# Contents

**1 Lecture 1 *<2017-09-05 Tue>* 5**
1.1 Homework . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
1.2 Quizzes . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.3 Proofs . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.3.1 Example Proofs . . . . . . . . . . . . . . . . . . . . . . 7

**2 Lecture 2 *<2017-09-07 Thu>* 8**
2.1 Things that are infinite . . . . . . . . . . . . . . . . . . . . . 8
2.2 Mathematical induction . . . . . . . . . . . . . . . . . . . . . 8
2.2.1 Deduction: . . . . . . . . . . . . . . . . . . . . . . . . . 9
2.2.2 Induction: . . . . . . . . . . . . . . . . . . . . . . . . . 9
2.2.3 Inductive or Deductive? . . . . . . . . . . . . . . . . . . 9
2.3 Handout . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.3.1 Recursive (inductive) definition: . . . . . . . . . . . . . . 10
2.3.2 Theorem . . . . . . . . . . . . . . . . . . . . . . . . . . 12

**3 Lecture 3 *<2017-09-12 Tue>* 12**
3.1 Set theory . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
3.1.1 Ex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
3.1.2 Definitions . . . . . . . . . . . . . . . . . . . . . . . . . 13
3.2 Tuples: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
3.2.1 Cross-product . . . . . . . . . . . . . . . . . . . . . . . . 14
3.2.2 Relations . . . . . . . . . . . . . . . . . . . . . . . . . . 14

**4 Lecture 4 *<2017-09-14 Thu>* 17**
4.1 Recap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.2 More on sets . . . . . . . . . . . . . . . . . . . . . . . . . . 17
4.2.1 Cardinalities . . . . . . . . . . . . . . . . . . . . . . . . 17
4.2.2 Proof (by contradiction): . . . . . . . . . . . . . . . . . . 18

**5 Lecture 5 *<2017-09-19 Tue>* 20**
5.1 Recap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
5.2 Cardinality . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
5.2.1 Countable: . . . . . . . . . . . . . . . . . . . . . . . . . 20
5.2.2 Uncountable . . . . . . . . . . . . . . . . . . . . . . . . 21
5.3 Formal Systems (GEB CH. 1) . . . . . . . . . . . . . . . . . . 21
5.3.1 Examples . . . . . . . . . . . . . . . . . . . . . . . . . . 21
5.3.2 MIU-System: . . . . . . . . . . . . . . . . . . . . . . . . 22

**6 Lecture 6 <*2017-09-21 Thu*> 25**
6.1 Quiz prep 25
6.2 Review 25
6.3 Decision Procedure 26
6.4 pq-system 26
6.4.1 Interpretation: 27
6.4.2 More Interpretations 27
6.5 geq-system: 28
6.6 Infinite Prime Numbers 29
6.6.1 Proof (by contradiction): 29

**7 Lecture 7 <*2017-09-26 Tue*> 29**
7.1 Quiz review 29
7.2 Decision procedure 30
7.3 FS for addition 31
7.4 FS for multiplication 31
7.5 Recursively enumerable set (r.e.) 32
7.6 Recursive set 32

**8 Lecture 8 <*2017-09-28 Thu*> 33**
8.1 Theorems 33
8.1.1 E.g. 33
8.2 PQ*-system 35
8.3 Propositional Logic 36
8.3.1 Formula trees: 36

**9 Lecture 9 <*2017-10-03 Tue*> 38**
9.1 Propositional Logic 38
9.2 Well-formed formulas (wff) 38
9.2.1 Interpretation 39
9.2.2 Truth tables 40
9.2.3 Inferences 42
9.2.4 Natural Deduction (Syntax) 43

**10 Lecture 10 <*2017-10-05 Thu*> 43**
10.1 Natural Deduction 43
10.2 Exercise 45

**11 Lecture 11 *<2017-10-10 Tue>* 46**

11.1 Recap . . . . . 46

11.2 Axiomatic Proof System . . . . . 46

11.3 GEB proof system . . . . . 48

11.3.1 Ex. . . . . . 48

11.4 Relations of systems . . . . . 48

11.4.1 Soundness Theorem . . . . . 49

11.4.2 Completeness Theorem . . . . . 50

**12 Lecture 12 *<2017-10-12 Thu>* 50**

12.1 Recap . . . . . 50

12.2 First Order Logic (FOL) . . . . . 51

12.2.1 Term . . . . . 51

12.2.2 wff . . . . . 52

12.2.3 First Order Arithmetic (FOA) . . . . . 52

**13 Lecture 13 *<2017-10-17 Tue>* 54**

13.1 Quiz Review . . . . . 54

13.2 Tarski & Quantifiers . . . . . 55

13.3 Syntax . . . . . 56

**14 Lecture 14 *<2017-10-19 Thu>* 57**

14.1 FOA . . . . . 57

14.2 Substitution . . . . . 58

14.3 Proofs . . . . . 59

14.3.1 Natural deduction calculus . . . . . 59

**15 Lecture 15 *<2017-10-24 Tue>* 61**

**16 Lecture 16 *<2017-10-31 Tue>* 62**

16.1 Typographic Number Theory (TNT)/Dedekind Peano Arithmetic (DPA)/FOA . . . . . 62

16.1.1 Theorems . . . . . 63

16.1.2 $\omega - incomplete$ . . . . . 65

**17 Lecture 17 *<2017-11-02 Thu>* 65**

17.1 MIU . . . . . 65

17.1.1 Theorem . . . . . 65

17.1.2 Proof by strong induction . . . . . 66

17.2 G\"odel numbering . . . . . 66

**18 Lecture 18** *<2017-11-07 Tue>* **68**
18.1 Turing Machines . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
18.1.1 TM as functions . . . . . . . . . . . . . . . . . . . . . . . 71
18.2 The Halting Problem . . . . . . . . . . . . . . . . . . . . . . . . . 72
18.2.1 Claim . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
18.2.2 Proof (by contradiction) . . . . . . . . . . . . . . . . . . 73

**19 Lecture 19** *<2017-11-09 Thu>* **74**
19.1 Reminder: Turing Machines & Computers . . . . . . . . . . . 74
19.2 Class of functions . . . . . . . . . . . . . . . . . . . . . . . . . . 74
19.2.1 E.g . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
19.3 Bloop-Programs . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
19.3.1 Ex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
19.3.2 Syntax . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77

**20 Lecture 20** *<2017-11-14 Tue>* **77**
20.1 Bloop . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
20.2 Floop . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 78
20.3 Bounded minimization . . . . . . . . . . . . . . . . . . . . . . . 78
20.4 Unbounded minimization . . . . . . . . . . . . . . . . . . . . . 79
20.5 Partial recursive functions . . . . . . . . . . . . . . . . . . . . 80

**21 Lecture 21** *<2017-11-16 Thu>* **82**
21.1 Turing Machine on Handout 5 . . . . . . . . . . . . . . . . . . 82
21.2 Turing Machine for addition . . . . . . . . . . . . . . . . . . . 84
21.3 Turing Machine for multiplication . . . . . . . . . . . . . . . . 85
21.4 The Halting Problem . . . . . . . . . . . . . . . . . . . . . . . . 85

**22 Lecture 22** *<2017-11-21 Tue>* **86**
22.1 Quiz Review . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
22.2 Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
22.3 Representability . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
22.3.1 Ex. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
22.3.2 Thm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
22.3.3 Thm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
22.4 Theories of Arithmetic . . . . . . . . . . . . . . . . . . . . . . . 87
22.5 Proof-Pairs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
22.6 Substitutions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88

**23 Lecture 23 *<2017-11-23 Thu>* 89**
23.1 Ex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
23.2 General Version of G\"odel's First Incompleteness Theorem . 91

**24 Lecture 24 *<2017-11-28 Tue>* 92**
24.1 Summary of G1 . . . . . . . . . . . . . . . . . . . . . . . . . . 92
24.1.1 Hilbert's Program . . . . . . . . . . . . . . . . . . . . 93
24.1.2 Thm . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
24.1.3 Proof . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
24.2 G2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
24.3 Church Thm . . . . . . . . . . . . . . . . . . . . . . . . . . 94

# 1 Lecture 1 *<2017-09-05 Tue>*

Logic & Computability Prof. Dirk Schlimm

- Find out what Schlimm means for the next lecture
- Great for people in computer science, but everyone else too
  - Essential material that everyone should know
  - Stable material, as the material is old
  - Very abstract and technical material, even if it does not require a solid mathematical background
  - Hard course
    - Important to give feedback to the professor

This course complements the textbook, Godel, Escher Bach.

## 1.1 Homework

is not graded, just checked if done.

- Why?
  - To motivate us to do homework exercises
  - Practice is important, the course is hard
  - TAs don't need to correct them, so they can hold more office hours

Discussion board on MyCourses. Do not email the professor, ask questions on discussion board so everyone can see the answer (incase they have the same question).

Some times there will be intentional mistakes on the board.

- To make it easier to ask questions
- To motivate us to pay attention

There are no stupid questions, even if you ask the same question as the person before you. Perhaps the professor's answer was unclear. The most stupid question is the one not being asked.

### 1.2 Quizzes

- 3 quizzes throughout this course
- Dates will be on the schedule on professor's website
- In class, 15-20 minutes long
- Each one is graded and worth 10%
- Fairly straightforward, some are even definitions
- To make sure you've done your work

Midterm is 25%, Final is 40%

### 1.3 Proofs

We will see lots of proofs, different kinds of proofs.

- Direct: Go from assumption towards the theorem
- Indirect: Negation of claim → contradiction → claim
  - Sometimes called proof by contradiction
- Biconditional: 2 claims, $p, q$
  - Start with $p$ and prove $q$ but also start with $q$ and prove $p$
- By cases: Split claim into several cases
  - case 1, case 2, case 3

    - Each one proves the same conclusion
    - If the cases were exhaustive, then you have proved the claim

- Induction (to be taught next lecture)

So you can split a proof into subproofs of these kinds.

### 1.3.1 Example Proofs

1. Thm $\sqrt{2}$ is nor rational.

    - Rational: Fractions: $\frac{x}{y}$

    If you have a square with sides of length 1, the length of the diagonal is $\sqrt{2}$ Pythagoras proved this.

    (a) Def 1 A natural number $a$ is even if and only if (iff) $\exists$ a natural number $b$, such that $a = 2b$.

    (b) Lemma 1 For any number $a$, $a^2$ is even iff $a$ is even.

    - Biconditional, since iff

    i. Proof: $\leftarrow$ Assume: $\underline{a \text{ is even}}$. So there is: $a = 2b$ (by def. 1) $a^2 = (2b)^{=}4b^2 = 2(\underbrace{2b^2}_{c})$ (square) So, $\underline{a^2 \text{ is even}}$, since it is 2 times $c$, a natural number.
    $\rightarrow$ (DIY) Assume: $\underline{a^2 \text{ is even}}$. So there is: $a^2 = 2b$ (by def. 1)

    (c) Lemma 2 For any rational number $x$, there are natural numbers $a$ and $b$, not both even, s.t. $x = \frac{a}{b}$

    - If they were both even, you could simplify the fraction by dividing by 2.

    Proof omitted for this lemma.

    (d) Proof of Thm: Indirect proof. (Contradiction) Assume (for reductio/contradiction): $\sqrt{2}$ is rational. By Lemma 2, $\exists$ natural numbers $a$ and $b$ not both even s.t. $\sqrt{2} = \frac{a}{b}$
    Square: $2 = \left(\frac{a}{b}\right)^2 = \frac{a^2}{b^2}$ $a^2 = 2b^2$
    So $a^2$ is even (by def. 1) $\underline{a \text{ is even}}$ by (lemma 1) If $a$ is even, we can write: $a = 2c$ (by def 1) Square: $a^2 = (2c)^2 = 4c^2 = a^2$
    $4c^2 = 2b^2$

Divide by 2: $2c^2 = b^2$ So $b^2$ is even (def. 1) $\underline{\text{b is even}}$ (by lemma 1)
Contradiction! Assumption is false, therefore $\sqrt{2}$ is $\underline{\text{not}}$ rational. □

2. **TODO** Download Handout and read it

# 2 Lecture 2 *<2017-09-07 Thu>*

Last class we talked about proofs & types of proofs. Next week we'll be talking about sets and countability and comparing them all. Will talk about density of rationals and irrationals, to say which is bigger.

## 2.1 Things that are infinite

- Natural numbers
- Rational numbers
- Infinite lists

1. How do you prove things about infinitely large things?
   - Counter example
     - All swans are white
       - Show one that isn't white
   - Pick an arbitrary example and show that it works for that
     - Use particular properties about an arbitrary object to show that something works for all of them

## 2.2 Mathematical induction

- Inference (step):
  - Certain number of assumptions/premises $A_n \ldots A_n$
  - Conclusion

### 2.2.1 Deduction:

- It is impossible for the premises of an inference step to be true and the conclusion false.
- The conclusion follows necessarily from the premises. (reformulation of above)

1. e.g. $\frac{\text{if } A \text{ then } B \quad A}{B}$
    - MODUS PONENS (type of deductive inference)

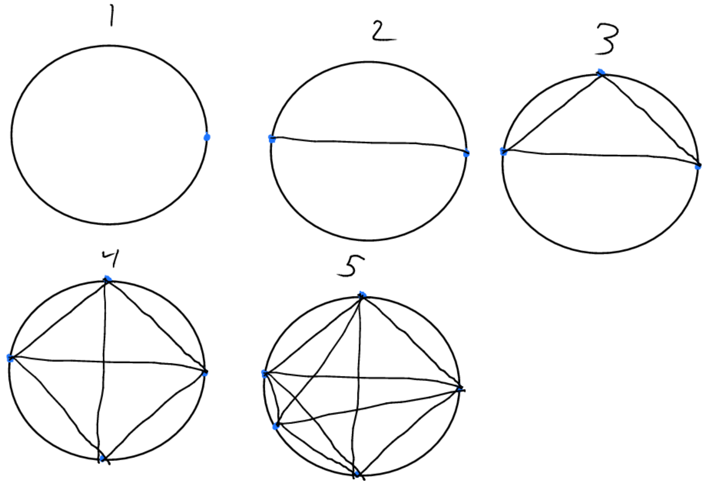
### 2.2.2 Induction:

- The premises make the conclusion more likely
- My cat is smart, my friend's cat is smart, my parent's cat is smart, so all cats are smart
    - Inductive argument, makes it more likely, but doesn't see them all

### 2.2.3 Inductive or Deductive?

1. Claim Let $n$ be the number of points on a circle. Then the number of regions obtained by pairwise connecting each point is $R = 2^{n-1}$
    (a) Argument

| $n$ | $R$ |
|---|---|
| 1 | $1 = 2^0$ |
| 2 | $2 = 2^1$ |
| 3 | $4 = 2^2$ |
| 4 | $8 = 2^3$ |
| 5 | $16 = 2^4$ |
| 6 | 31 |



- Inductive argument
- What was wrong with the argument?
  - We saw a pattern, but. . .
    - There's no reason for the jump from each $n$ to have something in common
    - If they had something in common, then it would continue holding for the next one
  - Therefore induction makes the premise more likely
    - But does not establish it deductively
  - So in order to rigorously prove something inductively, we need mathematical induction

## 2.3 Handout

### 2.3.1 Recursive (inductive) definition:

1. Base clause(s) defines basic elements.
2. Inductive clause(s): How to build up complex elements from parts
3. Final clause: Nothing else is an element (bookkeeping)

1. E.g.
   (a) $\mathbb{N}$
   - Base clause 0 is in $\mathbb{N}$

    - Inductive clause: if $x \in \mathbb{N}$ then $s(x)$ (successor of $x$) then $s(x)$ is in $\mathbb{N}$
    - Final clause: Nothing else is in $\mathbb{N}$
    - So natural numbers are:
        - $0, s(0), s(s(0)), \ldots$

   (b) Even numbers or odd numbers

    - Take successor of successor, take 0 as base clause for even, 1 for odd

   (c) Lists

    - Empty list is a list
    - What you get from adding to a list is also a list

   (d) Dominoes

    - When you have a domino, you can place one 2 cm behind it
    - Push first one, they all fall
        - To prove they all fall, have to show they all have a certain amount of space between them
            * Relates to proof by mathematical induction

2. Proof by mathematical induction

   (a) Base case: Show that the property holds of the basic elements.

   (b) Inductive step:

      i. Assume that the property holds for some element $n$ (Inductive Hypothesis)

      ii. Show: holds for elements generated from $n$ by inductive clauses.

   (c) Conclusion: Property holds for all elements.

   This is **deductive inference**! What are the premises?

   - For natural numbers:
       - $\dfrac{\overbrace{P(0)}^{\text{Base case}}\ \overbrace{P(n)}^{\text{IH}}\ \overbrace{\rightarrow}^{\text{Ind. step}}\ P(s(n))}{\forall x P(x)}$

3. Variant (strong/complete induction):

   - Ind. Step.

     (a) Assume that $P$ holds for all elements less than $n$

     (b) Show: $P$ holds of $n$

   - No base case
   - See example 5.5!

### 2.3.2 Theorem

For any nat. number $n \geq 1$, the sum $\underbrace{1+2+\ldots+n}_{\sum_i=1^n i} = \frac{n(n+1)}{2}$ (If you do a proof for your homework or on an exam, always include many details. You can even use a template to structure your proofs the same way, useful for steps for induction.)

1. Proof (by math. ind)

   (a) Base case: Show claim holds for $n = 1 = \frac{1(1+1)}{2}$

   (b) Ind. step.

      i. I.H. The claim holds for $m$: $\sum_{i=1}^{m} i = \frac{m(m+1)}{2}$

      ii. Show: The claim holds for $m+1$

   2 strategies, either $\frac{n(n+1)}{2} \rightarrow 1+2+\ldots+n$ or $1+2+\ldots+n \rightarrow \frac{n(n+1)}{2}$. Will be doing 2nd. $1+2+\ldots+(m+1) = \sum_{i=1}^{m+1} i = \sum_{i=1}^{m} i + (m+1)$

   $= \frac{m(m+1)}{2} + (m+1)$ (by I.H.)

