CS70 - Lecture 13 Notes

Name: Felix Su SID: 25794773

Spring 2016 GSI: Gerald Zhang

Countability Summary:

- S is countable if there is a bijection between S and some subset of $\mathbb{N}$.
- If the subset of $\mathbb{N}$ is finite, S has finite cardinality.
- If the subset of $\mathbb{N}$ is infinite, S is countably infinite.
- Bijection with natural numbers $\implies$ countably infinite.
- Enumerable (listing) $\equiv$ countable.
- Subset of countable set is countable.
- All countably infinite sets have the same cardinality
- Natural numbers have finite number of digits

Digonalization:

- The set of all subsets S_i of $\mathbb{N}$ (powerset of $\mathbb{N}$ is not countable
  - Arbitrary Listing: L
  - Diagonal set D: For each index i of L, if $i \notin S_i$, put i in D, otherwise omit i
  - D is not in L by construction: D is different from each ith set S_i in L, for every i
  - D is a subset of N: every element in D is a natural number
  - L does not contain all subsets of N: Contradiction

Diagonalization Algorithm:

1. Assume that set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list $\implies$ t is not in the list.
5. Show that t is in S.
6. Contradiction.

Cardinalities:

Continuum Hypothesis:

- Goedel proved this hypothesis cannot be proven with math we currently know
- There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

Uncountable Sets:

- Prove equivalence between cardinalities
- Show bijection exists between two sets: uncountable sets cannot be enumerated
- Create function $f : B \to A$ (can include multiple casesfor certain domains)
- Prove mapping is one to one by testing on arbitrary values: x, y (Need to validate for multiple cases)
  - Example: $|[0, 1]| \equiv |\mathbb{R}|$
  - $f : \mathbb{R}^+ \to [0, 1]$

Undecidability:

Russell's Paradox:

- Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

- NST : $y =$ the set of elements that satifies $P(x)$
- Make statement: $P(x) = x \notin x$
- By NST: There exists a y that satisfies above statement for $P(x)$
- Plug in $x = y$ to NST

$$y \in y \iff y \notin y \tag{2}$$

- Methematicl system is broken, because conditions and statements are false and contradictions

HALT: DNE

- $HALT(P, I)$: P = program, I = input to program
  - Theoretically determines if $P(I)$ halts or loops forever

Halt Turing Proof:

- Assume $HALT(P, I)$ exists
- Set $P = Turing(P)$
- Use Diagonalization

```
def Turing(P):
    if(HALT(P,P)): #halts
        go into infinite loop
    else
        halt immediately
```

- Assume $Turing(Turing)$ halts
- Run $HALTS(Turing, Turing)$
  - if 'halts', Turing(Turing) 'goes into infinite loop'
  - if loops forever, Turing(Turing) 'halts immediately'
- Contradiction, so $HALT(P, P)$ does not exist

Halt Diagonlization Proof:

- Program and input are both enumerable (fixed length strings)
- Program either halts or loops on any input
- Create list: $P_i \to P_j(P)$ where $i, j \in \mathbb{N}$
- Each entry of list is arbitrarily $HALT$ or $LOOP$
- Diagonal exists, so create $Turing()$ s.t. it returns opposite values along the diagonal
- This means $Turing()$ is not in the list $\implies$ $Turing()$ is not a prgoram
  - $Turing$ is a simple function constructed from $HALT$
  - $\therefore$ $Turing()$ DNE $\implies$ $HALT()$ DNE

Undecidable Problems

-