Message Sequence Chart의 오토마타 시멘틱 정의 및 시멘틱 오토마타 추출 알고리즘

김태효○ 차성덕 배두환
한국과학기술원 전산학과

Automata-Based Semantics of MSCs and Algorithm for the Construction of Semantic-Automata

Tai-hyo Kim, Sung-duck Cha, and Doo-hwan Bae
Dept. of Computer Science, KAIST

요 약

Message Sequence Charts는 요구 사항 및 설계 단계에서 시스템의 시나리오 정보를 명세하는 수단으로 많이 사용되어 왔다. 하지만, 그 정형 시멘틱에 대한 연구가 미흡하여, 자신의 완결성 및 다른 명세와의 일치성 등의 검증이 힘든 상황이다. 본 논문에서는 Message Sequence Charts의 오토마타 기반 정형적 시멘틱을 정의한다. 제안된 오토마타 기반 정형적 시멘틱은 MSC에서의 모든 경로에 대한 행위를 유지하고, 그 오토마타는 MSC에서 발생할 수 있는 이벤트의 연속과 일치한다. 또한 본 논문에서는 오토미타의 추출을 위한 알고리즘을 제공한다. 이를 통하여 오토마타 추출과정이 자동화 됨으로써 다른 상태 기반 언어와의 검증등이 용이하게 되고, MSC의 자체 검증에도 도움을 준다.

1. 개요

MSC(Message Sequence Chart)는 시스템 개발시 요구 사항의 시나리오 정보를 표현하기 위해서 많이 쓰여왔다. 하지만, MSC의 경우 그 정형 시멘틱(Formal Semantics)에 대한 연구가 미흡하여, 그 시멘틱의 정의 또한 현재 많이 연구 되고 있다

본 논문에서는 오토마타 이론에 기반을 둔 MSC의 시멘틱을 정의하고, MSC에 해당하는 행위 오토마타를추출하는 알고리즘을 제안 하고자 한다.

본 논문의 구성은 우선 2.장에서 MSC의 정형적 시멘틱을 위한 연구들을 소개하고, 3.장에서는 행위 오토마타를 추출하기 위한 중간 단계인 단순 메세지 흐름 그래프의 제안 및 MSC에서의 전환과정을 제시하겠다. 또한, 이 단계에서 본 논문에서 제한하고 있는 MSC의 요소들을 기정한다.

다음으로, 4.장에서는 단순 메세지 흐름 그래프에 해당하는 행위 오토마타의 정의와 행위 오토마타 추출 방법을 제시하고, 5.장에서 결론을 맺도록 한다.

2. 관련 연구

MSC의 종류는 크게 문법구조와 시멘틱에 따라서 나눌 수 있다. 문법의 경우 ITU의 Z.120표준에서의 Basic MSC와 High-level MSC를 생각할 수 있고, 시멘틱의 경우 ITU의 Z.120의 프로세스 대수학(proccss algebra)에 기반을 둔 표준 시멘틱[3]과 다른 접근 법으로써, Ladkin과 Leue의 MFG[1], 그리고 Damm과 Harel의 LSC에 대한 연구[2]등을 들 수 있다.

프로세스 대수학으로 정의된 표준 시멘틱의 경우, 많은 명세 언어들이 상태 기계(state machine)를 기반으로 하고 있고, 그렇기 때문에 다른 명세들과의 관계를 검증하기 힘들다는 단점이 있다.

MFG는 MSC를 Buchi 오토마타의 형태로 생성하여 형평성(fairness)을 표현하기 위한 연구이고, LSC는 MSC의 존재성(existenctial)과 항상성(universal)에 대한 구분을 통하여 MSC의 의미를 더 명확히 하기 위한 연구이다.

하지만, LSC의 경우 아직 정형 시멘틱이 미흡한 실정이고, MFG의 경우 Buchi 오토미타의 종료 상태에 대한 정의를 사용자가 개입하여 정의하기 때문에 완전히 자동적이지 못하다.

본 논문에서는 MFG에서의 접근 방법을 기반으로 하여, Buchi 오토마타가 아닌 유한 상태 기계(finite state machine)으로 MSC의 시멘틱을 표현함으로써, 형평성을 제외하였고 간단한 전환 알고리즘을 통하여 오토마타 생성을 자동화 하고자 한다.