   $= \frac{m(m+1)+2m+2}{2} = \frac{(m+1)(m+2)}{2}$

   (a) Conclusion: The claim holds for all $n \geq 1$ □

# 3 Lecture 3 *<2017-09-12 Tue>*

## 3.1 Set theory

All that is being said here is taken from the reading mathematical introduction to logic chapter zero.

A set is a thing with elements. We can present sets in two ways:

- Extensional:
  - Presentation
  - $\{1, 2, 3\}$
- Intensional:
  - Given a set $A$, and a property $P$: $\{x \in A | P(x)\}$

### 3.1.1 Ex

$\mathbb{N}$: the set of natural numbers.

$D = \{x \in \mathbb{N} | x \text{ is prime}\} = \{2, 3, 5, 7, 11, \ldots\}$

### 3.1.2 Definitions

- $A \subseteq B \iff \forall x, x \in A \implies x \in B$
- $A = B \iff (A \subseteq B) \wedge (B \subseteq A)$
- $A \underbrace{\subset}_{\text{Proper subset}} B \iff (A \subseteq B) \wedge (A \neq B)$
- Empty set: $\emptyset, \{\}$
  - When is $x \in \emptyset$? Never.
  - $\emptyset \subseteq X$? Always.
    - * Since all elements of the empty set are in $X$.
  - $\emptyset \in X$?. If $X$ contains $\emptyset$.
    - * E.g. $X = \{\{\}, 4\}$

$A = \{2, 4, 8\}, B = \{a, 4, z\}$

- $\underbrace{A \cap B}_{\text{intersection}} : \forall x, (x \in A) \wedge (x \in B)$
  - $A \cap B = \{4\}$
- $\underbrace{A \cup B}_{\text{union}} : \forall x, (x \in A) \vee (x \in B)$
  - $A \cup B = \{2, 4, 8, a, z\}$
- complement: $\bar{A}$ : all elements that are not in $A$ (from the universe of discourse, the universe we're talking about)
- Power set $\mathfrak{P}(A)$ : the set of all subsets of $A$
  - E.g. $\mathfrak{P}(B) = \{\emptyset, \{a\}, \{4\}, \{z\}, \{a, 4\}, \{a, z\}, \{4, z\}, \{a, 4, z\}\}$
  - If $A$ has $n$ elements, $\mathfrak{P}(A)$ has $2^n$ elements.
- Is $\{\emptyset, a\} \subseteq \{a, 4, z\}$? No.

### 3.2 Tuples:

Like sets, but order matters.

- Ordered pair: $\langle a, b\rangle = \{\{a\}, \{a, b\}\}$
- $\langle a, 4, z\rangle \neq \langle 4, a, z\rangle$

#### 3.2.1 Cross-product

$A \times B \iff \{\langle x, y\rangle | x \in A \wedge y \in B\}$

- If $A$ has $n$ elements, $B$ has $m$ elements
- then $A \times B$ has $n \cdot m$ elements
- and there are $2^{n \cdot m}$ relations between $A$ and $B$
  - Since this is essential just the cardinality of the power set of the cross product
  - E.g. $n = 5, m = 5$. 2 pairs of 5 friends. How many relations are possible? $2^{25} = 33,554,432$

#### 3.2.2 Relations

- $A = \{$John, Paul, George$\}$
- $B = \{$guitar, bass$\}$
- $\{\langle$John, guitar$\rangle, \langle$Paul, bass$\rangle, \langle$George, guitar$\rangle\} = R_1$
- $R_2 = \{\langle$John, bass$\rangle\}$

A relation $R$ on $A$ and $B$ is a subset of $A \times B$.

- Elements of relations are tuples.

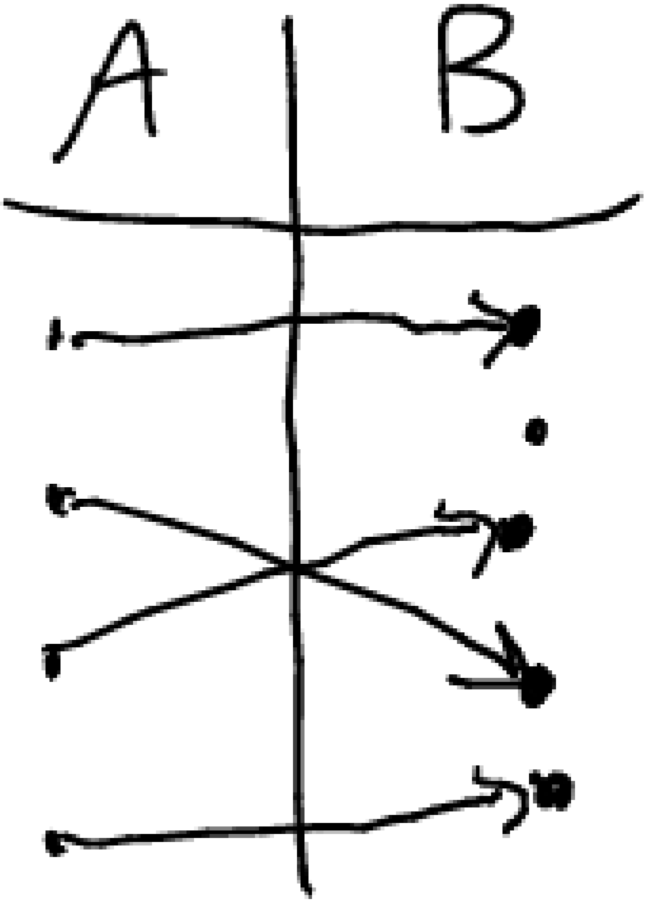
Domain of a relation $R : \{a|$there is a b, s.t. $\langle a, b\rangle \in R\}$

- domain of $R_1 : \{$John, Paul, George$\}$
- domain of $R_2 : \{$John$\}$

Range of a relation $R$: $\{b|$there is an a, s.t. $\langle a, b\rangle \in R\}$

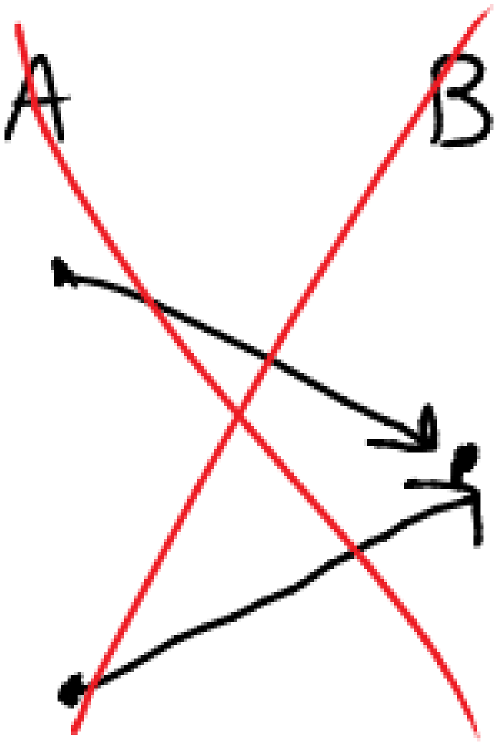
1. Functions A total function $f : A \to B$ is a binary relation $R$, on $A$ and $B$ such that.

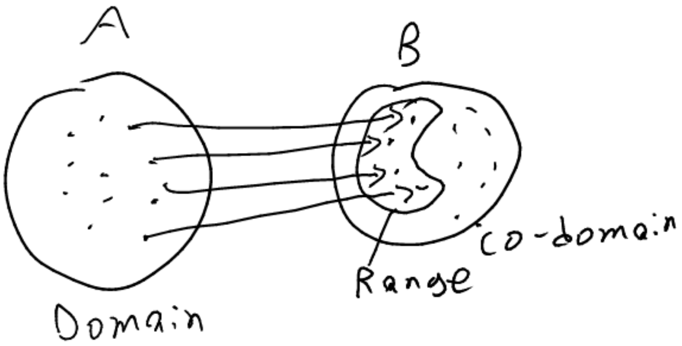
- It is single-valued
  - Every element in $A$ is mapped to exactly one element in $B$
- The domain of $R$ is $A$



(a) Definitions

- A function is injective (one-to-one), if each element in the range is mapped to by exactly one element.
  - To show this: Assume $f(x) = f(y)$
    - Show $x = y$
  - So you don't have the situation that:





- A function is surjective if the *range* = *codomain*.
- A function that is both injective and surjective is bijective.

# 4 Lecture 4 *<2017-09-14 Thu>*

## 4.1 Recap

We talked about sets last class, such as:

- $\{1, 4, z\}$
- $|\{1, 4, z\}| = 3$ (Cardinality)

---

Are there more students or chairs in this class?

- There are more chairs.
- Matched students with chairs and to see what is left
- f(students) → chairs
  - Injective function (can't have 2 students on one chair)
  - Every element of the range must be mapped to something
  - No element in the range can map to two elements
- $\implies |S| \leq |C| \iff$ there is an injective function from $S$ to $C$.

Cantor: $|A| = |B| \iff |A| \leq |B|$ and $|B| \leq |A| \iff$ there is a bijection between $A$ and $B$.

## 4.2 More on sets

### 4.2.1 Cardinalities

A set D is finite if its cardinality is a natural number.

- $D \leftrightarrow \{1, \ldots, n\}$ (bijective function with natural numbers exists)

A set is countably infinite (denumerable), if it is equinumerous to ℕ (bijection from this set to all the natural numbers).

- $E = \{2, 4, 6, 8, \ldots\}$
  - $|E| = |\mathbb{N}| = |\mathbb{Z}| = |\mathbb{Q}| < |\mathbb{R}|$

| $\mathbb{N}$ | 1 | 2 | 3 | 4 | $\ldots n$ |
|---|---|---|---|---|---|
| $E$ | 2 | 4 | 6 | 8 | $\ldots 2n$ |

$f(x) = 2x, \mathbb{N} \rightarrow E$

$\mathbb{Z} = \{\ldots - 3, -2, -1, 0, 1, 2, 3, \ldots\}$

- Is this bigger than the cardinality of the natural numbers?
  - No, it's the same size, bijection. Even to positive, odds to negative.

| $-3$ | $-2$ | $-1$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| 5 | 3 | 1 | 0 | 2 | 4 | 6 |

$\mathbb{Q}^+ = \{\frac{x}{y} | x, y \in \mathbb{N}\}$

- Are there more?

| | 1 | 2 | 3 | 4 | 5 | 6 | ... |
|---|---|---|---|---|---|---|---|
| 1 | 1/1 | 1/2 | 1/3 | 1/4 | 1/5 | 1/6 | ... |
| 2 | 2/1 | 2/2 | 2/3 | 2/4 | 2/5 | 2/6 | ... |
| 3 | 3/1 | 3/2 | 3/3 | 3/4 | 3/5 | | |
| 4 | 4/1 | 4/2 | 4/3 | 4/4 | | | |
| 5 | 5/1 | 5/2 | 5/3 | | | | |
| 6 | | | | | | | |

There are duplicates here though. So, instead of counting left to right, count diagonally.

- i.e. 1:1/1, 2:1/2, 3:2/1, 4:3/1, 5:2/2, 6:1/3, ...

$\mathbb{R} = \mathbb{Q} \cup \{\text{irrationals}\}$

- What are real numbers? All numbers that can be expressed via decimal expansion.
- $x.xxxxx\ldots$
- Is this countably infinite? No.

### 4.2.2 Proof (by contradiction):

Assume $|\mathbb{N}| = |\mathbb{R}^{0.1}|$. ($\mathbb{R}^{0.1} = \{x \in \mathbb{R} | 0 < x < 1\}$.)

Therefore, there is a bijection $f : \mathbb{N} \rightarrow \mathbb{R}^{0.1}$

Then, we can build the following table:

| $\mathbb{N}$ | |
|---|---|
| 0 | $f(0) = 0.12345\ldots$ |
| 1 | $f(1) = 0.33333\ldots$ |
| 2 | $f(2) = 0.5000\ldots$ |
| 3 | $f(3) = 0.011101\ldots$ |
| 4 | $\ldots$ |
| $\ldots$ | $\ldots$ |
| $n$ | $f(n) = 0.112\ldots = z$ |

Can you explain this table? Not really. Why?

- Construct new number $z$:
  - $z = 0.z_1 z_2 z_3 z_4 \ldots$
  - Rule for constructing $z$:

$$z_i = \begin{cases} 1 & \text{if } f(i)_j \neq 1 \text{ where } f(i)_j \text{ is the ith digit in the decimal expansion of f(i)} \\ 2 & \text{otherwise} \end{cases}$$

- $f(1)_1 = 3$
- $f(2)_2 = 0$
- $f(3)_3 = 1$
- $\implies z = 0.112\ldots$
- By construction, $z$ is a real number between 0 and 1.
- So it must be in the table, say in line $n$.

What is $z_n$?

- Two cases:
  - $z_n = 2 = f(n)_n$ if $f(n)_n = 1$ $\lightning$
  - $z_n = 1 = f(n)_n$ if $f(n)_n \neq 1$ $\lightning$
  - $\implies$ contradiction!
    - The assumption is false.

# 5 Lecture 5 *<2017-09-19 Tue>*

- Quiz in one week
- 20 minutes, in class
- 8-10 questions, very simple
  - Everything that was said in class
  - Everything done in the homework
  - Readings

## 5.1 Recap

Last class, we proved: (Size of Natural numbers) $\aleph_0 < |\mathbb{R}| \implies$ Diagonalization

- We had a table and changed every element on the diagonal in order to get a new element
- We will see many more proofs by diagonalization
- Homework question: $|\mathfrak{P}(\mathbb{N})| > |\mathbb{N}|$
  - In general though, $|\mathfrak{P}(x)| > |x|$ (Cantor's theorem)
    - What does this imply? There are infinite amount of infinite cardinalities (power set is bigger, power set of the power set is even bigger, ...)
    - $|\mathbb{N}| < |\mathfrak{P}(\mathbb{N})| = |\mathbb{R}| = 2^{\aleph_0} < |\mathfrak{P}(\mathfrak{P}(\mathbb{N}))|$

## 5.2 Cardinality

Things that have the same cardinality:

### 5.2.1 Countable:

$|\mathbb{N}|$:

- E
- $\mathbb{Z}$
- $\mathbb{Q}$

- English words
- Sentences
  - Finite objects that you can list
  - Why doesn't diagonalization work on sentences?
- Programs
- MIU strings
- MIU theorems
- If you can list them, they're countable

### 5.2.2 Uncountable

$|\mathfrak{P}(\mathbb{N})|$

- $\mathbb{C}$
- $\mathbb{R}$
- Functions from $\mathbb{N} \to \mathbb{N}$
  - From $\mathbb{N} \to \{0, 1\}$

---

## 5.3 Formal Systems (GEB CH. 1)

### 5.3.1 Examples

- Programming languages
- Logic
- Computation: TM
- Formal arithmetic

---

2 things in formal systems:

- The distinction between the two is very important

  - Important concepts in this course:
    * Induction
    * Diagonalization
    * Distinction between these 2 things

| Syntax | Semantics |
|---|---|
| - Grammar | - Meaning |
| - Formal Structure | - Context |

13 -> What is this?

- 13 is a numeral
  - The meaning of this numeral is the number 13 (abstraction)
- Why this example? We looked at the syntax of 13 but we said it was the number 13 (the meaning)
  - During everyday life we don't often make the distinction
- dog
  - Syntactically, has 3 letters
  - Semantically, has fur

### 5.3.2 MIU-System:

Alphabet: MIU
Strings (sequences of elements from the alphabet).
Rec. def:

- Base clause: $\emptyset$ is a MIU-string
- Inductive clause:
  1. If $x$ is a MIU-String, then $xM$ is a MIU string
     - Is $x$ an MIU-String? No, that's it's meaning, not it's syntax. It's a letter (also a meta-variablee)
  2. $xI$
  3. $xU$
- Final clause: Nothing else.

1. MIU-Theorems

    (a) Axiom: MI.

    (b) Inference rules:
    I. xI → xIU
    II. Mx → Mxx
    III. xIIIy → xUy
    IV. xUUy → xy
    (for x,y MIU strings, possibly empty)

    (a) Def. Derivation A derivation is a sequence of strings such that each element is either an axiom or obtained by applying an inference rule to an element earlier in the sequence
    The last element in a derivation is a theorem.

    - This is a recursive definition.
        - Includes base clause
        - Inductive clause
        - Recursive clause

    (b) Ex.

        i. MI is a derivation
        ii. MIU by I on line 1.
        iii. MII by II on line 1.
        iv. MIUIU II on line 2.

    These are all theorems since they're the last element of a derivation.

    (c) Random theorems

    - MIIII
    - MIIIIU
    - MIUU
    - MIUUIUU
    - MIIUU
    - They all have something in common, all start with M

2. Reasoning Reason inside (M-mode)

    - Generate theorems "within" the formal system

- Can be done by a machine
- Object language

Outside a system (I-mode)

- Show properties of the system, reason about it
- Meta-language
  - All theorems start with M
  - Looking on the outside
  - When do we use other languages to describe another language
  - Using English to talk about programming
  - Using English to talk about Mandarin
  - If you speak English, then you're reasoning inside

3. Bijection with Natural Numbers MIU strings are countably infinite. You can construct a bijection like:

   (a) M
   (b) I
   (c) U
   (d) MM
   (e) MI
   (f) MU
   (g) II
   (h) IM
   (i) IU
   (j) UU
   (k) . . .

   ---

   MIU theorems are also countably infinite? Why?

