Statechart로 구현된 명세의 정형 검증 기법*

방기석○ 박명환 남원홍 최진영
고려대학교 컴퓨터 학과

Formal Verification for a Statechart Specification

Ki-Seok Bang○ Myung-Whan Park Won-Hong Nam Jin-Young Choi
Department of Computer Science and Engineering, Korea University

요 약

Statechart는 다른 정형 명세와는 달리 그림으로 시스템을 명세하기 때문에 정형기법에 익숙하지 않은 사람도 쉽게 이해할 수 있다. 또한 시스템의 동작을 보다 명확하고 가시적으로 시뮬레이션 할 수 있는 장점이 있다. 그러나 이 명세 방법은 시스템의 특성을 증명하는 정형 검증의 기능은 제공하고 있지 못한 것이 단점으로 지적되고 있다. 이러한 단점을 해결하기 위해 본 논문에서는 statechart로 기술된 명세를 정형 검증 언어인 SMV 및 PROMELA로 변환하여 검증하는 방법에 대해 논한다.

1. 서론

시스템이 복잡해지고 대형화 되면서 그 정확성이 더욱 중요하게 대두되고 있다 특히 원자력 발전소나 비행기 컨트롤 시스템과 같은 safety critical system의 경우 작은 오류가 시스템 전체 및 사용자에게 매우 위험한 결과를 초래할 수 있는 것이 사실이다. 따라서 시스템의 개발에 있어서 정확성과 안전성을 보장하기 위해서 많은 testing을 시도하고 있다. 그러나 이러한 testing에 들어가는 시간과 비용이 시스템의 크기에 비례해서 기하급수적으로 증가하기 때문에 대부분의 시스템이 이러한 안정성을 완벽하게 보장하지 못하고 있는 현실이다.

이러한 문제를 해결하기 위해서 최근에는 정형기법(Formal method)[1]을 이용해서 시스템의 설계단계부터 그 안정성을 확인하려는 노력을 가하고 있다. 정형기법은 수학과 논리학을 기반으로 하여 시스템의 동작을 표현하는 정형명세(Formal specification)와 명세된 시스템이 만족해야 할 특성(property)을 역시 논리적인 방법을 이용해서 증명하는 정형검증(Formal verification)으로 이루어져 있다.

이번 연구에서는 많은 정형 명세 방법 중 시스템의 동작을 상태 기계(state machine)의 형식으로 표현하는 statechart[3]를 사용하였다. Statechart는 다른 정형 명세와는 달리 그림으로 시스템을 명세하기 때문에 정형기법에 익숙하지 않은 사람도 쉽게 이해할 수 있다. 또한 시스템의 동작을 보다 명확하고 가시적으로 시뮬레이션 할 수 있는 장점이 있다. 그러나 이 명세 방법은 시스템의 특성을 증명하는 정형 검증의 기능은 제공하고 있지 못한 것이 단점으로 지적되고 있다.

이러한 단점을 해결하기 위해 본 논문에서는 statechart로 기술된 명세를 정형 검증 언어인 SMV 및 PROMELA로 변환하여 검증하는 방법에 대해 논한다. 본 논문에서는 간단한 예제로 Auto pilot system을 위한 statechart 명세를 이용하였다. 2장에서는 본 연구의 대상인 *auto-pilot system*에 대해 간략히 설명한다. 3장에서는 statechart에 대한 간략한 설명과 statechart를 이용한 Auto-pilot system 명세를 보인다. 그리고 4장에서는 구현된 명세를 SPIN과 SMV의 입력언어로 변환시켜 검증한 결과를 분석하고 마지막으로 결론을 맺는다.

