적 정형기법 언어 SyncCharts

정형 기법은 여러 가지 장점에도 불구하고 명세 및 검증이 쉽게 이해하기 어려운 수리식으로 표현되기 때문에 시스템의 설계 및 구현에 참여하는 사람들이 접근하기 어려운 경향이 있었다. 따라서 최근에 이르러서는 동시적인 수행을 표현하는 상태 다이아그램으로 행위를 묘사하고 있는 Statechart 나 Modechart, Timed-automata,

* 본 연구는 과학기술부 원자력연구개발사업 중 원전계측제어시스템 연구개발사업 위탁연구 과제로 수행되었음

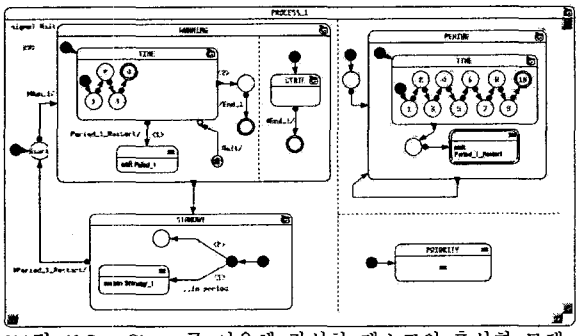
CRSM(Communicating Real-time State Machine)과 같은 정형적이고 도식적인 언어가 개발되기에 이르렀다. 본 논문에서는 이와 같은 도식적 언어의 하나인 SyncCharts 를 이용한다. SyncCharts 는 Reactive System 의 Textual 한 모델링 언어인 Esterel 의 Semantics 에 Statechart 의 외형적 특성을 결합한 도식적 언어로 모델링의 시각적 작성과 formal 한 semantics 의 두 가지 장점을 통해 시스템을 모델링하며, 모델 체킹 기법을 통해, 명세된 시스템에 대해 어떤 특성(Property)의 만족여부를 확인해봄으로써 시스템을 검증해볼 수 있다.

3. Rate Monotonic 알고리즘과 스케줄 가능성 분석

Rate Monotonic[3] 알고리즘은 실시간 시스템에 적용되는 스케줄링 기법으로, 태스크의 중요성 여부와는 관계없이 실행주기가 짧은 태스크일수록 더 높은 우선순위를 갖는 알고리즘이며, 스케줄가능성 분석(schedulability analysis)이란 어떤 스케줄링 알고리즘으로 스케줄되는 실시간 시스템이 데드라인(deadline)내에 일을 끝낼 수 있는지를 검사하는 것이다. 주기적으로 반복되는 시스템에서 어떠한 데드라인도 미스(miss)되지 않는다면 (즉, 데드라인을 만족하며 계속 수행된다면) 태스크들은 같은 동작을 무한히 반복하며 스케줄 가능함을 의미한다.

4. 시스템의 모델링

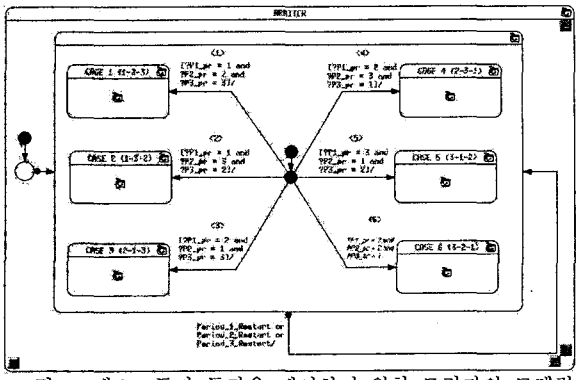
본 논문에서는 Rate Monotonic 알고리즘으로 스케줄링되는 3 개의 태스크로 구성된 실시간 시스템을 대상으로 모델링하며 SyncCharts 를 이용한 정형 검증을 통해 시스템의 스케줄 가능성을 분석한다.



