SyncCharts를 이용한 UML Statecharts 의미론+

*이수영° *김진현 **이장수 *최진영
*고려대학교 컴퓨터학과
**한국원자력 연구소
*{sylee°, jhkim, choi}@formal.ac.kr, **jslee@kaeri.re.kr

Semantics Of UML Statecharts by SyncCharts

*Su-Young Lee° *Jin-hyun Kim **Jang-Soo Lee *Jin-Young Choi
*Dept. of Computer Science Korea University
**Korea Atomic Energy Research Institute

요 약

Statecharts는 UML에서 시스템의 행위를 표현하기 위한 핵심적인 언어로서 다양한 분야에 응용되고 있다. 그 의미론은 수학적인 방법으로 기술되어 있으나 실제로 응용하여 구현하는데에는 상당히 많은 과정을 거쳐야 한다. 본 논문에서는 UML Statecharts와 유사한 언어인 SyncCharts로 정의한다. SyncCharts는 Esterel의 정형명세 언어에 기반한 도식적인 언어로서 그 의미론은 물론 내장형 시스템의 코딩을 위해 잘 정의되고 진화된 언어이다. 본 논문에서는 SyncCharts를 이용하여 Statecharts의 의미론을 정의한다. 특히 실시간적인 행위 측면에서의 동기적 시간 의미론과 비동기적 시간 의미론을 모두 정의한다. 이렇게 함으로써 UML Statecharts의 실시간과 관련된 의미론을 정의한다. 그에 더하여 SyncCharts의 명세를 통해 어떻게 구현이 가능한지를 보임으로서 실제 Statecharts를 이용한 검증 및 구현 과정을 보인다.

1. 서 론

내장형 시스템을 비롯한 다양한 소프트웨어 시스템의 설계에 UML언어가 사용된다. 그 중 하나인 Statecharts[1],[3],[4]는 UML 객체의 행동을 명세하기 위해 사용한다. Statecharts는 상태도의 확장으로서, 설계의 편리함 및 이해도로 인해 반응형 시스템의 개발에 널리 사용되고 있다. 하지만 도식적인 언어는 설계가 복잡하고 커지면 그 이해가 어려워지기 때문에 그 설계를 확인(Validation)하기 위해 시뮬레이션이 반드시 필요하다. 하지만 UML Statecharts는 수학적 의미론은 가지고 있지만 확인 검증을 위한 시뮬레이션과 검증 기법이 마련되어 있지 않다. 따라서 본 논문은 Statecharts를 정형명세 언어인 Esterel을 도식화한 SyncCharts[2]로 행위적 의미론을 정의한다. SyncCharts는 Statecharts와 유사한 설계 언어이며, 시뮬레이션 및 정형검증 도구가 지원된다. 이러한 SyncCharts로 Statecharts의 행위적 의미론을 정의함으로서 설계의 행위적인 시뮬레이션을 수행하고 또한 정형검증함으로서 설계의 완전성 및 정확성을 보인다. 더 나아가 SyncCharts 도구를 이용하여 UML Statecharts를 어떻게 구현할 수 있는지 그 방법을 보임으로, UML 언어를 이용하여 내장형 시스템 소프트웨어의 구현 방법을 제시한다.

2장은 Statecharts 및 SyncCharts에 대해 각각 문법 및 의미론에 대해 소개하고, 3장은 SyncCharts를 이용한 Statecharts의 행위적 의미론에 대해 정의를 내리고 watchdog을 예제로 Statecharts로 구현된 행위를 SyncCharts로 변환한다. 마지막으로 4장에서 결론 및 향후 과제를 제시한다.

+ 본 논문은 "원자력 연구개발 중장기사업인 원전계측제어 사업단"에 의해 지원되었음.

2. Statecharts 와 SyncCharts 소개

2.1 Statecharts

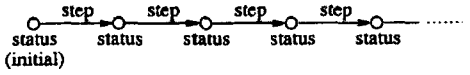
Statecharts는 시스템의 요구/설계 사항에 대한 시스템의 행위를 상태(state)와 전이(transition)들로 표현한다.

