<컴퓨터과학이 여는 세계> 2차 과제 2011- 박소현

신과 인간, 그리고 컴퓨터

인문대학 불어불문학과
2011- 박소현
2012.05.08

1. 서론: 400년의 축적, 그 이후

강의의 PART I에서 살펴보았듯이, 컴퓨터는 수학자들의 오랜 노력과 꿈 위에서 탄생했다. 직접적인 산물이라기보다는 부산물에 가깝지만, 이러한 역사 없이는 컴퓨터도 탄생하지 못했을 것이다. 이 모든 역사의 산물인 컴퓨터는 오늘날에 이르러 모든 분야의 빠질 수 없는 도구가 되었다. 그렇다면 매우 단순한 조작을 통해 움직이는 보편 튜링 기계(Universal Turing Machine)가 어떻게 이렇게까지 빠르고 정확하게 여러 가지 일을 할 수 있게 된 것일까? 이 에세이를 통해서 컴퓨터를 구성하는 계층 구조, 그리고 컴퓨터를 조종하는 알고리즘과 그 복잡도를 살펴보고, 이것들이 오늘날 가지는 의미를 평가하고자 한다.

2. 본론

2.1. 컴퓨터의 본질은 논리

컴퓨터의 본질은 논리이다. 부울의 저서 <사고의 법칙에 관한 탐구 (An Investigation of the Laws of Thought)>에는 인간 논리를 수학 연산처럼 계산 가능하게 정리하려는 시도가 담겨 있다. 부울 대수에서는 정확히 정의된 명제라면 몇 가지 연산 기호(∧, ∨, ¬: AND, OR, NOT)를 통해서 명제의 참과 거짓을 판명할 수 있다. 부울이 정리한 인간 사고의 논리 연산은 스위치 회로로 모두 구현될 수 있다. 이 업적을 이룬 것이 바로 섀넌이다. 이러한 논리 연산의 회로화를 통해 인간 논리는 자동화·기계화되었고, 이것이 바로 컴퓨터를 작동하게 하는 회로의 가장 기본적인 단위가 되었다.

그렇다면 AND, OR, NOT의 연산을 어떻게 회로로 구현할까? 일단 참과 거짓이라는 두 가지 상태가 있다고 가정한다. 전기 회로에서는 참일 때는 스위치가 닫혀 전류가 흐르고, 거짓일 때는 스위치가 열려서 전류가 끊기게 한다. 이런 구조에서는 AND, OR, NOT을 각각 직렬, 병렬, 뒤집기 스위치로 표현할 수 있다. AND연산을 예로 들어 생각해보자. A∧B가 참이기 위해서는 A와 B 양쪽 다 참이어야만 한다. A스위치와 B스위치를 직렬로 연결하는 경우, A스위치와 B스위치 양쪽이 닫혀야만 A∧B에 전류가 흐른다. 그러므로 AND연산은 전기 회로에서는 직렬 연결이다.

이 때, 전기가 흐르지 않거나 흐르는 두 가지 상태를 0과 1이라 표현한다. 이 때문에 사람들은 컴퓨터의 본질은 숫자라고 착각하기도 한다. 그러나 이것은 편의상 임의로 정의한 것이며, 중요한 것은 두 가지

상태로 표현될 수 있다는 것이다. 이 두 가지 상태의 셋을 비트(bit)라고 부른다. 숫자가 아닌, 비트가 컴퓨터 논리의 기본 단위이다. 1비트로는 두 가지 상태를 표현할 수 있고, 2비트로는 네 가지 상태(00, 01, 10, 11)를, 그리고 n비트로는 2^n가지의 상태를 표현할 수 있다. 그래서 컴퓨터는 두 가지 상태만 필요한 것이다. 세 가지, 혹은 그보다 더 많은 상태를 표현하는 기계일지라도, 비트를 늘린다면 두 가지 상태로 모두 시뮬레이션이 가능하기 때문이다.