[그림 1] SyncCharts 를 이용해 작성한 태스크의 추상화 모델

[그림 1]은 실시간 시스템이 가지는 태스크의 행위를 추상화해서 모델링한 그림이다. 해당 태스크는 자신만의 요구 실행시간(time unit to execute)과 주기(period)를 가지고 있으며 자신의 현재 상태를 지속적으로 보고하면서 일련의 작업을 수행한다. 태스크들은 수행(Running), 대기(Standby), 정지(Suspended)의 3 가지 상태를 가질 수 있으며, 수행상태에 있는 동안은 해당 태스크에 주어진 로컬 시그널의 값을 증가시키는 연산을 수행하며 작업중임을 나타낸다.

SyncCharts 로 명세한 위 시스템은 [그림 1]과 같은 태스크 3 개로 이루어지며 Rate Monotonic 알고리즘에 따라 각각의 작업들을 수행하되 공유 자원은 사용하지 않고 CPU 만을 사용하는 경우로 한정한다.

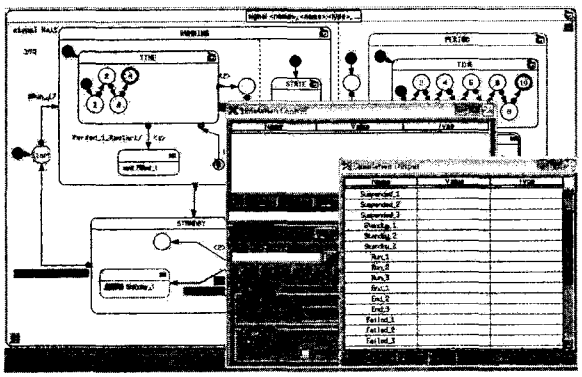


[그림 2] 태스크들간 동작을 제어하기 위한 조정자의 모델링

[그림 2]는 [그림 1]에서 모델링한 태스크들이 Rate Monotonic 알고리즘의 규칙을 따르며 작업을 할 수 있도록 도와주는 일종의 조정자(Arbiter)를 명세한 그림이다. 조정자는 3 개의 태스크들이 방출하는 우선순위를 기준으로 태스크들의 행동을 제어할 수 있도록 적절한 시그널을 발생시키는 작업을 수행한다.

SyncCharts 를 통해 모델링한 전체 시스템은 모두 4 개의 매크로 스테이트들로 이루어져 있으며 3 개의 태스크와 1 개의 조정자로 구성되어 있게 된다. 각각의 매크로 스테이트들은 서로간에 브로드캐스팅되는 시그널들을 통해서 동기화하며 주어진 작업을 수행한다.

5. 시뮬레이션 및 검증

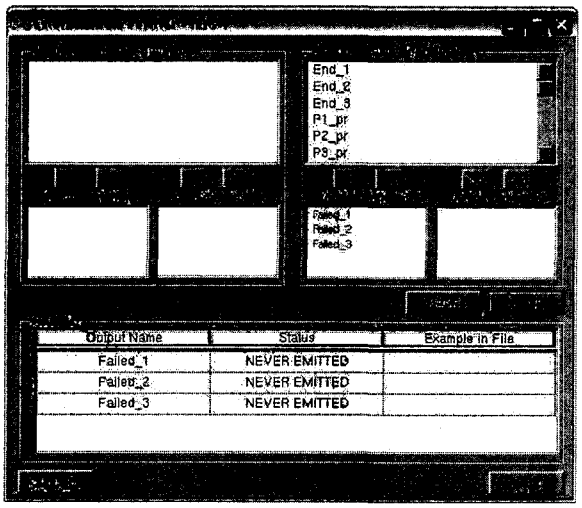


[그림 3] 시뮬레이션 화면

SyncCharts 에서는 명세한 시스템이 올바르게 작동할 수 있는지를 확인해 볼 수 있는 수단으로 시뮬레이션과 검증의 두 가지 방법을 제공한다. 시뮬레이션은 시스템에 어떤 입력 값을 주고 그 입력 값에 따른 스테이트 다이어그램의 상태 변화와 출력되는 시그널을 통해서 결과가 올바른지를 확인해 보는 것이다.

