STSR의 실시간 내장형 시스템 명세

김진현[O] 최진영
고려대학교
{jhkim[O], choi}@formal.korea.ac.kr

Specification of Real-time Embedded System using STSR

Jin Hyun Kim [0] Jin Young Choi
Dept. of Computer Science, Korea University

요 약

원자력 발전소 안전계통이나 의료 시스템과 같은 실시간 내장형 시스템의 설계는 그 안전성을 분석하기 위한 정형 명세가 요구된다. 이러한 실시간 내장형 시스템이 명세를 위해 본 논문에서는 Statecharts를 확장하여 시간적 명세 및 분석에 용이하고 하드웨어/소프트웨어 통합 설계에 유리한 언어를 제시한다. 그리고 그 언어의 특징을 보일 수 있는 프로토콜 예제를 제시하고 기존의 잘 알려진 언어와 비교 분석함으로 실시간 명세 능력을 보인다.

1. 서론

원자력 제어 계통이나 항공 시스템 혹은 의료 시스템과 같은 시스템은 일반적으로 시간적 정확성이 요구 되는 실시간 시스템이다. 근래 들어 이러한 실시간 시스템은 내장형 시스템으로 개발되고 있다. 실시간 시스템의 정확성은 결과의 정확성 뿐 아니라 시간적 제약 조건에 달려 있다. 따라서 이러한 시스템의 설계는 시간적 명세가 반드시 포함되어야 한다. 최근에 이러한 고안전성 실시간 시스템을 명세하기 위해 다양한 정형 명세 언어가 개발되고 있다. 특히 설계자 및 구현자 혹은 그 외 시스템의 설계와 관련된 사람들 간의 이해 및 의사소통을 원활하게 하기 위해 도식적 언어가 개발되고 있다. 이러한 도식적 언어 가운데 Statecharts[1,2]는 가장 잘 알려진 반응형 시스템 명세 언어로서 State-diagram에 계층성과 평행성을 포함하여 반응형 시스템을 보다 쉽게 설계하도록 고안된 언어이다. 하지만 Statecharts는 실시간 시스템을 명세할 수 있는 명확한 문법을 지니고 있지 않다. 즉 시간적 흐름 및 자원의 사용 등에 관한 명확한 문법 및 의미론을 지니고 있지 않다. 따라서 시스템의 시간적 분석에 상당한 어려움을 지니고 있다. 특히 근래에 들어 하드웨어 및 소프트웨어의 통합 설계에 관심이 더해지고 있는 시점에서 시간적 분석에 어려움을 지닌 Statecharts는 적합하지 않다.

본 논문에서는 이러한 문제를 해결하기 위해 Statecharts with Timed Shared Resource(이하 STSR)를 제안하고 이를 기반한 실시간 명세 기법을 Statecharts 명세 기법과 비교 분석함으로 그 가능성을 보이고자 한다. 또한 하드웨어/소프트웨어 통합 설계가 STSR로 어떻게 가능한지를 제안하고자 한다.

본 논문의 구성은 다음과 같다. 1장에서는 STSR의 언어에 대해 기술하고 2장에서는 이를 통한 실시간 명세 기법을 기술한다. 3장에서는 상용화되어 사용하고 있는 Statecharts의 실시간 명세와 STSR의 명세를 비교 분석하여 실시간 명세의 능력을 보이고 4장에서는 STSR을 이용한 하드웨어/소프트웨어 통합설계가 어떻게 가능한지를 보인다. 5장에서는 본 연구의 결론 및 향후 연구 과제를 기술한다.

2. STSR 언어

I-Logix 사의 STATEMATE MAGNUM은 Statecharts를 설계언어로 사용하는 반응형 시스템 개발 도구이다. 이 도구의 Statecharts는 실시간 시스템의 시간적 특성을 명세하기 위한 의미론으로 Synchronous time scheme 와 Asynchronous time scheme[2]을 갖는다. 첫번째 Synchronous time scheme는 하나의 time-unit에 하나의 전이 스텝을 수행하는 시스템을 가리킨다. 두번째 Asynchronous time scheme은 time-unit 내의 하나 이상의 스텝을 수행하는 시스템을 가정한다. 본 논문의 STSR은 하드웨어/소프트웨어 통합 설계를 위해 이 두 가지 의미론 중 하나의 Asynchronous time scheme을 확장한다. 본 논문에서는 STATEMATE MAGNUM의 Asynchronous time scheme을 지닌 Statecharts을 다음과 같이 확장한다.