   - Subset of MIU strings
   - Why not finite? Inference II, can keep expanding

4. Theorem All MIU-theorems begin with M.

   - This is a proof about the MIU-system, not within

(a) Proof By induction (strong induction) on the length of derivations: (number of steps to derive)

- Base case: Derivation of length 1: MI (good)
- Induction step:
  - IH: The claim holds for all derivations of length $< n$
  - Show: The claim holds for a derivation of length $n$
  - Line n is either an axiom or derived by rule I, II, III or IV.
- Case 1: Line n is an axiom: MI (you can write an axiom at any step, reverting back to MI)
- Case 2: Line n is derived from an earlier line (say $m < n$ ) by Rule I. By IH, the theorem in line $m$ begins with M. Rule I doesn't change the first letter, so it is also an $M$.
- Case 3:
- Case 4:
- Case 5:
- (DIY)

# 6 Lecture 6 *<2017-09-21 Thu>*

## 6.1 Quiz prep

- What is derivation?
- What is a theorem?
- How many infinite cardinalities are there?
- Can a set have the same cardinality as its power set?
- Is the empty set a subset of every set?
- How do you prove something is inductive?

## 6.2 Review

- Is U a MIU-theorem?
  - No, it doesn't start with M
- Is MU a MIU-theorem?

## 6.3 Decision Procedure

Guarantees a yes or a no answer in a finite amount of time

- A set/question that has a decision procedure is decidable
- If given 2 functions with inputs and outputs, can we tell if they're identical? Is it decidable?
  - No, infinite amount of inputs
- Decision procedure for getting someone's cellphone number?
  - Try all combinations until the phone rings, if no ring, no number
  - Not feasible, but we care what you can do in principal, as long as its finite
- Given a computer program
  - Can we decide if it terminates in 10 minutes?
    - Yes, just wait
  - Can we decide if it terminates in finite time?
    - No, if it doesn't stop, you'll never know

## 6.4 pq-system

- Alphabet: p, q, -
- Axiom(s): xp-qx-
  - What is x here? An arbitrary number of hyphens, meta-variable (used to describe system, not part of the system)
  - How many axioms? $\aleph_0$, x can be uncountably many
    - The written axiom is more like an axiom "template"
  - Is there a **decision procedure** to check if something is an axiom?
    - Yes, just count number of hyphens.
  - Axioms in a formal system **have to be decidable**
- IR: If xpyqz is a thm, then xpy-qz- is a thm
  - E.g. --*p*--*q*-----

### 6.4.1 Interpretation:

pq-system: p (plus) q (equals) - (1) -- (2) ---(3)

| plus | equals | 1 | 2 | 3 | (Semantics) | Math structure $\langle \mathbb{N}, +, = \rangle$ |
|---|---|---|---|---|---|---|
| p | q | -- | -- | --- | (syntax) | Typographical structure $\langle \{\text{-}, \text{--}, \text{---}\}, p, q \rangle$ |

- GEB: Calls this an Isomorphism
  - Misleading, because in mathematics, it's a structure preserving bijection
  - $\langle \mathbb{N}, + \rangle$ is isom $\langle Even, + \rangle$ by $f(x) = 2x$
  - $a + b = c \iff f(a) + f(b) = f(c)$

Are $\langle \mathbb{N}, + \rangle$ and $\langle \mathbb{N}, x \rangle$ isom?

- $a + b = c \iff f(a) \times f(b) = f(c)$
- $f(x) = 2^x$
  - $3 + 5 = 8 \to 2^3 \times 2^5 = 2^8$
  - Does this work? No. Not surjective.
  - Is there a bijection?

If an interpretation makes all axioms and thm true, it is called a model.

- Is our interpretation a model?
- Yes, argue by saying it makes axioms true and IR keeps it true.
  - E.g. --$p$--$q$-----
  - $2 + 3 = 5$

### 6.4.2 More Interpretations

Change p to times. It's still an interpretation, but not a model. It's false, as all axioms and theorems must be true.

- Keep p to plus, but change all dashes to negative integers. Is it also a model?
  - Yes. We can still have multiple models for

pq-system: p (equals) q (taken from) - (2) -- (4) ---(6)

- E.g. --*p*--*q*-----
- 6 = 4 taken from 10
- Still a model!
- Formal system can have many models, just depends on interpretation

### 6.5 geq-system:

- Thms:
  - - geq -
  - -- geq -
  - --- geq ---
- Model: geq -> $\geq$
  - - 1
  - -- 2
  - ...
- Soundness: Every **thm** in a formal system is **true** under an interpretation
- Completeness: Every **truth** in an interpretation is a **theorem**
- Soundness and completeness relate semantic and syntactic notions with each other
- Our interpretation is sound and complete
- Different interpretation, if we make geq -> =
  - Is it sound? No. Some theorems are true, but some, like -- geq - are not true
  - Is it complete? Yes. Every equality that can be expressed via this interpretation is a theorem.
  - Soundness and completeness are independent, so it was possible for us to get completeness and not soundness, but it's also possible to get the opposite <u>(think of an example)</u>

- model $\iff$ sound
- Syntax and semantics can be switched, which we'll see later
  - We'll be looking at formal systems of numbers/arithmetic that mean different things

### 6.6 Infinite Prime Numbers

Theorem: There are infinitely many prime numbers.

#### 6.6.1 Proof (by contradiction):

Assumption: There are finitely many prime numbers:

- $\{p_1, p_2, \ldots, p_n\}$
- So there is a greatest prime number, say $p_n$
- Define: $g = (p_1 \times p_2 \times \ldots \times p_n) + 1$
  - Is $g$ a prime number?
    - Case 1: Yes. Then $g > p_n$ ↯
    - Case 2: $g$ is not prime.
      - But then it must be divisible by some prime number.
      - But, it cannot be divisible by $p_1, p_2, \ldots, p_n$, as there will be a remainder of 1 ↯
- So, assumption is false.

## 7 Lecture 7 *<2017-09-26 Tue>*

### 7.1 Quiz review

- Try to use diagonalization on rational numbers?
  - Produced number may not be rational.
- Show 2 sets have the same cardinality
  - Prove there's a bijection
- Show a set has less than or the same amount of cardinality

  - Injection

- Adding something to a set of size $\aleph_0$
  - Size is still $\aleph_0$

- What is a derivation?
  - Sequence of formulas such that each element is an axiom or obtained from an axiom

- Theorem?
  - Last element in a derivation

- When is a formal system complete?
  - Every truth in the interpretation is a theorem

- When is a formal system sound?
  - Every theorem is a truth in the interpretation

## 7.2 Decision procedure

- What does it mean for a set to be decidable?
  - If it has a decision procedure.
    - Algorithm that gives a yes or no answer in a finite amount of time.

- A set $X$ is decidable if there is a decision procedure for it.
  - Characteristic function:
    -

$$C_x(n) = \begin{cases} 1 & \text{if } n \in X \\ 0 & \text{if } n \notin X \end{cases}$$

Later we will see that decidable set $\iff$ characteristic function computable.

## 7.3 FS for addition

Write down properties of addition to try and come up with a formal system to fit that interpretation.

Recursive definition of addition:

- $x + 1 = (x + 1)$
- $x + (n + 1) = (x + n) + 1$
  - This allows you to compute any addition
  - $x + 4 = (x + 3) + 1 = (x + 2) + 1 + 1 = (x + 1) + 1 + 1 + 1$

Now translating it to the pq system:

| Axiom | xp-qx- | x+1=(x+1) |
|---|---|---|
| IR | If xpyqz is a theorem then xpy-qz- | x+(n+1)= (x+n)+1 |

## 7.4 FS for multiplication

Want FS for multiplication (tq system) x t y q z

- $x \times 1 = x$
- $x \times (n + 1) = (x \times n) + x$

Translating:

- Axiom: *xt-qx* ($\aleph_0$ axioms)
- Inference Rule: If *xtyqz* is a thm, then $xt\overbrace{y\text{-}}^{n+1}qzx$

---

Modification of tq system above.

- Alphabet: t q - C
- Second Inference Rule: If *x-ty-qz* is a theorem then $Cz$ is a theorem. (x,y,z non-empty strings of -)
- Example theorems:
  - -t-q-
  - --t--q----

  - C----
  - ---t--q------
  - C------

Cx is true if $x$ is a composite number (not a prime)

- If Cx is not a Ctq theorem, then Px (prime)
  - Can we add this as another inference rule? No. Not saying how to get primes, just what isn't a prime. It's not an inference rule, you have to be able to apply an inference rule mechanically (has to be decidable)

### 7.5 Recursively enumerable set (r.e.)

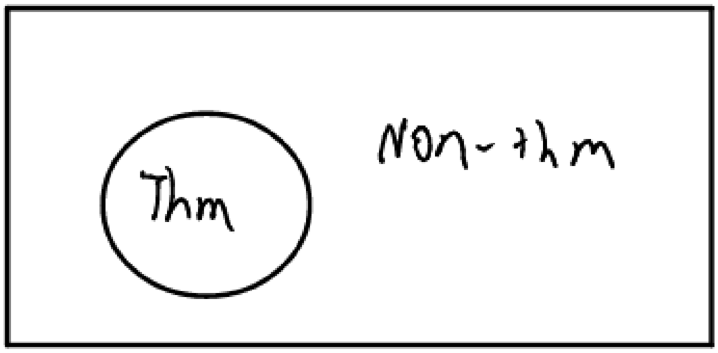
A recursively enumerable set can be generated as theorems of a formal system.

- Ex. Natural numbers

### 7.6 Recursive set

A set is recursive if it is r.e. and its complement is also r.e. Only want to talk about the complement in a clearly defined realm (universe).

- Well-formed expressions:



- In GEB, he calls the circle the figure and the ground the non-thms that are the non-thms of the circle but they are theorems themselves.

- Recursive sets are decidable. Why?
- Can a set be recursively enumerable but not recursive?

# 8 Lecture 8 *<2017-09-28 Thu>*

## 8.1 Theorems

$P$ is equivalent to $Q$ relative to $A_1 \ldots A_n$:

- $A_1 \ldots A_n, P$ prove $Q$
- $A_1 \ldots A_n, Q$ prove $P$

### 8.1.1 E.g.

Relative to Euclid's axioms:
Proclus' axiom

- If a line intersects 2 parallels it must intersect the other

Playfail's axiom

- If a line is parallel to a point, then there exists one parallel containing that point

Parallel postulate
Are equivalent.

- If you cannot prove $P$ from $A_1 \ldots A_n$ then $P$ is independent of $A_1 \ldots A_n$

  Parallel postulate is independent from the axioms of Euclid.

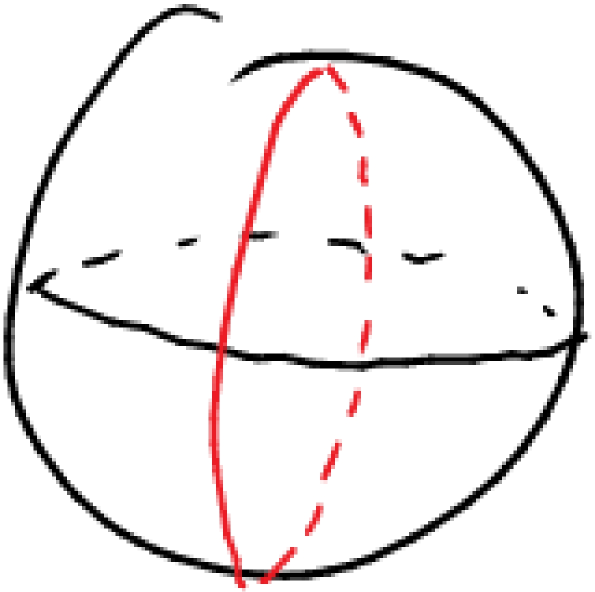
- One way to show:
  - Give a model for $A_1 \ldots A_n$ in which $P$ is false. (If a model makes $P$ false, then $P$ cannot be a theorem.)
- How to show that there are certain models for Euclid's axioms where the parallel postulate is false? Well, we have to come up with a model.

1. Playfail's axiom There exists exactly one parallel to a given line through a given point.
   - What would it mean for this to be false?

  - Playfair's axiom can be false in 2 ways:
    a) More than one parallel exists
    b) No parallel exists

b)

- Line -> great circle on a sphere



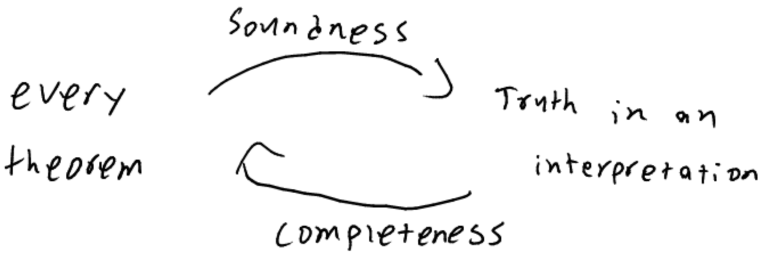
All great circles intersect, no parallels (Elliptic geometry)

- Point -> Point and it's antipode

a)

- Line -> line inside disc
- Point -> point inside the disk

Infinitely many parallel lines (Hyperbolic geom.)



## 8.2 PQ*-system

- Ax. schema 1: xp-qx-
- IR: xpyqz → xpy-qz-
- Ax. schema 2: xp-qx
- Interpretation:
  - p -> plus
  - q -> equals
  - - -> unit
- -p-q--
- Now, --p-q-- is a thm
- Meaning: 2+1 = 2, which is false.
- Complete but not sound with respect to interpretation 1 (p -> plus)
  - Different interpretation (2):
    - p -> plus
    - q -> greater or equal
    - - -> unit
    - Sound but not complete with respect to interpretation 2.
      - Ex. ---p--q- is not a theorem, but $3 + 1 \geq 1$ is a truth
      - Axiom schema 2 only gives you things greater by 1
  - Different interpreation (3):

  - ∗ p -> plus
  - ∗ q -> greater by 1 or equal
  - ∗ - -> unit
  - ∗ Sound and complete with respect to interpretation 3

## 8.3 Propositional Logic

- Today's presentation is harder for those who already know propositional logic, next week will be the standard presentation.

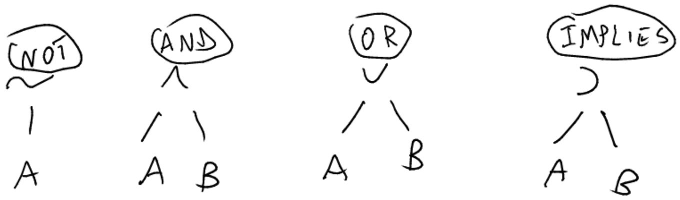
### 8.3.1 Formula trees:

- Language: Propositional variables: $P_0, P_1, P_2, \ldots$
  - Unary connective: $\sim$ (negation)
  - Binary connectives: $\wedge$ (conjunction)
    - $\vee$ (disjunction)
    - $\supset$ (implication)

1. Inductive Definition

   (a) Base clause: A prop. variable is a formula tree

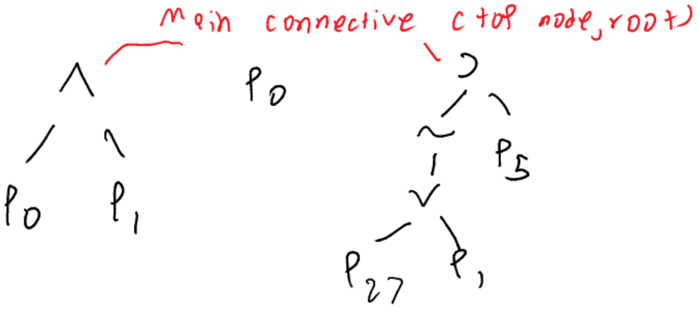
   (b) Inductive clauses: If $A, B$ are formula trees then

   are also formula trees (with A, B as subtrees)

   (c) Nothing else is a formula tree

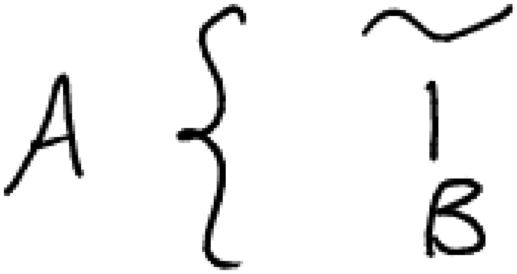
2. E.g.



The tree to the right has 5 subtrees (main connective is not a subtree of itself)

3. Truth value assignment A truth value assignment is a function from propositional variables to $\{T, F\}$ (True, False). The truth value of a formula tree A under the truth value assignment f is:

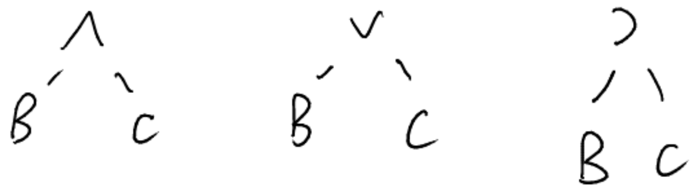
- Case 1: A is a propositional variable: f(A)
  - E.g. $f(P_0) = T$, $f(P_1) = F$, $f(P_{27}) = T$, $f(P_5) = F$
- Case 2: A is of the form:



  - Truth values:

| B | A |
|---|---|
| T | F |
| F | T |

- Case 3: A is of the form



| B | C | A$_1$ | A$_2$ | A$_3$ |
|---|---|---|---|---|
| T | T | T | T | T |
| T | F | F | T | F |
| F | T | F | F | T |
| F | F | F | F | T |

# 9 Lecture 9 *<2017-10-03 Tue>*

## 9.1 Propositional Logic

Most of this is on handout 2b.

Let's define logic as a formal system.