Statecharts의 상태는 OR 상태, AND 상태, 기본 상태 3가지 타입이 있다. OR 상태는 배타적인 관계의 상태를 의미하고 AND 상태는 병렬적인 관계의 상태를 나타내며 기본 상태는 상태 계층의 가장 하위에 있는 상태로 더 이상 하부로 나누어지지 않는 상태이다. 부모 상태는 하나 이상의 기본 상태를 가진 상위 계층의 상태를 의미한다. Statecharts의 전이는 e[c]/a로 표현된다. e는 이벤트로, 상태변화를 유발하고, c는 조건(condition)으로, 참 또는 거짓의 값을 가지는 것으로 상태변화의 조건이 된다. a는 동작(action)으로, 이벤트 발생 또는 데이터 아이템 값 변화 또는 조건값 변화를 발생시킨다.

Statecharts의 특징은 다음으로 간략하게 표현할 수 있다.

Statecharts =
상태도(State-diagram) + 직교성(Orthogonality) +
계층성(Hierarchy) + 브로드캐스팅(Broadcasting)

본 논문에서는 Statecharts의 행위를 정의하기 위해 가능한 런(run)의 집합으로 정의한다. 런이란 외부 환경으로부터 주어진 일련의 외부 자극에 대한 시스템의 반응을 의미한다. 런은 시스템이 가진 상태의 일련의 스냅샷으로 이루어지는데 이를 'status'라 한다. 런은 [그림 1]과 같이, status와 스텝으로 구성되는데, status는 시스템의 행위에 대한 특정 인스턴스에 대한 스냅샷이며 이러한 스냅샷은 스텝에 의해 진행된다. 이러한 스텝은 외부 자극에 의해 유발된다. Statecharts의 이러한 행



[그림 1] Statecharts의 전이 행위

위에 다음과 같은 제약을 가한다.

1) 내,외부 이벤트들에 대한 반응과 스텝에서 일어나는 변화는 그 스텝이 종료 후에만 감지된다.
2) 이벤트들은 그들이 발생한 다음 오직 한 스텝에만 유효하며 다음 스텝들에는 기억되지 않는다.
3) 스텝의 초기 상태를 기반으로 한 스텝의 연산(Calculation)이 일어난다.

다음 절에서는 이러한 의미론을 바탕으로 SyncCharts를 이용하여 행위에 대한 의미론을 기술한다.

2.2 SyncCharts

SyncCharts는 Statecharts에서 시스템의 행위를 표현하는 방법과 같이 상태와 전이를 이용한다. 상태는 가장 하부 상태인 기본 상태와 계층 구조를 갖는 매크로 상태가 있는데 매크로 상태는 하부 컴포넌트의 종류에 따라 다시 도식형 매크로 상태(grapic macro state)와 서술형 매크로 상태(textual macro state)로 구분된다. 전이는 t[c]/e로 t는 동기(trigger)로 시그널 상태에 대한 불린(true이면 수행) 표현식이다. 해당 시그널 발생 유무에 따라 참 또는 거짓을 가지며, "not","and","or"등의 연산자를 갖는다. c는 조건으로 연산자나 함수 호출을 이용해 변수나 시그널들을 결합하는 데이터 표현식이다. e는 결과(effect)로, 전이가 이루어졌을때 방출시킬 시그널들을 나타낸다.

SyncCharts의 특징 역시 Statecharts와 마찬가지로 다음과 같이 표현한다.

SyncCharts =
상태도(State-diagram) + 직교성(Orthogonality) +
계층성(Hierarchy) + 브로드캐스팅(Broadcasting)

Statecharts에 동기적 가설(Synchrony Hypothesis)을 적용, SyncCharts를 이용하여 위의 명세된 의미론을 다음 장에서 정형적으로 기술하고 시뮬레이션 및 정형검증을 한다.

3. SyncCharts를 이용한 Statecharts의 행위적 의미론 정의

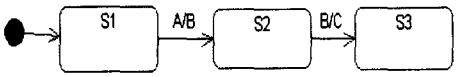
이 장에서는 Statecharts의 상태, 전이를 SyncCharts의 상태, 전이로 정의하는 규칙을 제시하고 Statecharts의 행위에 대해 2.1절에서 제시한 제약조건들을 만족하도록 SyncCharts를 이용하여 정형적으로 표현한다.

규칙1) 상태들

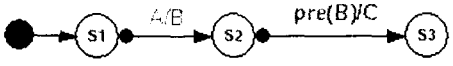
모든 Statecharts의 상태들은 SyncCharts의 상태들로 명세한다.

규칙2) 이벤트들과 동작들

전이를 일으키는 동기 이벤트가 내부적으로 발생한 것이라면 pre()(이전 tick에서 발생한 시그널을 기억한다.)를 붙이고 외부에서 발생한 동기 이벤트는 pre()를 붙이지 않는다. 동작은 Statecharts와 동일하게 표현한다.