STSR = Statecharts with Asynchronous time scheme + Shared Resource + Priority

STSR은 asynchronous time scheme에 자원 및 우선순위를 포함시킴으로 다음과 같은 특징을 지닌다.

첫째, 시간을 사용한 상태와 시간을 사용하지 않는 상태를 명확히 구분함으로 실시간 시스템의 설계에 대한 시간적 분석을 용이하게 한다. 둘째, 자원 사용에 대한 우선순위를 기술함으로 공유 자원에 대한 경쟁관계를 명확히 명세하며 스케줄러와 공유자원 분배에 관한 기술을 단순화하여 시스템 설계를 단순화 한다. 셋째, 하드웨어/소프트웨어 통합 설계를 위해 시간을 사용하는 부분을 명확히 함으로 통합 설계 이후, 하드웨어와 소프트웨어 부분을 나누기가 보다 분명하고 수월해진다.

STSR 언어는 다음과 같이 정의된다.

Def 2.1 (State) : $S = \{BS, AS, OS, TRS\}$
State는 각각 label을 가지며, 다음과 같이 정의한다.
Base state ($bs \in BS$) : instantaneous 한 상태이다.
And-state ($as \in AS$) : Orthogonal state, 두 개 이상의 동시에 수행되는 상태를 의미한다.
Or-state ($os \in OS$) : 두 개 이상의 orthogonal하지 않은 상태들을 의미한다.
Timed-resource state ($trs \in TRS$): 우선순위를 갖는 자원을 갖거나 혹은 가지지 않고 시간만을 소모하는 상태를 의미한다.

Def 2.2 (전이 : *Transition*)
Transition, T는 다음과 같이 정의 한다.

$$T : s_1 \xrightarrow{\delta} s_2 \quad (s_1, s_2 \in S,\ \sigma \in \Sigma)$$

Def 2.3 (*Timed Transition*)
전이 가운데, 시간적 제약이 가해지는 전이를 *Timed Transition*,라 하고, *timed transition* 의 집합을 *TT*라 한다.

Def 2.4 (*Atomic Time Constraint*)
실수 값을 갖는 *atomic time constraint*, *tc*는 전이 t ($t \in T$) 에 대한 시간적인 제약으로 정의한다. *atc*는 다음과 같이 표시한다.

$atc ::= \sim n$, for $\sim \in \{\geq, \leq, =\}$, $n \in R$.

*TC(TT)*는 모든 *TT*에 대한 time constraint의 집합을 가리킨다.

Def 2.5. (*Clock variable*)
Clock variable, *CV* 는, *atomic time constraint*, *tc*를 만족하는 시간이 할당되는 clock 변수의 집합이라 한다.

Def 2.6 (*Time interpretation*)
Time interpretation, *v*는 *tc*를 만족하는 시간 t ($t \in R$)을 $cv (cv \in CV)$ 를 할당하는 함수라 한다.

$V : cv \rightarrow t \quad (cv \in CV,\ t \in R)$

Def 2.7 (*Assignments*) : *A(I)*
I를 정수 값을 갖는 변수의 집합이라 할 때, I에 대한 정수 *assignment*는 $\langle v, c_1, c_2 \rangle$ 의 tuple로 $v = c_1 \cdot v + c_2$, $v \in I$이고 $c_1, c_2 \in Z$이다. *A(I)*는 *I*에 대한 모든 assignment의 power-set를 가리킨다.

Def 2.8 (*Action*)
Action 은 이벤트를 내보내는 것과 *A(I)*를 포함한다.

Def 2.9 (*Act*)
전이 시, 받는 이벤트 및 수행되는 *Action*의 집합을 *Act*이라 하자. *ACT* 다음과 같이 표현된다.
$e[c]/a$, for $e \in E(Event)$, $a \in A(Action)$, $c \in condition$.

Def 2.10 (*Resource Label*)
*R*을 *resource*의 집합이라고 하고, *P*를 정수 값의 우선 순위라 할 때, *Resource label*, *rl*은 다음과 같은 형태를 갖는다.