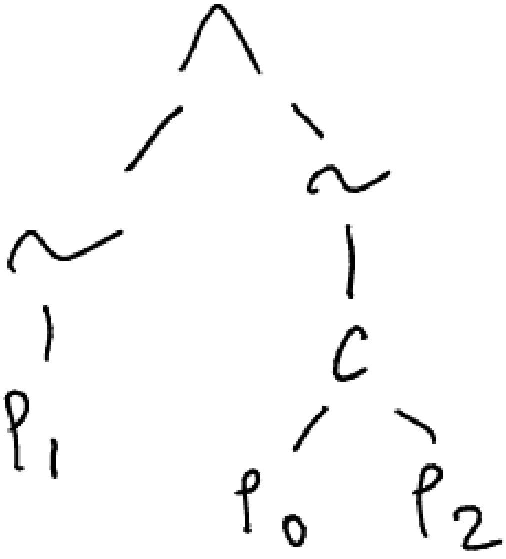
- Alphabet: $P_0, P_1, P_2, \ldots$ ($\aleph_0$ propositional variables)
- Connectives: $\wedge$ (conjunction), $\vee$ (disjunction), $\supset$ (implication), $\sim$ (negation)
- Parentheses

## 9.2 Well-formed formulas (wff)

1. Base clause: $P_i$ is a wff ($i \in \mathbb{N}$) (called atomic)
2. Inductive clause: If $A$ and $B$ are wffs, then so are:
   - $\sim A$ "not"
   - $(A \wedge B)$ "and"
   - $(A \vee B)$ "or"
   - $(A \supset B)$ "implies"
3. Nothing else is.

E.g.

- $P_0 \supset P_1$ is **not well formed**, lack of parentheses.
- $(P_0 \supset P_1)$
- $(P_{27})$ is **not well formed**, shouldn't have parentheses in atomic form.
- $(\sim P_1 \wedge \sim (P_0 \supset P_2))$
- Can be shown as:



Convention: outer parens are omitted (except if asked for a well formed expression explicitly, as parentheses are required to comply with rules that give us the nice structure)

### 9.2.1 Interpretation

Propositional variables → truth values:

| True | False |
|---|---|
| T | F |
| 1 | 0 |
| T | $\perp$ |

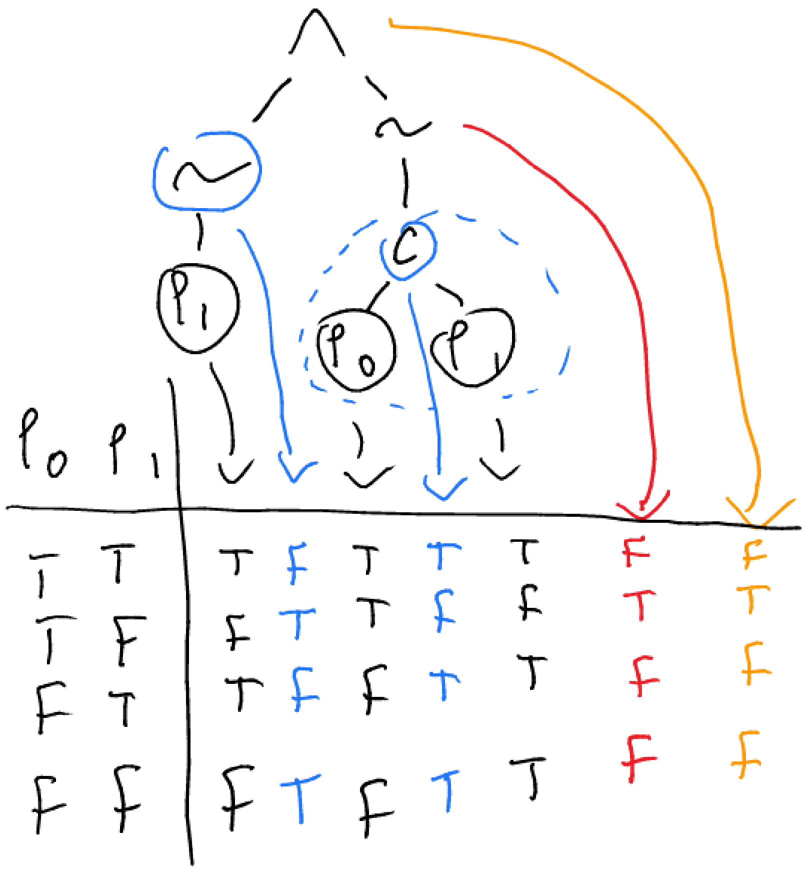
(Bivalence)

### 9.2.2 Truth tables

A,B (metavariables that stand for wff):

| A B | A ∧ B | A ∨ B | ∼ A | A ⊃ B |
|---|---|---|---|---|
| T T | T | T | F | T |
| T F | F | T | F | F |
| F T | F | T | T | T |
| F F | F | F | T | T |

- $A \supset B$ -> if . . . then . . .
  - A is the antecedent
  - B is the consequent
  - This is material implication, not causal implication
  - The light (B) can be on even if I didn't flip the switch (A)
- Ex.



| $P_0$ | $P_1$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| T | T | T | F | T | T | T | F | F |
| T | F | F | T | T | F | F | T | T |
| F | T | T | F | F | T | T | F | F |
| F | F | F | T | F | T | T | F | F |

| $P_1$ | $\wedge$ | $\sim$ | $P_1$ |
|---|---|---|---|
| T | F | F | T |
| F | F | T | F |

| $P_1$ | $\vee$ | $\sim$ | $P_1$ |
|---|---|---|---|
| T | T | F | T |
| F | T | T | F |

Something that is always true is a **tautology**.

- Two wff are logically equivalent if their TV agrees on all possible TV-assignments:

| A | B | $\mathrm{A} \supset \mathrm{B}$ | $\sim \mathrm{A} \vee \mathrm{B}$ |
|---|---|---|---|
| T | T | T | T |
| T | F | F | F |
| F | T | T | T |
| F | F | T | T |

Do we read $\sim A \vee B$ as $(\sim A) \vee B$ or $\sim (A \vee B)$?

- $(\sim A) \vee B$ due to the way we defined wff

Minimal sets of connectives:

- $\{\sim, \vee\}$
- $\{\sim, \wedge\}$
- $\{\sim, \supset\}$
- $\implies$ Sheffer-Stroke

For a wff with $n$ prop. vars, the truth table has $2^n$ lines.

Is finding out if a proposition is a tautology decidable or not? Yes, just write out the truth table.

### 9.2.3 Inferences

An inference is **valid** if it is impossible for all the premises to be true and the conclusion false at the same time.

| | | Premises | | Conclusion |
|---|---|---|---|---|
| A | B | A | $\mathrm{A} \supset \mathrm{B}$ | B |
| T | T | T | T | T |
| T | F | T | F | F |
| F | T | F | T | T |
| F | F | F | T | F |

This is a valid inference, when both premises are true, the conclusion is also true. Thus:

- $\mathrm{A}, \mathrm{A} \supset \mathrm{B} \models \mathrm{B}$
  - Where $\models$ is the (semantic) consequence
  - Can check if something is semantically implied by checking the truth table and when all premises are true.

Generalizing: $A_1, \ldots A_n \models B$

- $\models$ B (tautology)

### 9.2.4 Natural Deduction (Syntax)

Introduced by Gentzen, 1934.
$\frac{\text{Premises}}{\text{Conclusion}}$

- $\frac{A \ A \supset B}{B}$ ⊃ Elimination (Implication Elimination) or MODUS PONENS
- $\frac{A \ B}{A \wedge B}$∧ Introduction (since it introduces conjunction)
- $\frac{A \wedge B}{A}$∧ Elim
- $\frac{A \wedge B}{B}$∧ Elim
  - Not the same as the rule above! You cannot get to $B$ from the first one, you must use this one.
  - Also, $A \wedge B$ and $B \wedge A$ are not the same! They might have the same meaning, but they are different as strings, syntactically
- $\frac{A}{A \vee B}$∨ Intro
- $\frac{A}{B \vee A}$∨ Intro

Missing:

- ∨ Elim
- ⊃ Intro
- ∼ Intro
- ∼ Elim

# 10 Lecture 10 *<2017-10-05 Thu>*

## 10.1 Natural Deduction

- Proof system for propositional logic
- In the land of syntax when we do this
  - Remember that if something looks different, it is different
  - $A \wedge B$ is not the same as $B \wedge A$!

---

Some rules:

- $\frac{A \ A\supset B}{B} \supset Elim$
- $\frac{A \ B}{A\wedge B} \wedge Intro$
- $\frac{A\wedge B}{A} \wedge ElimR$
- $\frac{A\wedge B}{B} \wedge ElimL$
- $\frac{A}{A\vee B} \vee Intro$
- $\frac{A}{B\vee A} \vee Intro$

Note that $A, A \supset B \models B$ consists of semantics, not **syntax**. The above rules mentioned are rules to infer other things syntactically.

$\frac{\frac{A\wedge B}{B}\wedge ElimL \ \ \frac{A\wedge B}{A}\wedge ElimR}{B\wedge A} \wedge Intro$ can be abbreviated as: $A \wedge B \vdash_{ND} B \wedge A$ (ND stands for natural deduction, don't confuse this symbol with the semantic one)

- If you can get from $A$ to $B$, then you can box A (canceling this assumption $A$) with a subscript of the amount of steps.

$$\dfrac{\begin{array}{c}[A]_n\\ \vdots\\ B\end{array}}{A \supset B} \supset Intro_n$$

- $\underbrace{\dfrac{[A]_2 \ [B]_1}{\frac{B\supset A}{A\supset(B\supset A)} \supset Intro_2} \supset Intro_1}_{\vdash_{ND} A\supset(B\supset A)}$
  - A and B don't prefix $\vdash$ since they were eliminated (boxed)

  - You can get final result from no assumptions

- $\underbrace{\dfrac{A\ [B]_1}{B \supset A} \supset Intro_1}_{A \vdash_{ND} B \supset A}$

$$\dfrac{A \vee B \qquad \begin{matrix}[A]_1 \\ \vdots \\ C\end{matrix} \qquad \begin{matrix}[B]_1 \\ \vdots \\ C\end{matrix}}{C} \vee Elim_1$$

- A and B both have subscript 1 because they're eliminated at the same time
- This is like a formal definition for a proof by cases

### 10.2 Exercise

Prove: $A \vee C, A \supset B, C \supset D \vdash_{ND} B \vee D$

Assumptions

$$\dfrac{A \vee C \qquad \dfrac{\dfrac{[A]_2 \quad A \supset B}{B} \supset Elim}{B \vee D} \vee Intro\ R \qquad \dfrac{\dfrac{[C]_2 \quad C \supset D}{D} \supset Elim}{B \vee D} \vee Intro\ L}{B \vee D} \vee Elim_2$$

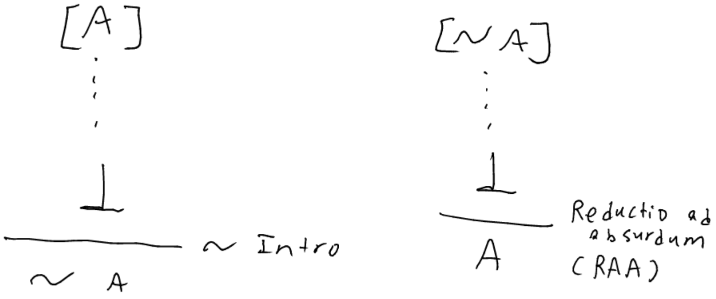
✓

---

- New symbol: $\perp$ false/falsum
- Generated by:

  - $\frac{A \ \sim A}{\bot} \sim Elim$

- Can use to:
  - $\frac{\bot}{A} Exfalsum$ (can introduce anything)



-

  RAA is a proof by contradiction.

Prove: $\vdash_{ND} \sim A \supset (A \supset B)$

# 11 Lecture 11 *<2017-10-10 Tue>*

## 11.1 Recap

- A, A ⊃ B ⊨ B
  - It is impossible for $B$ to be false and $A$ and $A \supset B$ to be true.
- A, A ⊃ B $\vdash_{ND}$ B
  - $\frac{A \ A \supset B}{B} \supset E$
  - How many rules in natural deduction system? About 10. How many axioms? None.

## 11.2 Axiomatic Proof System

- IR: From A and A ⊃ B, infer $B$ (MODUS PONENS)
- Substitution rule: Any wff can be substituted for A, B, C
- Axiom schemata: (has meta variables, infinitely many axioms. Difference between axioms is important, good quiz question)

1. $\sim A \supset (A \supset B)$
2. $B \supset (A \supset B)$
3. $(A \supset B) \supset ((\sim A \supset B) \supset B)$
4. $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$
5. $(A \supset (B \supset (A \wedge B)))$
6. $(A \wedge B) \supset A$
7. $(A \wedge B) \supset B$
8. $A \supset (A \vee B)$
9. $B \supset (A \vee B)$
10. $((A \vee B) \wedge \sim A) \supset B$

- Don't need to memorize these axioms or the natural deduction rules. Will be given on an exam if required.

- A proof of $A$ from assumptions $A_1, \ldots, A_n$ is a sequence of wffs such that each element of the sequence is:

  1. An instance of an axiom
  2. An assumption (since it's a proof from assumptions)
  3. The result of MP (the inference rule) to earlier elements in the sequence

- 1 & 2 are base clauses, 3 is the inductive clause. This is a recursive definition. Since proofs are defined inductively, we can prove things about proofs using induction.

Prove: A $\supset$ A $\vdash_{AX}$ B $\supset (A \supset A)$

1. $A \supset A$ Assumption

2. $(A \supset A) \supset (B \supset (A \supset A))$ $Ax2[(A \supset A/B, B/A)]$ $(A \supset A)$ replaces B in the axiom, $B$ replaces $A$ in the axiom.

3. $B \supset (A \supset A)$ MP on 1 and 2

Prove: $A \supset (B \supset C) \vdash_{AX} (A \supset B) \supset (A \supset C)$

1. $A \supset (B \supset C)$ Assumption

2. $(A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))$ Ax 4

3. $(A \supset B) \supset (A \supset C)$ MP 1,2

## 11.3 GEB proof system

### 11.3.1 Ex.

$\vdash_{ND} (A \land B) \supset (B \land A)$

- $\frac{\frac{A \land B}{B} \land Elim \quad \frac{A \land B}{A} \land Elim}{\frac{B \land A}{(A \land B) \supset (B \land A)} \supset Intro} \land Intro$

$\vdash_{GEB} (A \land B) \supset (B \land A)$ [ push

- $A \land B$ ass
- $A$ sep
- $B$ sep
- $B \land A$ joining

] pop $(A \land B) \supset (B \land A)$

- Won't go over GEB system in class, not as interesting, but you can look it over in handout or in GEB.

## 11.4 Relations of systems

How do these 3 systems relate to each other?

What we have so far:

1. Semantic consequence: $\Gamma \models A$, read as $\Gamma$ entails $A$ or $A$ is a logical consequence of $\Gamma$
   - Check via truth tables
2. Syntactic: $\Gamma \vdash A$, read as $\Gamma$ proves $A$ or $A$ is derivable from $\Gamma$
   - Check via natural deduction
   - or axiomatic proof
   - or GEB proof
   - All 3 are equivalent

To show that the natural deduction system and the axiomatic system are equivalent (proofs in one system can be acquired in the other system):

1. Prove all axioms from the natural deduction rules. (Modus Ponens is an inference rule in natural deduction)

2. Show that each natural deduction rule corresponds to an axiomatic proof

---

Proving axiomatic axioms:

1. $\vdash \sim A \supset (A \supset B)$

$$\dfrac{\dfrac{\dfrac{\dfrac{[\sim A]_1, \quad [A_2]}{\bot}\ \bot\text{ Intro}}{B}\ \text{Ex falso}}{A \supset B}\ \supset\text{ Intro } 2}{\sim A \supset (A \supset B)}\ \supset\text{Intro}_1$$

$$\dfrac{\dfrac{[A]_1, \quad [B]_2}{A \supset B}\ \supset\text{Intro } 1}{B \supset (A \supset B)}\ \supset\text{Intro}_2$$

2.

You can do this for all axioms and you'll get that they're equivalent. The fact that we know they're equivalent now allows us to use which one we prefer and go from one to the other, you won't lose any logical power.

---

How does the semantic consequence relate to the syntactic consequence?

### 11.4.1 Soundness Theorem

If $\Gamma$ is a set of wff and $A$ wff then:

- If $\Gamma \vdash A$, then $\Gamma \models A$

- To show, show that all axioms are tautologies. We already know that modus ponens is a semantic consequence as well.

We also have if $\vdash A$, then $\models A$, i.e. if derivations with no or closed assumptions, then tautology, via soundness theorem.

### 11.4.2 Completeness Theorem

If $\Gamma \models A$, then $\Gamma \vdash A$

- If $A$ is a logical consequence of $\Gamma$, then there is also a proof of $A$ from $\Gamma$

Proof of these theorems done in PHIL 310 and MATH 318

if $\models A$, then $\vdash A$, tautologies can be derivations with no or closed assumptions

# 12 Lecture 12 <*2017-10-12 Thu*>

- There will be a proof question on the quiz (natural deduction/axiomatic)
  - You will be given the rules, don't need to memorize
- Most important thing is terminology

## 12.1 Recap

- Propositional logic
  - Syntax (formula trees):
    - Language
    - (Inference) rules
    - Proof-systems
  - Semantics:
    - Truth tables
  - Soundness and completeness theorem (connects them)
- Now we will be doing the same thing (structurally), but we'll be doing first order logic
  - No truth tables, but structures and interpretations instead

## 12.2 First Order Logic (FOL)

Language $\mathfrak{L}$ of FOL:

1. Variables: $X_0, X_1, X_2, \ldots$
2. n-ary (can have $n$ arguments) function symbols: $f_0, f_1, f_2, \ldots$
   - 0-ary fun symbols: constant, don't take in any arguments
3. Predicate symbols (n-ary): $P_0, P_1, P_2$
4. Pred. symbol $=$
5. Propositional connectives: $\wedge, \vee, \supset, \sim, \perp$
6. Quantifiers: $\forall$ ("for all", universal), $\exists$ ("exists", existential)
7. Parentheses: ()

2 and 3 are called extra-logical, since they aren't fixed, need to be given meaning, must be interpreted when we give the semantics.