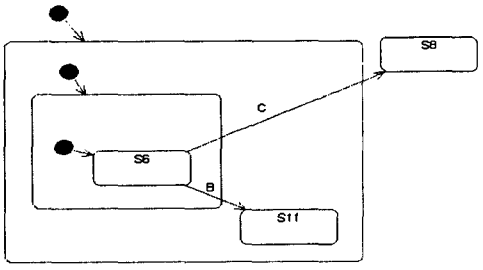
[그림 2] Statecharts의 Events and actions



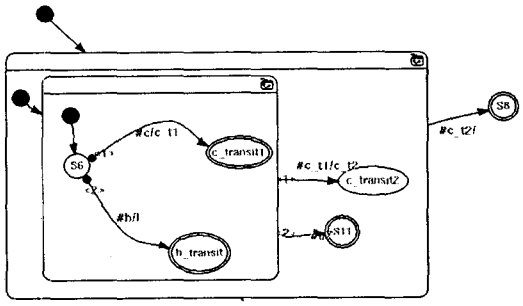
[그림 3] SyncCharts의 Events and actions

[그림 2]는 Statecharts로 S1에서 외부 이벤트를 받아 S2에서 B를 발생하고 S3는 이전 S2 상태에서 발생한 B를 내부 이벤트로 받아 S3에서 C를 방출하고 종료한다. 이 행위를 [그림 3]과 같이 SyncCharts를 이용하여 변환함으로서 내,외부에서 발생한 이벤트에 대한 Statecharts와 SyncCharts의 행위는 같게 된다. SyncCharts에서 S3으로의 전이에서는 pre()을 붙여줌으로 Statecharts의 제약조건 1)을 만족시킨다.

[그림 4]와 [그림 5]는 내부계층전이(interlevel transition)을 표현한 것이며, 계층적으로 전이가 일어나는데 가장 바깥쪽이 상위레벨이다. [그림 4]의 Statecharts는 S6에서 이벤트 B와 C를 동시에 받을 경우 전이는 S6에서 S8로 전이된다. [그림 5]의 SyncCharts에서는 내부계층전이를 허용하지 않는다. 그러므로 Statecharts의 행위를 구현하기 위해 중간에 노드(c_transit)를 삽입하여 내부계층전이를 표현한다.



