안전한 프로그램: 엄밀함과 느슨함으로

김태은

2022.04.14

Abstract

전통적 의미의 안전한 프로그램은 현실 세계의 안전과 직접 연결되는 좁은 개념이었다. 그러나 최근, 다양한 프로그램의 사용이 일상화되며, 안전한 프로그램에 대한 정의가 넓어졌으며 수요는 더욱 늘어났다. 따라서 프로그램이 안전한지 검사할 수 있는 기술이 필요한데, 그것은 바로 엄밀한 검증 기법(Formal Verification Method)이다. 엄밀한 검증 기법 그 자체로는 성능이 확실하지만 사용하기가 어려운 단점이 있다. 하지만 인공지능의 보조와 함께 사용된다면 확실한 성능과 우수한 사용성을 동시에 갖추어, 안전한 프로그램에 대한 넘치는 수요를 감당 할 수 있는 방법 중 하나가 될 것이다.

전통적인 의미에서의 안전한 프로그램은 말그대로 현실 세계에서의 안전과 직접 연결되는 좁은 개념이었다. 비행기 항법 장치, 원자로 제어 프로그램 등의 프로그램은 잘못 설계되었을 시 대형 사고로 이어지며 무수한 인명피해가 발생하게 된다. 따라서 과거에는 이러한 프로그램들을 안전하게 만드는 것이 사람들의 주요 관심사였다. 그렇기에 자연스럽게 안전한 프로그램의 공급자와 수요자는 대부분 특수한 분야의 전문가들로 구성되었다. 반면, 안전한 프로그램에 대한 대중적인 필요는 미미하였다. 왜냐하면 대부분의 사람들의 프로그램 사용은 문서 작업이나 스프레드시트 작성에 그쳤기 때문이다.

하지만 사람들의 일상에 다양한 프로그램이 들어오는 시대가 열리며 안전한 프로그램의 적용 범위가 이전보다 넓어지고 있다. 이제 사람들에게 프로그램은 문서 작업 도구 이상의 의미를 가진다. 사람들은 채팅 프로그램을 통해 소통하고, 프로그램을 통해 물건을 구매하며, 은행 업무도 프로그램으로 처리하고, 프로그램이 운전을 보조하는 자동차를 탄다. 이렇게 일상적으로 사용하는 프로그램 중에는 문제가 일어나도 살짝 불편할 뿐인 사소한 것들도 있지만, 재산 피해나 인명 피해로 이어질 수 있는 프로그램도 존재한다. 더 나아가, 이제는 개인 정보 유출 등, 그 피해의 정도를 측정하기 어려운 문제도 발생할 가능성이 있다. 따라서 모든 사람들이 안전한 프로그램에 대한 수요자가 되었다. 안전한 프로그램의 수요는 이전과 비교할 수 없을 만큼 확장되었고, 지켜져야 하는 안전함의 종류도 더욱 다양해졌다.

좋은 소식은, 안전한 프로그램은 위한 다양한 기술들이 지금까지 연구되어 왔다는 것이다. 그 중 가장 보편적으로 사용되는 기술은 동적 테스팅 기법일 것이다. 동적 테스팅은 프로그램이 정상적으로 돌아가는지 실제 실행을 통해 확인하는 것이다. 여기서 정상이라 함은 개발자의 의도에 맞게 작업을 수행하고 또 종료하는 것을 말한다. 즉, 안전함을 확인하는 것이다. 동적 테스팅의 장점은 특별한 전문 지식 없이도 할 수 있다는 것이다. 프로그램의 기능에 대해 알고 있는 사람이면 입출력 예제를 잘 구성하여 실행할 수 있기 때문이다. 앞서 언급된, 문제가 일어나도 살짝 불편할 뿐인 프로그램들이 안전한지 확인할 때는 동적 테스팅만으로도 충분할 것이다. 아주 가끔씩 문제가 발생해도 괜찮기 때문이다. 하지만 조금의 위험이라도 감수할 수 없는 프로그램들에게 동적 테스팅은 탐탁치 않은 방법이다. 동적 테스팅의 가장 큰 문제점이 프로그램의 안전을 온전히 보장할 수 없는 점이기 때문이다. 어느 정도 복잡한 프로그램은 거의 무한한 경우의 수로 실행이 가능하고, 이 모든 것을 직접 실행하여 확인하는 것은 불가능하다. 따라서 동적 테스팅 외에도 프로그램이 안전한지 확실하게 보장할 수 있는 기술이 필요하다.

