TSCR를 이용한 실시간 시스템의 정형 명세

최수진* 강인혜** 양승민**
sjchoi@realtime.soongsil.ac.kr {kang, yang}@computing.soongsil.ac.kr
*숭실대학교 대학원 컴퓨터학과 **숭실대학교 정보과학대학 컴퓨터학부

Formal Specification for Real-Time Systems Using TSCR

Soojin Choi* Inhye Kang** Seungmin Yang**
*Dept. of Computing, Soongsil University **School of Computing, Soongsil University

요 약

실시간 시스템은 신뢰성이 중요하므로 실시간 시스템을 설계하는데 있어서 각 태스크의 기능적인 정확성의 검증은 물론 모든 태스크에 대한 시간적인 요구사항들을 보장해야만 한다. 본 논문에서는 이러한 조건 즉, 실시간 시스템의 정확성과 시간성 보장을 위하여 기존의 객체 지향적 모델을 기반으로 하는 실시간 시스템을 위한 정형명세 언어인 TSC(Timed State Chart)에 자원(resource)의 개념을 추가한 TSCR (Timed State Chart with Resource)를 제안한다. TSCR을 통하여 실시간 시스템의 다양한 요구사항을 명세하며, 태스크들의 시간성 보장을 위한 스케줄링 가능성 분석(schedulability analysis)을 할 수 있다. 실시간 시스템의 대표적인 예로 철도 건널목 제어 시스템을 TSCR를 이용하여 명세한다.

1. 서 론

실시간 시스템은 신뢰성이 중요하므로 설계할 때 시스템의 정확성과 안정성을 입증할 수 있는 방법이 필요하다. 특히 고안전성 시스템(safety-critical system)은 안전과 직결되므로 높은 신뢰도가 요구된다. 이러한 실시간 시스템은 계산 결과의 논리적 정확성 뿐만 아니라 결과가 나오기까지 주어진 마감시간(deadline)도 만족해야 하는 시스템이다. 만일 결과가 마감시간에 맞추지 못한다면 시스템에 고장이 발생할 수 있다. 이러한 신뢰성 보장을 위해 시스템은 개발단계의 초기에서부터 정확히 명세되어야 하고, 시스템의 특성이 미리 검증되어야 한다. 이와 같이 엄격한 설계와 검증을 요구하는 실시간 시스템을 정확하게 명세하기 위해서 근래에는 정형기법(formal method)[1]을 많이 이용한다. 정형기법은 명세에서 발생할 수 있는 애매모호함(ambiguity), 불완전성(imcompleteness), 불일치성(inconsistency) 등을 수학을 사용함으로써 제거할 수 있다.

본 논문에서는 신뢰성 높은 시스템을 개발할 수 있도록 객체 지향 설계 기법을 위한 정형 명세 언어인 TSC(Timed State Chart)[2]을 확장한 TSCR(Timed State Chart with Resource)를 제안하고, TSCR를 이용하여 실시간 시스템의 예제인 철도 건널목 제어 시스템을 명세한다.

TSCR은 Statecharts와 같이 상태 기계(state machine)를 계층적 구조로 확장한 정형 명세 언어인 TSC에 자원(resource)의 개념을 추가한 것이다. TSCR을 통하여 실시간 시스템의 다양한 요구사항의 명세는 물론 태스크들의 시간성 보장을 위한 스케줄링 가능성 분석(schdulability analysis)을 할 수 있다. 실시간 시스템을 검증하기 위한 스케줄링 가능성 분석은 어떤 스케줄링 알고리즘으로 스케줄되는 실시간 시스템이 데드라인(deadline)내에 일을 끝낼 수 있는지를 검사하는 것이다[3].

