Informational logic

1 Primitive recursive function 04/10

- Primitive recursive function is a recursion that changes the "n" condition into "n-1" condition
- Primitive recursive function is defined by zero, succ, proj, Composition and primitive recursion
- Primitive recursive predicate is a primitive recursive function that outputs 0 or 1.
- Functions that we often use is written in primitive recursive function.

2 Recursive function and while programming 04/17

Today, I learnt recursive function mostly about difference from primitive recursive function. Plus, I learnt introduction to while programming.

- It is defined by zero, succ, proj, Composition, Primitive recursive, and Minimization
- The difference from primitive recursive function is the minimization
  - it is like not bounded minimization
  - Ackerman function is not primitive recursive but recursive function
- What is while programming
- For any recursive function, there is a while program that computes it.
  - use induction
- Define normilized while programming

3 While progtramming and Church thesis 04/24

Today, I learnt the equality between while programming and recursive function and its prrof. Then did the proof that undecidability of halting problem by introducing the universal recursive function.

- Every while programming can be normalized since it is equivalent to recursive function.
- Computability is equivalent to recursive function
- Define universal recursive function

$$\boldsymbol{\lambda}\,(x_0,\ldots,x_{m-1})\,.\quad comp\,(p_f,\langle x_0,\ldots,x_{m-1}\rangle)$$

comp means interpreter

- Halting problem is undecidable
  - It is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running or continue or run forever.

4 Recursion theorem 05/08

Today, I learnt the function that is recursive but not terminate in finite time. This is what recursive enumerable represents. Plus, Rice's theorem is important theorem to prove some other theorem. It represents the limitation of what program code can represent.

- S-m-n theorem
  - foundation of partial function
- Recursion theorem
  - if f is a total recursive function, there exists r s.t r and f(r) represents the same code.
- Rice's theorem
  - The restriction that we take when using code.
- Recursively enumerable
  - outputs yes and undefined
  - can be written in 5 forms
- Negation theorem
  - "P is recursive" and "P and $N^m \setminus$ P are both RE" are equivalent

5 Equalitional Logic 05/15

- Three steps are taken to judge the equality in a computer
  - Fix a language
  - Specify axioms
  - Derive the equality
- In this three steps, terms and variables are used
  - Term is like a funciton in a semantics
  - Make sure that variable is used in two layers, object levels and meta levels
  - We also define a substitusion
- After judging the equality, we have to specify the assumed equality
  - In this step, we are yet interpret equational formulas
  - Defune algebraic specification as a pair of a signiture(Σ) and a set of equational formulas(E)
  - Starting from E, we can create a proof tree by applying a derivations
- Semantics
  - In this steps, we interpret equational formulas
  - In other words, we make the syntax into the mathematical objects

6 Syntax vs Semantics 05/22

- Soundness is (all the equation on the syntax tree) $\rightarrow$ (all the equation that can be proven in algebra)
- completeness is (all the equation that can be proven in algebra) $\rightarrow$ (all the equation on the syntax tree)
- Soundness in equational logic can be proven by induction over the syntax structure
- Completeness in the equational logic can be proven by taking a contradiction

7 Propositional logic syntax 05/29

- prove the completeness of the equational logic
- Any formulae in equational logic are defined inductively
- Next, introduce the derivarion rules
  - Some styles that are equivalent (Hilbert style, Natural deduction, etc)
  - "Sequent" is an object that composed of two sequents and delimiting symbol ($\Rightarrow$)
  - How to make proof trees
- introduction of semantics in the propositional logic

8 Propositional Logic Semantics Completeness and Soundness 06/05

- Valiation function is a function that J:

$$PVar \rightarrow \{tt, ff\}$$

- Soundness is easy to prove
- completeness
  - First, we introduce the consistent pair (U,V)
  - We can create maximal consistent pair from one consistent pair using induction
  - we can prove contraposition
- Compactness in the propositional logic

9 Predicate logic syntax and semantics 06/12

- Definition of FnSymb, PdSymb and Fml.
- Alpha equivalence and capture avoiding substitution
  - Free variable is different if name is different, however bounded variable independent from their name ($\forall x.P(x) = \forall z.P(z)$)
  - $A[t/x]$means replace free occurence of x in A with t

- Semantics of predicate logic

10 Predicate logic soundness and completeness 06/19

- Soundness is proved using the structual induction
- Completeness is much more difficult
- Cut elimination
  - cut elimination is like a syllogism
  - What can be proved using cut eliminations are proves without cutelimination
- Skolemization
  - this is a way that express $\exists$ in the other way
  - Skolemization is used for proving completeness in an easy case, Herbrand Structure

11 Herbrand structure and Resolution principle 06/26

- Prove completeness of the closed formula using Herbrand structure
  - Converting predicate logic into propositional formula, we prove the completeness.
  - We defined the Herbrand structure
- Resolution principle
  - definitnion of literal, clause, and Horn clause
  - c(S) is denoted by

$$\left(\bigvee c_1\right) \wedge \cdots \wedge \left(\bigvee c_n\right)$$

12 Resolution princeiple (predicate logic) 07/03