컴퓨터의 본질에 대한 통찰

2015410047 컴퓨터학과 진상수

컴퓨터의 본질은 무엇일까. 다르게 말하면 컴퓨터의 작동 원리를 정의하는 것은 무엇일까. 많은 사람에게 가장 먼저 떠오르는 것은 프로세서 안에서 셀 수 없이 일어나는 연산 하나하나일 것이다. 입력에서 데이터를 읽어 원하는 결과가 나올 때까지 연산을 하고, 최종적으로 목표한 결과를 출력하는 것이 컴퓨터의 목적이라는 정의는 구체적이며 우리에게 익숙하기까지 하다. 1학년 때의 컴퓨터 프로그래밍 강의가 컴퓨터가 정확하고 효율적인 연산을 할 수 있게 하는 방법을 배우는 것이 목적이었던 것처럼, 연산은 컴퓨터에 절대적인 존재였으며 이는 당연한 사실이었다.

그러나 계산 이론 강의는 지금까지 당연하다고 생각했던 사실에 대해 의문을 가지게 했다. 무엇보다 computation의 기본 원리를 다루는 계산 이론에서 연산은 지금까지 알고 있던 것과 너무 달랐다. automata와 같이 추상적으로 정의된 시스템은 현재 입력에 대한 결과를 연산하는 것이 아닌 시스템 내부의 상태 변화 과정을 연산하며, 연산한 결과를 출력하는 것이 아닌 입력을 평가해 정의된 시스템이 받아들이는지를 확인했다. (튜링 머신을 제외하면) 물리적인 출력이 없는 연산, 입력이 아닌 내부 상태를 바꾸는 연산이 어떻게 실제 컴퓨터에서 가능할 수 있을까. 상태를 변화 시키는 연산이 가장 기본적 연산이라면 이를 이용해 어떻게 일반적인 연산을 구현할 수 있을까.

이산 수학에서 공부한 내용도 혼란스럽기는 마찬가지였다. 괴델의 불완전성 정리와 정지 문제(halting problem) 모두 알고리즘, 즉 연산의 한계와 관련된 내용인데, 불완전성 정리는 무한 집합의 성질을 이용해 수학적으로, 정지 문제는 귀류법을 이용해 논리적으로 증명하였다. 이제 계산 이론의 상태 변화 연산에 이어 수학적 추론 연산, 논리적 추론 연산까지 컴퓨터의 본질을 구성하는 요소로 봐야 한다. 각 요소가 서로 매우 달라 양립할 수 없어 보이는데 이러한 요소들이 컴퓨터를 구성할 수 있다는 것이 의심스러웠고 각 요소들이 실제 프로그램과 어떤 관계를 가지고 있는지 알 수 없었다.

이광근 교수님이 지은 '컴퓨터과학이 여는 세계' 책은 컴퓨터의 본질을 이루는 세 가지 요소들과 이산 수학과 계산 이론의 내용들이 복잡하게 얽혀 충돌하고 있는 이 상황을 명쾌하게 풀어 정리해 주었다. 상태 변화 연산, 수학적 추론 연산, 논리적 추론 연산. 이 세 가지 요소는 컴퓨터에서 본질적으로 모두 연결되어 있으며, 컴퓨터의 본질, 즉 컴퓨터의 시작은 자동으로 수학적 사실을 증명할 수 있는 기계에서 시작되었다는 것이다. 서로 매우 달라 보이는 것이 사실 하나라는 사실은 책을 읽고 난 뒤에도 매우 당황스럽게 다가오는데, 어떻게 이런 일이 가능할까?

컴퓨터의 시작은 이산 수학에서 배운 불완전성 원리와 직접적으로 연결된다. 당시 수학자들에게 수학적 증명은 증명된 수학적 사실에서 시작해 추론 규칙을 통해 완성되는 것이었다. 만약 증명이 단지 추론 규칙을 적용하는 일련의 과정이라면 추론 규칙을 모두 발견한다면 이를 통해 자동적으로 추론 규칙을 적용한다면 자동으로 수학적 증명을 할 수 있을 것이다. 그러나 괴델을 통해 자동적으로 모든 수학적 증명을 도출해 낼 수 없다, 즉 알려진 모든 추론 규칙을 적용해도 증명할 수 없는 수학적 명제가 존재한다는 사실이 증명되었다.

이러한 격변의 상황에서 앨런 튜링은 괴델의 불완전성 정리의 증명을 접했고, 자신만의 방법으로 불완전성 정리를 증명해보기로 했다. 바로 튜링이 사용한 '자신만의 방법'이 자동적인 방법(기계적인 방법)으로 수학적인 명제를 만드는 추상 기계였다. 추상적인 '컴퓨터'의 개념이 탄생한 것이다. 이때 정의한 추상 기계가 계산 이론에서 배운 튜링 머신이며, 테이프를 통해 내부 상태를 변화 시키며 읽고 쓰는 튜링 머신에서 상태 변화 연산이 정의되었다. 튜링은 이 과정에서 더 나아가서 튜링 머신을 확장시켜 임의의 튜링 머신이 하는

작업을 할 수 있는 '보편 만능의 튜링 머신'이 정의하였다. 이제 보편 만능의 튜링 머신은 모든 튜링 머신이 할 수 있는 작업을 할 수 있다.