* 본 연구는 1998년 과학기술기초 중점지원사업으로 이루어졌습니다

2. 예제 : Auto Pilot System

전투기를 조종하는 조종사가 기동 중 자주 경험하는 현상으로 black out이 있다. Black out이란 항공기가 급기동할 때 발생하는 중력가속도(G-force)로 인해 조종사의 머리에 있는 피가 아래로 몰리면서 눈의 신경이 작용을 하지 못하여 사물이 보이지 않는 상태를 말한다. 이 상태에서 더 많은 중력가속도가 가해지면 조종사는 의식을 잃게 되고 항공기는 추락을 하게 된다. 현재의 최신 항공기는 대부분 AFCS(Automatic Flight Control System)를 가지고 있어 자동비행이 가능하지만 이런 시스템도 조종사가 직접 AFCS를 작동하여야만 한다. 따라서 앞의 경우처럼 조종사가 의식을 상실했을 경우에는 AFCS는 무의미해진다. 이런 상태를 방지하기 위해서 본 논문에서는 조종사가 의식을 상실했을 경우 이것을 감지하여 항공기 제어를 자동적으로 AFCS에게 넘겨주는 Auto-pilot 시스템을 제안한다. Auto-pilot 시스템은 G-meter와 stick sensor, 그리고 beeper와 controller로 구성되어 있다 G-meter는 현재 항공기에 걸리는 중력가속도의 값을 측정하고 stick sensor는 현재 조종사가 stick을 잡고 있는지에 대한 정보를 제공한다. controller는 이 정보를 받아서 조종사가 현재 의식을 상실했는지를 판단하고 beeper를 울려 조종사에게 경고 메시지를 보낸다. 그리고 조종사가 반응을 하지 않으면 항공기 control을 AFCS로 넘긴다.

3. Statechart를 이용한 Auto-pilot 명세

본 절에서는 위에 제안된 예제를 statechart를 이용하여 명세한다. 이러한 명세는 STATEMATE[4] 도구로 사용하였고 시뮬레이션을 이용하여 간단히 test해 보았다.

먼저, *transition*의 *trigger*는 *event*와 *in condition*만 사용하였고 interlevel transition을 금지하였다. 그리고 closed system을 만들기 위하여 외부 event들은 시스템 내부에서

event generator를 사용하여 생성하였다. 또한, statechart의 timeout은 SPIN이나 SMV에서 적용되지 않기 때문에 sub-state들의 transition으로 이것을 해결하였다.

3.1 G_meter 명세

G_meter는 급격한 기동이 요구되는 항공기에 필수적으로 장착된 계기로서 항공기에 걸리는 중력가속도를 감지하여 조종사에게 알려준다. Auto-pilot 시스템에서 G_meter는 N_G 상태와 O_G 상태로 구성되어 있다. N_G 상태는 정상적인 중력가속도로서 이 상태에서 조종사는 절대로 의식을 상실하지 않는다. 그러나 O_G 상태는 항공기가 급격히 기동하고 있는 상태로서 언제든지 조종사는 의식을 상실할 수 있다.

이 두 상태간의 전이는 G_generator가 생성하는 두가지 event(N_G, O_G)들에 의해서 발생한다. 이 event들은 transition에 trigger가 없기 때문에 nondeterministic하게 발생한다. 이런 성질은 시스템이 도달할 수 있는 모든 가능한 경우를 검사하는 SPIN이나 SMV에서 유용하게 사용된다

3.2 S_sensor 명세

S_sensor는 조종사가 현재 stick을 잡고 있는 지에 대한 정보를 제공한다. Auto-pilot 명세에서는 조종사가 의식을 상실하면 stick을 놓는다고 가정한다. GR 상태는 조종사가 stick을 잡고 있는 상태이고 UN상태는 조종사가 stick을 잡고 있지 않는 상태이다.

3.3 BEEPER 명세

BEEPER는 ON과 OFF 상태가 있다. 이 두 상태간의 전이는 CONTROLLER가 보내주는 event들에 의해 발생한다.