스케줄링 가능성을 분석하는 대표적인 예로는, ACSR-VP(Algebra of Communicating Shared Resources with Value Passing)[4]를 이용하는 방법이다. 프로세스 대수의 일종인 ACSR-VP는 명세가 된 시스템이 정확한지를 검사하는 검증 방법으로 바이시뮬레이션(bisimulation)과 스케줄링 가능성 분석을 이용한다. 실제로 스케줄링 가능성 분석은 VERSA(Verification, Execution and Rewrite System for ACSR)[5]툴을 이용하여 구현 명세가 정확성 명세와 동가(equivalence)라는 것을 보이고 다음에 정확성 명세가 IDLING TASK와 동가 관계이 있는지 보임으로써 구현 명세가 정확하다는 것을 검증한다. 즉, 시스템이 데드라인내에 일을 끝낼 수 있는지를 검증한다. ACSR-VP를 이용한 스케줄링가능성 분석은 유용한 방법이기는 하나 명세하는 것이 어렵다는 단점을 가지고 있다.

Real-Time UML[6], ROOM[7], SAT(Scedulability Analysis Tool)[8]은 객체 지향적 분석 및 실시간 소프트웨어 개발의 생산성 향상을 목적으로 스케줄링 가능성을 분석해주는 방법과 도구이다. 스케줄링 가능성 분석 도구는 개발자가 태스크들의 분석과 그 태스크들의 스케줄링 가능성을 검증할 수 있으므로, 예측 가능한 실시간 응용분야를 설계하는데 매우 유용하다.

본 논문의 구성은 다음과 같다. 이어지는 제 2장에서는 TSC의 기반이 되는 실시간 객체 모델인 dRTO 모델에 대해서 설명하고, 3장에서는 실시간 객체의 행위(behavior) 명세하기 위한 정형 명세 언어인 TSCR를 설명한다. 4장에서는 실시간 시스템의 예로 철도 건널목 제어 시스템을 설명하고 TSCR를 이용하여 명세한다. 마지막으로 5장에서는 결론 및 앞으로의 연구 방향에 대해 기술한다.

2. dRTO 모델

dRTO 모델[9]은 신뢰도 높은 실시간 시스템의 구축을 위하여 객체 지향과 실시간성, 그리고 결함허용의 세가지 기본 개념을 단일 모델에 수용하고 있다. dRTO 모델에서는 실시간 시스템의 모든 컴포넌트가 실시간 객체(RTO 또는 Real-Time Object)로 추상화된다. 실시간 객체는 실시간 자료와 실시간 메쏘드로 구성된다.

실시간 메쏘드는 구동 특성에 따라 시간구동 메쏘드(Time-triggered Method)과 메시지구동 메쏘드(Message-triggered Method)로 구분된다. 시간구동 메쏘드는 실시간 객체의 생성시 단일 쓰레드로 생성되며 명세된 주기마다 활성화되어 작업을 수행한다. 시간구동 메쏘드는 주기, 최대수행시간, 우선순위, 마감시간, 그리고 중요도와 같은 시간 특성을 갖는다. 이 쓰레드는 명세 되어진 마감시간 이전에 작업을 완료하여야 한다. 메시지구동 메쏘드는 시간구동 메쏘드나 다른 메시지구동 메쏘드의 호출에 의해 수행된다. 이 때 각각의 호출에 대한 독립적인 쓰레드가 생성되며 생성된 쓰레드는 호출시 주어지는 마감시간 이전에 작업을 완료하여야 한다. 메시지구동 메쏘드는 전달인자 이외에 최대수행시간, 우선순위, 마감시간, 그리고 중요도와 같은 시간 특성을 가진다.

그림 1은 철도 건널목 제어 시스템을 dRTO로 나타낸 것이다. 실시간 객체 RailroadCrossingControlSystem는 객체 Track, Controller, Gate로 구성되고 Track은 여러 개의 객체 $Track_1$, ..., $Track_n$으로 구성된다. 객체 내의 메쏘드는 그림 1과 같이 두 가지로 표현되는데 $TrainBehavior_i$, Controlling, ControlGate는 시간구동 메쏘드이고, $Sensing_i$, GetInfo, PutCommand는 메시지구동 메쏘드를 나타낸다. [1]

