계산 가능성 이론 : 튜링 기계와 람다 대수

이상학

개요

계산 가능성 이론은 어떤 함수가 효과적으로 계산 가능한지에 대해 연구하는 수리논리와 계산 이론의 한 분야입니다. 이는 1930년, 1차 논리의 증명 가능성을 결정하는 문제를 해결하기 위해 만들어진 두 계산 모형, 람다 대수와 튜링 머신으로부터 시작되었습니다. 이 세미나는 이 두 체계의 정의와 성질을 자세히 설명하고 얼마나 많은 것을 표현할 수 있는지 소개하기 위한 것입니다. 튜링 기계는 이론 컴퓨터의 원형으로, 상당히 많은 것을 계산할 수 있으며 어떤 합리적인 계산 모형도 이보다 강력하지 않지만(처치-튜링 논제), 정지 문제와 같이 계산적으로 해결 불가능한 문제도 존재합니다. 바쁜 비버 함수는 계산 불가능 함수의 대표적인 예시로 튜링 기계가 얼마나 많은 것을 계산할 수 있는지에 대한 지표입니다. 한편 람다 대수는 논리에서 출발한 계산 모형으로, 모든 것을 함수로 간주하는 단순한 직관에서 시작하지만 놀라울 정도의 표현력을 갖고 있습니다. 고정점 연산자를 사용하면 모든 재귀 함수를 구현할 수 있으며, 람다 대수는 사실상 튜링 기계와 동치가 됩니다.

0 도입

기원

20세기 초, 수학의 기초를 찾기 위한 노력의 최전선에는 힐베르트 프로그램이 있었다. 1928년 거기서 나온 한 가지 질문, 결정 문제를 해결하기 위한 노력은 계산 가능성 이론으로 이어졌다. 어떤 1차 논리의 문장이 타당함을 기계적으로 판정할 수 있는가?

'계산'의 형식화

문제는 '기계적으로 판정'이 엄밀히 무슨 의미냐는 것이다. 이는 궁극적으로 어떤 자연수 위의 함수가 '효과적으로 계산 가능한지' 여부로 환원된다. 문장이

든 뭐든 유한적인 객체는 대부분 자연수로 인코딩할 수 있기 때문이다. 그러한 계산 가능성의 정의, 즉 계산 모형은 여러가지가 있다. 대표적인 예시가 바로 튜링의 튜링 기계와 처치의 람다 대수이다. 1936년 둘은 그러한 체계를 통해 독립적으로 결정 문제의 해결 불가능성을 증명하였다.

1 튜링 기계

정의

튜링 기계는 무한히 긴 테이프와 그 테이프 위를 움직이는 헤드로 구성된다. 헤드는 유한 개의 상태를 가질 수 있고, 테이프의 각 칸에는 유한 개의 기호가 쓰일 수 있다. 기계의 '프로그램'은 시작 상태와 뭘 읽었을 때 어떻게 움직일지 알려주는 명령의 집합으로 구성된다. 기계는 시작 상태로부터 명령에 따라 한 단계씩 진행하며 작동하며, 명령에 없는 상태/기호를 받으면 정지한다.

표준 세팅은 상태가 알파벳, 기호가 $\{0, 1\}$, 모두 0이 쓰인 테이프 0번째 칸에서 시작하는 것이다. 예를 들어 명령이 $A0 \to 1RB$와 $B0 \to 0RA$인 기계는 모든 음이 아닌 짝수에 1을 쓰는 기계이다. 그 외에도 12-상태 제곱수 기계와 25-상태 피보나치 수 기계도 있다.

자연수 함수의 계산 가능성을 정의하기 위해 자연수를 1의 위치로 표현한다. 따라서 n에 쓰인 1로 시작하여 $f(n)$에 쓰인 1로 멈추는 기계가 존재할 때 함수 f를 튜링 계산 가능하다 한다. 여러 개의 1로 입력을 주면 다변수 함수의 계산 가능성도 정의할 수 있다.

계산의 예시

다음은 $f(n) = 2n$에 대한 기계이다.

$$A0 \to 1RB \ / \ B0 \to 0RB \ / \ B1 \to 1RC$$
$$C0 \to 1LD \ / \ C1 \to 1RC \ / \ D0 \to 1LE$$
$$D1 \to 1LD \ / \ E0 \to 0RC \ / \ E1 \to 0RF$$
$$F0 \to 1LH \ / \ F1 \to 0RF$$

우선 A에서 시작하여 0번째 칸에 마킹하고 B로 입력인 n번째 칸에 있는 1로 이동한다. 그런 다음 오른쪽 C로 추가, 왼쪽 D로 움직이고 E로 판정하며 0번째 칸의 1을 찾는다. 찾았다면 F로 지금까지 1을 제거하고 $2n$에 답을 쓴다. 따라서 f는 계산 가능하다.

