Abstraction in Software Verification

20170504 이주안

전산학 수업을 많이 들어본 것은 아니지만, 그간 들었던 6개의 전산학 과목에선 항상 ‘Abstraction’을 강조했었다. 오죽하면 Abstraction이 전산학의 전부라고 주장하시는 교수님도 계셨고, 전산학부 4년간 배워야 할 유일한 것이 Abstraction이라고 말씀하신 분도 계셨다. 아직 1학년이고 새내기에, 전산학부에 진학하지도 않은 내가 깨닫기엔 어려운 개념이었지만, 양홍석 교수님의 수업도 그 Abstraction에 대한 내용으로 보였다.

교수님은 수업에서 Software Verification이라는 분야에 Abstraction을 담아 내는 과정을 보여주셨다. Software Verification은 문자 그대로 소프트웨어를 검증하는 과정으로, 프로그래머의 실수 또는 여타 과정(보통은 프로그래머의 실수이거나 논리의 오류인 경우가 많다.)에서 발생한 컴파일 에러나 런타임 에러를 사전에 찾아내는 기술이었다. 튜링의 논문에서 시작한 Software Verification은 현대에 이르러 Abstraction이 본격적으로 적용되며 automation 기술이 발전되었다.

솔직히, Software Verification에 대해 처음 들어보는 기술이었고 abstraction에 대한 이해도 많이 부족했기에 수업 내용을 따라가기 힘들었다. 다만, 그 중에서도 개인적으로 익힌 부분이 있다면 항상 한 줄 한 줄 확인해 보는 것이 전부였던 디버깅 과정과 유사한 Software Verification이 전산학에서 크게 연구되는 분야라는 것과, 해당 분야가 나에게 꽤나 흥미로웠다는 사실이다. 특히, 자동으로 Software Error을 측정하는 부분은 과거 새내기 세미나 수업에서 비슷한 강의를 들은 적이 있었기에 더욱 흥미롭게 들을 수 있었다. “Testing and Pruning”이란 부분도 결국, try&catch의 논리와 유사한 것 같은데, 최근 대두되는 빅 데이터나 머신 러닝과 비슷한 원리로 알고 있어 더욱 흥미로웠다.

또한, Abstraction의 이해에 한발짝 더 다가갈 수 있는 기회가 된 것 같아 좋았다. Abstraction으로 이해한 것은 verifying 과정에서 실제 값이 아닌 State를 명시함으로써 추상화 시키는 과정이었는데, even + even, odd + odd 등의 예시에서 와 닿는 게 조금이나마 있어 나중에 다시 공부하는 과정에서 이해에 도움을 줄 수 있을 것 같았다.

프로그래밍 언어 본 수업에 대한 진도도 어렵다고 생각하던 참이었고, hw 를 통해 수업에서 악착같이 따라가던 길이 맞는지 하나하나 확인하던 과정이었다. 그러던 참에 추가적으로 듣게 된 강연은 솔직히, 처음엔 걱정이었다. 하지만 강연이 끝난 지금 software verification과 abstraction이라는 분야에 대해 조금이나마 알게 된 것 같아 뿌듯함이 남는다.