엄밀한 검증 기법(Formal Verification Method, 정형 검증으로도 불린다)은 프로그램이 안전하다는 것을 확실하게 보장한다. 엄밀한 검증은 프로그램을 수학적으로 증명하고자 했던 노력으로부터 시작되었다. 그 방법을 간단히 설명하자면 다음과 같다. 엄밀한 검증은 세 가지 요소로 이루어진다. 먼저, 프로그램에 대한 요구사항을 수학적으로 기술한 스펙(specification), 프로그램 그 자체, 그리고 프로그램이 스펙을 만족한다는 수학적 증명. 스펙이 제대로 작성되었다면, 엄밀한 검증 기법은 수학적 증명을 통해 프로그램이 안전한지 정확하게 판단할 수 있다. 최근에는 엄밀한 검증 기법이 마이크로커널, 웹 브라우저, 클라우드 등 다양한 곳에 적용되어 적재적소에서 안전한 프로그램을 보장하고 있다.

그러나 엄밀한 검증 기법의 가장 큰 단점은 사용하기가 너무 어렵다는 것이다. 프로그램을 작은 단위로 쪼개어 스펙을 일일히 정의해주고 이를 증명해야 하는데, 상용 프로그램은 나날이 복잡해지고 커지고 있다. 사람들은 이에 대응하기 위해 프로그램의 중요한 부분만 선별적으로 검증하거나, 프로그램이 돌아가는 기반인 커널만이라도

검증하여 최소한의 노력으로 전체적인 안전을 확보하려는 시도를 하고 있다. 하지만 여전히 일반 개발자에게는 그 최소한의 노력이 부담스럽다. 프로그램의 스펙을 수학적으로 정의하는 것 자체가 낯설고 어려운 작업이기 때문이다. 따라서 누구나 쉽게 사용할 수 있는 엄밀한 검증 도구를 개발하는 것은 어렵지만 반드시 완성해야 하는 과제이다.

그 해결책의 단서는 엄밀함의 반대 진영인 인공지능에서 찾을 수 있다. 엄밀한 프로그램 검증의 한 방법인 연쇄적 검사 방식(Inductive Assertion Method)을 생각해보자. 연쇄적 검사는 프로그램의 각 부분이 안전한지 검증함으로써 연쇄적으로 전체 프로그램의 안전을 확인하는 기법이다. 이를 위해서는 각 프로그램 조각들의 시작 조건(precondition)과 마무리조건(post-condition), 그리고 각 반복문에 대해 항상 유지되는 조건(loop invariant)이 필요하다. 이 각각의 조건들은 프로그램이 실행되는 환경과 프로그램에 요구되는 바가 수학적으로 표현된 것이다. 사람이 이를 일일히 정의하는 것은 굉장히 어렵고 번거로운 작업이다. 반면 인공지능은 이런 작업을 잘 할 수 있다. 일정한 패턴을 학습하여 문맥을 파악하고 그에 상응하는 답을 내놓는 것이 인공지능이 하는 일이기 때문이다. 따라서 인공지능의 도움을 받아 자연어 설명, 혹은 코드를 기반으로 필요한 조건들을 자동으로 도출할 수 있다면 프로그램 검증에 큰 도움이 될 것이다. 마치 NP 문제처럼, 답을 찾는 것은 어렵지만 답이 맞다는 것을 확인하는 것은 쉽기 때문이다. 증명은 검증기가 엄밀하게 진행하는 대신, 그 증명에 필요한 조건식들은 인공지능이 합성하는 것이다.

엄밀한 프로그램 검증 기술의 사례에서 보았듯이 앞으로 연구자들은 쉽게 안전한 프로그램을 만드는 것을 목표로 삼아야 할 것이다. 안전은 본질적으로 귀찮고 어렵기 때문이다. 원자로 제어 프로그램을 만든다고 하면 어떻게든 개발자를 혹사 시키고 자본을 더 투입해서 안전함을 추구할 것이다. 그러나 메신저 프로그램에도 같은 노력과 자본을 쏟을 수 있을까? 요즘 유행하는 메타버스의 개발자에게 같은 요구를 할 수 있을까? 쉽지 않을 것이다. 하지만 이제는 이런 프로그램들도 안전해야 하는 시대가 왔다. 따라서 최소한의 노력으로 최대한 안전한 프로그램을 만들 수 있는 기술이 더욱 필요할 것이다.