추상 시간 기계를 기반으로 한 시간 명세와 분석

노경주°, 박지연, 이문근
전북대학교, 컴퓨터과학과
{kjuno, jypark, mklee}@cs.chonbuk.ac.kr

Timing Specification and Analysis based on Abstract Timed Machine

Kyoung-ju Noh°, Ji-yeon Park, Moon-kun Lee
Dept. of Computer Science, Chonbuk National University

요 약

최근 대부분의 시스템은 즉각적인 정보를 요구하는 실시간 시스템이다. 실시간 시스템은 오류에 대한 유연적인 대체를 필요로 하는 내고장성 시스템(fault tolerant system) 의 개념을 지향한다. 내고장성 시스템은 보다 정확한 시스템 설계가 중요하며, 특히 실시간 시스템에서 필요로 하는 여러 종류의 시간제약에 대한 풍부한 표현력과 명세된 시간에 대한 검증이 중요하다. 본 논문은 특정 시점과 시간 간격, 주기, 이산 시간, 다수의 타이머에 의한 동작 등의 시간 관련 동작을 ATM(*Abstract Timed Machine*)에 기반으로 명세하고 명세된 시간에 대한 분석을 살펴본다.

1. 도입

최근의 금융, 의료, 에너지, 제조업, 국방 등의 사회 각 분야에서 사용되는 대부분의 시스템은 시간과 같은 속도의 정보를 요구한다. 이러한 실시간 시스템은 에러에 대한 유연적인 대체 동작을 필요로 하는 내고장 허용 시스템(*fault tolerant system*)으로 시스템 동작에 대한 정확한 설계가 매우 중요하다.

특히나 실시간 시스템의 시간을 소요하는 동작(*timed action*)에서는 시스템에서 요구하는 여러 종류의 시간 제약을 정확히 명세 할 수 있어야 하며 이에 대한 정확한 분석을 기초로 전체 시스템이 디자인되고 구현되어야 한다. 이러한 동기로 의해서 실시간 시스템의 명세와 명세에 대한 검증을 위해 개발된 ATM을 기반으로 하여 여러 분류의 시간 명세 방법과 시간 명세를 분석하도록 한다.

2. 관련 연구

시스템의 명세를 위해 CRSM(Communicating Real -time State Machines)[1], Statecharts[2], Timed Petri Net[6], Modechart[7] 등과 같은 다양한 정형 기법들이 개발되었다.

특히나 실시간 시스템의 시간을 명세하기 위해 Timed Petri Net[6] 과 같은 명세언어의 대부분은 전이의 레이블에 시간 제약을 추가로 명시하고 있다. 하지만 표현되는 시간 명세가 특정 전이나 지속시간(duration)과 시간간격(interval)표현에 국한되어 있다. 또 분산 실시간 시스템을 위한 명세 언어[5]는 within, until, from 과 같은 시간에 대한 한정사를 명세에 직접 사용하여 이해가 용이한 반면 그래픽 명세 언어에서 사용하기에는 어려운 점이 있다.

이러한 상황으로 그래픽 명세 언어에서 시간에 대한 제약 조건을 명세하기에 보다 풍부하고 간결한 표현에 대한 연구가 필요하다.

한국과학재단 특정기초연구(과제번호 1999-2-003-3) 지원으로 수행되었음.

3. ATM에서의 시간 명세

시스템 명세에서 고려 되는 시간의 종류로 1) 특정 시간의 시점과 일정 간격의 시간을 표현하기 위한 시점(time points)과 시간간격(time interval), 2) 다수의 타이머에 의한 지역 시간과 전역 시간의 구분, 3) 실세계 시간의 실제 개념인 연속적(continuous) 시간과 연속적인 시간을 표현하기 위한 이산적(discrete) 시간의 표현[3]이 있으며 각각에 대해서 ATM 에서의 표현 방법에 대해서 살펴본다.