Expressions:

| Prop. | FOL |
|---|---|
| wff - T/F | wff - T/F |
| | Terms - like names |

### 12.2.1 Term

Def. A term is

1. A variable is a term
2. If $f$ is an n-ary fun symbol, and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.
3. Nothing else.

Sometimes write $t_1 f t_2$ for $f(t_1, t_2)$ (infix notation for binary function symbols)

### 12.2.2 wff

A wf-formula

1. If $P$ is an n-ary predicate symbol, $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ is a wff. (atomic formula)
   - In particular: $t_1 = t_2 = (t_1, t_2)$ is wff
2. If $A, B$ are wff then $\sim A, (A \vee B), (A \wedge B), (A \supset B)$ (we see that propositional logic is built into first order logic)
3. If $A$ is a wff, x a variable, then
   - $(\forall x : A)$ and $(\exists x : A)$ are wff. Here, $A$ is called the scope of the quantifier.
4. Nothing else.

### 12.2.3 First Order Arithmetic (FOA)

- Going to be using this a lot

To use FOL, you must specify the language $\mathfrak{L}$ which describes the function symbols.

1. Syntax
   - Fun symbols:

| Arity | 0 | 1 | 2 | 2 |
|---|---|---|---|---|
| | 0 | S | $+$ | $x$ |
| | Zero symbol | Successor symbol | Plus | Times |
| | Constant | S(x) | $t + t$ | $t \times t$ |

   - Terms: Ex: $s(s(0)), x_1, 0, +(x_1, x_2), 0 \times S(0)$
   - Formulas: $x_1 = x_2$ (Since a formula involves a predicate symbol and the only predicate symbol we have is =)
     - $0 = (x_1 + x_2)$
     - $\sim (0 = (x_1 + x_2))$
     - $(x_1 = x_2) \supset \sim (0 = (x_1 + x_2))$
     - $(\forall x_1 : x_1 = 0)$
     - $(\exists x_1 : x_1 = 0)$

    - These can all be true or false depending on what we're talking about

2. Semantics Structure = <universe of discourse; designated elements from UD (constants); n-ary operations (fun symbols); n-ary relations (predicate symbols)>

    - Structure depends on language you want to interpret. If your language has no constants, you won't need designated elements. If you have 5 fun symbols, you'll need 5 n-ary operations, etc.
    - An interpretation of a language $\mathfrak{L}$, is a structure and a mapping $\sigma$ of the extra-logical symbols into the structure.
    - An interpretation of FOA:
        - $\mathfrak{n}$ = <natural numbers; zero; successor; addition; multiplication> (natural number structure, how it was made to be interpreted. But you can interpret it differently. You can swap the mapping, but make sure they have the same arity: can switch addition with multiplication, but not successor with zero)
        - $\mathfrak{L}$: 0 s + ×
        - Map: 0 -> 0, s -> successor, + -> addition(+), × -> multiplication (×)
            - Make sure to understand the difference between the semantic level and the syntactic level. They look the same, one is the symbol and one is the meaning. One is the symbol for addition, but the other means addition.
            - Why don't we have n-ary relations? Because FOA has no predicate symbols. Shows how structure is dependent on language.
            - $\times^\sigma = \sigma(\times) =$ mult, shows what × maps to in mapping $\sigma$

3. Truth in a structure (Tarski 1933)

    (T1) If x is a variable, then $x^\sigma$ is given by the valuation (interpretation that also interprets the variables, i.e. map $x_1$, $x_2$, etc.).

    (T2) If f is an n-ary function symbol, $t_1, \ldots, t_n$ terms:

    - $(f(t_1, \ldots, t_n))^\sigma = f^\sigma(t_1^\sigma, \ldots, t_n^\sigma)$
        - e.g. $[s(s(0))]^\sigma = s^\sigma([s(0)]^\sigma) = s^\sigma(s^\sigma(0^\sigma))$

(F1) $P$ nary predicate symbol, $t_1, \dots, t_n$ terms

$$(P(t_1, \dots, t_n))^\sigma = \begin{cases} T & \text{if } P^\sigma \text{ holds of } t_n^\sigma, \dots, t_n^\sigma \\ F & \text{otherwise} \end{cases}$$

(F2)

$$(\sim A)^\sigma = \begin{cases} T & \text{if } A^\sigma = F \\ F & \text{otherwise} \end{cases}$$

(F3)

$$(A \supset B)^\sigma = \begin{cases} T & \text{if } A^\sigma = F \text{ or } B^\sigma = T \\ F & \text{if } A^\sigma = T \text{ and } B^\sigma = F \end{cases}$$

(F4)

$$(\forall x : A)^\sigma = \begin{cases} T & A^{\sigma(u/x)} = T \text{ for all } u \in UD \\ F & \text{otherwise} \end{cases}$$

# 13 Lecture 13 *<2017-10-17 Tue>*

## 13.1 Quiz Review

- Inductive definition of Strings
- Prime numbers recursive? Yes
- Tautology, semantic
- Proof system, syntactic
- Derivation, syntactic
- Truth table, semantic
- Inference rule, syntactic
- How do you know something can be proved by something? Use Completeness theorem and check truth table

## 13.2 Tarski & Quantifiers

$$(\forall x : A)^\sigma = \begin{cases} T & A^{\sigma(u/x)} = T \text{ for every } u \in UD \\ F & \text{otherwise} \end{cases}$$

$$(\forall x :\sim (0 = Sx))^{\sigma(u/x)} = T \text{ for every } u \in \mathbb{N}$$

How to check this?

Decompose

$$(\sim (0 = Sx))^{\sigma(u/x)} = T \text{ for every } u \in \mathbb{N}$$
$$(0 = Sx)^{\sigma(u/x)} = F \text{ for every } u \in \mathbb{N}$$
$$(\mathbf{0} = \mathbf{succ\ u}) = F \text{ for every } u \in \mathbb{N}$$

False for every $u \in \mathbb{N}$, $succ0 \neq 0, succ1 \neq 0, \ldots$

Can then define the following:

$$\exists x : A \iff \sim \forall x :\sim A$$

Read as "there exist an x . . . "

If one wanted to define it formally like the universal quantifier:

$$(\exists x : A)^\sigma = \begin{cases} T & \text{if } A^{\sigma(u/x)} = T \text{ for at least one } u \in UD \\ F & \text{otherwise} \end{cases}$$

$$\forall x : \exists y : M(x, y)$$

Structure for interpretation: $<$ all people ; is mother of$>$

- For all $x$, there is a y, such that x is the mother of y.
  - This is false, not everyone is a mother.
  - How can we make it true?

$$\forall x : \exists y : M(y, x)$$

- Everybody has a mother, true

What does the following mean?

$$\exists x : \forall y : M(x, y)$$

- There exists somebody who is the mother of everybody

- False

$$\exists x : \forall y : M(y, x)$$

- There exists someone such that everyone is their mother

We see that just switching around quantifiers changes meaning completely. So we must read quantifiers correctly, from left to right.

- We can say:
  - $\overbrace{\sigma}^{\text{valuation}}$ $\underline{\text{satisfies}}$ $\overbrace{A}^{\text{wff}}$ : $A^\sigma = T$
  - $\overbrace{\sigma}^{\text{valuation}}$ $\underline{\text{satisfies}}$ $\overbrace{\Gamma}^{\text{set of wff}}$ : $A^\sigma = T$ for all $A \in \Gamma$
- A is $\underline{\text{logically true}}$: A is satisfied by every valuation
  - $\models A$
- A is a $\underline{\text{logical consequence}}$ of $\Gamma$: every valuation that satisfies $\Gamma$ also satisfies $A$
  - $\Gamma \models A$
- $\Gamma$ unsatisfiable, no valuation that satisfies each $A \in \Gamma$ at the same time
- These are similar to what was said in propositional logic, but with different terms

This was all in the land of semantics (we talked about truth). Now we'll go back to the land of syntax.

### 13.3 Syntax

Variable $x$ is bound if it falls in the scope of a quantifier that has $x$ as a variable of quantification. $x$ is quantified in $A$.

$$\forall x : \underbrace{\overbrace{A}^{x \text{ is bound in } A}}_{\text{scope}}$$

$$\forall x_1 : \exists x_2 : (x_1 = x_2 + x_3)$$

- $x_1$ and $x_2$ are bound variables, while $x_3$ is a free variable.

$$(\forall x_1 : \exists x_2 : (x_1 = x_2 + x_3)) \vee (\underbrace{x_1}_{\text{free}} = 0)$$

The second $x_1$ is free since it's not in the scope of the quantifier.

$$\forall x_1 : \exists x_2 : (x_1 = x_2 + x_3) \vee (\underbrace{x_1}_{\text{free}} = 0)$$

In this case, both $x_1$ are bound.

# 14 Lecture 14 *<2017-10-19 Thu>*

- Good midterm question: What does $\Gamma \models A$ mean in one sentence?
  - $\Gamma$ is a set of wff and $A$ is a wff. $\Gamma \models A$ if everything in $\Gamma$ is true then $A$ is true or it is impossible that everything in $\Gamma$ is true and $A$ isn't
- $\Gamma \vdash A$
  - There's a proof of $A$ from $\Gamma$
- $\Gamma \models A$ (What does this mean in FOL?)
  - $\Gamma$ entails $A$
  - Every valuation that satisfies $\Gamma$ satisfies $A$
    - A lot harder to check than propositional logic, since you have to check **every** valuation

## 14.1 FOA

If UD = $\mathbb{N}$, Even(x) $\implies$ even

- What does: $\exists x \; Even(x)$ mean?
  - There exists at least one number that is even.
- $\exists x \exists y \; Even(x) \wedge Even(y)$
  - There exist a $x$ and $y$ such that $x$ is even and $y$ is even
  - There exists at least an even number ($x$ and $y$ can be the same)
- $\exists x \exists y \; (Even(x) \wedge Even(y)) \wedge \sim (x = y)$

  - There exists at least two even numbers

- $\exists x \exists y \; (Even(x) \wedge Even(y)) \supset\sim (x = y)$

  - True, since it's exists and we can pick our $x$ and $y$

- $\forall x : \forall y : x = y$

  - For all $x$ and for all $y$, $x = y$
  - False.
  - We can make this true if we use a universe of discourse with one element, such as $UD = \{A\}$

- $\forall x : \forall y :\sim (x = y) \supset [Even(x) \vee Even(y)]$

  - For 2 numbers that aren't equal, one of them is even
  - False

- $\forall x : \forall y :\sim (x = y) \supset [Even(x) \vee Even(y)]$

  - If two numbers aren't equal, then one is greater than the other
  - True

### 14.2 Substitution

Recall, a term is either a variable or a function symbol followed by the appropriate number of terms.

- A term is closed if it contains no variables.

  - Ex. $0, S0$ in FOA

- Otherwise it's called open

- Sentence: wff with no free variables, $0 = 0$, $\forall x_1 : \exists x_2 : x_1 = x_2$

- A term $\underline{t}$ is free for an occurrence of $\underline{x}$ in $\underline{A}$ iff

  - $x$ is a free variable in $A$
  - And $x$ does not lie within the scope of any quantifier, where $y$ is a variable that appears in $t$

- What do we want to do? Replace variables by terms.

Substitution:

- $s, t$ terms, $x$ var
- $x(t/x) = t$
  - Term $t$ replaces variable $x$
- $y(t/x) = y$
- $f(s_1, \ldots, s_n)(t/x) = f(s_1(t/x), \ldots, s_n(t/x))$
  - Since the $s_i$ are terms, we have to do the substitution on each term
- $\forall x : P(x)(t/x)$
  - We cannot do this, $x$ is not a free variable
  - It isn't $\forall t : P(t)$, because you cannot use a quantifier on a term, a term can be a constant.
- $\forall y : \underbrace{x}_{\text{free}} = y(y/x) \to \forall y : \underbrace{y}_{\text{bound}} = y$
  - These are not equivalent
  - Might end up making a false statement true, want to avoid this.
  - Don't want to switch from free to bound.

## 14.3 Proofs

Now we need a proof system for First Order Logic. Will see natural deduction in class, read about axiomatic and GEB on the handout.

### 14.3.1 Natural deduction calculus

We need introduction and elimination rules for quantifiers.

$$\forall \; Intro : \frac{\text{A(a/x)}}{\forall x : A}$$

Here, $a$ is a variable/Eigen variable.

- e.g.

$$\frac{\text{0=a}}{\forall x : (0 = x)}$$

$a$ must be free for $x$ in $A$, cannot bind it.

$$\forall\ Elim\ \frac{\forall x : A}{\text{A(t/x)}}$$

$t$ must be free for $x$ in $A$

- e.g.

$$\frac{\forall x : (x = x)}{s0 = s0}$$

$$\exists\ Intro\ \frac{\text{A(t/k)}}{\exists x : A}$$

- E.g.

$$\frac{\overbrace{\sim (0 = S0)}^{A(0/x)}}{\exists x : \underbrace{\sim (x = S0)}_{=A}}\ t = 0,\ A(x) \sim (x = S0)$$

$A(0/x) \implies \sim (O = S0)$

$$\exists\ Elim\ \frac{\exists x : A \qquad \dfrac{\dfrac{[\text{A(a/x)}]}{\vdots}}{\dfrac{\vdots}{\text{C}}}}{\text{C}}$$

$$\frac{\dfrac{\forall x : A(x)}{A(t)}\ \forall\ Elim}{\exists x : A(x)}\ \exists\ Intro$$

For all $x$, $A(x)$ means that there exists $x$ such that $A(x)$

1. Ex $\vdash \forall x : (A(x) \supset B) \supset (\exists x : A(x) \supset B)$

$$\cfrac{\cfrac{\cfrac{[\exists x : A(x)]_1 \qquad \cfrac{\cfrac{\forall x : A(x) \supset B}{A(t) \supset B}\,\forall\ \mathit{Elim} \qquad A(t)}{B}\supset\ \mathit{Elim}}{B}\,\exists\ \mathit{Elim}}{\exists x : A(x) \supset B}\supset\ \mathit{Intro}_1}{\forall x : (A(x) \supset B) \supset (\exists x : A(x) \supset B)}\supset\ \mathit{Intro}_2$$

Complicated proof, won't have something like this on midterm (since just introduced these rules), but maybe final.

2. FOL: Sound & Complete Proved by Godel in 1930

# 15 Lecture 15 *<2017-10-24 Tue>*

Reviewed quiz 2.

- Recursive Definition
    1. Base clause
    2. Inductive clause
    3. Final clause
- Recursive

Means that both the set and its complement are recursively enumerable (we can derive them as theorems in a proof system)

- Terms

Either a variable or an n-ary function with $n$ terms

- Atomic wff

Formula made up of only terms and predicates

- Decision procedure to see if $\Gamma \models A$
    - Make a truth table, see if when all of $\Gamma$ is true, then $A$ must be true

# 16 Lecture 16 *<2017-10-31 Tue>*

## 16.1 Typographic Number Theory (TNT)/Dedekind Peano Arithmetic (DPA)/FOA

Why were people so concerned with the consistency of this?

Say we start with geometry

- -> $\mathbb{R} \times \mathbb{R}$ (cartesian)
  - But then how do we know that real numbers are consistent?
  - -> $\mathbb{Q} + (sets)$
    - But then what about the rational numbers?
    - -> $\mathbb{N}$
      - How do we show the consistency of natural numbers and therefore the consistency of mathematics?
      - -> Logic (FREGE, invented predicate logic on the way)
      - -> Set theory (Cantor)

Hilbert's program (1920s):

- Blob with everything mathematicians do
  - Arithmetic -> Formalize: FOA
  - Analysis ->
  - Set Theory -> Formalize: ZFC

Used finitary arithmetic to show consistency of all 3

- Can look at finite elements to prove something about the other domains that are infinite
- Difference between DPA and FOA?
  - There's no difference! Just different names.

---

TNT:

- Language: 0, S, +, ×
- Axioms:
  1. $\forall x :\sim (Sx = 0)$, For all numbers successor of $x$ is not 0

2. $\forall x : \forall y : (Sx = Sy) \supset x = y$, succ is injective
3. $\forall x : (x + 0) = x$
4. $\forall x : \forall y : (x + Sy) = S(x + y)$
5. $\forall x : (x \cdot 0) = 0$
6. $\forall x : \forall y : (x \cdot Sy) = ((x \cot y) + x)$

### 16.1.1 Theorems

A theory $T$ is said to be wff, s.t. $s \in T$ iff $T \vdash S$ (closed under deduction)

- A theory $T$ is axiomatic if there exists a recursive set $A_k$ st
- $T = \{S | A \vdash S\}$
- A theorem is consistent if it does not contain both $A$ and $\sim A$ for some wff $A$.

$$\begin{tikzcd}
\Gamma \vdash A \arrow[r, bend right, "\text{soundness}"'] & \Gamma \vDash A \arrow[l, bend right, "\text{completeness}"']
\end{tikzcd}$$

A theory $T$ is complete, if for every wff $A$, either $A \in T$ or $\sim A \in T$. Both can be true, but then it'll be complete and inconsistent.

Can we come up with something not derivable from a theory?

- (Axioms 1.-6.) $\nvdash \forall x :\sim (Sx = x)$
  - True in standard interpretation
  - Claim: Cannot derive from these axioms
  - How do we show this?
  - How do we show $\Gamma \nvDash A$?
    - We're looking for an interpretation. Want one that satisfies $\Gamma$ but does not satisfy $A$.