이제 튜링 계산 가능한 함수의 집합을 C_T라고 하자. C_T는 얼마나 클까? 일단 분명한 점은 C_T는 가산이므로, 여기 속하지 않는 함수도 존재한다. 정지 문제 등이 있다. 그러나 작지도 않은 게, 튜링 기계는 합성함수나 재귀도 충분히 돌릴 수 있다.

한편, 다른 계산 모형 M에서도 계산 가능성을 정의하여 C_M을 만들 수 있을 것이다. C_M과 C_T 중 어느 것이 더 클까? 즉, 튜링 기계와 M 중 무엇이 더 강력할까? 놀라운 대답은, 현재까지 개발된 모든 계산 모형은 튜링 기계보다 강하지 않다. 실제로 어느 정도 강력하면 튜링 기계와 동치가 된다 - 이를 튜링 완전이라 한다. 즉, 효과적으로 계산 가능한 절차를 갖는 모든 함수는 튜링 기계로 계산할 수 있음이 강하게 추측된다. 이를 처치-튜링 논제라 한다.

웬만한 함수는 튜링 머신으로 계산할 수 있다. 명령을 명시적으로 쓰기가 귀찮을 뿐이다. 팩토리얼 같은 것도 n의 1을 $-n$에 복제한 다음 곱 인자를 하나씩 옮겨가며 1에 곱하면 된다. 그런 튜링 기계를 만들 수 있음을 확신할 수 있어야 한다. 그리고 처치-튜링 논제를 갖다 써라.

바쁜 비버

바쁜 비버 함수는 튜링 기계가 얼마나 많은 것을 계산할 수 있는지를 나타내는 함수이다. 여러 변종이 있지만 여기서는 n 상태 튜링 기계의 최대 움직임 함수 $S(n)$을 고려한다. 즉, 정지하는 튜링 기계가 정지할 때까지 단계 수 최댓값이며, 시작 상태는 모두 0이다. $S(1) = 1$임은 자명하고, $S(2) = 6$, $S(3) = 21$도 금방 얻을 수 있다. 기계 개수가 $(2(2n+1))^{2n}$일 뿐만 아니라, 기계의 정지 여부를 모르기에 계산이 어렵다. 사실 이는 빠르게 증가하는 함수인데, 모든 계산 가능 함수를 궁극적으로 지배하기 때문이다.

$S(4) = 107$임은 알려져 있었고, $S(5)$는 재작년이 되어서야 47176870으로 밝혀졌다. 5-상태 바쁜 비버를 구축하는 데에는 콜라츠 추측과 유사한 수열이 사용된다. $S(6)$은 $10 \uparrow\uparrow 15$ 이상임이 증명되었고, 콜라츠 추측과 유사한 문제를 풀어야 구할 수가 있다. 이와 같은 간단한 수학 난제의 형식화는 바쁜 비버를 더욱 흥미롭게 만든다. 예를 들어, 25-상태의 '두 소수의 합으로 표현 불가능한 짝수를 만나면 정지하는 기계'를 고려하라. 리만 가설이 거짓이면 정지하는 744-상태 기계나 ZFC가 모순적이면 정지하는 745-상태 기계도 있다.

2 람다 대수

도입 및 정의

두 번째 계산 모형인 람다 대수는 '함수'의 개념을 극도로 밀어붙인 결과이다. 처음에는 논리를 형식화하기 위한 형식 체계였지만, 계산 이론에서의 엄청난 유용성이 밝혀졌다. 아이디어는 모든 것을 함수로 보는 것이며, 이를 위해 두 가지 단순화를 사용한다. 하나는 함수에 굳이 이름을 붙이지 않는다는 것인데, 여기에 λ 표기법을 사용한다. 두 번째는 커링으로, 모든 것은 단항으로 취급하는 것으로 이항은 단항 함수의 연쇄일 뿐이다.

람다 대수에서 다루는 대상의 엄밀한 정의는 변수에 람다 추상화와 적용을 몇 번 적용시킨 모든 것이다. 람다 추상화는 가능한 넓은 범위에 적용되며, 적용은 왼쪽부터라는 규칙으로 괄호를 생략할 수 있다. 람다 추상화의 종속 변수는 더미일 뿐이며, 마음대로 바꿔도 된다. 이를 알파 동치라 한다. 람다 추상화에 항을 적용하면 함수를 '계산'한다, 즉, 변수를 항으로 치환한다. 이를 베타 축약이라 한다. 마지막으로 $\lambda x.fx$와 f를 서로 바꿀 수 있는데, 이를 에타 변환이라 한다. 이 변환으로 전이 가능한 항들은 사실상 같은 항으로 봐도 무방하다. 베타 축약이 더 안 되는 항을 표준형이라고 하며, 이는 처치-로서 정리에 의해 존재한다면 유일하다.