모델링한 실시간 시스템에서 작동되는 태스크들은 각자의 주기에 따라서 요구되는 시간동안 CPU 를 점유하며 작업을 수행하게 되며, 데드라인 내에 해당 작업을 마치지 못할 경우에는 **Failed** 시그널을 발생시킨다. 따라서 해당 시스템이 어떠한 **Failed** 시그널도 발생시키지 않는다면 시스템은 주어진 일을 계속 수행하면서 스케줄 가능함을 의미한다.

위의 [그림 3]은 시스템의 스케줄 가능성을 분석하기 위해 시뮬레이터를 사용한 결과 화면이다. 도식적으로 표현되는 태스크들의 상태와 발생되는 시그널들을 통해서 태스크들이 주기와 요구실행시간을 만족시키며 작업을 하고 있음을 보여준다.



[그림 4] 검증 결과

[그림 4]는 검증 기법을 통해서 해당 시스템이 스케줄 가능한지를 분석하는 그림이다. SyncCharts 에서 제공하는 검증기법은 시스템 사용 시 발생 가능한 모든 상태를 표현하는 유한상태기계(Finite State Machine)로 시스템을 바꾼 후 등가관계(Bi-simulation equivalence)기법을 통해 유한상태기계의 상태를 줄인 후에, 상태 탐색[4] 기법을 이용하여 특정 조건 하에서의 특정 시그널 발생 여부를 확인하는 것이다. 해당 조건과 요구되는 시그널은 시제 논리로 표현할 수 있으며 이러한 시그널의 발생 유무를 확인하는 방법을 통해 시스템의 Safety 나 Fairness 를 검증할 수 있다.

위의 [그림 4]의 결과는 앞서 모델링한 실시간 시스템은 해당 시스템이 가질 수 있는 어떠한 상태에서도 결코 Failed 신호를 발생시키지 않는다는(Never emitted) 사실을 보여준다. 이를 통해 이 실시간 시스템은 데드라인을 만족시키며 자신의 작업을 계속해서 수행해 나갈 수 있음을 나타낸다.

6. 결론 및 향후 과제

실시간 시스템은 그 성격상 설계부터 구현까지 그 안전성과 신뢰도에 대한 충분한 검증이 필요하다. 본 논문에서는 실시간 시스템의 모델링에 정형 기법을 도입해 봄으로써 보다 신뢰성 있는 시스템을 모델링 할 수 있는 사실을 확인해 보았다.

본 논문에서 명세한 시스템은 Rate Monotonic 알고리즘을 사용한 시스템이지만, 마찬가지의 방법으로 EDF 나 LLF 와 같은 알고리즘을 사용한 시스템 역시 동일한 방법으로 명세 가능하다. 또한 본 논문에서 명세하고 검증한 실시간 시스템은 자원을 사용하지 않고 CPU 만을 사용하는 경우만으로 한정했으나, 앞으로는 태스크들이 자원을 사용하는 경우와 그러한 경우 발생할 수 있는 우선순위 역전(priority inversion)현상을 해결할 수 있는 Priority Inheritance Protocol 이나 Priority Ceiling Protocol 과 같은 알고리즘을 도입한 시스템을 명세해 봄으로써 좀 더 복잡한 실시간 시스템을 명세하고 설계할 수 있는 방법을 제시할 계획이다.

7. 참고문헌

[1] Joost-Pieter Katoen, *Protocol validate*
[2] Charles ANDRE, SyncCharts: A Visual Representation of Reactive Behavior, 1996
[3] John Lehoczky, Lui Sha, Ye Ding, The Rate Monotonic Scheduling Algorithm, 1989
[4] Edmund M.Clake, Orna Grumberg, Doron A.Peled. *Model Checking*. The MIT press 1999
[5] Amar Bouali. XEVE - " an Esterel Verification Environment" . INRIA.France.1997
[6] Bernard Dion, Modeling and Implementing critical Real-Time systems with SyncCharts
[7] Jin-Young Choi, Insup Lee, and Hong-Liang Xie, " The Specification and Schedulability Analysis of Real-Time Systems Using ACSR" , Real-Time System Symposium, IEEE,1995
[8] David B.Golub, " Operating System Support for Coexistence of Real-Time and Conventional scheduling" , School of Computer Science, Carnegie Mellon University, Nov. 1994