    - ∗ Recall that $\Gamma \not\models A$ means that every interpretation that satisfies $\Gamma$ must satisfy $A$ (no truth tables as we're no longer in propositional logic)
    - ∗ So $\Gamma \not\models A \implies \Gamma \nvdash A$ from soundness (opposite direction because of negation)

- UD: $\mathbb{N} \cup \{*\}$
  - 0 -> zero
  - + -> $+^*$
    - ∗ $n \in \mathbb{N}$

| $+^*$ | $n$ | $*$ |
|---|---|---|
| $n$ | $n + n$ | $*$ |
| $*$ | $*$ | $*$ |

  - × -> $\times^*$
    - ∗ $n \in \mathbb{N}$

| $+^*$ | 0 | $n$ | $*$ |
|---|---|---|---|
| 0 | 0 | 0 | $*$ |
| $n$ | 0 | $n \times n$ | $*$ |
| $*$ | 0 | $*$ | $*$ |

  - s -> $s^*$
    - ∗ $s^*(n) = s(n)$ if $n \in \mathbb{N}$
    - ∗ $s^*(*) = *$

$$\mathfrak{n} = \langle \mathbb{N}; 0, succ, +, \times \rangle$$

$$\mathfrak{n}' = \langle \mathbb{N} \cup \{*\}; 0, s^*, +^*, \times^* \rangle$$

Looking at the axioms:

- Is 2 true in both? Yes.
- Checking them all you will see that all the axioms are true in $\mathfrak{n}'$ as well.
- But (1.-6.) $\nvdash \forall x :\sim (sx = x)$ is not true in the new interpretation (because $s^*(*) = *$), when it is true in the old one. Which is why these axioms are insufficient to prove things as simple as this.

So these are in fact not all the axioms of TNT.

### 16.1.2 $\omega - incomplete$

- (1.-6.) $\nvdash (0 + 0 = 0)$
  - $\ldots (0 + s0 = s0)$
  - $\ldots (0 + ss0 = ss0)$
- But $\nvdash \forall x : (0 + x = x)$ (not the same as axiom 3)

This is called $\omega - incomplete$
Need an additional axiom
7.Axiom schema of induction

$$A(0) \supset [\forall x : (A(x) \supset A(sx)) \supset \forall x : A(x)]$$

Called a schema because it can make countably infinite amount of axioms
Now with all 6 axioms and the axiom schema we have TNT.

# 17 Lecture 17 *<2017-11-02 Thu>*

## 17.1 MIU

Is MU a MIU theorem? Recall:
I. xI -> xIU
II. Mx -> Mxx
III. III -> U
IV. UU -> ∅
Axiom: MI

- If it is a theorem, then you can prove it.
- If it isn't though, you can do MIU proofs for the rest of your life and never know the answer
- One way to answer this is find properties of these theorems

### 17.1.1 Theorem

The number of I's in a MIU theorem is not a multiple of 3 (and 0 is a multiple of 3, so if this follows then MU is not a theorem)

### 17.1.2 Proof by strong induction

- Let $t$ be a theorem, obtained from a proof of length $n$ (theorem at line $n$ of our proof)
- IH. The I-count of all theorems in lines $1-(n-1)$ is not a multiple of 3
- Show: Claim holds for line n

t can be:

1. An axiom: MI: 1 is not a multiple of 3
2. $t$ is obtained by Rule I from theorem $s$ $(1 \leq s < n)$. By IH. the I-count of $s$ is not a multiple of 3. So the I-count of $t$ is also not a multiple of 3.
3. $t$ is obtained by Rule II . . .
4. $t$ is obtained by Rule III . . .
5. $t$ is obtained by Rule IV . . .

Conclusion: The claim holds.

So we get the answer, to is MU a MIU-theorem -> No, because 0 is a multiple of 3. This here is a proof about proofs, which we'll be seeing a bit more of later with proofs about proofs in the TNT system.

## 17.2 G\"odel numbering

Continuing with the MIU system:

- We have the symbols
  - M->3
  - I->1
  - U->0

Now we can use these proofs for arbitrary strings

| | | | |
|---|---|---|---|
| 1. | MI | Axiom | 31 (Godel #) |
| 2. | MII | Rule II | 311 |
| 3: | MUI | Rule II | 3111 |
| 4. | MUI | Rule III | 301 |
| 5. | MUIU | Rule I | 3010 |

5 can be represented as -> a) x1 -> x10 (typographical) ; b) A number whose remainder, when divided by 10 is 1, can be multiplied by 10.

- Two ways of talking about it once we transform it into a number.

So G\"odel Numbering allows us to go from a formal system to number theory.

| Formal System -> | Number Theory |
|---|---|
| MIU | |
| MIU-thm | MIU-producible numbers |

MIU-producible numbers are recursively enumerable, since we can get them from the MIU formal system

But a more interesting transition is going from Number theory to FOA/TNT, another formal system.

| Number Theory -> | FOA/TNT (formal system) |
|---|---|
| e.g. 2 -> | SSO |
| MIU-producible numbers | theorems |

| FS | Number Theory | FOA |
|---|---|---|
| MU is a MIU-thm | 30 is a MIU-prod. number | $MON(\overbrace{SS\ldots S}^{30}0)$ is a TNT-theorem |

MON is a predicate standing for the MIU-producible numbers. Looks something like $\forall x : \exists y : \ldots$. Can look specifically at one system to show it for another. Like a translation, but nothing is lost.

Now let's start with the TNT system:

| Formal System | Number Theory | FOA |
|---|---|---|
| TNT | | |
| TNT-thm | TNT-producible numbers | theorems |
| _ is a TNT-thm | _ is a TNT-prod.number | _ |

How do we talk about the English language? "short" is short.

- TNT-formula: $\forall a :\sim Sa = 0$ is a theorem of TNT
- Go #: 626 262 636 223 123 262 111 666 is a TNT number -> $F(SS\ldots S0)$ is a TNT-thm
- -> $\sim F(G)$ is a TNT-thm -> says that "I am not a TNT-thm"

Can go from TNT to number theory to TNT. Reaching something different in the end will lead us to G\"odel's theorem

In a TNT proof, every line can be mapped to a Godel # (since you can translate formulas into numbers) and then you can get a unique Godel number for an entire proof

TNT-PROOF-PAIR(x,y): Predicate between a pair of numbers. True if x codes (is the godel number of) a proof of the wff coded by y

TNT-PROOF(x): $\exists y : TNT - PROOF - PAIR(y, x)$

# 18 Lecture 18 *<2017-11-07 Tue>*

## 18.1 Turing Machines

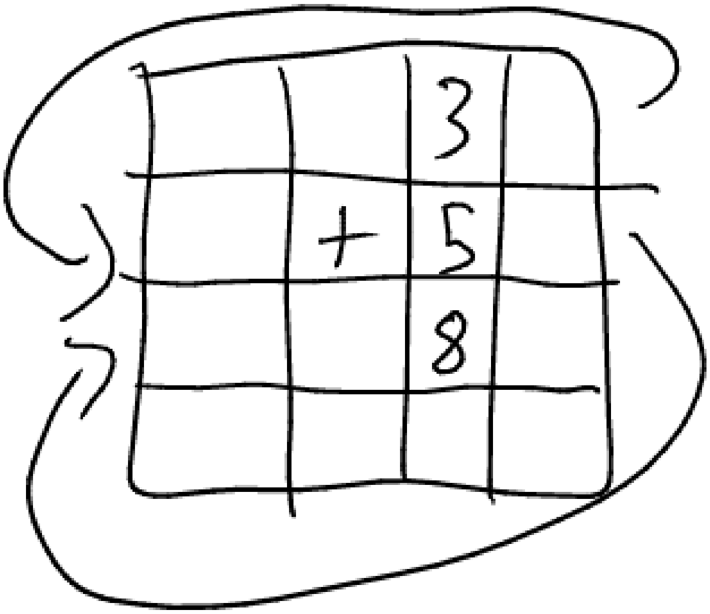
- Formalizations of computability
  - What is computable? Need some sort of handle on studying this mention, formalizing is a good way to do so, you can then check if something is valid in the formal system or not valid. Can also meta-reason about them and such.
- Different formal systems for computability have been made
  1. Representability in FOA
     - If it can be represented in FOA it's computable
  2. Arithmetic descriptions: $\mu -$ recursive fun
  3. $\lambda -$ calculus
  4. Machine-like descriptions: TM, Post, register

These 4 were done in 1930s. All 4 can be proved to be equivalent.

- Going from these formal systems to computability (and vice versa) is sometimes called the Church-Turing thesis
  - Can this be proved? No, because the Church-Turing thesis isn't well posed, going from something formal to something informal (computability). How can you go from one to the other? That's why it's a thesis not a proof.
- Alan Turing (1912-1954) invented Turing Machines in 1936

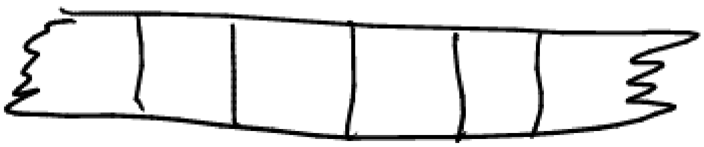
---

Say you have a paper with numbers on it:



Conditions:

- Boundedness: finite # of symbols and possible actions
- Locality: finite range for reading and writing
- Tape:



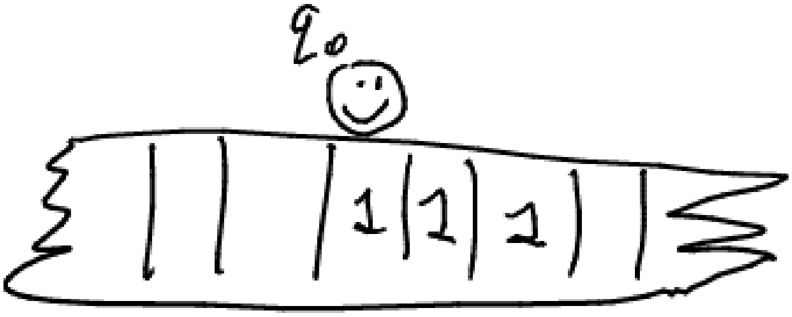
  - finite alphabet, e.g. every square is either 1 or $B$(lank)
  - Head: over one square

- read symbol (that it's currently looking at)
- write & delete symbol
- move left or right: L R
- States: $q_0, q_1, q_2, \ldots$
- Instructions:
  - $(q_i, S_y, O_p, q_j)$
  - $q_i$ is current state
  - $S_y$ is symbol read
  - $O_p$ is operation, either $1, B, L, R$
  - $q_j$ is next state
- Always begin in state $q_0$

Now we can write Turing Machine programs.

TM1 =

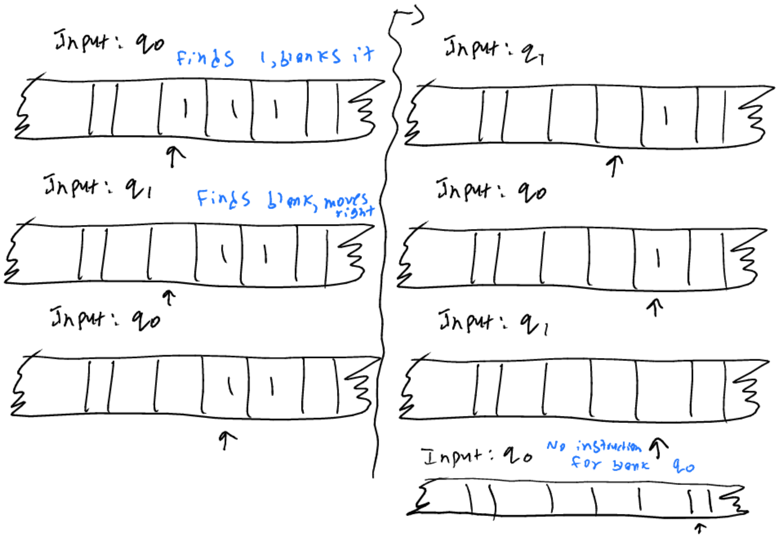
- $(q_0, B, R, q_0)$, if at a blank, go right
- $(q_0, 1, B, q_0)$, if at a 1, write a blank
- Erases tape and never halts



- Now we want a TM that erases all 1s

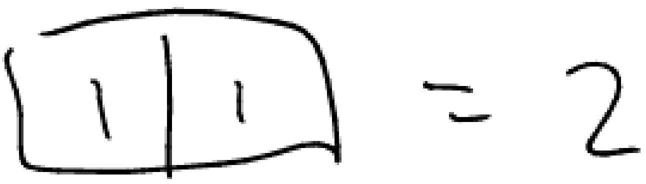
TM2 =

- $(q_0, 1, R, q_1)$
- $(q_1, B, R, q_0)$
- Erases consecutive 1s

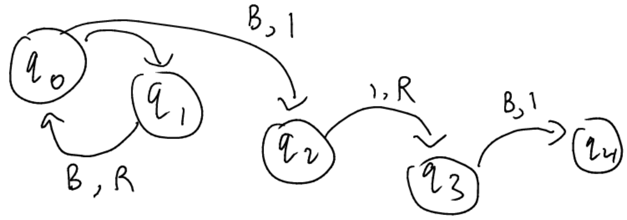
- 

### 18.1.1 TM as functions

TM as functions from $\mathbb{N} \to \mathbb{N}$ (input is only 1 block of consecutive 1's)

- 

- TM2: $f(x) = 0$
- TM3: What if we want $f(x) = 2$
  - Can reuse TM2, erase everything
  - Once everything is erased, write 2 1's
  - TM3 =
    - $(q_0, 1, R, q_1)$
    - $(q_1, B, R, q_0)$
    - $(q_0, B, 1, q_2)$
    - $(q_2, 1, R, q_3)$
    - $(q_3, B, 1, q_4)$, $q_4$ doesn't exist, will stop at this point

- Alternate notation:
    - 

- TM4: $f(x) = x + 2$
    - Skip all 1's and then add 21's
    - Alternatively, just go left and add 21's
- TM5: $f(x, y) = x + y$
    - Sequence of 1's and then a blank and then another sequence of 1's
        - Delete first 1 and then go right until you see a blank and then write a 1 there
        - Need to know conventions about number of blanks in between
- TM6: $f(x, y) = x \cdot y$
    - Have to copy $yx$ amount of times after $y$ and then delete $y$, more complicated, but a good exercise

How many functions are there from $\mathbb{N} \to \mathbb{N}$? Uncountably infinite. How many turing machines are there? Countably infinite. Why? Because there are finitely many instructions with a finite amount of states in your turing machine.

- This means that not every function can be represented as a turing machine, i.e. some are not computable.

## 18.2 The Halting Problem

We can enumerate all Turing Machines based off a fixed alphabet. So we can write: $T_1, T_2, T_3, \ldots, T_n, \ldots$.

- $T_n(m) : T_n$ on input $m$
- $K = \{x \in \mathbb{N} | T_x(x) \text{ halts}\}$

- TMs can either halt (stop) or go on forever.

The Halting Problem is as follows: Is there a TM that decides whether $y \in K$ or $y \notin K$? In other words:

$$TM_k(y) = \begin{cases} 1 & \text{if } y \in k \\ 0 & \text{if } y \notin k \end{cases}$$

### 18.2.1 Claim

This function is not computable. Then by Church-Turing thesis, a computer cannot compute this.

Note that $K$ is r.e.

- Make a table with $T_1$ on input 1, $T_2$ on input 2, ...
- Check at each step if it's still running or halts, to see if they eventually halt
- Go through this table diagonally like was done for the rational numbers. This way we can list all elements of $K$.

### 18.2.2 Proof (by contradiction)

Assume $T_H$ decides the set $K$. i.e.

$$T_H(x) = \begin{cases} 1 & \text{if } x \in k,\ T_x(x) \text{ halts} \\ 0 & \text{if } x \notin k,\ T_x(x) \text{ loops} \end{cases}$$

Now define a new machine:

$$T_G(x) = \begin{cases} 0 & \text{if } T_H(x) = 0 \\ \text{loops} & \text{if } T_X(x) = 1 \end{cases}$$

So $T_G(x) = T_n(x)$ for some $n$ (since we enumerated all Turing machines). Want to know output of $T_G(n)$

- So machine can either give 0 or loop
- It is 0 if $T_H(n) = 0$, i.e. $n \notin k$ and $T_n(n)$ loops, so $T_G(n)$ loops. Can't loop and give 0 at the same time! ↯
- Loops if $T_H(n)$ halts, i.e. $n \in k$ $T_n(n)$ halts, $T_G(n)$ halts. Can't loop and halt! ↯

This is diagonalization!

# 19 Lecture 19 *<2017-11-09 Thu>*

## 19.1 Reminder: Turing Machines & Computers

| TM | Computer |
|---|---|
| tape | memory/disc |
| head | processor/pointers |
| instructions | program |
| □ (square) | 1 bit |

Everything you can do on a computer you can do on a Turing Machine. Things you can't do on a Turing Machine you cannot do on a computer, i.e. The Halting Problem

Turing Machines were supposed to be a formalization of "computability"

- Another way of analyzing computability is with functions
  - Primitive recursive functions, which we'll try and extend to other functions

## 19.2 Class of functions

Inductive def of a class of functions: $\mathbb{N} \to \mathbb{N}$

Primitive Recursive Functions

1. $z(x) = 0$ (zero function)
   - $s(x) =$ the next number in the sequence following $\mathbb{N}$ (successor)
   - $P^i_k(\underbrace{x_1, \ldots, x_k}_{\vec{x}}) = x_i, 1 \leq i \leq k$ (projection)
2. Composition: Given $g$ with $m$ arguments and $h_1, \ldots, h_m$ with $\underbrace{k \text{ arguments}}_{\vec{x}}$
   - $f(\vec{x}) = g(h_1(\vec{x}, \ldots, h_m(\vec{x}))$
   - Schema of prim. rec. ($h$ is primitive recursive)
     - $f(0) = d$
     - $f(n+1) = h(f(n), n)$
   - Generalizing: $g, h$ are prim. rec.
     - $f(0, \vec{x}) = g(\vec{x})$
     - $f(n+1, \vec{x}) = h(f(n, \vec{x}), n, \vec{x})$

### 19.2.1 E.g

1. $id(x) = x = P^1_1(x)$
2. $plus(0, x) = x$
    - $plus(n + 1, x) = S(plus(n, x))$
    - $\implies g(x) = p^1_1(x))$
        - $h(u, v, w) = S(P^1_3(u, v, w))$
        - $u$ is $plus(n, x)$ , $v = n, w = x$
3. $mult(x, y)$, factorial, exponentiation
4. $pred(0) = 0, pred(n + 1) = n$ (only need the simple scheme, not the generalized one)
    - $h(u, v) = P^2_2(u, v)$
5.

$$cond(x, y, z) = \begin{cases} y & \text{if } z = 0 \\ x & \text{otherwise} \end{cases}$$

1. For relation $R(\vec{x})$: (Characteristic function)

$$C_R(x) = \begin{cases} 1 & \text{if } R(\vec{x}) \\ 0 & \text{otherwise} \end{cases}$$

1. Def by cases:

$$f(x) = \begin{cases} h_1(x) & \text{if } x \text{ satifies } A_1 \\ \ldots \\ h_n(x) & \text{if } x \text{ satisfies } A_n \end{cases}$$

$$f(x) = (h_1(x) \cdot C_{A_1}(x)) + \ldots + (h_n(x) \cdot C_{A_n}(x))$$

Assuming they're disjoint, only one will happen at a time, i.e. you get one of these values (as that's the only one satisfied).

