엄밀 검증 : 버그로부터 자유로운 소프트웨어를 향한 길

20180368 안해찬

Abstract

버그로 부터 자유로운 소프트웨어를 만들 수 있을까? 우리는 수학적 증명의 힘을 프로그램의 버그 없음을 보이는 데 사용할 수 있다. 이를 **엄밀 검증(formal verification)**이라 부른다. 엄밀 검증이란 (1) 프로그램(program)이 (2) 명세(specification)를 따른다는 (3) 증명(proof)을 하는 행위다.프로그램이 명세를 따른다는 증명은 **자동 검증기(automated verifier)**에 의해 자동화 혹은 반(半) 자동화 될 수 있지만, 복잡한 증명은 어렵다. 하지만 지금은 꽤 많은 엄밀 검증 도구들이 발전되었고, 실제 프로그램도 엄밀 검증이 되기 시작했다. 앞으로 엄밀 검증이 어떤 문제를 고려해야 하고, 어떤 분야에 활용되어야 할까. 첫째, 검증기의 버그를 증명하기 위한 검증기가 필요하지 않을까? 둘째, 버그가 없다는 증명보다 버그가 있다는 증명이 중요한 프로그램이 더 많으므로 그 방향으로 검증기를 발전시켜야 하지 않을까. 셋째, 인공신경망도 결국 프로그램이므로 이들의 검증이 필요한데, 요약 실행과 확률 모델을 바탕으로 검증할 수 있지 않을까. 위 문제들이 해결된다면 엄밀 검증이 널리 퍼지는 일은 꿈이 아니게 될지도 모른다.

버그로 부터 자유로운 소프트웨어를 만들 수 있을까? 소프트웨어 버그는 소프트웨어의 탄생과 함께 개발자의, 더 나아가 인류의 숙적이 되었다. 아무리 똑똑한 개발자들이 모여도 하트블리드, 멜트다운, Log4j 등 심각한 수준의 버그는 쉼없이 발생해 왔다. 개발자의 노력 만으로는 버그가 없음을 확신할 수 없다는 뜻이다. 좀 더 체계적이고 명확한 근거가 필요하다. 그런데 어떤 주장이 맞다는 것을 명확하게 보이는 것은, 수학 그리고 증명이라는 세계에서 일상적으로 일어나는 일이다. 우리는 수학적 증명의 힘을 프로그램의 버그 없음을 보이는 데 사용할 수 있다. 이를 **엄밀 검증(formal verification)**이라 부른다.

엄밀 검증이란 (1) 프로그램(program)이 (2) 명세(specification)를 따른다는 (3) 증명(proof)을 하는 행위다. 여기서 명세는 우리가 기대하는 프로그램의 행동 방식을 의미한다. 예를 들어, 리스트를 정렬하는 프로그램이 있다고 생각해 보자. (1) 프로그램은 정렬 프로그램 그 자체이다. (2) 명세는 "결과 리스트는 정렬되어있다.", "결과 리스트의 내용물은 원래 리스트의 내용물과 동일하다", "정렬 프로그램이 무한 루프에 빠지지 않는다" 등이다. (3) 증명은 프로그램이 명세를 따른다는 증명이다. 증명을 위해선 프로그램의 각 명령어가 수학적으로 어떤 의미를 가지는 지 정의해야 하고, 명세를 수학적으로 표현해야 한다. 그런데, 복잡한 수학적 증명을 사람이 어느 세월에 다 하나?

프로그램이 명세를 따른다는 증명은 **자동 검증기(automated verifier)**에 의해 자동화 혹은 반(半) 자동화 될 수 있지만, 복잡한 증명은 어렵다. 프로그램은 점점 더 커지고, 점점 더 복잡해졌다. 당연히 명세의 증명도 점점 더 어려워 지고 복잡해지고 있다. 복잡한 증명은 일단 증명을 하는데 시간도 많이 걸린다. 이에 더해 자동 검증기가 증명할 수 없는 부분을 인간이 수동으로 채워주어야 하기도 한다. 이런 문제로 인해 엄밀 검증은 아주 널리 퍼지진 못해왔다.

하지만 지금은 꽤 많은 엄밀 검증 도구들이 발전되었고, 실제 프로그램도 엄밀 검증이 되기 시작했다. Coq, Z3, HOL 등 사용자 친화적이고 효율적인 엄밀 검증 도구들이 우리의 손에 놓여 있다. 발전에 힘입어서, 실제 세계의 프로그램들도 조금씩 엄밀 검증이 되기 시작했다. Chrome과 Firefox 웹 브라우저의 일부분은 검증된 코드를 가지고 있다. seL4라는 마이크로커널이나 AWS의 분산 알고리즘도 엄밀 검증을 거쳤다. 이외에도 CompCert 프로젝트에선 C의 컴파일러를 검증하기도 하는 등, 엄밀 검증은 날갯짓을 준비하고 있다.