3. 단순 메세지 흐름 그래프(sMFG)

본 논문에서는 MSC의 행위 오토마타의 추출에 앞서, 추출을 위한 중간 단계인 단순 메세지 흐름 그래프를 정의하고자 한다. 이 단순 메세지 흐름 그래프는 Ladkin과 Leue의 MFG에 관한 연구[1]와 유사하나, 행위 오토마타 그래프를 추출하기 위한 위치 정보를 추가 하였다.

이 장에서는 본 논문의 범위인 MSC의 문법구조 정의를 3.1장에서, 그에 해당하는 단순 메세지 흐름 그래프를 3.2장에서 제안하고자 한다

3.1 MSC의 문법구조 제한

MSC의 문법구조는 다양한 형태를 가진다. 그 예로, ITU Z120의 basic MSC와 high-level MSC[4], 그리고 UML의 Sequence Diagram등을 예로 들 수 있다.

기본 MSC의 경우, 몇 가지의 요소를 제외하면 단순히 인스턴스(Instance) 사이의 메세지(Message)의 전달구조로 가정할 수 있다. 즉, 메세지를 어떠한 이벤트(event)로 가정하였을 때, MSC

는 단순한 이벤트의 연속(sequence)이다.

이와 같은 가정을 위하여 본 논문에서는 기본 MSC의 몇 가지 요소를 제한하였다. 그 요소로는 1)동적 요소*, 2)조건(Condition), 3)시간(Timer), 4)동시지역(Coregion)이 있다. 동적 요소및 시간의 관련 연구는 아직 미흡한 실정이고, MSC의 행위 의미구조를 위한 연구외에 많은 논점을 가지므로 제외하였고, 조건의 경우 이벤트 이외의 변수(variable)에 대한 고려를 필요하고, 동시지역의 경우 동시성(Concurrency) 문제의 정의를 필요로 하나 본 논문의 방법에서 쉽게 확장될 수 있다.

3.2 단순 메세지 흐름 그래프의 정의 및 시멘틱

3.1장에서 정의 한 바와 같이 MSC는 하나의 이벤트의 연속으로 간주 될 수 있다. 이 가정을 바탕으로 3.2장에서는 MSC의 행위 오토마타 추출을 위한 중간 단계인 단순 메세지 흐름 그래프를 제안한다. 단순 메세지 흐름 그래프는 Ladkin과 Leue의 메세지 흐름 그래프에 관한 연구[1]와 유사하나, 행위 오토마타 추출을 위한 위치정보를 추가하였다. MSC에서 단순 메세지 흐름 그래

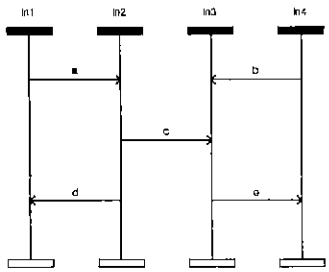

그림 1: MSC예제

프를 추출하는 과정을 간단히 설명하면 다음과 같다. 우선, 그림1에서의 메세지를 중심으로 메세지를 주는 위치와 받는 위치에 각각의 노드를 생성한다. 그리고, 한 인스턴스에서 시작하는 위치에 Top노드와 끝나는 위치에 Bottom노드를 생성한다. 한 노드의 위치 값의 경우 각 인스턴스에 대하여 순서대로 자연수를 부여한 것이고, Bottom노드의 위치 값만 ∞로 표현하였다. 또한, 한 인스턴스안의 노드들의 경우 연속 이벤트로 분류하였고, 인스턴스 사이의 메세지의 경우 신호 이벤트로 분류하였다. MSC의 메세지 m에 해당하는 신호 이벤트의 경우, 발생하는 노드에는 !m 그리고 받는 노드에는 ?m의 이벤트 이름을 갖는다. 이와 방법으로 그림 1을 전환한 예제는 그림2와 같다. 그림2에서 실선은 신호 이벤트를, 점선은 연속 이벤트를 뜻한다.