3.4 Controller 명세

Controller는 S_sensor와 G_meter의 정보를 받아서 현재의 조종사 상태를 파악하고 만약 조종사가 의식을 잃었다면 항공기 control을 AFCS에게 넘긴다. 초기 상태는 NORMAL로서 S_sensor가 GR이고 G_meter가 N 상태일 경우이다. 즉, 정상적인 중력가속도에서 조종사가 stick을 잡고 있는 상태를 말한다 이 상태에서 O_G가 발생하면 READY 상태로 전이된다. 이것은 항공기가 급격히 기동하게 되어 조종사가 의식을 상실할 수도 있는 상태가 된다. READY 상태에서 조종사가 stick을 놓으면(UNGRIP event가 발생) WARNING 상태로 들어간다. READY 상태에서 정상적인 중력가속도로 바뀌면 MONITOR 상태로 가게 되는데 이 상태는 많은 중력가속도를 경험한 후 힘이 빠진 조종사를 시스템이 감시하고 있는 상태이다. MONITOR State의 3개의 sub-state는 시간의 흐름을 나타내며 이 시간동안 조종사가 계속 stick을 잡고 있으면 정상이라고 간주하고 NORMAL 상태로 가지만 stick을 놓으면 의식을 상실했을 수도 있으므로 WARNING 상태로 들어간다. WARNING 상태에서는 beeper가 켜지면서 조종사에게 stick을 다시 잡으라고 경고메시지를 보낸다. WARNING State의 3개의 sub-state가 전이되는 시간까지 조종사가 stick을 잡지 않으면 의식을 상실했다고 가정하고 AFCS 상태로 들어간다 만약 stick을 잡으면 그 때의 중력가속도의 상태에 따라 NORMAL이나 READY 상태로 돌아간다. AFCS 상태는 시스템이 항공기를 조종하는 상태이다. 이 상태에서는 beeper가 계속 켜져 있으며 조종사가 stick을 잡으면 NORMAL 상태로 가면서 beeper는 꺼진다.

3.5 전체 명세

그림 1은 STATEMATE를 이용해 설계한 Auto pilot system의 statechart 명세이다.

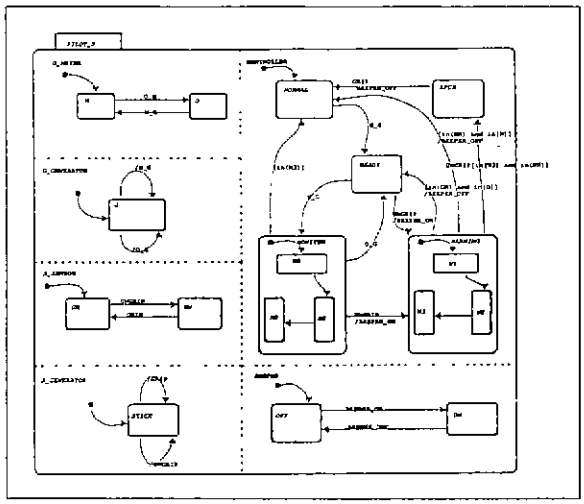

그림 1: Auto pilot system의 statechart 명세

4 . Statechart설계의 정형검증

위에서 설명한 바와 같이 statechart는 시스템을 정형 명세하고 시뮬레이션 하는 기능은 제공해 주지만 검증하는 기능은 없다. 이러한 단점을 지원하기 위해 본 연구에서는 SPIN과 SMV와 같은 정형 검증 도구를 이용하여 명세된 시스템을 검증하는 방법을 제안하고자 한다. 우선 statechart명세를 정형 검증 도구의 입력으로 변환하기 위해 STATEMATE로 만들어진 statechart명세를 EHA(extended hierarchical automata)로 변환하여 statechart의 여러가지 요소를 분석한다 이렇게 분석된 결과는 statechart 명세에 나타나는 상태, 이벤트, 그리고 전이 관계이다. 이것을 SPIN의 입력 언어인 PROMELA와 SMV의 입력 언어로 구현한다. 이렇게 구현된 코드를 SPIN과 SMV에 입력하여 검증할 수 있다.