1. Bounded minimization:

$$f(\vec{x}) = \min y \leq n[h(\vec{x}, y) = 0]$$

- Tries out all $y$'s from 0 to $n$ and as soon as $h$ is 0, it will return $y$

- e.g. $h(3,0) = 7$
    - $h(3,1) = 4$
    - $h(3,2) = 0$
    - With $n = 10$, $f(3) = 2$
    - If condition is not met (none of the values up to $n$ are 0, then it will return $n$)
- Essentially this is like searching, like a for-loop

---

Are all computable functions primitive recursion functions

- All primitive recursion functions can be enumerated (since they're theorems of a formal system)

Prim. rec: $f_1, f_2, f_3, f_4, \ldots$

| | 1 | 2 | 3 | ... |
|---|---|---|---|---|
| $f_1$ | $f_1(1)$ | $f_1(2)$ | $f_1(3)$ | |
| $f_2$ | $f_2(1)$ | $f_2(2)$ | | |
| $f_3$ | ... | | | |

Let $g(x) = f_x(x) + 1$. If $g$ were primitive recursive, then $g(x) = f_n(x)$ for some $n$.

So what is $g(n)$? $g(n) = f_n(n)$ but by definition of $g$, $g(n) = f_n(n) + 1$. A function cannot be the same value as a function and that function plus 1. ↯

Therefore, $g$ is <u>not</u> primitive recursive.

Primitive recursive functions are really nice and most if not all we've seen in our lives are primitive recursive.

## 19.3 Bloop-Programs

Bloop -> Bounded loop. This is the definition used in GEB instead of defining primitive recursive functions like we just did

### 19.3.1 Ex

- LOOP 4 TIMES
- BLOCK: BEGIN

- . . .
- BLOCK: END

TisLoopy from course webpage to compile Bloop-Programs and practice without having to define a primitive recursive function.

### 19.3.2 Syntax

```
DEFINE PROCEDURE "MINUS" [M,N]:
BLOCK 0: BEGIN
      OUTPUT <= 0;
      IF M<N, THEN: QUIT BLOCK 0;
      LOOP AT MOST M+1 TIMES:
      BLOCK 1: BEGIN
           IF OUTPUT+N=M, THEN ABORT LOOP 1;
    OUTPUT <= OUTPUT + 1
      BLOCK 1: END
BLOCK 0: END
```

| M | N | OUTPUT |
|---|---|---|
| 5 | 2 | 0 |
| | | 1 |
| | | 2 |
| | | 3 |

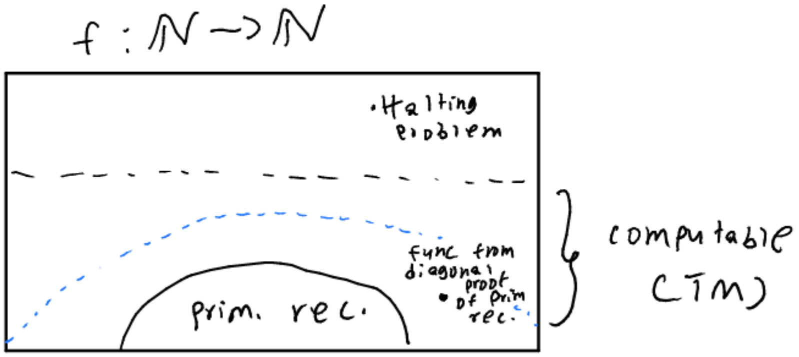
# 20 Lecture 20 *<2017-11-14 Tue>*

## 20.1 Bloop

Bloop = primitive recursive functions. What makes them nice?

- Computable
  - Note that, as seen in last class, not all computable functions are primitive recursive functions
- Evaluation always terminates
- Total, defined on all inputs

Big picture:



- Want to expand our definition of primitive recursive functions. How?

Recall that bloop has loops as follows:

- `LOOP ... TIMES`
- `LOOP AT MOST ... TIMES`
- Both of the loop iterations are bounded.

## 20.2 Floop

Like Bloop but have a new block:
MU-LOOP:

```
MU-LOOP:
BLOCK:BEGIN
ABORT LOOP
BLOCK:END
```

Then we get programs, such that some terminate and some do not terminate (loop forever).

## 20.3 Bounded minimization

$f(x) = \min y \leq n \overbrace{[h(x,y) = 0]}^{\text{condition}}$

- How to compute?
- Start with $y = 0$, keep evaluating and iterating until it's 0 (then you output $y$) or you reach the end and you output the end ($n$).

$$prime(x) = \begin{cases} 1 & \text{if } x \text{ is prime} \\ 0 & \text{otherwise} \end{cases}$$

E.g. $nextprime(x) \min y \leq (x! + 1)[x < y \text{ AND } prime(y) = 1]$

- $listprimes(0) = 2$
- $listprimes(n + 1) = nextprime(listprimes(n))$

### 20.4 Unbounded minimization

$f(x) =$ the least $y$, such that $\overbrace{y + x = 5}^{\text{condition}}$

| $x$ : | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $f(x)$ : | 5 | 4 | 3 | 2 | 1 | 0 | undefined ↑ |

(↑ means undefined)

This is called the $\mu -$ operator (least search operator):

- $\mu y[f(\vec{x}, y) = 0] \implies z$ iff $f(\vec{x}, z) = 0$ and for every $y < z, f(\vec{x}, y)$ is defined (and $> 0$)

E.g. $\mu y[f_i(y) = 0]$

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | ... |
|---|---|---|---|---|---|---|---|---|
| $f_1(x)$ | 3 | 2 | 1 | 0 | 0 | 0 | 0 | ... |
| $f_2(x)$ | 2 | 0 | ↑ | 3 | ↑ | 0 | 0 | ... |
| $f_3(x)$ | 2 | 3 | ↑ | 1 | 0 | ↑ | ... | |

- $\mu y[f_1(y) = 0] \implies 3$
  - Because $f_1(3)$ is the first to be 0 and the previous values are defined.
- $\mu y[f_2(y) = 0] \implies 1$
- $\mu y[f_3(y) = 0] \implies \uparrow$
  - It is undefined because not all $f_3(y)$ are defined before $f_3(z) = 0$

Can we make a Turing Machine that acts like this $\mu$ operator?

- Yes. The Turing Machine goes through a row until it either hits 0 or ↑ (loops forever)

  - Note that we can't determine in finite time if something loops forever

- Recall that Turing Machines can loop forever, which may happen if a function keeps evaluating to something that isn't 0 but is defined.

- $\implies$ computable

- $f(x)\uparrow \iff$ TM loops forever

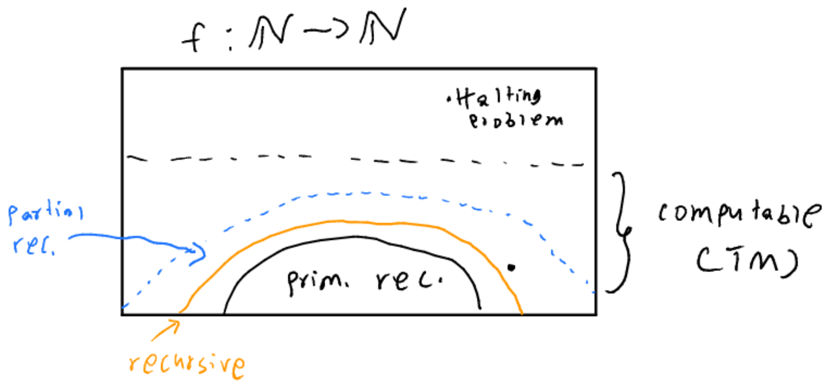
### 20.5 Partial recursive functions

- Primitive recursive functions

- $\mu$ – operator

Some partial rec. fun. are defined (has a value) for all arguments: total functions (so we can distinguish between partial rec. fun and total partial rec. fun)

- We call these recursive functions



Alternatively: $\begin{array}{l|l l|}\text{prim. rec} & \text{rec.} & \text{not total part. rec}\\ \hline \text{total} & & \text{not total}\end{array}$ Are primitive functions syntactic or semantic? Syntactic. The definition is typographical. Partial rec. functions are also syntactic.

- But recursive functions are semantic because you can't just look at a function and determine if its recursive. It needs to be defined, needs a meaning.

How did we show that there was a computable function that wasn't primitive recursive? Diagonalization. Can you use diagonalization to create a computable function that isn't partial recursive.

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $\varphi_0$ | $\varphi_0(0)$ | $\varphi_0(1)$ | $\varphi_0(2)$ | $\varphi_0(3)$ |
| $\varphi_1$ | $\varphi_1(0)$ | $\varphi_1(1)$ | $\dots$ | |
| $\varphi_2$ | $\dots$ | | | |

Define: $\psi_1(x) = \underbrace{\varphi_x(x)}_{\text{might be undefined}} +1$

$$\psi_2(x) = \begin{cases} 1 & \text{if } \varphi_x(x) \uparrow \\ 0 & \text{if } \varphi_x(x) \downarrow \end{cases}$$

The Halting Problem for partial recursive functions is not recursive: There is no recursive function that tells us whether $varphi_x(x)$ is defined or not. Assume:

$$f(x) \simeq \begin{cases} 1 & \text{if } \varphi_x(x) \downarrow \\ 0 & \text{if } \varphi_x(x) \uparrow \end{cases}$$

is a total partial rec. function

- (this is recursive, since $f(x)$ always gives you a value).

Define

$$p(x) \simeq \begin{cases} \uparrow & \text{if } f(x) = 1 \\ 0 & \text{if } f(x) = 0 \end{cases}$$

$p(x) = \mu y[y + f(x) = 0]$

- Then $p(x)$ is partial recursive because it's just the $\mu-$operator applied on another partial recursive function.

Because $p(x)$ is partial recursive, it must occur in the enumeration such that $p(x) = \varphi_y(x)$

- What is $p(y)$? $p(y) = \varphi_y(y)$. It is undefined if $f(y) = 1$ and $f(y) = 1$ if $\varphi_y(y)$ is defined. Contradiction ($p(y)$ is defined and undefined at the same time). ↯

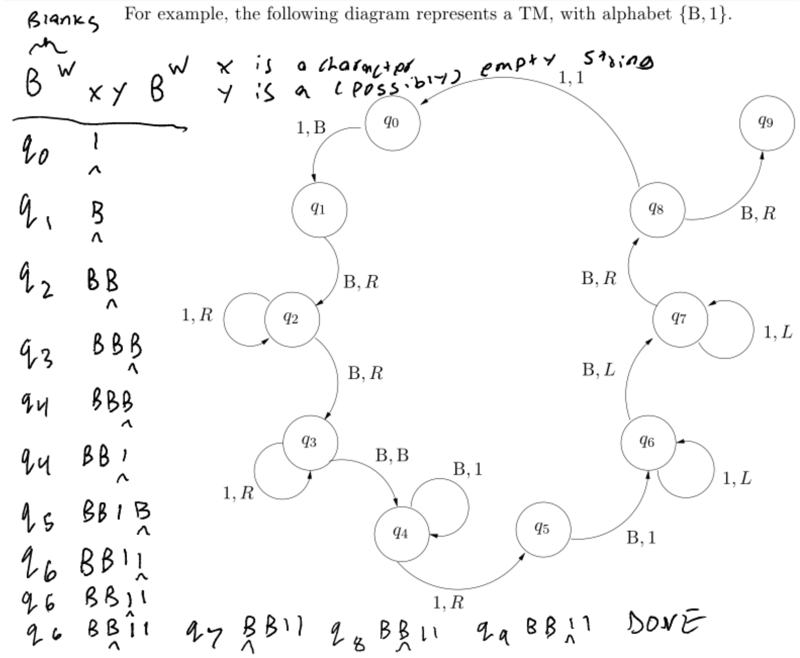
# 21 Lecture 21 *<2017-11-16 Thu>*

Review of Turing Machines

## 21.1 Turing Machine on Handout 5

This machine doubles a string of 1's

- Doubling 1:



For example, the following diagram represents a TM, with alphabet {B, 1}.

Blanks

$\underbrace{B^w}$ $xy$ $B^w$ x is a character, y is a (possibly) empty string

$q_0$ $\underset{\wedge}{1}$

$q_1$ $\underset{\wedge}{B}$

$q_2$ $B\underset{\wedge}{B}$

$q_3$ $BB\underset{\wedge}{B}$

$q_4$ $BB\underset{\wedge}{B}$

$q_4$ $BB\underset{\wedge}{1}$

$q_5$ $BB1\underset{\wedge}{B}$

$q_6$ $BB1\underset{\wedge}{1}$

$q_6$ $BB\underset{\wedge}{1}1$

$q_6$ $B\underset{\wedge}{B}11$ $q_7$ $\underset{\wedge}{B}B11$ $q_8$ $B\underset{\wedge}{B}11$ $q_9$ $BB\underset{\wedge}{1}1$ DONE

- Doubling 3 (part of execution):

$q_0 \quad \underset{\wedge}{1}11$

$q_1 \quad \underset{\wedge}{B}11$

$q_2 \quad B\underset{\wedge}{1}1$

$q_2 \quad B1\underset{\wedge}{1}$

$q_2 \quad B11\underset{\wedge}{B}$

$q_3 \quad B11B\underset{\wedge}{B}$

$q_4 \quad B11B\underset{\wedge}{1}$

$q_5 \quad B11B1\underset{\wedge}{B}$

$q_6 \quad B11B1\underset{\wedge}{1}$

$q_6 \quad B11B\underset{\wedge}{1}1$

$q_6 \quad B11\underset{\wedge}{B}11$

$q_7 \quad B1\underset{\wedge}{1}B11$

$q_7 \quad B\underset{\wedge}{1}1B11$

$q_7 \quad \underset{\wedge}{B}11B11$

$q_8 \quad B\underset{\wedge}{1}1B11$

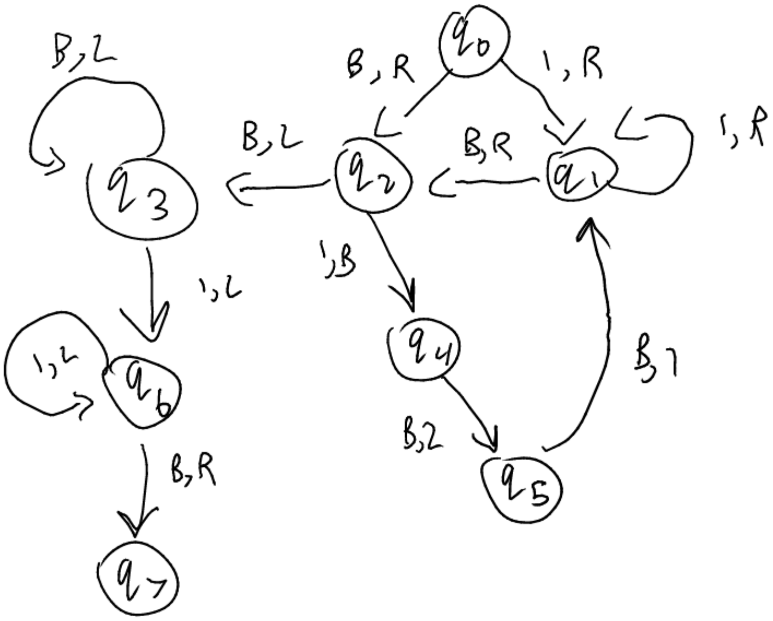
$q_0 \quad B\underset{\wedge}{1}1B11$

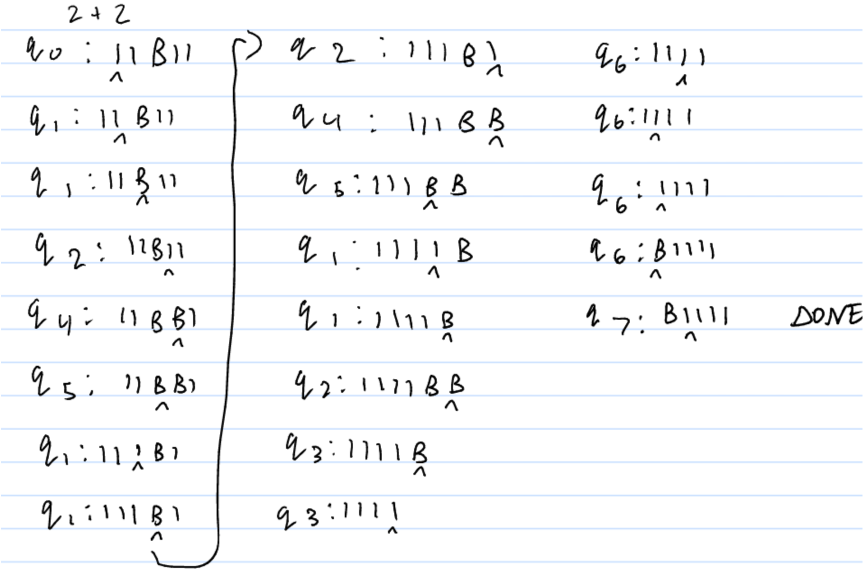
.... $q_0 \quad BB\underset{\wedge}{1}B1111$

