각자 얻은 성과는 정말로 우연히 합쳐지는가?

2017-17088

박찬정

요약

알게 된 것

튜링 기계는 사실 괴델의 불완전성 정리를 튜링 자신의 방식대로 재확인하는 과정에서 고안되었다. 모든 참인 명제를 차례로 만들어내는 튜링 기계가 존재한다면 멈춤 문제를 푸는 기계도 만들 수 있다. 하지만 멈춤 문제를 푸는 기계가 존재한다면 튜링 기계의 수는 자연수의 개수보다 많아져야 하고, 이는 모순이므로 멈춤 문제를 푸는 기계는 존재하지 않는다.

논리학 분야에서도 수학의 논리 전개에 대한 시도는 많이 있어 왔다. 이러한 체계를 오래 전부터 modus ponens라 불러왔다. 이전부터 논리 전개의 과정을 수식화하고자 하는 시도는 많이 있어왔지만, Frege의 표기법이 이에 대한 열쇠가 되어주었다. 결과적으로 말로 할 때 같아 보이던 if-then 절은 세 가지 방식으로 나타나게 된다.

Gentzen의 natural deduction과는 별개로 등장한 Church의 lambda calculus는 natural deduction을 기계적으로 단순화시키는 단서가 된다. 보통 수학에서 함수를 equation으로 정의하는 것과 달리, Church는 lambda calculus에서 모든 계산을 함수의 실행으로 생각한다. Lambda calculus에 대한 관심이 높아지며 계산만이 아니라 각종 수학적 개념들마저 lambda calculus의 영역으로 끌어들이게 된다. Church는 기존의 natural deduction에 type 개념을 추가하고, 이로써 function application 과정을 Gentzen의 natural deduction을 활용하여 표현할 수 있게 된다. 결국 lambda calculus의 term은 논리학에서의 proof와 일대일 대응 관계이다. 그리고 Church는 lambda term이 기계로 계산 가능한 모든 함수를 표현함을 알게 된다.

느낀 것

고등학생 때, 친구들 앞에서 문제 풀이를 발표를 할 때 일부러 전혀 동떨어진 논리에서부터 풀이를 시작한 적이 있다. 이는 당연히 문제 전반을 잘 이해하고 있었기 때문에 가능한 일이었다. 비

슷하게, 지금은 튜링과 많은 사람들의 연구 성과가 잘 정리되어 있다. 하지만 그 연구자들은 복잡한 시행착오와 군더더기 속에서 오랫동안 헤엄치며 성과를 얻은 것이다.

컴퓨터의 역사를 설명하는 글이나 매체를 접하게 되면 보통 물리적인 방식으로 구현된 계산기에서 이야기를 시작하는 경우가 많다. 그러면서 튜링 기계와 같은 컴퓨터의 이론적인 역사는 적은 비중을 차지하게 된다. 그래서 이번 기회에 컴퓨터의 이론적인 역사를 새로이 알게 되면서 컴퓨터에 대한 시각이 달라졌다.

힐베르트가 수학이 완전성과 무모순성을 보이고자 했다는 것과, 그것이 괴델에 의해 좌절되었다는 것은 수학의 역사에 대한 배경지식으로 이미 알고 있었다. 반면 튜링에 대해서는 단편적으로만 알고 있었다. 그래서 튜링 기계가 괴델의 불완전성 정리를 재확인하는 과정에서 고안된 것임을 알았을 때 둘 사이의 관계성을 쉽게 떠올릴 수 없었다. 튜링 기계의 개념을 이해할 땐 현대 컴퓨터의 구조를 떠올리니 쉬웠다. 하지만 논리학자들의 성과를 컴퓨터와 관련하여 이해하는 것은 어려웠다. 그래도 지금까지 수업에서 들은 내용들과 과제로 경험한 것들을 바탕으로 둘 사이의 연관성이 어느 정도는 가늠이 된다.

질문하고 싶은 것

수학에 대한 근본적인 물음에서 또 다른 획기적인 결과물이 나올 수 있을까? 여러 연구자들 각자의 성과가 나중에 합쳐지는 것을 보며 놀라워하지만, 그 부분에 대한 이해가 높아지고 나면 '그럴 만했다'는 생각이 들지는 않을까?

