Homework 5

2018-14245 김익환

1. 알게 된 것

Gentzen 의 natural deduction

Gentzen 은 Frege 의 논리학을 정형화하였다. Frege 의 함축 논리는 3 개의 axiom 을 기반으로 하고, 특히 3 번째 axiom 은 직관적으로 이해하기 힘들었다. 하지만 Gentzen 은 assumptions 를 추가함으로써 Frege 의 judgement 를 일반화하였고, 이를 통해, 함축 논리를 조금 더 자연스럽게 만들었다. Gentzen 의 자연 연역은 Id rule(tautology), Introduction rule, Elimination rule 을 기반으로 이루어진다. 같은 $\{\mathrm{A},\mathrm{B}\} \vdash A \wedge B$을 증명하더라도, {A, B}로부터 직접 증명하는 방법이 있고, $\{\} \vdash (\mathrm{B} \wedge \mathrm{A}) \rightarrow (\mathrm{A} \wedge \mathrm{B})$로부터 modus ponens 로 증명하는, 돌아가는 방법이 있다. 방금과 같은 경우, {A, B}로부터 직접 증명하는 것이 더 단순한데, Gentzen 은 임의의 $\Gamma \vdash \mathrm{A}$의 증명이 Γ와 A 에 있는 명제만 포함하도록 단순화될 수 있다는 subformula property 를 sequent calculus 에 대해 증명하였다. 하지만, subformula property 를 natural deduction 에 대해 증명하지는 못했다. 이에 대한 해결 방법은 Church 의 lambda calculus 에서 아이디어를 얻는다.

Church 의 lambda calculus

A. Church 는 논리의 새로운 형식화를 위해 lambda calculus 를 고안하였다. 하지만, 이는 컴퓨터 프로그램을 고안하는데 사용된다. Lambda calculus 는 기계적으로 계산될 수 있는 모든 함수를 나타낼 수 있다는 것이 증명되었고, 동시대에 Turing 이 고안한 Turing machine 이 동일한 역할을 한다는 것이 증명되었다. 기계에 의해 계산 가능한 수에 대한 모든 함수는 lambda abstraction, function application, variables, notion of reduction 으로 구성된 lambda term 으로 표현된다.

Typed lambda calculus

Church 는 다른 논리를 괴롭히던 역설들을 피하기 위해 lambda calculus 의 typed version 을 고안하였다. 하지만 신기하게도 Typed lambda calculus 의 프로그램은 Gentzen 의 natural deduction 과 유사하다는 것을 파악한다. 예를 들어, $\mathrm{B} \rightarrow \mathrm{A}$는 natural deduction 에서 A 가 B 를 함축한다는 것을 나타내지만, Church 의

시스템에서는 B type 에서 A type 으로 가는 함수의 type 을 나타낸다. 또한, $B \wedge A$는 natural deduction 에서 B 와 A 의 연언 결합을 나타내지만, Typed lambda calculus 에서 (B, A)로 construct 된 pair 의 type 을 나타낸다. Typed lambda calculus 또한 Id rule(tautology)과 함수, pair 각각에 대한 Introduction rule 과 Elimination rule 로 이루어진다.

Curry-Howard correspondence

Type derivation 이 주어졌을 때, 이로부터 대응되는 natural deduction proof 를 만들 수 있다. 또한, natural deduction proof 로부터 대응되는 type derivation 을 유도할 수 있다. 이는 terms 와 proofs 가 일대일 대응이 된다는 것을 의미한다. 또한, term reduction 은 proof simplification 에 대응된다. 예를 들어, $\{A, B\} \vdash A \wedge B$를 $\{\} \vdash (B \wedge A) \rightarrow (A \wedge B)$로부터 modus ponens 로 돌아가서 증명하는 방법은 $(\lambda z. < z.snd, z.fst >)(< y, x >)$의 lambda term 에 대응되고, 이 proof 의 단순화는 해당되는 lambda term 의 reduction sequence 에 대응된다. 이러한 natural deduction 과 typed lambda calculus 의 대응이 Curry-Howard correspondence 이다.

컴퓨터과학의 원천 아이디어가 나오기까지: 튜링의 1935 년 이야기

튜링 기계는 사실 "기계적으로는 모든 참인 명제를 만들 수 없다."를 증명하는데 사용된 소품에 불과하다. 위 명제를 처음 괴델이라는 수학자가 증명하였고, 튜링은 괴델의 증명에 대한 수업을 듣고 이를 자신의 방식으로 해석하였다. 먼저, "기계적"을 증명하기 위해 네가지 부품(기록 테이프, 읽고 쓰는 헤드, 상태 표시기, 부호 테이블)으로 이루어진 튜링 기계를 고안하였다. 중요한 점은 임의의 튜링 기계가 정해진 유한 개의 부호로 이루어진 글로 표현되기 때문에 유한한 수에 대응된다는 것이다. 만능 튜링 기계는 임의의 튜링 기계를 테이프에 실어 테이프에 기록된 튜링 기계를 그대로 실행하는 보편적인 기계이다.