단순 메세지 흐름 그래프를 정형적으로 정의하면 다음과 같은 튜플(tuple)로 정의 될 수 있다

$G = (E, N, NE, SIG, Top, Bottom, IN)$

- E : a set of events ,N : a set of nodes
- NE : $NE \subseteq (N \cup Top) \times (N \cup Bottom)$ (연속 이벤트)
- SIG $SIG \subseteq N \times N$ (신호 이벤트)
- Top(Bottom) : a set of Top(Bottom) nodes
- IN : a set of instance names

또한, node는 다음과 같은 튜플로 정의 된다.

$N =< I, EN, L >$

- in : instance name ($in \in IN$)
- en : event name (!en or ?en s.t. $en \in E$)
- l. location ($l \in \aleph \cup$ '+')†

본 논문에서는 설명을 위하여 다음과 같은 함수를 정의하였다.

- 인스턴스 추출 : i_name(n) = in, if $n =< in, en, l >$ s.t. $n \in N$
- 이벤트 추출 : e_name(n) = tail‡(en)
- 위치 값 추출 : num(n) = l

위의 정의를 바탕으로 단순 메세지 흐름 그래프의 위치 값은 다음을 만족한다

1. Top노드 t_i와 Bottom노드 b_i에 대하여 :
   - $t_i =< i, \emptyset, 0 >$, $if\ i \in IN,\ t_i \in Top$
   - $b_i =< i, \emptyset, + >$, $if\ i \in IN,\ b_i \in Bottom$
   - $|Top| = |Bottom| = |IN|$ (각각의 인스턴스에 하나씩의 Top과 Bottom노드가 존재한다.)
2. 한 인스턴스에 해당하는 노드는 연속된 위치 값을 갖는다 :
   - $num(n_1) + 1 = num(n_2)$ and $i\_name(n_1) = i\_name(n_2)$, if $ne = (n_1, n_2)$ and $n_2 \notin Bottom$ s.t. $ne \in NE$

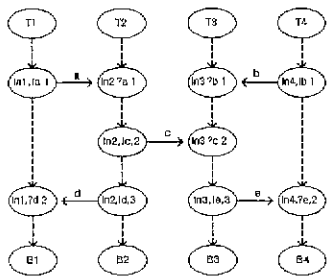

그림 2: 단순 메세지 흐름 그래프의 예제

4. Simple Automata

3.장에서는 MSC에 대한 행위 오토마타를 생성하기 위한 단순 메세지 흐름 그래프를 정의하였다. 4 장에서는 단순 메세지 흐름 그래프를 바탕으로 MSC에 대한 행위 오토마타를 정의하고 행위 오토마타를 추출하기 위한 알고리즘을 제시한다

단순 메세지 흐름 그래프에서 행위 오토마타로 전환하는 과정은 각각의 인스턴스에 해당하는 위치 정보에 대한 벡터(vector)를 상태로 간주하여, 한 이벤트가 일어 날 때 바뀌어 진 위치 정

*인스턴스의 동적 생성 및 삭제, 메시지의 발견(found)및 유실(loss)

†위치 값은 자연수나 +기호를 갖는다

‡일의 기호를 제외한 것

보에 대한 벡터를 다음의 상태로 정의하여 이루어 진다. 예를 들어 설명하면 다음과 같다.

초기 상태는 어떠한 이벤트도 일어 나지 않은 상태이다. 따라서, 그림2 에서 모든 인스턴스의 위치 값이 1인 경우이다. 따라서, 이러한 상태인 $<1,1,1,1>$이 단순 행위 오토마타의 초기 상태가 된다.

이 상태에서 일어날 수 있는 이벤트는 a와 b가 있으므로 각각이 일어 났을 때의 상태인 $<2,2,1,1>$과 $<1,1,2,2>$가 행위 오토마타의 상태에 추가된다. 일어 날 수 있는 이벤트인지 결정하는 것은 신호 이벤트의 정의에 의해서 획득되는데, 신호 이벤트의 집합인 SIG에서 발생하고 받는 노드들의 위치 값이 현재 상태의 인스턴스 위치 값들과 일치 하면 발생할 수 있는 이벤트이다. 즉, 그림2에서 a와 b의 이벤트에 해당하는 신호 이벤트는 $\{<In1,!a,1>,<In2,?a,1>\}$과 $\{<In4,!b,1>,<In3,?b,1>\}$이다. 여기서의 모든 위치 값들은 모두 1이고, 현재 상태의 모든 인스턴스의 위치 또한 모두 1이다[§]. 따라서, a와 b의 이벤트는 모두 일어 날 수 있다. 이와 같은 방법을 통하여 얻어진 행위 오토마타는 그림3과 같다.