.... $q_8 \quad BBB\underset{\wedge}{B}111111$

$q_9 \quad BBBB\underset{\wedge}{1}11111$

### 21.2 Turing Machine for addition




2 + 2

q0 : 11B11

q1 : 11B11

q1 : 11B11

q2 : 11B11

q4 : 11BB1

q5 : 11BB1

q1 : 111B1

q1 : 111B1

q2 : 111B1

q4 : 111BB

q5 : 111BB

q1 : 1111B

q1 : 1111B

q2 : 1111BB

q3 : 1111B

q3 : 1111

q6 : 1111

q6 : 1111

q6 : 1111

q6 : B1111

q7 : B1111 DONE

Try: `http://turingmachinesimulator.com/shared/tsgfopdqwb`

## 21.3 Turing Machine for multiplication

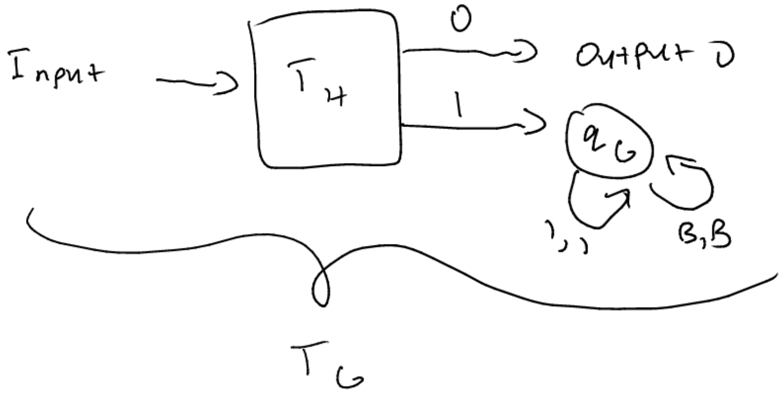
We're not actually going to make one, but what if we wanted to? We would reuse our Turing Machine for addition, don't reinvent the wheel from scratch each time. Note that:

$$n \times m = \begin{cases} 0 & \text{if } n = 0 \\ (n-1) \times m + m & \text{if } n > 0 \end{cases}$$

Can recursively add in order to multiply.

## 21.4 The Halting Problem

This helps you understand the Halting Problem more.



- Turing Machines can be inputted as an argument for another
- Review of Halting Problem
- Key points:
  - Assuming existence of $T_H$
  - Constructing the $T_G$ through elementary operations on $T_H$
  - Self-reference (feeding $T_G$ to itself)-> ↯

# 22 Lecture 22 *<2017-11-21 Tue>*

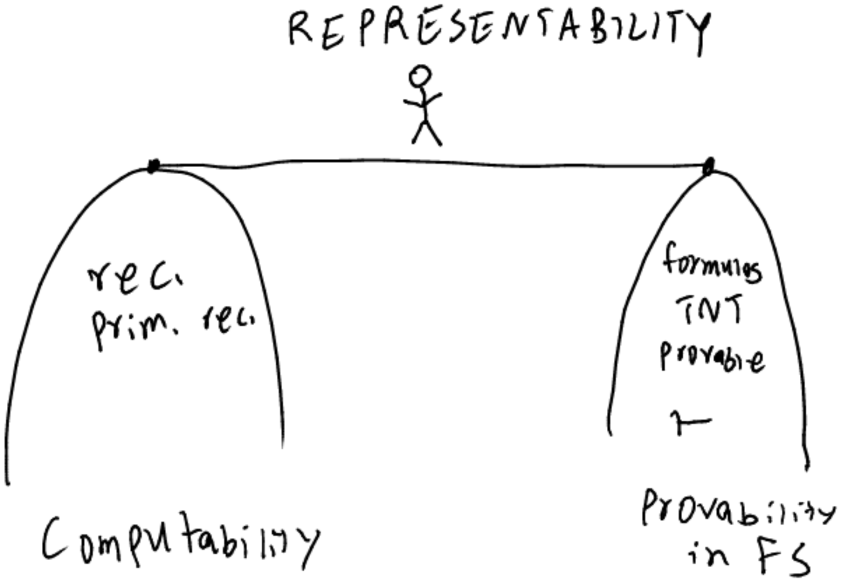
## 22.1 Quiz Review

- Non logical symbols in FOA: $0, S, +, \times$
- Halting set: $K = \{n \in \mathbb{N} | T_n(n) \text{ halts}\}$
- Partial recursive def
- $\mu y[S(y) \text{ op } 1 = 0]$
  - $f(1) = 0, f(0) = \uparrow$
- $\aleph_0$ Turing Machines/Primitive Recursive Functions

## 22.2 Functions

| | | |
|---|---|---|
| Prim. rec | BlooP | - |
| Part. rec | FlooP | TM |
| Rec. | terminating FlooP | TM always halt |

## 22.3 Representability



Today we will learn how to go from computability to provability in a formal system.

A predicate is represented in a FS:

1. All true instances are theorems in FS
2. All false instances are not thms

### 22.3.1 Ex.

Express: Evenness

$$Even(x) = \begin{cases} 1 & \text{if } \exists y : x = 2 \cdot y \\ 0 & \text{otherwise} \end{cases}$$

Represented in TNT:

- $TNT \vdash Even(SS0)$
- $TNT \nvdash Even(S0)$
- $TNT \vdash \sim Even(S0)$

### 22.3.2 Thm

Any primitive recursive predicate (BlooP) is representable in FOA.

- Recall: What is a primitive recursive predicate? A predicate is primitive recursive if its characteristic function is primitive recursive
- $Even(x) = \forall x : \exists y : \ldots$

### 22.3.3 Thm

Any recursive predicate is representable in FOA

## 22.4 Theories of Arithmetic

(Handout 8)

| | | | |
|---|---|---|---|
| $\Pi_0$/baby | $\Pi_1$/junior | $\Pi_2$/Q | $\Pi$/FOA |
| 4 axiom schemas | 4+3 axiom schemas | finitely many (9) axioms | 4 schema + induction |

$\Pi_0 \nvdash 0 \neq S0$

$$\Pi_0 \subset \Pi_1 \subset \Pi_2 \subset \Pi$$

## 22.5 Proof-Pairs

TNT-PROOF-PAIR(n,m), where $n$ is the G\"odel # of a proof of $m$ and $m$ is the G\"odel # of the theorem

The property of being a proof-pair is primitive recursive (it's like proof checking, just decompose it and see if it works): proof-pair(x,y)

- This means that it's representable
  - $\implies$ There is a FOA-formula Bew(x,y) (German word Beweis = proof) that represents it.

Being a theorem-number:

- $theorem - number(x) = \exists y : proof - pair(y, x)$
  - i.e. $x$ is a theorem number if it's a G\"odel number of a theorem
  - Is this primitive recursive? Not clear. You have to find a proof rather than check a proof.

## 22.6 Substitutions

| Formula | G\"odel # |
|---|---|
| $a = a$ | 262 111 262 |
| $SS0 = SS0$ (substitute by term) | 123 123 666 111 123 123 666 |

SUB(formula with free variable, numeral (term), resulting formula) -> In our example, SUB(262 111 262, 2, 123 123 666 111 123 123 666)

- This is primitive recursive
  - This means that there is a formula in TNT that represents it
- All we have to do to check this is just decode the first formula, plug in the numeral into it and see if it's equal to the resulting formula

SUB(x,y,z)

- Good exam question, which numbers represent x,y,z?

ARITHMOQUINE(a,b) = SUB(a,a,b)

## 23 Lecture 23 *<2017-11-23 Thu>*

What is a theory? Set of sentences closed under deduction. $x \in T \iff T \vdash x$, where $T$ is a theory.

A theory $T$ is complete, if for all sentences $A$: either $T \vdash A$ or $T \vdash \sim A$ What we will show next is that TNT is not complete (incomplete)!

- Proof: Produce $G$, s.t. $TNT \nvdash G$ and $TNT \nvdash \sim G$
- This is (more or less) G\"odel's First Incompleteness Theorem
  - Very important, perhaps the highlight of the course

ARITHMOQUINE(a,b) = SUB(a,a,b)

- $a$ is a G\"odel number of a formula with a free variable, the second $a$ is the number that stands for a certain numeral and $b$ is the G\"odel number of the formula from the first $a$, but with the free variable replaced by the numerable represented by the second $a$

### 23.1 Ex

$A(x)$ is $\sim (Sx = 0)$

- We want to get the ARITHMOQUINE of this number
- G\"odel number is: $a = 223362123262111666323$
- ARITHMOQUINE(a,b)
  - Take $A(a) => \sim (S\underbrace{SS\ldots S}_{a-many}0 = 0)$
    - Now take the G\"odel number of this and you get your $b$
- Arithmoquine is certainly recursive and terminates. We can easily make a computer program to mimic it.

More mathematically: Arithmoquine is usually called: d (diagonal function)

$$d( \underbrace{\#\alpha}_{\text{the Godel \# of } \alpha} = \begin{cases} \#\alpha(S_{\#\alpha}) & \text{if } \#\alpha \text{ is the Go of a form } \alpha \text{ with one free variable} \\ \#\alpha & \text{otherwise} \end{cases}$$

- E.g. $d(5) = 5, d(a) = b$

---

- $Even(x)$ expresses "x is even"
- $Even(SSSS0)$ expresses "4 is even"
- $A(x)$ "x has the property A"
- $A(SSSS0)$ "4 has the property A"

A number (ex. student id), can just be a number but it can represent a student.

IS-TNT-AXIOM(x) => property of the number $x$ or property of the formula with G\"odel # x

- $\forall a : \exists b : \ldots x \ldots$

G's aunt:

- $\sim \exists a : \exists a' : \underbrace{TNT-PROOF-PAIR(a, a')}_{a \text{ is a proof of } a'} \wedge \underbrace{ARITHMOQUINE(a'', a')}_{a' \text{ is the arithmoquinification of } a''} =$
  $F(a'')$ with G\"odel # $u$
  - Equation has one free variable, $a''$
  - Putting them together we get $a$ is a proof of the arithmoquinification of a”
- Now G's aunt is:
  - there is no ($a$, s.t. $a$ is a) TNT proof of ($a'$ and $a'$ is )the arithmoquinification of a”

Now we plug in $u$ as the free variable $a''$

- $\sim \exists a : \exists a' : TNT{-}PROOF{-}PAIR(a, a') \wedge ARITHMOQUINE(\underbrace{SS \ldots S0}_{u \text{ many}}, a') =$
  $F(u) = G$
- This is now a sentence, no more free variables

What is the G\"odel # of $G$? The arithmoquinification of $u$.

- We have done the same thing we did earlier with the smaller example except for a longer formula.

What does $G$ express? There is no TNT-proof of $\underbrace{\text{the arithmoquinification of } u}_{G}$.

=> $G$ says: "I am unprovable (in TNT)"

- Is G a TNT theorem? Assume TNT $\vdash G$. Since we're working in a system that is sound, then $G$ must be true. But $G$ expresses that it is not a TNT theorem. ↯
- => TNT $\nvdash G$
- Is $\sim G$ a TNT-theorem? Assume TNT $\vdash\sim G$. So, $\sim G$ true, i.e. $G$ is provable in TNT. So TNT $\vdash G$ => TNT is inconsisten.
- If TNT is consistent then TNT $\nvdash\sim G$

So we end up with: If TNT is consistent, then TNT is not complete! **G\"odel's First Incompleteness Theorem**

- TNT was important here to tell us that the functions we used (i.e. TNT-PROOF-PAIR) are representable
- We did not need TNT directly, we could have used a weaker theory, like $\Pi_1$

## 23.2 General Version of G\"odel's First Incompleteness Theorem

$G1$: Every theory that includes $\Pi_1$ is consistent (or else you prove everything, including $G$) and axiomatizable (has an axiom system, or else we wouldn't have anywhere to start, we need to be able to define what a proof is) is incomplete.

- How could you make $G$ provable in TNT?
  - Just turn it into an axiom
  - So we can do something like TNT+$G$
    - But this still includes $\Pi_1$ and is axiomatizable (because TNT has axioms)
    - But then you can do the same thing with this new system and get something similar, say $G'$
    - This will once again, be incomplete
  - As long as you're axiomatizable, you'll still be incomplete, no matter how much we try to extend TNT.

# 24 Lecture 24 *<2017-11-28 Tue>*

## 24.1 Summary of G1

Consistent theory that is axiomatizable and includes $\Pi_1$ is incomplete (can't prove everything)

We've seen two ways of defining a consistent theory

- Syntactic definition:
  - Recursive set $A$ (axioms)
  - Theory $T = \{X | A \vdash X\}$
- Semantic definition:
  - Structure $\mathfrak{M}$ (can be natural numbers or something like that)
  - $\text{Th}(\mathfrak{M}) = \{X | X \text{ is true in } \mathfrak{M}\}$
  - e.g. $\text{Th}(\mathfrak{N}) = \Omega$ (all the true statements about the natural numbers)
    * Is this consistent? Yes. If there are natural numbers, a statement, i.e. $SS0 = S0$ is either true or false, can't be both. Also note that if you are inconsistent, you can prove anything, i.e. everything is true (because from a contradiction you can get anything, natural deduction)
    * Does it include $\Pi_1$? Yes. Since $\Pi_1 \subset$ TNT and TNT is built for all of its true statements to be true for the natural numbers, then the statements of $\Pi_1$ are true too.
    * Complete, since it either has $A$ or $\sim A$ for each sentence.
    * All true statements in FOA will be in $\Omega$. All false statements' negations are true and in $\Omega$.
    * By $G1$, $\Omega$ cannot be axiomatizable (if you assume it's axiomatizable you get a contradiction)
    * So we cannot come up with a system of axioms that contains all of $\Omega$

Recall, theories of arithmetic:

$$\Pi_0 \subset \Pi_1 \subset \Pi_2/Q \subset \Pi/TNT/GPA \subset TNT + G \subset \dots$$

i.e. you can keep expanding by adding axioms. But since $\Omega$ is not axiomatizable, you cannot reach it by constantly adding axioms, it is out of reach

Can we prove the consistency of TNT? (Note that consistency is a syntactic property, it means you cannot $\underline{\text{deduce}}$ both $\underline{A}$ and $\underline{\sim A}$, for some $A$)

### 24.1.1 Hilbert's Program

Assume some very small theory of arithmetic (finitary) and use it to show the consistency of TNT. Thus we have reduced the problem of consistency to this finitary problem (smaller problem).

---

Idea: Formalize "consistency"

- NEG$(x, y)$ $\iff$ $x$ is the G\"odel # of wff, $y$ is the G\"odel # of $\sim A$. These two numbers code an inconsistent pair.
- $CONS_{TNT} = \forall x : \forall y : \forall z : \forall w \sim$ [NEG$(x, y)\wedge$ TNT-PROOF-PAIR$(z, x)\wedge$ TNT-PROOF-PAIR$(w, y)$]
  - i.e. There are no numbers such that there is a proof of $x$ and a proof of $y$ when $x$ and $y$ are negations.
  - If this is true, then the system (TNT) is consistent, i.e.
  - $CONST_{TNT} \implies \sim (TNT \vdash A \wedge TNT \vdash \sim A)$

Alternatively,

- $\sim (0 = 0)$ must not be provable in TNT
- $CONS'_{TNT} = \sim \exists a :$ TNT-PROOF-PAIR$(a, \quad \overbrace{S \ldots 0}^{\text{Go \# of expression above}} \quad )$
- $CONS_{TNT}$ expresses: TNT is inconsistent
- Question: $TNT \vdash CONS_{TNT}$? No.

### 24.1.2 Thm

If TNT is consistent, then TNT $\nvdash CONS_{TNT}$

### 24.1.3 Proof

By $G1$, there is a sentence $G$, such that TNT $\nvdash$ G (i.e. $G$ is not provable in TNT)

- So, if $CONS_{TNT}$, then $G$
- One can show (proof omitted here, also note that in the proof we need TNT):

  - $TNT \nvdash CONS_{TNT} \supset G$

- Assume (for contradiction): $TNT \nvdash CONS_{TNT}$
  - Apply Modus Ponens, getting $TNT \nvdash G$ ↯
- Therefore our assumption was wrong, so $TNT \nvdash CONS_{TNT}$. i.e. TNT cannot prove the formula that says it is consistent. Then clearly something that is smaller than TNT (like Hilbert's Program), cannot prove it either, so Hilbert's Program does not work.

### 24.2 G2

Any theory $T$ that is:

- Consistent
- Axiomatic
- Includes Π(TNT)

Cannot prove $CONS_T$ (which expresses that $T$ is consistent). In other words, you cannot prove your own consistency.

Which axiomatic theory $T$ can prove that TNT is consistent?

- $TNT + CONS_{TNT} = TNT'$, but this can't prove that it itself is consistent.
- $TNT' + CONS_{TNT'} = TNT''$
- ...

We have to assume something stronger in order to prove TNT's consistency, but then the stronger things are even more doubtful. So we cannot prove that TNT is consistent without assuming consistency of a strong theory, which we cannot prove consistency for without assuming something even stronger and so on ...

So we just assume that TNT is consistent, but we have no proof of that.

### 24.3 Church Thm

FOL (all the theorems that follow from FOL) is undecidable.

- Is $x$ a thm of the axioms of FOL? No computable procedure that gives you an answer.