3.1 Time Points & Time Intervals

시스템 명세에서 요구되는 시간 제약의 대부분은 특정시간에 동작을 수행하라는 명령이나 특정 시간 간격을 두고 동작하라는 명령이다. 이때 사용되는 시간은 절대시간과 상대시간 모두의 표현을 요구한다. 이러한 것에 대해서 ATM에서는 다음과 같이 표현한다.

3.1.1 At

ATM에서는 특정 시점을 나타내는 At의 사용은 절대시간을 표현하기위해 사용하는 것을 해당 된다. 상대적 시간은 각 모드와 전이의 표현에 의해서 각각에 제어의 도착 시점을 상대적 시간으로 가정하기 때문이다.

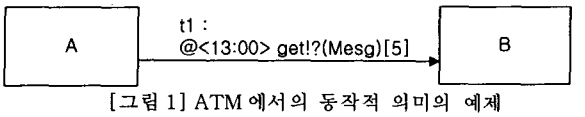
AT의 표현은 @<Aug 1 2000 13:00>, @<13:00> 와 같이 전이와 모드의 레이블 앞에 @ 를 추가하여 명시한다.

3.1.2 After & Be fore

ATM에서는 특정 시간의 전후 관계를 명시할 필요는 없다. 이것은 상대시간의 At을 표현 할 필요가 없는 것과 같이 전이와 모드에 의한 그래픽적 표현에 의해서 이미 동작의 전후 관계는 결정되어 지고, 필요에 따라서 @ 을 사용하여 특정 시간의 전후관계를 명시 할 수 있기 때문이다.

[그림 1] 의 ATM 예제에서 모드 A가 동작 중이거나 A의 동작이 완료된 후 대기하고 있을 때 <13:00> 의 시점에 전이

*t1*이 발생한다. 따라서 *A*는 *Before* <13:00> 의 시간에 동작되어야 하고, *B*는 *After* <13:05> 분에 실행 되어야 한다는 것이 동작적 의미(operational semantics)에 의해서 내제 되어 있다.



[그림 1] ATM 에서의 동작적 의미의 예제

3.1.3 Until

*Until*은 특정 시간까지 동작이 지속 되어야 하는 것을 말한다. ATM 에서는 현재로부터의 특정 시간 까지를 나타내는 시간인 경우에는 지속시간과 같은 개념이라고 할 수 있다. 특정 절대 시간까지 동작의 지속을 명시할 경우에는 다음 동작의 레이블에 @ 으로써 시간을 명시하게 되므로 다음 동작의 @ 에 명시된 시간 전까지 동작한다.

3.1.4 Duration & Execution

모드와 전이가 동작을 지속하는 시간을 말한다. *Duration*은 모드와 전이로 제어가 진입함과 동시에 시작되어 그 시간 안에 해당 동작이 종료 되어도 *duration*동안은 다음 전이나 모드로 진입하지 못한다.

3.1.5 Between & Deadline

특정 시간 사이에 동작이 시작되어 완료 되는 것을 나타낸다. [*lower-bound, upper-bound*] 사이에 동작이 완료되었을 경우에는 *upper-bound*가 완료되기 전이라도 동작이 완료되면 즉시 다음 모드와 전이로 진행 된다. 또 동작이 *upper-bound* 이내에 끝나지 않더라고 *upper-bound*의 시간이 지나면 제어는 강제적으로 다음 모드와 전이로 진입한다.

3.1.6 Delay & Ready Time

*Delay*와 *ready time* 은 특정 동작이 시작 되기 전에 동작이 지연되어야 하는 시간을 나타낸다. 지연의 시작은 모드와 전이에 제어가 진입한 즉시 시작된다.

3.1.7 From-to

특정 시간부터 특정 시간까지 동작이 발생하는 것을 말하는 것으로 *Between*과 구별된다. *From-to*의 개념은 동작이 *upper-bound*전에 한번 완료되어도 *upper-bound*에 도달하기까지 동작을 반복한다. *From-to* 사이에 반복되는 횟수는 *duration*의 지수로 표현된다. 예를 들어 단위 시간 10의 실행 시간을 갖는 작업이 *upper_bound*전에 시작하여 *upper_bound*를 포함하여 *n*번 수행되는 작업을 명세할 경우에는 $10^{[n]}$, *upper_bound*를 지나는 작업은 포함시키지 않고 *n*번 수행되는 작업은 10^n으로 표현한다.