결국, 컴퓨터는 같은 논리 구조를 사용하기만 한다면 굳이 전기 회로와 스위치가 아닌, 수도관과 밸브나 나무막대기와 끈으로도 만들 수 있다. 다만, 복원 논리라는 것을 지킬 수 있는 매체가 가장 이상적이다. 컴퓨터는 그 기본 단위는 단순하지만, 총합해서 사용하는 기본 단위의 수는 천문학적이다. 이 때 전달되는 신호에 하나라도 왜곡이 있다면 컴퓨터가 제대로 작동할 수 없다. 그래서, 아무리 멀리 나아가더라도, 아무리 입력되는 신호가 작더라도, 출력되는 신호의 세기가 계속 복원·유지되는 매체가 필요하다. 전류와 수돗물은 그것이 가능하지만, 나무 막대기를 밀어서 작동하는 컴퓨터는 그렇지 않다. 물론, 수도관과 밸브로 컴퓨터를 만드는 것은 크기의 문제가 있다. 복원 논리를 지키면서도 가장 작고 빠르게 회로를 구현하는 기술이 반도체이기 때문에 지금 모든 컴퓨터가 반도체로 만들어진 것이다.

2.2. 기능적 추상화: 무한한 계층의 가능성

지금까지 살펴본 AND, OR, NOT은 기본적으로 논리적인 연산이기 때문에, 이를 추상화해서 생각하는 것이 가능하다. 그 연산들을 통해 0과 1로만 이루어진 논리 블록을 정의할 수 있다. 한 번 정의된 논리 블록은 더 이상 생각할 필요가 없기 때문에 편리하다. 힐리스의 비유처럼 레고 블럭 같은 것이다. NOR이라는 새로운 블록도 만들 수 있다. 메모리, 디코더, 멀티플렉서도 가능하다. 심지어는 AND조차 OR과 NOT을 적절히 조합하면 만들 수 있는 논리 블록이다. 어찌되었든 간에, AND, OR, NOT으로 다수결 결정을 하는 논리 블록, 가위바위보 승패를 판정하는 논리 블록 등을 만들 수 있다. 그리고 그 논리 블록들은 이제 사용자의 마음대로 이용할 수 있는 블록이 된다.

현실에서는 시간이라는 요소는 무시할 수 없는 부분이다. 이러한 블록 구조에서 시간의 흐름을 고려해서 만들어진 유한 상태 기계라는 블록이 있다. 유한 상태 기계는 이전에 실행된 조작을 기억하는 메모리가 필요하고, 그 기계의 상태는 지금까지의 조작의 결과이다.

유한 상태 기계를 구성하는 데에 매우 중요한 레지스터라는 블록이 존재한다. 레지스터는, 앞서 살펴본 논리 연산으로 이루어진 부울 논리 블록의 출력을 시간에 따라 기록(write)하고 상태를 바꾸는 일을 한다. 간단히 말하자면, 유한 상태 기계는 부울 논리 블록과 레지스터를 연결한 것이다. 이 레지스터에서 상태가 바뀌는 속도가 컴퓨터의 속도를 기본적으로 결정한다고 볼 수 있다.

여담이지만 힐리스가 제시한 소총 부대 문제를 조금 생각해 보게 되었다. 해답은 생각보다 간단한 곳에 있었다. 해법에는 강의시간에 다룬 정렬 문제가 큰 힌트가 되었다. 각 병사를 기본 단위로 생각하면, 한 열을 반으로 계속 쪼개다가 쪼개지는 조각이 $n \le 1$이 되는 순간 발포하면 되는 것에 착안하여, 1:3의 속도로 움직이는 두 가지 신호를 전달한 뒤, 양 끝의 병사들은 신호를 되돌려 보내는 기능을 주었다. 신호가 서로 부딪힐 때마다 양쪽으로 똑같이 1:3의 속도로 움직이는 두 가지 신호를 보낸다면 각 조각은 계속 반으로 쪼개지고, 1이 되는 순간 다같이 발포하면 된다.

2.3. 거북이를 노래하게 하는 것은?

3장에서 힐리스는 프로그래밍 언어를 소개한다. 프로그래밍 언어는 인간으로 하여금 원하는 바를 컴퓨터에게 지시할 수 있도록 만들어진 언어이다. 그가 우리에게 간단한 예로 제시한 것이 로고라는 프로그래밍 언어이다. 로고는 어린이들에게 프로그래밍을 교육시키기 위해 만들어진, 다루기 쉬운 언어이다. 그러나 알뜰하게도 빠진 것 없이 다 들어가 있다고 한다. 백문이 불여일견, 로고를 프로그래밍하는 동시에 결과를 보여주는 응용 프로그램인 ACSLogo를 다운받아서 로고를 직접 다루어보았다. 과연 기본적인 지시사항(forward, right, left)는 충분히 쉽게 익힐 수 있었다. 반복되는 명령어를 REPEAT함수로 간단하게 쓰는 것도, 새로운 함수를 정의하는 것도 어느 정도의 시행착오 후에 사용할 수 있게 되었다. 거북이가 다니는 길의 너비와 색도 쉽게 바꿀 수 있었다.