예를 들어 $S = \lambda xyz.xz(yz)$, $I = \lambda x.x$일 때 $M = SII$는? $K = \lambda xy.x$, $N = \lambda x.(\lambda xy.y(xy))$일 때 SNK는? MM 같이 표준형이 존재하지 않는 것도 있다. 참고로 이 SKI는 모든 람다 항을 표현하기에 충분히 강력하다 - 즉 이는 튜링 완전하다. 심지어 $I = SKK$라서 S와 K만으로도 충분하고, $\iota = \lambda x.xSK$는 홀로 튜링 완전하게 된다.

표현력

이제 계산 가능성을 정의하기 위해, 자연수를 표현한다. 처치 인코딩이라는 방법은 자연수 n을 $n\,f\,x = f \cdots f\,x$인 항 n, 즉 n번 반복자로 정의한다. 그러면 덧셈이나 곱셈은 쉽게 정의된다:

$$\begin{aligned}\mathrm{add}\,m\,n &= \lambda fx.(m\,f)(n\,f\,x)\\ \mathrm{mult}\,m\,n &= \lambda fx.m\,(n\,f)\,x\end{aligned}$$

이렇게 $F'n' =' f(n)'$이 되도록 하는 람다 항 F가 있을 때 함수 f를 람다 계산 가능하다 한다. 뺄셈은 어떤가? 여기서부터 쉽지 않은데, 참고로 $m > n$이면 $n - m = 0$으로 계산되어야 한다.

이를 위해서는 $\mathrm{pred}\, n = n - 1$ 함수를 먼저 만들고, $\mathrm{minus}\, n\, m = n\, \mathrm{pred}\, m$으로 두면 된다. pred 함수는 순서대로 0 0 1 2 3 4$\cdots$를 반환해야 하는데, n을 반복자로 사용하면 $x, f(x), f(f(x)), \cdots$ 밖에 못한다. 그래서 사용하는 아이디어는 순서쌍 $(0,0), (0,1), (1,2), \cdots$를 사용하는 것이다. 순서쌍 (x, y)는 $\lambda z.zxy$로 형식화하는데, 그러면 $\mathrm{first} = \lambda p.p(\lambda xy.x)$, $\mathrm{second} = \lambda p.p(\lambda xy.y)$이다. 이제

$$f\, p = \mathrm{pair}(\mathrm{second}\, p)(\mathrm{succ}\, (\mathrm{second}\, p))$$
$$\mathrm{pred}\, n = \mathrm{first}(n\, f\, (\mathrm{pair}\, 0\, 0))$$

으로 정의할 수 있다. 이런 식의 재귀 함수 표현은 거의 제한이 없고 팩토리얼이나 피보나치도 구현 가능하다. 쌍을 반복 적용하면 리스트 처리도 가능하며 논리나 정수 등도 표현할 수 있다.

고정점 연산자

일단 람다 대수에서 if문을 처리하는 법을 알아보자. $\mathrm{isZero}\, n\, p\, q$라는 함수를 만들 건데, n이 0이면 p이고 다른 자연수면 q가 되도록 하고 싶다. 이는 isZero$n$이 0을 받으면 $\lambda xy.x$를, 아니면 $\lambda xy.y$를 반환하도록 $n(\lambda x.(\lambda xy.y))(\lambda xy.x)$로 두면 된다. 이제 m을 n으로 나눈 나머지 함수 $\mathrm{mod}\, n\, m$을 구축하고자 한다. $\mathrm{mod}\, n$ 함수를 f로 나타내면, f는 $f\, m = \mathrm{isZero}(\mathrm{minus}(\mathrm{succ}\, m)n)m(f(\mathrm{minus}\, m\, n))$을 만족해야 한다.

어떻게 해야 할까?

$$F\, f\, m = \mathrm{isZero}(\mathrm{minus}(\mathrm{succ}\, m)n)m(f(\mathrm{minus}\, m\, n))$$

라는 항 F를 구축한 다음, $F\, f = f$인 f를 찾으면 된다. 이는 F의 '고정점'이다 - 즉 고정점을 찾아주는 람다 항 Y만 찾으면 사실상 임의의 재귀를 구현할 수 있다. $MM = MM$을 다시 고려하면, 이는 자기 참조 역설의 표준적인 형태, 콰인이다. 따라서 $Y\, f = (\lambda x.(f(xx)))(\lambda x.(f(xx)))$로 두면 되는데, 이를 커리 고정점 combinator라고 한다. 그러면 $Y\, F$는 원하던 함수 $\mathrm{mod}\, n$이므로,

$$\mathrm{mod} = \lambda n.(Y(\lambda fm.\mathrm{isZero}(\mathrm{minus}(\mathrm{succ}\, m)n)m(f(\mathrm{minus}\, m\, n))))$$

이다. 고정점 연산자를 사용하면 람다 대수는 튜링 기계와 동일한 계산력을 가짐을 보일 수 있다.