4.1 SPIN을 이용한 검증

우선 statechart명세를 정형검증도구 SPIN[5]의 입력언어인 PROMELA[5]로 변환을 한다 SPIN은 상태공간의 축약기술을 이용해서 상태 공간을 축약하고 LTL[6]을 이용해서 특성을 검증하는 모델체커이다. PROMELA 역시 Buchi automata의 동작을 이용하기 때문에 상태의 전이에 따라 동작하는 시스템의 기능을 잘 표현할 수 있다. MOCES에 의해 생성된 EHA를 이용하여 PROMELA의 형식으로 변환시키면 statechart로 명세된 시스템과 동일한 동작을 명세한 PROMELA 코드를 구현할 수 있다. 이렇게 변환된 모델을 SPIN을 이용해서 시뮬레이션 하고, 이 시스템에 만족해야 할 특성에 대해 LTL을 적용시켜 검증을 한다. 본 연구에서는 시스템의 안정성을 보장하기 위해 다음과 같은 특성을 검증하였다

(1) 조종사가 의식이 있는 경우 AFCS 상태로 가지 않는다. 조종사가 의식이 있다는 것은 stick을 잡고 있는 것을 의미하므로, 시스템이 항상 stick을 잡고 있는 것을 나타내는 GR 상

태와 AFCS 상태에 동시에 있지 않음을 의미한다. 이 특성을 LTL로 표현하면 다음과 같다.

! [] ! (in_GR && in_AFCS)

(2) 조종사가 의식을 잃고 일정 시간이 지나도 깨지 않으면 AFCS 상태로 전이한다. 여기서 일정 시간은 항공기가 조종사의 제어 없이도 추락하지 않을 정도의 시간이다. 즉, 이 특성은 제한 시간안에 AFCS 상태로 전이 하지 않으면 항공기가 추락할 수도 있는 safety property이다. 이 특성을 LTL로 표현하면 다음과 같다.

! [] ((in_ON && in_UN && in_W3 && gen_UNGRIP) →<> in_AFCS)

위의 식을 이용하여 검증한 결과 Auto Pilot System이 안정성 조건을 만족하고 정상적인 작동을 보이는 것을 알 수 있다.

4.2 SMV를 이용한 검증

SMV[7] 시스템은 유한 상태 시스템(finite state system)이 CTL[6]로 표현된 요구 명세를 만족하는지를 검증하는 정형 검증 도구이다. SMV의 입력 언어는 유한 상태 시스템을 명세하기 위해 만들어 졌으며, 시스템은 이 입력 언어를 통해서 동기적인 밀리 머신(Mealy machine)이나, 비동기적인 네트워크로 손쉽게 명세될 수 있다. CTL은 safety, liveness, fairness, dead lock freedom을 포함한 풍부한 종류의 시간적 특성들을 간단한 문법을 이용하여 표현하는 것이 가능하다. SMV는 입력 언어로 나타내어진 모델이 CTL로 표현된 요구 명세를 만족하는지의 여부를 효율적으로 검사하기 위해서, OBDD(Ordered Binary Decision Diagram)[8]를 기반으로 하는 심볼릭 모델 체킹 알고리즘을 사용한다. 이러한 SMV를 이용하여 대형 하드웨어나 소프트웨어 시스템 명세를 검증하는 예가 연구되고 있다[2]. 이와 같은 SMV를 이용하여 Simple Auto Pilot System의 안전성을 위하여 다음과 같은 몇 가지 특성을 검증한다.

(1) 조종사가 의식이 있는 경우 AFCS 상태로 가지 않는다. 조종사가 의식이 있다는 것은 stick을 잡고 있는 것을 의미하므로, 시스템이 항상 stick을 잡고 있는 것을 나타내는 GR 상태와 AFCS 상태에 동시에 있지 않음을 다음과 같은 CTL 문장을 이용하여 검증한다.