본문

알게 된 것

튜링 기계는 사실 괴델의 불완전성 정리를 튜링 자신의 방식대로 재확인하는 과정에서 고안되었다. 튜링 기계는 유한한 개수의 심볼과 무한한 길이의 테잎, 테잎에 쓰인 심볼을 읽거나 기록하는 장치, 장치의 작동규칙표로 이루어진다. 임의의 튜링 기계는 일렬의 테잎 위에 표현될 수 있고, 결국 하나의 튜링 기계만 있어도 무한한 가지수의 튜링 기계를 흉내내도록 만들 수 있다. 이렇게 테잎 위에 표현된 튜링 기계는 고유의 자연수로 변환할 수 있으므로, 튜링 기계의 수는 자연수의 개수를 넘지 못한다.

모든 참인 명제를 차례로 만들어내는 튜링 기계가 존재한다면 멈춤 문제를 푸는 기계도 만들 수 있다. 모든 참인 명제를 만들어내는 과정에서, 입력으로 들어온 프로그램이 멈추는지에 대한 명제도 만들어낼 것이기 때문이다.

하지만 멈춤 문제를 푸는 기계가 존재한다면 튜링 기계의 수는 자연수의 개수보다 많아져야 하고, 이는 모순이므로 멈춤 문제를 푸는 기계는 존재하지 않는다.

논리학 분야에서도 수학의 논리 전개에 대한 시도는 많이 있어 왔다. 이러한 체계를 오래 전부터 modus ponens라 불러왔다.

이전부터 논리 전개의 과정을 수식화하고자 하는 시도는 많이 있어왔지만, Frege의 표기법이 이에 대한 열쇠가 되어주었다. Frege는 assertion에 대한 judgement를 표현하기 위하여 ⊦ 표기법을 고안했다. 이후 가로선을 통해 premise와 conclusion을 표현하는 표기법도 고안되었다. 이후 Gentzen은 Frege의 표기법에서 가정에 대한 부분을 추가하였다. 또한 premise에서 conclusion으로 넘어가며 사용한 규칙에 대해서도 표기할 수 있도록 하였다.

결과적으로 말로 할 때 같아 보이던 if-then 절은 세 가지 방식으로 나타나게 된다. Proposition은 화살표, judgements는 ⊦, premises and conclusion은 inference rule로 표기된다.

Gentzen의 natural deduction과는 별개로 등장한 Church의 lambda calculus는 natural deduction을 기계적으로 단순화시키는 단서가 된다. Lambda calculus는 논리를 수식화하는 새로운 방식을 고안하며 등장하였다. 비록 첫 등장에서는 결함이 있었지만, 이후 발전하여 논리학계에 큰 영향을 끼치게 된다.

보통 수학에서 함수를 equation으로 정의하는 것과 달리, Church는 lambda calculus에서 모든 계산을 함수의 실행으로 생각한다. 덧셈이나 곱셈과 같은 사칙연산의 과정도 lambda calculus에서는 function이 apply되는 과정으로 표현하게 된다. 이 때 reduction의 순서와 관계없이 동일한 결과가 나온다는 것을 증명했고, 이를 Church-Rosser 정리라고 부른다.

Lambda calculus에 대한 관심이 높아지며 계산만이 아니라 각종 수학적 개념들마저 lambda calculus의 영역으로 끌어들이게 된다. 숫자도 function으로 정의할 수 있는데, 해당 숫자만큼 인자로 들어온 함수를 연속하여 수행하는 함수로 정의할 수 있다. 이에 따라 사칙연산도 정의할 수 있게 된다. 또한 자료구조와 관련된 포용성을 갖고자 pair도 정의하여 사용하게 된다. 인자가 여럿인 함수를 구현할 때는 인자를 받아 함수를 반환하는 함수를 중첩하는 방식인 currying을 사용하도록 제안되기도 한다.