그러나 임의의 변수를 정해서 그 변수를 자유자재로 다루는 부분은 조금 어려웠다. 토씨라도 하나 틀리게 입력하거나 파라미터를 제대로 정의해주지 않으면 거북이가 말을 듣지 않거나 길을 잃기 일쑤였기 때문이다. 컴퓨터에게 무엇이든 간에 지시하기 위해서는 매우 엄밀하고 정확하게 원하는 것을 말해야 된다는 것을 직접 경험해본 것이다. 대신, 제대로 정의된 변수의 활용은 모든 명령을 하나씩 입력해야 하는 수고를 덜어 주는, 마치 마법과도 같은 효과가 있었다. 한 번만 수고를 해서 여러 번의 수고를 덜 수 있었다.
컴퓨터 속 거북이를 움직일 수 있는 언어는 하위 레벨부터 더 높은 레벨까지 다양한 계층으로 존재한다. 매우 기초적인 부분까지 조작할 수 있는 C언어는 아주 작은 부분까지 프로그래머가 조절할 수 있지만, 모든 부분에 대해 책임을 져야 하기 때문에 실수가 잦을 수 있다. 좀 더 고차원적인—상위 레벨의—언어들도 존재하고, C언어보다는 접근성이 높다고 말할 수 있다.

무엇보다 중요한 것은, 프로그래밍 언어를 통해서 단순히 거북이를 움직이게만 할 수 있는 것이 아니다. 거북이가 네모를 그리게 하는 것은 누구나 할 수 있는 것이다. 그러나 같은 언어로, 좋은 프로그래머는 거북이를 춤추게 할 수도 있고, 노래하게 할 수도 있다. 자연어에서도 마찬가지이다. 이 세상에서 영어를 모국어로 갖는 사람만 해도 360,000,000명이다. 또한, 르네상스 영시의 작법은 매우 엄격해서, 5음절으로 이루어진 행들이 'abab-cdcd-efef-gg'의 운율을 가진 것만 소네트로 인정되었다. 그러나 영어를 안다고 해서, 그리고 소네트의 규칙—매우 단순하다—을 안다고 해서 누구나 워즈워스나 셰익스피어가 되는 것은 아니다. 음악의 언어 역시 그렇다. 16세기에 정립된 대위법(counterpoint)는 당시의 음악을 지배하던 매우 엄격한 작법으로, 규칙 체계를 익히는 데는 두어 달이면 족하다. 그러나 그 규칙을 안다고 해서 모두 바흐나 비발디가 되는가? 오히려 규칙을 알면 알수록 거장의 위대함에 저절로 고개가 숙여질 뿐이다. 규칙이 정해 주는 틀 안에서 날아오르는 천재들이 있기 마련이고, 컴퓨터 프로그램도 그렇다.

기능적 추상화라는 계층 구조는 이러한 프로그래밍 언어로부터 시작해서 실제로 전기 회로의 스위치를 움직이는 것까지 연결한다. 그래서 컴퓨터 동작의 핵심이라는 것이다. 상위 계층부터 가장 하위 계층까지 일사불란하게 통제할 수 있게 해 주는 것이 기능적 추상화이다.

2.4. 컴퓨터와 인간: 보편만능의 기계를 넘어서

지금까지의 컴퓨터는 모두 한 가지로 축약 가능하다. 튜링이 발명한 보편 튜링 기계가 모든 컴퓨터의 기본적 구성이다. 그리고 위에서 말했듯이, 0과 1의 두 가지 상태만 요구한다. 다른, 더 좋은 방법의 컴퓨터는 존재하지 않는 것인가? 힐리스는 그 가능성을 양자컴퓨팅에서 찾고 있다. 컴퓨터도 이 세상의 기계이

기에 이 세상의 물리적 법칙을 따르지 않을 수 없다. 그리고 거시 세계에서의 물리량은 아날로그적이다. 그러나 미시 세계로 가면 얘기가 달라진다.