---

1 실시간 객체 실시간 자료 시간구동 메쏘드 메시지구동 메쏘드

자세한 것은 [9] 참조

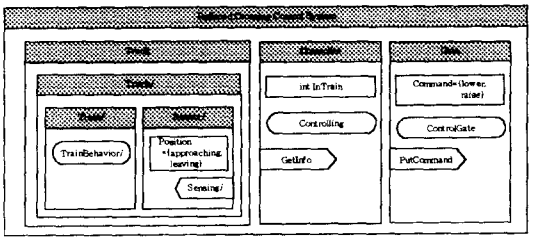


그림1. 철도 건널목 제어 시스템의 dRTO 모델

3. TSCR (Timed State Chart with Resource)

TSCR은 정형 명세 언어인 TSC에 자원의 개념을 추가한 것으로 대부분 TSC와 유사하다. 그러므로 우선 TSC에 대해 간단하게 설명하고 TSCR에서의 추가된 자원에 대해 설명하기로 한다.

3.1 TSC(Timed State Chart)

TSC는 dRTO 모델을 비롯한 객체 지향 기반의 실시간 시스템의 행위를 명세하기 위한 정형 명세 언어이다. TSC에서는 객체, 메쏘드를 모두 상태의 형태로 표현하고 그 상하관계에 따라 계층적으로 표현한다. 또한 실시간 시스템의 다양한 시간 제약사항과 시간의 흐름에 따른 실시간 객체의 상태 변화를 표현하기 위하여 클럭 변수(clock variable)를 도입한다. 클럭 변수들은 보통 변수와 마찬가지로 그 값을 비교하고 변경할 수 있으나 시간의 흐름에 따라 그 값이 같은 비율로 증가한다는 차이가 있다. 이 클럭 변수를 사용하여 주기, 마감시간 등 다양한 실시간 제약사항을 표현할 수 있다. TSC의 시멘틱스(semantics)는 실행(execution)으로 정의하여 각 명세가 정확한 의미를 가지고 있도록 한다. 실시간 시스템을 dRTO로 모델링하고 각 메쏘드의 행위는 TSC로 명세함으로써 전체 시스템의 의미를 모든 가능한 실행들로 나타내며, 이 실행들에 대하여 정확성을 검증함으로써 전체 시스템의 정확성을 보장을 할 수 있다[2].

3.2 자원(Resource)

TCSR은 TSC를 바탕으로 시스템의 계층적 설계 및 분석, 다양한 시간 제약사항, 객체간의 통신, 자원, 우선순위 등 실시간 시스템에 필요한 여러 개념을 지원하므로 실시간 시스템을 명세하고 검증하는데 적합하다. TSCR를 이용함으로써 여러 프로세스들이 한정된 공유자원을 사용하는 시스템을 묘사할 수 있고 여러 프로세스들이 동시에 수행되는 상황과 프로세스들간의 통신을 표현할 수 있다.

TSCR에서는 다양한 종류의 많은 자원들을 표현하기 위해 ACSR-VP[4]에서 제안한 방법을 사용하며, 공유 자원들간에 우선순위를 다음과 같이 명세한다.

$$\{(r1,p1), (r2,p2), \ldots\}$$

여기에서 r은 자원이며, p는 그 자원의 우선순위이다. 예를들어, 자원과 우선순위가 {(r1,2),(r1,1)}인 경우 공유 자원들(r1) 간의 우선순위는 높은 수(2)를 가지는 것이 우선순위가 높은 것이며, 동적(dynamic)으로 바뀔 수 있고, 우선순위의 상속(inheritance)이 가능하다.

또한, 자원 할당 함수(resource mapping function 또는 rmf)를 두어 단계적으로 실시간 객체, 메쏘드, 상태 별로 자원을 구별한다. 자원 할당 함수(rmf)를 정의하면

$$\begin{aligned} rmf:\ & Obj \rightarrow 2^R \\ & Method \rightarrow 2^R \\ & States \rightarrow 2^R \end{aligned}$$

와 같다. Obj는 시스템 내에 있는 모든 객체들의 집합, method는 시스템 내에 있는 모든 메쏘드들의 집합, states는 시스템 내에 있는 모든 상태들의 집합을 말한다. 또한 R은 자원들의 집합이다.

제약사항으로는 $O \in Obj$, $M \in Method$, $S \in States$이고, S가 M내에 있고, M이 O내에 있다면 $rmf(S) \subseteq rmf(M)$이 되고 $rmf(M) \subseteq rmf(O)$의 관계가 성립된다.