Church는 기존의 natural deduction에 type 개념을 추가하고, 이로써 function application 과정을 Gentzen의 natural deduction을 활용하여 표현할 수 있게 된다. 그리고 function application 과정

은 derivation tree를 작게 축소시키는 역할을 한다는 것도 알게 된다. 이 때 derivation tree를 단순화하는 과정은 결국 끝나게 된다.

결국 lambda calculus의 term은 논리학에서의 proof와 일대일 대응 관계이다. 이제 Gentzen의 natural deduction은 lamda calculus를 통해 단순화할 수 있게 된다. 각각의 발견은 비슷한 시기에 일어났지만, 둘 사이의 연관성을 깨닫고 적극적으로 사용하게 되기까지는 꽤 오랜 시간이 걸렸다.

그리고 Church는 lambda term이 기계로 계산 가능한 모든 함수를 표현함을 알게 된다. 이로써 비슷한 시기에 논문을 썼던 튜링과의 연결고리 또한 생기게 되며, 둘은 함께 연구를 진행하게 된다.

느낀 것

고등학생 때, 친구들 앞에서 문제 풀이를 발표를 할 때 일부러 전혀 동떨어진 논리에서부터 풀이를 시작한 적이 있다. 문제와는 관련 없는 것처럼 보이다가 갑작스레 문제를 푸는 실마리가 등장하더니 번뜩 답을 내려버리는 듯 연출한 것이다. 듣는 사람들 입장에서는 문제에 대한 해답이 예상치도 못한 곳에서 내려쳐 등장한 것처럼 보였을 것이다.

이는 당연히 문제 전반을 잘 이해하고 있었기 때문에 가능한 일이었다. 그 문제에 대한 모든 것을 잘 알고 있었기 때문에 어디서부터 시작해야 가장 극적일지, 또는 이해가 잘 될 지를 파악하고 나름대로 구성할 수 있었던 것이다.

비슷하게, 지금은 튜링과 많은 사람들의 연구 성과가 잘 정리되어 있다. 잘 정리되어 있다는 말은 처음 그것들을 생각해낼 때의 시행착오와 군더더기를 제거하고 중심이 되는 핵심 내용만 따라갈 수 있게 되었다는 뜻이다.

하지만 그 연구자들은 복잡한 시행착오와 군더더기 속에서 오랫동안 헤엄치며 성과를 얻은 것이다. 처음부터 깔끔한 논리로 핵심을 향해 달려간 것은 아닐 것이다. 자신이 알고 있는 것의 범위를 부단히 넓혀가다 새로운 성과를 얻고 나면 그것에 이르는 길을 다시 정리했을 것이다. 하늘에서 뚝 떨어지는 지식은 없고, 모두 기존의 지식에 대해 자신의 방식대로 이해하고 한 번 더 질문하는 과정에서 확장되는 것이라는 생각이 든다.

컴퓨터의 역사를 설명하는 글이나 매체를 접하게 되면 보통 물리적인 방식으로 구현된 계산기에서 이야기를 시작하는 경우가 많다. 물론 컴퓨터의 역사에서 초창기의 계산기 또한 무시할 수 없는 부분이긴 하나, 이는 실제로 존재했던 대상을 바탕으로 설명하는 것이 일반 대중이 받아들이

기 쉽다는 점을 고려했을 것이다.

그러면서 튜링 기계와 같은 컴퓨터의 이론적인 역사는 적은 비중을 차지하게 된다. 그나마 튜링 기계는 그림 자료로 보여줄 수라도 있기에 언급은 되고 넘어가지만, lambda calculus나 natural deduction에 대한 내용은 교양서에 함부로 싣기에는 큰 부담이 될 것이다. 그렇기에 컴퓨터와 관련된 전공을 한다는 나이지만 발전의 역사에 있어 이론적인 부분은 거의 알지 못했다.

그래서 이번 기회에 컴퓨터의 이론적인 역사를 새로이 알게 되면서 컴퓨터에 대한 시각이 달라졌다. 지금까지는 컴퓨터를 실용적인 관점에서만 바라보고 있었다. 컴퓨터가 이 세상에 끼친 영향을 바라보며 하나의 유용한 도구로서만 인식하고 있었다. 하지만 이제는 수학과 논리학의 거대한 도전을 좌절시키며 등장한 커다란 개념으로서도 볼 수 있게 되었다.