임의의 소립자는 편재한다. 우리가 어떠한 물체를 관찰해서 그 위치를 안다고 말하는 것은, 양자 세계에서는 단지 그곳에 존재할 확률이 높은 것일 뿐, 그곳뿐만 아니라 다른 곳에도 동시에 존재하고 있다. 그리고 우리가 하나의 입자를 관찰한다는 것은, 그곳에 있지 않은 다른 입자에게도 동시에 영향을 끼친다. 사실은, 그곳에 둘 다 존재할 확률이 있다. 듣기만 해도 머리가 핑핑 돌게 되는 이야기이지만, 양자를 통해서 '디지털'의 본질에 가까워질 수 있을 것이라고 사람들은 생각한다.

다행인지 불행인지는 모르겠지만, 인간 뇌를 이루고 있는 뉴런의 작동이 양자의 법칙을 따른다는 증거는 없다. 인간의 뇌에서 일어나는 모든 사고 과정을 현재 우리가 갖고 있는 보편 튜링 기계로 모두 시뮬레이션이 가능할 수도 있다는 얘기이다. 몇몇 사람들은 그런 추측에 대해 불편해하거나 심지어는 화낼 수도 있을 것이다. "아니, 그럼 인간이 컴퓨터랑 똑같단 말이야!"하고 말이다. 그러나 그런 식의 반응은 신대륙이 발견되었을 때에도, 다윈이 진화론을 발표했을 때에도 있었다. 컴퓨터가 인간 뇌를 똑같이 따라할 수 있다고 해서 인간 고유의 가치가 사라지는 것은 아닐 것이다.

우리가 살고 있는 세상은, 이 자연은, 이 우주는 하나이다. 우리의 우주를 지배하는 법칙이라면 우리도 그 법칙 아래에 있고, 그 법칙을 안다면 시뮬레이션은 더욱더 가능할 것이다. 이 때문에 모든 우주를 당장이라도 수학적, 물리적 법칙으로 환원시킬 기세의 사람들도 존재한다. 그러나 인간은 그러한 보편만능의 법칙을 찾아내기까지는 아직 멀었다. 양자역학의 발전과 함께 화학은 물리학의 영역으로 완전히 편입되었다고들 말하지만, 사실 양자역학으로 정확히 설명할 수 있는 화학 결합은 H-H정도밖에 없고, 나머지는 아직 연구 중이라고 한다. 더욱이 중요한 것은 '창발성'이라는 개념이다. 말하자면 수소 원자 두 개와 산소 원자 하나의 성질을 모두 규명했다고 해서 물의 성질을 모두 예측할 수는 없다는 뜻이다. 인간을 구성하는 모든 분자가 주어진다고 해서 세포 하나 창조할 수 없다. 그래서 환원주의를 결코 부정하고 싶지도 않지만, 당장 모든 것을 치환시킬 수 있다는 관점도 아직은 이르다고 믿는다.

2.5. 유레카! 알고리즘과 휴리스틱

알고리즘의 복잡도는 현실적인 문제이다. 세상에는 다항식 시간(polynomial time) 내에 풀 수 있는 문제들과, 그럴 수 없는 미정다항시간문제(Nondeterministic Polynomial time problem, NP)가 있는데, 이 NP를 P시간으로 풀기 위한 노력이 끊임없이 있어 왔다. NP 중에서 여러 가지 문제는 우리에게 매우 유용하기 때문에 컴퓨터 알고리즘을 연구하는 사람들은 P=NP인지 아닌지 밝히려고 노력하고 있다. 실제로 P=NP 문제는 클레이 연구소에서 제시한 밀레니엄 난제 7문제 가운데에 있고, 이것을 푸는 사람에게는 100만 달러의 상금과 함께 엄청난 영광이 돌아간다. 컴퓨터 과학자들은 아마도 P와 NP는 등치가 아닐 것이라고 예상하고 있다.