$rl ::= \{(r_0, p_0), \ldots (r_n, p_n)\}$, $r \in R$, $P \in Z$

Def 2.11 (*Statechart with Timed Shared Resource*)
STSR = $\{S, \Sigma, R\}$
S : A set of State.
Σ : $S \times TC(TT) \times Act \times S$: A set of transition labels.
R : A set of resources

이를 통한 실시간 시스템의 명세는 [그림 2-1]과 같을 수 있다. 이 그림은 Timeout interrupter를 명세한 STSR이다. 이것은 PR 프로세스가 r 이라는 자원은 적어도 2 time-unit 동안 사용하고 a 라는 이벤트를 받아 프로세스를 종료하는 것을 의미한다. 만약 2 time-unit이 지나고 5 time-unit이 지났음에도 a가 오지 않으면 timeout을 통해 PR 프로세스를 종료 시킨다. 이 명세에서의 특징은 시간을 사용한 부분이 명확히 드러난다는 것이 특징 될 것이다. 다음 절에서는 실시간 모델체커 벤치마킹 프로토콜인 Fischer의 프로토콜을 Statecharts와 STSR로 명세하여 비교 분석한다.

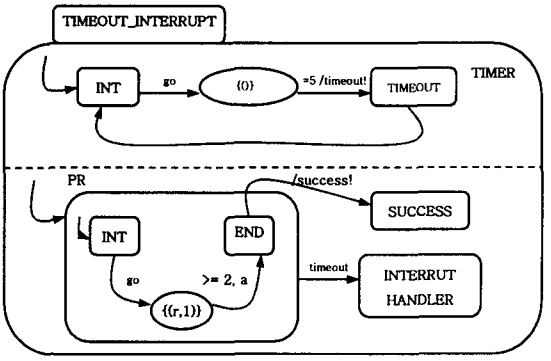

그림 2-1 STSR로 명세한 TIME INTERRUPTER

3. 실시간 시스템 명세를 위한 Statecharts와 STSR 비교

실시간 시스템의 명세를 위해 본 논문에서는 Fischer의 프로토콜을 명세한다. Fischer의 프로토콜[3]을 Timed 오토마타로 명세한 그림은 [그림 3-1]과 같다.

본 논문에서는 STSR과 비교하기 위한 Statecharts로 I-Logix사의 STATEMATE MAGNUM의 Statecharts를 사용한다. 이것은 가장 널리 사용된 상용화된 도구에서 사용하는 언어로 잘 정리된 의미론을 지니고 있기 때문이다. 위의 Fischer의 프로토콜을 Statecharts로 명세한 그림이 [그림 3-2] 이다.

이 명세에서는 우측에 각 프로세스가 임계구역으로 들어갈 수 있도록 스케줄링을 하는 프로세스를 볼 수 있다. 이 스케줄러는 f2가 f1보다 우선순위가 높다고 가정하고 스케줄링하게 된다. 이 명세에서 볼 수 있는 것처럼 시간이 들어간 전이 문법 외에는 시간이 소모하는 부분을 명확히 볼 수 없을 뿐더러 어떤 공유자원을 사용하는지 그리고 공유자원에 대한 우선순위를 프로세스 별로 명세할 수 없다. 따라서 스케줄러의 명세에서만 프로세스의

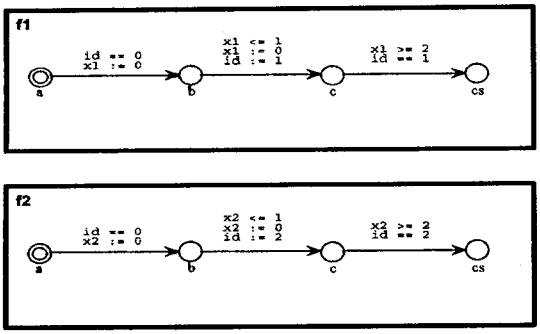

그림 3-1 Fischer' s protocol[4]

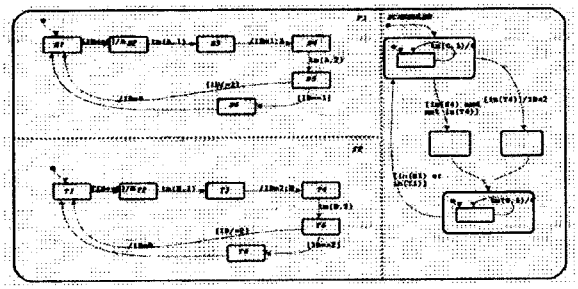

그림 3-2 Statecharts로 명세한 Fischer's protocol

우선순위를 명세해야 한다.