예를들어, 철도 건널목 제어 시스템에서의 실시간 객체가 사용하는 자원들의 집합은 rmf(RailroadCrossingControlSystem)={$cpu1_i$, $cpu2_i$, cpu3, cpu4, gate}로 정의되며, 이 시스템을 실행하기 위해서 $cpu1_i$, $cpu2_i$, cpu3, cpu4, gate의 자원들을 사용한다는 의미이다.

4. 예제 : 철도 건널목 제어 시스템

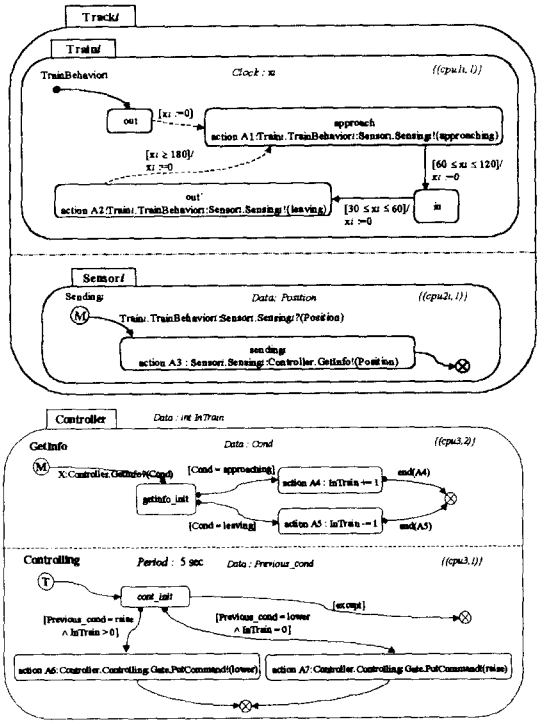
철도 건널목 제어 시스템(railroad crossing control system)은 철도 건널목의 게이트를 제어하는 시스템이다. 철도 건널목 제어 시스템은 고신뢰도를 요구하는 실시간 시스템으로서 시스템의 동작 중에 발생하는 논리적, 물리적, 시간적 오류는 바로 인명 피해를 유발하는 교통사고로 연결될 수 있다. 예제로써 이 시스템을 dRTO로 모델링하며 추출된 각각의 실시간 객체들의 행위를 TSCR를 이용하여 명세한다.

철도 건널목 제어 시스템의 요구사항으로는 철도 건널목은 기차가 들어오고 나가는 경계 영역을 감지하여, 건널목에 있는 게이트를 안전하고 실용적으로 열고 닫는 데 목적을 둔다. 안전하다는 것은 기차가 횡단하고 있을 때에는 게이트는 내려져 있어야 하며, 실용적이라는 것은 기차가 횡단하고 있지 않을 때에는 되도록 게이트는 올라가 있어야 한다. 전체 제어 시스템은 그림 1에서와 같이 실시간 객체 구성되어 있다.

- Track: 건널목의 각 트랙에서 기차가 들어오고 나가는 행위를 감지하는 객체. Track은 n개의 트랙을 나타내기 위해서 n개의 객체 $Track_1$, $Track_2$, ..., $Track_n$으로 구성된다. 각 $Track_i$는 다음과 같은 객체로 구성된다.
  - $Train_i$: i번째 트랙에서 건널목을 오가는 기차의 행위를 시뮬레이트하는 객체
  - $Sensor_i$: i번째 트랙에서 기차의 위치를 감지하여 Controller에게 정보를 준다.
- Controller: Sensor로부터 기차의 상태(approaching, leaving)에 대한 정보를 받아 게이트를 제어하는 객체.
- Gate: 철도 건널목의 게이트를 Controller의 명령(lower, raise)에 따라 올리고 내리는 역할을 하는 객체.