그래도 우리 생활에 제기되는, 얼핏 간단해 보이는 많은 문제들이 NP문제들이다. 그 문제들을 P시간 안에 풀 수 없으니, 휴리스틱이라는 새로운 방법으로 해결하려는 시도가 일어나고 있다. 휴리스틱(heuristics)의 어원을 분석하면, '찾다' 혹은 '발견하다'는 뜻의 어근이 있다. 아르키메데스가 외쳤다는 유명한 "유레카!"와 같은 어원이다. 이렇듯 휴리스틱은 어느 정도 운과 우연에 의지하는 프로그램이다. 계산 가능한 부분에서 가장 정답일 확률이 높은 선택을 빨리 계산해 내는 방법이다. 체스 게임을 하는 컴퓨터 프로그램이

그 예이다. 힐리스가 1장에서 제시한 틱택토 게임과는 달리, 체스 게임의 모든 수를 짧은 시간 내에 내다볼 수 있는 컴퓨터는 아직 없다. 그래서 게임을 유리하게 만들 수 있는 수를 확률적으로 계산해서 두는 것이다. 휴리스틱은 그래서 간혹 틀릴 수도 있다. 그러나 여러 가지 요소를 고려한 점점 더 정교한 휴리스틱으로 어느 정도까지는 정확성을 끌어올릴 수 있다. 그 증거로 97년도에는, 체스의 세계 챔피언 카스파로프가 슈퍼컴퓨터 딥 블루에게 패배하는 대사건이 발생했다. 휴리스틱은 단지 의미 없는 찍기가 아니라, 실제로 문제 해결에 적용될 수 있는 해법이다.

3. 결론: 신, 인간, 컴퓨터

지금까지 컴퓨터라는 기계를 어떻게 우리가 원하는 마법의 기계가 되게 하는지, 그 방법을 살펴보았다. 작은 블록을 반복하고 조합해서 쌓고 쌓은 거대한 피라미드와 같은 계층 구조, 그리고 그 구조를 움직이게 하는 마법의 주문인 알고리즘. 게다가 너무나도 인간적인 휴리스틱이라는 문제 해결 방식. 이 모든 것을 알고 나니 컴퓨터가 단지 차갑고 기계적인 사물로만 느껴지지가 않는다. 제우스의 머리를 쪼개고 나온 아테나처럼, 인간의 머리에서 튀어나온 컴퓨터인 것을 느낀다.

어느 나라의 신이든 간에, 신은 자신의 형상으로 인간을 창조한다. 그러나 신 역시 인간이 창조한 것이 아닌가? 컴퓨터도 인간이 자신의 형상을 창조하고 싶은 욕구로부터 발생한 것이 아닐까? 그러나 자신들이 만들어낸 신을 두려워하는 인간처럼, 컴퓨터 역시 두려워하고 있다. 아니, 이제는 오히려 막연한 신보다 실체로 성큼 다가선 컴퓨터를 더욱 두려워하는 듯하다. 사람들은 <매트릭스>와 같은 공상 과학 영화들에서처럼 로봇이 인간들보다 더 똑똑해져, 인간들을 지배하려 하지 않을까 하는 막연한 공포심에 사로잡혀 있는 듯하다. 그래서 인간의 지능과 컴퓨터의 인공지능을 비교하려는 시도에 불쾌감과 거부감부터 드러낸다. 끊임없이 컴퓨터를 자신과 닮게 하려는 노력을 하면서도, 컴퓨터와는 절대적인 차별성을 두려 한다.

이 두려움과 이중성은 언제쯤 극복할 수 있을까? 인간은 이러한 것에 대해서 어느 정도의 본능적인 거부감이 있는 것 같다. 코페르니쿠스와 갈릴레오, 케플러 등이 지동설을 주장했을 때에도, 신대륙과 새로운 인간을 발견했을 때에도, 그리고 다윈의 진화론이 제시되었을 때에도, 사람들은 흔들리고 두려워했다. <종의 기원>이 출판된 지 150년이 지난 지금에도, 아직 진화론은 거짓말이라고 부인하며 살아가는 사람들이 있을 정도니까.

그렇지만 인류의 대부분은 그런 두려움을 극복하고 새로운 호기심을 통해서, 지식과 창조로 나아가기를 원하는 사람들이다. 최소한 나는 그렇게 믿는다. 컴퓨터가 지능을 얻는다고 해서, 과거와 다르게 갑자기 인간의 본질이 평범하고 진부한 것이 되리라고는 생각하지 않는다. 오히려 인간을 닮은 컴퓨터를 통해, 우리는 인간의 본질을 비추는 하나의 거울을 얻은 것이 아닐까?

참고문헌

대니얼 힐리스,『 생각하는 기계 』. 노태복 역. (사이언스북스, 2006)