AG !(in_GR & in_AFCS)

(2) 조종사가 의식을 잃고 일정 시간이 지나도 깨지 않으면 AFCS 상태로 전이한다. 여기서 일정 시간은 항공기가 조종사의 제어 없이도 추락하지 않을 정도의 시간이다. 즉, 이 특성은 제한 시간안에 AFCS 상태로 전이 하지 않으면 항공기가 추락할 수도 있는 safety property이며 다음과 같은 CTL 문장으로 나타내어서 검증한다.

AG (in_ON & in_UN & in_W3 → AX in_AFCS)

검증 결과 위의 두 가지 경우를 모두 시스템이 만족하는 것으로 검증되었다. 즉, 본 논문에서 제안한 Simple Auto Pilot System은 조종사가 의식이 있는 경우 AFCS 상태로 가지 않으며, 조종사가 의식을 잃고 일정 시간이 지나도 깨지 않으면 AFCS 상태로 전이한다는 것을 검증하였다

그러나, 1차 검증 시 CONTROLLER 부분에서 명세의 모호함이 발견되었다. 즉, CONTROLLER가 WARNING 상태에 있고 3초가 지나는 순간 조종사가 의식을 되찾은 경우, 명세의 모호함이 SPIN과 SMV 두 도구를 이용한 검증 과정에서 모두 발견되어 수정하였다.

5 . 결론

정형 기법은 수학과 논리학에 기반을 둔 방법으로 정형 명세는 자연어가 내포하는 애매모함과 불확실성을 배제할 수 있으며, 정형 검증은 시스템이 어떤 특성을 만족하는지를 검증하기 때문에 최소한 검증된 특성에 대해서는 완전히 믿을 수 있다. 본 논문은 Simple Auto Pilot System을 statechart를 이용하여 정형 명세하였다. Statechart는 graphical한 기호를 사용하는 장점이 있는 반면 정형 검증의 기능은 제공하지는 않는다. 이러한 단점을 극복하고 정형검증을 지원하기 위해 statechart의 명세를 MOCES를 이용하여 정형검증 도구 SPIN과 SMV의 입력 언어인 PROMELA와 SMV로 변환하였다. 이렇게 변환된 입력언어를 SPIN과 SMV를 이용하여 몇 가지 safety 특성에 대해서 검증했다. 검증 결과 명세된 시스템의 안정성을 증명할 수 있었다 statechart로 명세된 시스템에 대해 정형 검증의 기능을 제공할 수 있었다. 이 과정에서 statechart와 모델 체커간의 상이점으로 인해 statechart 언어를 몇 가지 제한하였다. 이러한 제한 중에서 timeout, scheduled와 같은 시간에 관련된 condition과 level이 다른 state간의 전이인 interlevel transition, 그리고 action에서의 assignment 사용 등 본 연구에서 제한된 statechart 언어의 모델 체커 입력 언어로의 변환 등이 향후 과제라고 하겠다.

참고 서적

[1] Andre. M. van Tilborg, *Foundations of Real-Time Computing-Formal Specifications and Methods*, Kluwer Academic Publishers, 1991

[2] William Chan, et. al., "Model Checking Large Software Specifications", IEEE Transaction on Software Engineering, vol. 24, no.7, pp.498-519, July 1998.

[3] David Harel, "Statecharts:A Visual Formalism for Complex Systems", Science of Computer Programming 8, 1987.

[4] David Harel and Amnon Naamad, "The STATEMATE Semantics of Starecharts", ACM Trans. Soft. Method, 1996.

[5] Gerald J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall, 1991

[6] Zohar Manna and Amir Pnueli, *The Temporal Logic of Reactive and Concurrent Systems - Specification*, Springer-Verlag, 1996

[7] Kenneth L. McMillan, *SYMBOLIC MODEL CHECKING*, Kluwer Academic Publisher 1993.

[8] R E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Transaction Computers, vol. 35, no. 6, pp.677-691, Aug. 1986.