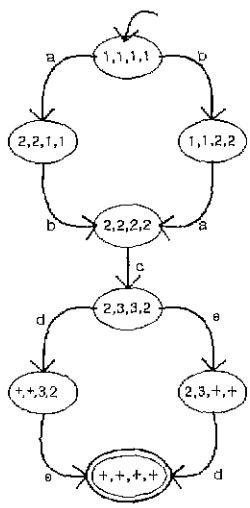

그림 3: 행위 오토마타의 예

본 논문에서는 MSC에 대한 행위 오토마타를 유한 상태 기계(Finite State Machine:FSM)으로 정의하였다. 행위 오토마타의 FSM은 다음과 같은 튜플로 정의된다.

$M = (Q, \Sigma, \delta, q_0, F)$

- Q : a set of state
- Σ . a set of input events
- δ : $\delta \subseteq Q \times \Sigma \times Q$
- q_0 . initial state
- F final state

어떠한 단순 메세지 흐름 그래프 G = (E, N, NE, SIG, Top, Bottom, IN)가 있을 때, 그에 해당하는 행위 오토마타는 다음과 같이 정의 할 수 있다.

초기 상태 벡터는 $q_0 =< 1,1,\cdots,1 >$이고, $|IN| = n$일때 n개의 요소를 가진다. 또한, 종료 상태 벡터는 $F =< +,+,\cdots,+ >$와 같다. Σ는 일어 날 수 있는 이벤트와 같으므로 $\Sigma = E$를 만족한다. 이와 같은 성질을 바탕으로 Q와 δ를 구하는 알고리즘은 다음과 같다

```
Q = δ = ∅;
do
 let c = < i1, i2, ···, in > s.t. c ∈ Q;
 for ∀sig ∈ SIG
  let sig = (n1, n2) s.t.
    n1 =< in1, !e, l1 >, n2 =< in2, ?e, l2 >)
    and p(q)th instance name is in1(in2);
  if ip = l1 and iq = l2 then
    Q = Q ∪ new s.t.
      new = < ···, ip + 1, ···, iq + 1, ··· >;
    δ = δ ∪ (c, e_name(n1), new);
  fi
 rof
while some changes in Q
```

5. 결론 및 향후 연구방향

본 논문에서는 MSC에 대한 상태 기계를 기반한 시멘틱을 정의하였다. 이를 위하여 중간단계인 단순 메세지 흐름 그래프를 정의하였고, 행위 오토마타로의 전환을 위한 알고리즘을 제안하였다. 본 논문에서 제안한 행위 오토마타는 기존 MSC의 모든 가능한 경로를 보유하고 있고, 간단한 알고리즘을 통하여 자동화될 수 있는 장점이 있다.

향후 연구 방향으로는 본 논문의 문법 구조 확장 및 상응하는 시멘틱의 확장 을 들 수 있고, 또한 상태 기계를 기반으로 한 다른 명세 언어, 즉 Statechart등과 의 일치성 검증에 대한 연구가 진행 중이다.

참고 문헌

[1] P B. Ladkin and S Leue, "Interpreting Message Flow Graph", *Formal Aspects of Computing*, 1994

[2] W. Damm and D. Harel, "LSCs: Breathing Life into Message Sequence Charts", *Proc of 3rd IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems*, 1999

[3] *ITU-TS Recommendation Z.120: Message Sequence Charts(MSC)*, ITU-TS, Geneva, 1996

[4] *ITU-TS Recommendation Z.120: Message Sequence Charts(MSC) - Annex B: Algebraic Semantics of Message Sequence Charts*, ITU-TS, Geneva, 1996

§ 상태 벡터가 $<1,1,1,1>$이므로