각 객체들이 사용하는 자원들을 살펴보면, 우선 Track의 자원 할당 함수는 rmf(Track)={$cpu1_i$, $cpu2_i$}이 되며, 이는 n개의 트랙을 지나가는 기차의 행위를 시뮬레이트하고 위치를 감지하는데 필요한 자원이다. Controller 객체는 rmf(Controller)={cpu3}로 1개의 자원을 사용하여 우선순위에 따라서 각각 메쏘드들이 실행한다. Gate 객체의 자원 할당 함수는 *rmf(Gate)={cpu4, gate}*로 Controller의 명령을 실행하기 위한 자원과 실제 게이트를 올리고 내리는데 사용하는 gate자원이 있다.

위의 실시간 객체들의 행위는 그림 2의 TSCR로 주어진다.



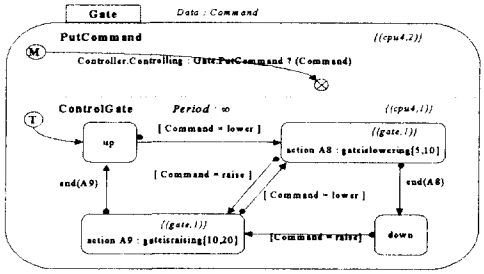


그림 2. 철도 건널목 제어 시스템의 TSCR

객체 $Train_i$는 $TrainBehavior_i$라는 메쏘드에서 기차가 건널목을 다가오고 나가는 반복적인 행위를 나타내는데 기차가 센서의 위치에 오면 객체 $Sensor_i$의 $Sensing_i$이라는 메쏘드를 구동하여 기차의 위치를 알린다. $Train_i$에 $cpu1_i$와 $Sensor_i$에 $cpu2_i$는 각각 자원을 병행적으로 사용하며 서로 공유하는 자원이 없으므로 우선순위는 1로 정의한다. 기차와 관련된 시간 제약조건은 기차간의 간격은 적어도 180초 이상이고, 기차는 센서가 있는 곳에서 건널목까지 최소 60초, 최대 120초이고, 건널목을 지나는 시간은 최소 30초, 최대 60초 이다. 이 제약조건은 out'에서 approach간의 전이 조건 $x_i \geq 180$, approach에서 in간의 전이 조건 $60 \leq x_i \leq 120$, in에서 out'간의 전이 조건 $30 \leq x_i \leq 60$으로 나타낸다.

객체 $Sensor_i$의 메시지구동 메시지 $Sensing_i$은 Position이라는 변수를 통해 기차의 위치를 $Train_i$의 $TrainBehavior_i$로부터 받아 Controller의 GetInfo에게 전해준다.

객체 Controller는 두개의 메쏘드와 지역내에 있는 기차의 개수를 저장하는 변수 InTrain으로 구성되어 있다. 메시지구동 메쏘드인 GetInfo는 $Sensing_1$, ..., $Sensing_n$중 하나로부터 기차의 위치를 받아 기차가 다가오면 InTrain을 증가시키고 나가면 감소시킴으로써 지역 내에 있는 기차 수를 센다. 시간구동 메쏘드인 Controlling은 5초 마다 구동하여 게이트의 위치와 지역내의 기차유무에 따라 객체 Gate에 게이트를 올리거나 내릴 것을 명령한다. Controller에서 두개의 메쏘드는 cpu3의 자원을 공유하며, 두 메쏘드가 동시에 cpu3을 요구할 시에는 우선순위가 높은 GetInfo 메쏘드가 자원을 사용하여 실행되고, Controlling 메쏘드는 GetInfo 메쏘드가 자원을 복귀할 때까지는 실행될 수 없다.

객체 Gate의 메시지구동 메쏘드 PutCommand에서는 Controller 객체로부터 명령을 받아서 Command에 저장한다. 또한 시간구동 메쏘드 ControlGate는 Command의 값에 따라 게이트를 올리거나 내리는 동작을 수행한다. 게이트를 올리는데 걸리는 시간은 10초에서 20초 사이, 내리는데 걸리는 시간은 5초에서 10초로 주어진다. 실제로 게이트를 올리고 내리는 상태에서는 gate라는 자원을 사용하며, 명령을 처리하기 위해서 cpu4의 자원을 사용하여 실행한다.