[그림 4] Statecharts의 내부계층전이



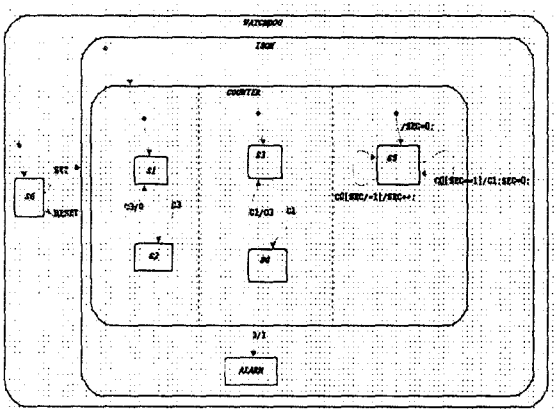
[그림 5] SyncCharts의 내부계층전이

3.1 SyncCharts에 의한 Statecharts의 정의

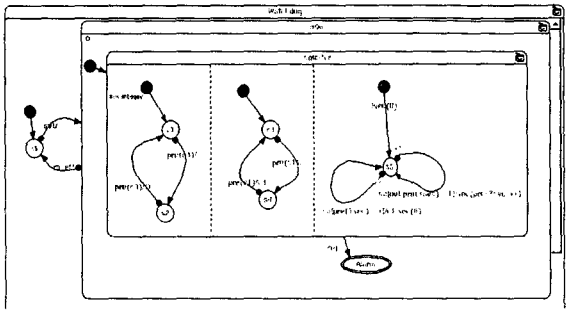
예제 watchdog은 기본적으로 SR 플립플립(inputs: set, reset)으로 ON 일때, c0이 12번 발생하면 시스템은 'alarm' 상태로 간다. 'reset'은 이 'alarm' 상태 밖으로 나가면 일어나야 한다. 이 예제는 계층성(hierarchy), 병렬성(parallelism), 선점(preemption), 시그널 브로드캐스팅(signal broadcasting)에 대해 표현하고 있다. [그림 6]과 [그림 7]은 Statecharts와 SyncCharts로 watchdog을 유사하게 명세한 것이다. Statecharts의 계층 표현방식은 SyncCharts에서와 마찬가지로 가장 바깥 쪽이 상위레벨이다. 즉, 상태 S6이 set 이벤트가 발생되면서 가장 먼저 전이가 일어난다. 그리고 매크로 상태 타이머(macro-state timer(counter))가 병렬적으로 실행된다. 상태 S5에서는 우선순위를 두어 상태로 전이를 일으킨다. SyncCharts는 Statecharts와 달리 자동으로 우선순위가 설정된다. Statecharts의 각 상태들은 SyncCharts와 직접적인 대응으로 변환 가능하다. Statecharts는 브로드캐스팅 방식을 사용하기 때문에 내,외부 자극에 대하여 해당하는 모든 상태가 상태변화 가능하다. SyncCharts는 동기화 방식을 사용하여 Statecharts와 같이 상태전이 하는 것이 가능하다. 상태간 전이는 pre() 연산자를 사용하여 Statecharts의 동기적인 의미와 같게 만들 수 있다. Statecharts에서는 스텝이 일어난 후 출력이 발생하는 반면 SyncCharts는 tick이 발생함과 동시에 출력이 방출되므로 Statecharts와 똑같은 행위를 구현하기 위해 pre() 연산자를 사용하여 해결한다.

[그림 7]과 같이 SyncCharts로 명세한 watchdog을 시뮬레이션하고 정형검증을 한 결과로 c0이 2tick을 라이징(rising)하고 SyncCharts의 경우 pre() 연산자를 사용하므로 다운(down) 할때 다음 c1의 tick이 된다. 그런식으로 c0가 2tick씩 될때마다 다운시 다음 tick때 시그널이 방출되고 다음 상태로 전이된다. [그림 8]의 웨이브폼은 tick이 될 때마다 입력값에 의한 출력값이 방출되는 것을 볼 수 있다.

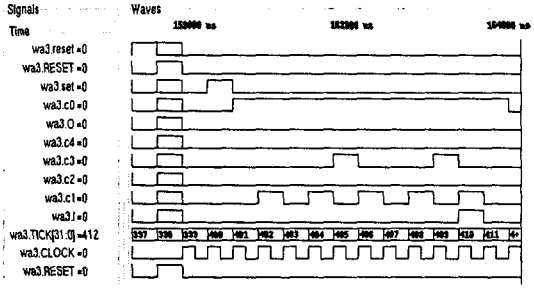
이와 같은 방법으로 변환된 SyncCharts는 Statecharts와 같은 행위를 가지며 변환된 SyncCharts로 UML Statecharts에서는 불가능한 시뮬레이션 및 정형 검증을 할 수 있다.



[그림 6] Statecharts로 명세한 watchdog



[그림 7] SyncCharts로 명세한 watchdog



[그림 8] SyncCharts로 명세한 watchdog 행위 waveform

4. 결론 및 향후 연구 과제

본 논문은 Statecharts를 정형 명세 언어인 Esterel을 도식화한 SyncCharts로 행위적 의미론을 정의해 보았다. 동기적 시간을 적용한 Statecharts의 행위적 의미론은 SyncCharts로 정의함으로 설계의 행위적인 시뮬레이션을 수행함으로 확인하였으며 정형검증함을 보일 수 있었다. 더 나아가 코드 생성을 지원하는 SyncCharts 도구를 이용하여 UML Statecharts를 어떻게 구현할 수 있는지 그 방법을 보임으로, UML 언어를 이용하여 내장형 시스템 소프트웨어의 구현 방법을 제시할 수 있다.

향후 과제로는 SyncCharts 도구를 이용하여 자동 생성된 내장형 코드를 이용해 내장형 시스템의 소프트웨어를 구현하고 이를 타겟 플랫폼에 이식할 수 있는 방법을 제시할 계획이다.

5. 참고 문헌

[1] Charles Andre, "SyncChart: A Visual Representation of Reactive Behavior", Technical report RR-95-52, 13S, 1995

[2] Demissie B. Aredo, "Semantics of UML Statecharts in PVS", Nordic Workshop on Programming Theory, Bergen, Norway, Oct 2000

[3] Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide", Addison Wesley Longman, Inc, 1999

[4] David Harel and Amnon Naamad, "The STATEMATE Semantics of Statechart", ACM Trans. Software Enginnering and Method, Vol. 5, No. 4, pp 293-333, Oct 1996.