드디어 튜링은 보편 만능의 튜링 머신을 이용해 증명할 수 없는 명제가 존재한다는 사실을 증명한다. 여기서 사용한 것이 이산 수학에서 다룬 정지 문제이다. 모든 튜링 머신과 가능한 입력은 셀 수 있다(countable). 만약 정지 문제가 증명 가능하다면 모든 튜링 머신과 모든 입력에 대해 정지 여부를 확인해주는 튜링 머신이 존재할 것이다. 튜링은 이 사실과 칸토어의 대각선 논증을 이용해 모든 튜링 머신에 대해 판정 가능하다고 가정한 정지 문제 튜링 머신이 판성할 수 없는 튜링 머신이 반드시 하나 존재한다는 것을 보였고, 이는 정지 문제가 증명 가능하다고 가정했기 때문에 발생한 문제이므로, 정지 문제는 증명 가능하지 않고, 따라서 증명할 수 없는 명제가 존재한다는 불확정성 원리를 증명하였다. 이렇게 튜링은 불확정성 원리를 튜링 머신으로 증명하면서, 튜링 머신의 상태 변화 연산과 수학적 추론 연산이 서로 연결되어 있음을 확인하였다.

튜링 머신을 통해 서로 연결된 상태 변화 연산과 수학적 추론 연산은 튜링 머신을 전자회로로 구현하면서 논리적 연산과도 연결된다. 새년의 논문으로 스위치 결합이 임의의 부울 논리식임이 증명되자 연구자들은 원하는 논리적 연산을 수행하는 스위치 회로를 디자인 할 수 있게 되었다. 이러한 진보는 기존의 데이터를 기억할 수 있는 메모리 회로를 발명했고, 메모리 회로가 발명되자 튜링 머신에서 상태를 기억하고, 저장된 규칙표를 확인하고, 테이프(기억장치)의 데이터를 수정하는 이론적 내용이 구체화가 가능해졌다. 이렇게 튜링 머신이 디지털 컴퓨터로 바뀌면서, 논리적 연산이 상태 변화 연산과 수학적 추론 연산에 끼어들어가게 되었다.

지금까지의 일련의 과정이 책에서 다룬 디지털 컴퓨터의 역사이다. 책은 컴퓨터의 역사를 통해 자동으로 증명할 수 있는 추상 기계가 디지털 컴퓨터로 구체화 되면서 내부 상태를 변화시키는 연산, 수학적으로 추론하는 연산, 그리고 AND, OR 같은 논리적 연산이 어떻게 하나로 합쳐질 수 있는지 직관적으로 보여주었다. 그리고 여기서 책은 하드웨어에서 넘어가 소프트웨어적 측면에서도 모든 연산의 요소들이 연결되어 있다는 통찰을 보여준다. 명령어 프로그래밍은 튜링 머신과 동등하다. 변수나, 메모리 값 같은 내부 '상태'를 바꾸는 연산(명령)을 실행하면서 목표하는 결과를 도출한다. 그러나 프로그램은 꼭 내부 상태를 바꿀 필요가 없다. 값과 값이 바뀌며 전달되는 함수처럼 수학적 추론 연산, 논리적 연산을 통해 내부 상태가 아닌 테이터를 바꾸는 이러한 과정을 람다 계산식이라 하며, 람다 계산식의 논리를 사용하는 프로그래밍을 함수형 프로그래밍이라고 한다. 놀랍게도, 명령어 프로그래밍과 함수형 프로그래밍은 서로 표현 능력이 완전히 같다. 프로그램을 보는 방식이 서로 다를 뿐이다. 하드웨어와 소프트웨어 모두에서 연산의 동등성. 이 메시지가 작가가 독자에게 전달하려는 가장 중요한 통찰이라고 생각한다.

2학년에 올라와 계산이론, 이산 수학 등 전공 강의를 들으면서 서로 조금씩 비슷한 내용을 다루고 있다는 느낌을 받았다. 서로 전혀 다른 내용을 다루는 것 같은데 각 과목들이 공유하는 공통점이 분명 존재하는 것 같았다. 그리고 이 책을 읽고 이 공통점은 생각보다 매우 크고, 서로가 서로와 깊게 얽혀 있다는 것을 알게 되었다. 그 전에는 이 과목들을 왜 공부해야 하는지 명확하게 알지 못했는데 책을 읽으면서 왜 이 과목들이 필요한지 확실히 느낄 수 있었다. 컴퓨터의 본질은 연산인데, 모든 종류의 연산이 서로 연결되어 있으므로 각 과목들을 이해해야 연산의 본질적 의미를 이해할 수 있기 때문이니까 말이다.