힐베르트가 수학이 완전성과 무모순성을 보이고자 했다는 것과, 그것이 괴델에 의해 좌절되었다는 것은 수학의 역사에 대한 배경지식으로 이미 알고 있었다. 하지만 그 이후의 이야기는 알지 못했다. 나는 백여 년 전의 힐베르트의 실패를 적당한 마음으로 안타깝다 여기고 그것을 무너뜨린 괴델의 대단함을 감히 상상할 수 없어 우러러볼 뿐이었다. 실패담의 뒷이야기에는 관심을 두지 않았다.

반면 튜링에 대해서는 단편적으로만 알고 있었다. 튜링이라는 이름을 들으면 일단 천재라고 불린다는 것이 먼저 떠올랐고, 다음으로 그의 이름을 딴 여러 용어들, 예를 들면 튜링 머신이나 튜링 테스트 등이 떠올랐다. 그리고 한 켠으로 유명하다지만 아직 보지는 않은 튜링을 다룬 영화의 제목이 떠올랐다. 그나마 튜링 머신에 대해서는 컴퓨터의 개념을 처음 제시했다는 의의 정도만 알고 있었다. 컴퓨터의 역사와 관련된 중요한 소재이기에 예의상 알고 있었다는 정도의 느낌이다.

그래서 튜링 기계가 괴델의 불완전성 정리를 재확인하는 과정에서 고안된 것임을 알았을 때 둘 사이의 관계성을 쉽게 떠올릴 수 없었다. 실패로 끝맺은 이후 역사의 흐름 속에 묻힌 줄만 알았던 이야기에서, 어떻게 어디선가 뚝 떨어진 천재로만 알고 있었던 사람에 대한 이야기로 넘어갈 수 있으며, 그것이 지금도 사용하고 있는 컴퓨터에 대한 이야기로 이어질 수 있는지 상상하기 힘들었다.

튜링 기계의 개념을 이해할 땐 현대 컴퓨터의 구조를 떠올리니 쉬웠다. 실제 예제도 직접 따라해보며 튜링 기계가 작동하는 방식을 잘 이해할 수 있었다. 튜링 기계 자체를 테이프 속에 표현하고 이를 사용하여 튜링 기계를 모방하는 만능 기계를 생각해내는 것도 Virtual Machine을 상상하니 편했다.

하지만 논리학자들의 성과를 컴퓨터와 관련하여 이해하는 것은 어려웠다. 지금껏 컴퓨터를 도구로서 이해해왔던 영향인지 증명과 컴퓨터의 동작 사이의 관계를 이해하기는 힘들었다. 여러 수식들을 보며 lambda calculus가 증명의 과정을 단순화시킨다는 것을 시각적으로는 어느 정도 이해할 수 있었으나 더욱 근본적인 부분에서의 이해는 힘들었다.

그래도 지금까지 수업에서 들은 내용들과 과제로 경험한 것들을 바탕으로 둘 사이의 연관성이 어느 정도는 가늠이 된다.

질문하고 싶은 것

수학에 대한 근본적인 물음에서 또 다른 획기적인 결과물이 나올 수 있을까? 힐베르트의 수학의 완전성에 대한 질문과 같이 수학에 대한 깊이 있는 논의는 무궁무진하게 이뤄질 수 있을 것이다. 그렇다면 그 과정에서 컴퓨터의 파급력에 필적하는, 혹은 더 큰 영향력을 갖는 무언가가 탄생할 수도 있지 않을까? 세상에 새로운 혁신을 가져올 수 있지 않을까?

여러 연구자들 각자의 성과가 나중에 합쳐지는 것을 보며 놀라워하지만, 그 부분에 대한 이해가 높아지고 나면 '그럴 만했다'는 생각이 들지는 않을까? 사실은 각자의 관심 분야가 필연적으로 이어질 수밖에 없는 것이었음을 알게 되지는 않을까? 지금은 우연으로 보이는 것들이 사실은 필연은 아니었을까?