3.1.8 Timeout

Duration, upper-bound, deadline 등의 특정 시간 조건을 만족하지 못했을 경우 *timeout*이벤트가 발생된다. 현재 ATM 에서는 *timeout* 이벤트는 예외 전이로써 발생된다. 예를 들어 단위 시간 10 의 데드라인을 만족하기 못하였을 경우 [10, -]의 시간 제약을 갖는 예외 전이가 발생하게 된다.

3.1.9 Release

동작을 위해 사용 중이던 자원을 특정 시간에 해제한다. 예를 들어 Buffer 의 사용을 단위시간 50 후에 해제해야 할 경우 Release(50) 으로 표현된다. Release 나 Blocking 과 같은 명령의 발생은 환경적인 요인으로 시스템에서 발생시키거나 ATM의 머신이 다른 머신의 동작을 제어하기 위해 발생된다.

3.1.10 Blocking

단위 시간 동안 특정 동작을 수행하는 머신을 Blocking 한다. 예를 들어 단위 시간 50 동안 블록킹 해야 할 경우에는 Blocking(50)으로 표현된다.

3.1.11 Periodic

일정 단위 시간 마다 특정 동작이 반복적으로 수행되는 주기적 작업인 경우에는 (delay, period, execution, deadline)의 시간 제약 순으로 명세 된다.

3.2 Discrete time & Continuous time

실세계 시간은 연속적(continuous)이다. 실세계 여러 과학 분야에서 실세계의 연속적인 시간을 특정 값으로 인식할 수 있는 시간(discrete time)로 표현한다.

연속적 시간에 따른 실세계의 변화의 예로 물탱크의 예를 들 수 있다[3]. 물탱크의 수위의 연속적인 변화는 시간에 대한 수량의 변화로 이해 할 수 있다. 이것은 다시 시간에 대한 연속적인 함수로 표현 될 수 있다. 만약 물탱크의 총 수용 능력이 23 리터이고 초당 4 리터의 물이 흘러 들어 오고, V_0을 초기에 물탱크에 수용된 물의 양이라면 시간의 흐름에 따른 탱크가 수용한 총 물의 양은 $V(t)=4*t+V_0$ 와 같은 함수로 표현 할 수 있다.

이산적인 시간은 일반적으로 정수를 사용하는데, 연속적인 변화를 나타내는 방정식 시스템의 해답을 제공하지 못한다는 단점이 있다. 예를 들어 위에 물탱크에서 물탱크의 총 수용량인 23 리터가 되기 위해 초기의 수용된 물이 없을 경우 약 5.75 시간으로 정수로써 나타낼 수 없는 경우가 발생하여 시스템 방정식의 해답을 제공하지 못한다.

이러한 문제를 해결하기 위해 단위 시간의 크기(granularity)를 조정 함으로써 방정식의 해답을 제공하게 된다. 물탱크 예제에서 실제 방정식의 해는 5.75 초이다. 이때 100분의 1 초를 단위 시간으로 한다면 575 의정수로 해당 방정식의 해을 제공할 수 있지만, 이러한 방법은 무한수의 해를 가지는 경우에는 해를 표현 할 수 없다는 단점이 존재한다.

이러한 이유로 ATM 에서는 단위 시간의 크기를 조정 하는 방법 외에 시간의 범위로써 정수로 표현할 수 없는 방정식의 해을 제공하게 된다. 예를 들어 between 을 나타내는 [5, 6] 의 표현은 5.75 의 실수를 커버한다. 이러한 방법은 정확한 해을 제공하지는 못하지만 해의 범위를 제공함으로써 보다 최소의 오차를 인정하여 시스템이 보다 유연적인 범위에서 동작하게 한다.