앞으로 엄밀 검증이 어떤 문제를 고려해야 하고, 어떤 분야에 활용되어야 할까: 자동 검증기를 검증해야 할까? 과연 얼마나 많은 "검증이 필요한" 프로그램이 세상에 있는가? 머신 러닝이나 딥러닝도 결국 프로그램으로서 구현되는데, 그들의 검증은 어떻게 할 것인가? 버그 없음의 확신도를 정량화할 수 있을까?

첫째, 검증기의 버그를 증명하기 위한 검증기가 필요하지 않을까? 검증기가 특별한 이유는 프로그램에 버그가 없음을 확신할 수 있게 해주기 때문이다. 그래서 느리더고 힘들더라도, 보안이나 안전에 관련된 프로그램은 검증을 거친다. 그런데 검증기 자체에 버그가 존재하면 그거야 말로 헛수고 아닌가. 이를 해결하기 위해선 검증기를 입력 프로그램으로 받는, 특화된 검증기가 필요하지 않을까? 다행히도 검증기 프로그램은 일반적인

프로그램에 비해 프로그램 논리가 간단하고, 더 안전한 언어를 사용하기 때문에 자동화된 검증이 어렵지는 않으리라 생각한다. 여기서 "검증기의 검증기"는 검증기보다 간단한 프로그램이어야 할 것이다. Ocaml로 구현된 C의 정적분석기(정적분석기도 큰 틀에서 보면 엄밀 검증기이다)를 Coq으로 검증하는 예시가 있겠다. 한 발짝 더 나아가, Coq 구현을 검증하는 한 단계 더 수학에 가깝고 추상적인 언어로 구현된 검증기를 쓸 수도 있지 않을까?

둘째, 버그가 없다는 증명보다 버그가 있다는 증명이 중요한 프로그램이 더 많으므로 그 방향으로 검증기를 발전시켜야 하지 않을까. 과연 얼마나 많은 "버그 없음"검증이 필요한 프로그램이 세상에 있을까? 오늘의 학식 메뉴를 보여주는 프로그램에서 1년에 한 번 꼴로 버그가 발생한다 해보자. 개인마다 다를 수 있지만, 대부분의 사람은 "오늘은 잘 안되네"라 생각하고 넘어갈 것이다. 지금 세상에 나와 있는 많은 프로그램들은 위와 비슷한 이유로 엄밀 검증을 통해 버그 없음을 증명하지 않는다. 그렇다면 버그가 완전히 없음을 증명하기 보단, 버그가 여기엔 무조건 하나 있음을 증명하는 검증기를 만드는 게 우선아닐까?

셋째, 인공신경망도 결국 프로그램이므로 이들의 검증이 필요하고, 요약 실행과 확률 모델을 바탕으로 검증할 수 있지 않을까. 검증이 필요한 프로그램은 언급했듯이 보안이나 안전에 연관된 것들이다. 그런데 인공신경망 기술이 최근 부상하면서 자율주행처럼 사람의 목숨이 달린 일에도 사용되기 시작했다. 인공신경망을 전통적인 프로그램의 관점에서 본다면 "당연히" 검증이 필요하다. 하지만 인공신경망은 설명 불가능함, 이해하기 어려움이 그 분석에 있어서 큰 문제다. 이를 해결하기 위해선 인공 신경망의 개별 층(layer)이나 퍼셉트론(perceptron)들을 요약(abstract)해야 할 것이다. 그리고 인공 신경망에 투입될 수 있는 가능한 (무한 개의) 입력들을 요약해서, 요약 실행(abstract interpretation)을 통해 성질들의 만족 여부를 확인한다. 또한 인공 신경망의 일부분과 비슷하게 작동하는 확률 분포 함수를 찾아서 요약 실행과 결합한다면 인공 신경망의 설명 불가능함을 보완할 수 있으리라 생각한다.

버그로 부터 자유로운 프로그램을 만드는 것은 개발자와 연구자들의 오랜 염원이었다. 프로그램의 엄밀 검증이 그 방법이다. 하지만 전통적으로는 엄밀 검증을 위해 필요한 증명이 너무 복잡해서 실제 세계의 프로그램엔 많이 적용되지 못했다. 다행히도 최근 많은 검증 도구가 마련되면서 점점 더 복잡한 프로그램의 검증이 이루어지고 있다. 앞으로 엄밀 검증은 검증기 구현 자체의 검증과, 버그가 있다는 검증 그리고 인공신경망 프로그램의 검증에 주의를 기울여야 할 것이다. 만약 이루어 진다면, 엄밀 검증이 널리 퍼지는 일은 꿈이 아니게 될지도 모른다.