이를 STSR로 명세한 그림이 [그림 3-3]이다. 여기에는 우선 시간이 사용되는 부분이 타원으로 표시되며 그 상태의 전이 레이블에는 반드시 시간 제약의 명세가 기술된다. 또한 프로세스가 사용하는 자원의 이름과 각 프로세스의 우선순위를 볼 수 있다. 따라서 스케줄러는 이를 참조하여 스케줄링하게 된다. 여기서는 [그림 3-2]에서 볼 수 있는 스케줄러를 명세할 필요가 없게 된다.

Statecharts와 STSR를 실시간 내장형 시스템의 명세를 위해 비교하면 [표3-1]과 같다.

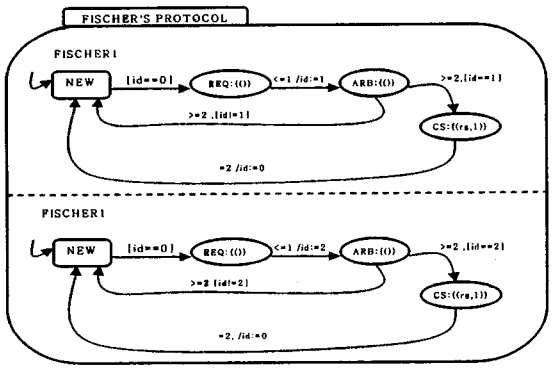

그림 3-3 STSR로 명세한 Fischer's protocol

	Statecharts (I-Logix)	STSR
시간표현	Timeout/schedule 외에 없음	시간 사용에 대한 다양한 문법 (==,<=,>= 등)
자원 및 우선순위 표현	없음	있음
설계내의 시간사용 분석	다소 어려움	시간이 사용되는 부분을 명확히 알 수 있음

표 3-1 Statecharts와 STSR의 실시간 시스템 명세 비교

4. STSR을 이용한 하드웨어/소프트웨어 통합설계

3장에서 보였듯이 STSR은 시간을 사용하는 부분을 제외한 모든 상태 전이는 Instantaneous 하게 전이한다. 이것은 하드웨어의 회로의 상태 전이와 유사한다. 하드웨어는 clock과 같은 외부 자극에 의해 모든 회로가 순간적(instantaneous)으로 반응한다. 하지만 소프트웨어는 CPU의 속도에 따라 수행하며 따라서 하드웨어 보다는 상대적으로 비 동기적이라 할 수 있다. 따라서 시간을 소모하는 노드에 소프트웨어의 코드를 명세할 수 있을 것이다. 그에 더하여 계층적 구조를 지닌 STSR 명세는 하드웨어 및 소프트웨어 분할을 위한 시간 및 비용에 관련된 분석을 계층적으로 할 수 있어 하드웨어/소프트웨어 통합설계를 가능하게 할 것으로 생각된다.

5. 결론 및 향후 연구 과제

본 연구에서 제시된 STSR은 실시간 내장형 시스템의 하드웨어 및 소프트웨어 통합 설계를 위해 I-Logix 사의 Statecharts의 asynchronous time scheme을 확장하고 있다. 이를 통해 명료한 시간 및 자원 사용을 명세하고 있으며 또한 우선순위의 명세를 통해 자원 경쟁관계를 명세함으로 실시간 운영체제와 같은 시스템의 명세에도 유용하게 사용될 것으로 사려 된다.

향후 연구 과제로는 이에 대한 도구 개발 및 실시간 모델체킹과 같은 정형 검증 도구를 통한 분석기법이 개발되어야 할 것으로 사려 된다.

6. 참고문헌.

[1] David Harel, "STATECHART: A VISUAL FORMALISM FOR COMPLEX SYSTEMS", Science of Computer Programming 8 (1987) pp231-274

[2] David Harel and Ammon Naamad, "The STATEMATE Semantics of STATECHARTs", ACM Trans. Soft. Eng. Method. Oct. 1996

[3] Leslie Lamport, A Fast Mutual Exclusion Algorithm. ACM Transactions on Computer Systems, 5(1):1-11 February 1987.

[4] Johan Bengtsson and Fredrik Larsson, UPPAL: A Tool for Automatic verification of Real-time Systems, Master of Science Thesis, January 15, 1996.