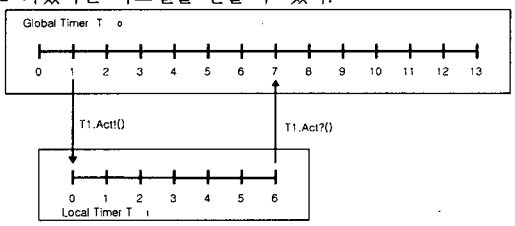
3.3 다수에 타이머에 의한 시간

선형(linear) 시간이란 과거에서 현재로, 현재에서 미래로 이어지는 실세계의 일반적인 시간 규칙이다. 선형 시간은 과거-현재-미래 중 한 단계를 건널 수 없고, 순서를 역행 할 수 없다. 시스템 명세에서도 이러한 선형 시간을 사용한다.

ATM에서도 이러한 선형 시간을 사용한다. 각 머신에 속한 타이머로는 지역 머신에 속한 지역(local) 타이머가 존재있고, 시스템 전체에 걸쳐서 활성화 되어 있는 메인 머신에 속하는 전역(global) 타이머가 있다. 각 타이머는 자신이 속한 머신이 활성화 될 때 동시에 활성화 된다.

[그림 2] 와 같이 전역 타이머 T_0이 동작하는 중에 T_0이 속한 머신에서 지역 타이머 T_1이 속한 머신을 활성화 시키면

내부 적으로 지역 타이머 T_1도 동시에 활성화 시그널을 받아서 활성화 된다. 전역 타이머와 지역 타이머는 같은 비율로 자신의 시간을 증가하여 지역 타이머 T_1이 6 단위 시간 동안 활성화 되었다가 종료한다면 T_1이 종료한 시간은 전역 타이머 T_0도 그 만큼의 시간이 경과된 시점에서 T_1이 종료 되었다는 시그널을 받을 수 있다.



[그림 2] 전역 타이머와 지역 타이머의 상호작용

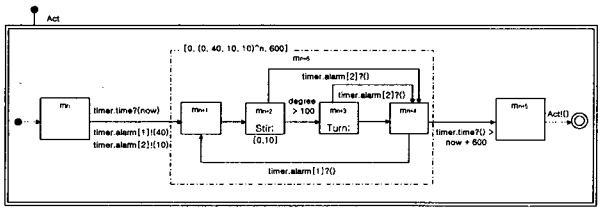
각 타이머는 활성화 포트, 타임 포트, 알람(alarm) 포트를 가진다. 활성화 포트는 머신이 활성화 될 때 자신의 타이머를 활성화 시키기 위한 포트이고, 타임 포트는 타이머의 시간을 전송하는 포트이다. 알람 포트는 alarm[1], alarm[2], …., alarm[n]와 같이 다수의 포트가 존재하여 특정 시간마다 시그널을 보내주기를 원하는 시간을 받아들여 해당 시간마다 소속 머신으로 시그널을 보낸다. 예를 들어 Timer.alarm[1]!(40)의 경우는 타이머의 1 번 알람 포트를 통하여 알람 시간을 단위 시간 40 으로 설정하고, Timer.alarm[1]?(40)는 타이머의 1 번 알람 포트로부터 단위시간 40 이 경과 할 때 마다 시그널을 받는 것을 나타낸다.