튜링은 먼저, 모든 참인 명제를 만드는 튜링 기계가 존재한다면, 그 기계로 멈춤 문제를 풀 수 있다는 사실을 증명하였고, 다음으로 멈춤 문제를 푸는 튜링 기계는 존재할 수 없다는 사실을 증명하여 "기계적으로는 모든 참인 명제를 만들 수 없다."를 증명하였다.

튜링이 막스 뉴만의 강의를 듣고 괴델의 증명을 자신의 방식으로 재구성한 그 1 년(1935 년)을 복기해보면 튜링의 생각과정을 유추해볼 수 있다. 먼저, 괴델은 무한한 것을 찾아서 불완전성을 증명하였다. 튜링은 이를 토대로 하여 자신의 기계 세계에서도 어떤 무한한 것이 불가능해야 함을 알았을 것이다. 그럼 그 무한한 것이

무엇인지 생각해보기 마련인데, 일단 모든 튜링 기계는 유한한 부호의 글로 표현되므로 유한하다. 실행과정은 무한할 수 있다(infinite loop). X = 증명불가(X)에서 증명불가()라는 함수는 기계가 기계를 입력으로 받는다. 또한, 입력으로 받은 기계가 무한히 돌지 않을지 결정할 수 있어야 한다. 위의 추론과정에서 기계가 기계를 입력으로 받는 것이 만능 튜링 기계의 역할임을 알 수 있고, 입력으로 받은 기계가 무한히 돌지 않을지 결정하는 기계가 멈춤 문제를 푸는 기계임을 알 수 있다. 또한, 멈춤 문제를 푸는 기계가 불가능하다는 것을 증명하는 것에도 괴델의 대각선 논법을 칸토르의 대각선 논법으로 변환하여 적용하였다.

2. 느낀 것

사실 이번 읽을거리를 읽으면서 배운 점에 비해 느낀 점은 많지 않다. "Proofs are Programs" 이 글은 The Curry-Howard correspondence 를 설명하기 위해 잘 조직된 글이란 것은 알겠으나, 이것을 표면적으로만 설명하여(정확한 증명은 없고, 이것을 관찰할 수 있다는 식으로 설명함) 이 원리를 어떻게 실용적으로 활용할 수 있을지는 잘 모르겠다. 물론 Conclusion 에서 이 원리가 TAL 과 PCC 등 보안에 활용되고, 어떤 프로그램이 제대로 동작할지 알아내는 프로그램 분석에도 활용된다는 것은 알려주었으나 이것이 어떻게 구체적으로 적용되었는지는 잘 모르겠다. "Proof 는 Program 과 같다" 이 한 문장으로는 할 수 있는 것이 많지 않다. 어떻게 natural deduction 이 typed lambda calculus 와 같은 지(정확한 증명), Church 가 typed lambda calculus 를 왜 고안하였는지, 피하려고 한 paradox 가 무엇인지 조금 더 깊은 공부를 해봐야 The Curry-Howard correspondence 가 시사하는 바가 무엇인지, 이를 어떻게 활용할 수 있을지에 대한 직관이 생길 것 같다.
가장 가슴에 와 닿았던 말은 "튜링은 천재가 아니다"라는 말이다. 2018 카오스 강연의 목적은 저 말 한마디를 전하기 위함이었을 것이다. 괴델의 증명을 기반으로 한 튜링의 생각 과정을 추론해가는데 필요한 최소한의 지식(튜링 기계, 만능 튜링 기계)을 전달하고, 이를 기반으로 괴델의 증명에 대응되는 튜링의 증명 부분을 각각 찾아내어 튜링의 사고 과정을 단계별로 추측하였다. 튜링처럼 누군가의 증명을 자신만의 방식으로 해석하는 것이 때때로는 획기적인 발견으로 이어질 수 있고, 누군가의 증명을 재해석하는 것은 그 증명을 제대로 이해한다면 어렵지 않게 할 수 있는 일이다. 아직 컴퓨터 과학 분야는 발달한지 100 년이 채 되지 않았고, 아직 확립할 학문적 기반이 많이 남아있기에 튜링처럼 꼭 천재적이지 않은 방식으로도 무엇이든지 도전하고 싶다는 마음이 든다.

3. 질문하고 싶은 것

Gentzen 의 natural deduction, Church 의 lambda calculus, The Curry-Howard correspondence, Turing machine 등에 대한 내용을 자세하게 정리한 책 또는 글을 추천받고 싶습니다.