TSCR에서 사용되는 객체간의 통신을 위한 채널들은 다음과 같다. $Train_i$의 $TrainBehavior_i$가 $Sensor_i$의 $Sensing_i$를 비동기호출을 하므로 채널

$$Train_i.TrainBehavior_i:Sensor_i.Sensing_i$$

을 통해 각각 호출한다. $Sensor_i$의 $Sensing_i$는 Controller의 GetInfo를 구동하므로 채널

$$Sensor_i.Sensing_i:Controller.GetInfo$$

을 통해 호출하고, Controller의 Controlling은 Gate의 PutCommand를 구동하므로 채널

$$Controller.Controlling:Gate.PutCommand$$

를 사용한다. 각 메쏘드를 TSCR로 나타내면 그림 2와 같다.

5. 결론 및 향후 연구 방향

본 논문에서는 실시간 객체 모델인 dRTO 모델을 비롯한 객체 지향 기반의 실시간 시스템의 행위를 명세하기 위한 정형 명세 언어인 TSCR를 제안하였다. TCSR은 TSC를 바탕으로 시스템의 계층적 설계 및 분석, 다양한 시간 제약사항, 객체간의 통신, 자원, 우선순위 등 실시간 시스템에 필요한 여러 개념을 지원하므로 실시간 시스템을 명세하고 검증하는데 적합하다. TSC는 실시간 시스템을 구성하는 실시간 객체 단위로 명세된다. TSC의 시멘틱스(semantics)에 대한 실행 모델을 정의함으로써 실시간 시스템의 시뮬레이션, 코드 생성 그리고 검증 도구의 개발을 위한 기반이 되며, TSC에 자원과 우선순위의 개념을 추가한 TSCR를 제안함으로써 실시간 시스템의 다양한 요구사항의 명세는 물론 태스크들의 시간성 보장을 위한 스케줄링 가능성 분석(schdulability analysis)을 할 수 있는 기반을 마련하였다. 또한, 실시간 시스템의 예제로 철도 건널목 제어 시스템을 TSCR를 이용하여 명세하였다.

현재 dRTO 모델을 위한 스크립터(scriptor), TSCR를 명세하기 위한 그래픽칼 편집기, 시뮬레이터 등을 개발 중이며, 스케줄링 가능성 검증 및 분석 도구에 대한 연구가 진행중이다. 향후 명세 된 실시간 시스템의 타당성(feasibility), 요구사항과 명세의 일관성(consistency), 실시간 제약사항에 대한 검증 방법, 자동 코드 생성 기법과 이를 위한 통합 자동화 도구를 개발할 예정이다.

6. 참고문헌

[1] Edmund M. Clarke and Jeannette M. Wing, "Formal Methods : State of the Art and Future Directions," *ACM Computing Surveys*, vol. 28, no. 4, pp. 626-643, December 1996.

[2] 강인혜, 정유성, 최수진, 양승민, "실시간 객체지향 모델링을 위한 정형 명세 언어 Timed State Chart," 한국정보과학회 논문지 심사중

[3] 임성묵, "ACSR-VP를 이용한 실시간 시스템의 스케줄링 기법에 대한 정형 명세와 검증," 한국정보과학회 논문지(A) 제25권, 제6호, pp.568-581, June 1998.

[4] 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.

[5] Duncan Clarke, Insup Lee, and Hong-Liang Xie, "VERSA: A Tool for the Specification and Analysis of Resource-Bound Real-Time Systems," *Journal of Computer and Software Engineering*, 3(2), April 1995.

[6] B.P. Douglass and S. Vasan, "Temporal Models in UML," Dr. Dobb's Journal, pp.74-85, Dec. 1999.

[7] M. Sakesena, A. Ptak, P. Freedman, P. Rodziewicz, "Schedulability Analysis for Automated Implementations of Real-Time Object-Oriented Models," Real-Time System Symposium, IEEE, 1998.

[8] Paolo Ancilotti, Giorgio Buttazzo, M.D. Natale, Marco Spuri, "Design and Programming Tools for Time Critical Applications," The Journal of Real-Time Systems, 14, pp. 251-267, 1998

[9] 이신, 손혁수, 양승민, "실시간 객체 모델 dRTO," 한국정보과학회 논문지 게재예정.