4. 시간 명세 예제 및 분석

다음의 예제는 [표 1][5] 을 ATM 으로 명세한 예제를 보이고 있다. [표 1] 은 now 로부터 10 분(600초) 동안(from-to)동안 매 40 초 마다(period) 10 초 안에(between) " stir;" 나 " turn;" 을 10 초(duration) 동안 수행하는 명세이다.

```
……
from now to now+ 10 min every 40 sec
execute 10 sec within 10 sec do
begin
stir;
if degree > 100 then
  turn;
end;
……
```

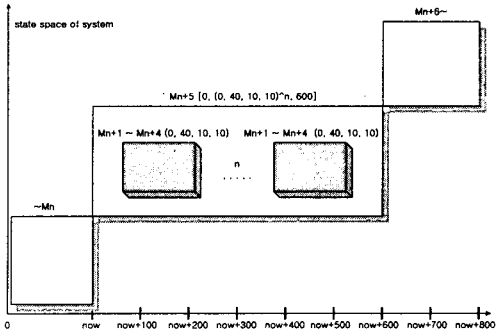
[표 1] 주기적 작업을 나타내는 예제[1]



[그림 3] 표 2 에 관한 ATM 명세 예제

[그림 3] 에서 알 수 있듯이 독립적인 동작을 수행하는 태스크는 하나의 머신으로 표현되고, 태스크 내의 주기적인 작업은 주기적인 시간 제약을 갖는 주제 모드로 표현된다. 주제 모드는 세부적인 시간 제약을 지니고 동작하는 하나 이상의 모드를 포함 한다.

시간을 기준으로 하여 특정 패턴을 지니고 동작하는 시스템의 상태를 모델링 했을 때, [그림 3] 에 대한 동작은 [그림 4] 와 같이 표현 될 수 있다. [그림 3] 의 머신이 활성화 되었을 때 머신 전체는 <~M_n. M_{n+5}. M_{n+6}~>와 같은 패턴을 가지고 동작하게 된다. 패턴의 한 요소 M_{n+5}는 다시 매40초 마다 반복적으로 실행되는 패턴을 갖는 n개의 작업으로 이루어진다.



[그림 4] 그림 3 에 관한 동작 모델

위와 같이 ATM 의 시간 명세는 주기적 작업과 같은 특정 패턴을 갖는 동작을 주제 모드와 같은 하나의 범위로써 포함한다. 시간의 범위에 대한 하나 이상의 동작의 집합에 대한 표현은 시간과 시스템에 대해 부분적으로 검증할 수 있다. 예를 들어 [그림 4] 에서 M_{n+5}와 같은 동작을 검증 하기 위해서는 now 에서 now+600 사이의 시스템의 동작을 검증하고, now 에서 now+600 사이의 시스템을 검증하기 위해서는 M_{n+5}를 검증하게 된다.

5. 결론 및 향후 연구

현재 본 연구는 명세 언어와 명세 도구로써의 개발을 완료하고 시스템 명세의 중요한 주제인 예외처리, 시간, 확률 등의 시스템의 여러 속성에 대한 연구와 검증에 필요한 명세로부터의 데이터 추출과 추출된 데이터로부터의 검증에 관한 연구가 진행 중이다. 향후에는 이와 같은 연구를 지속함과 동시에 검증에 관련된 도구 개발을 목표로 한다.

참고문헌

[1] A. Shaw, *"Communicating Real-Time State Machines,"* IEEE Transactions on Software Engineering, Vol. 18, No. 9, pp. 805-816, September 1992.

[2] D. Harel, "Statecharts: A Visual Formalism for Complex System," Science of *Computer Programming*. Vol. 8., pp. 231-274, 1987.

[3] Fernando Barber, Salvador Moreno, *"Representation of Continuous Change with Discrete Time,"* Proceedings of 4th International Workshop on Temporal Representation and Reasoning [TIME '97], May 1997.

[4] I. Kang and I. Lee, *"State Minimization for Concurrent System Analysis Based on State Space Exploration,"* Proceedings of Conference on Computer Assurance, Gaithersburg MD, June 1994.

[5] Insup Lee, *"Language Constructs for Distributed Real-Time Programming,"* Dec, 1985.

[6] Joachim fischer, Evegeni Dimitrov, Udo Taubert, *"Analysis and formal Verification of SDL'92 Specifications using Extended Petri Nets"*.

[7] S. Jahanian and A. Mok, *"Modechart : A Specification Language for Real Time Systems,"* IBM Technical Report: RC 15140, November, 1989.