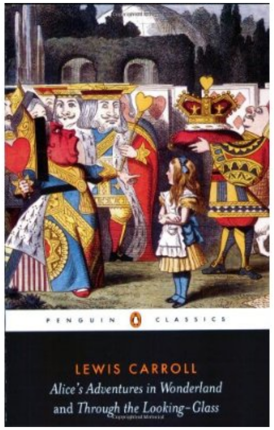
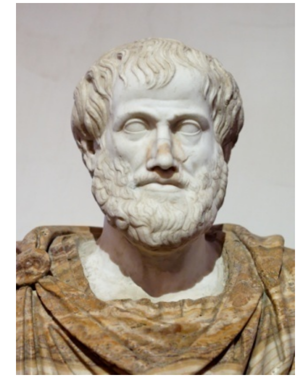


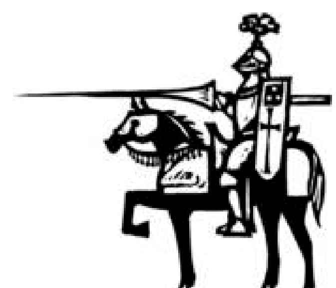


COMP 1002

Logic for Computer Scientists

Lecture 19

B 5 2 J



Puzzle: the barber club

- In a certain barber's club,
  - Every member has shaved at least one other member
  - No member shaved himself
  - No member has been shaved by more than one member
  - There is a member who has never been shaved.

- ***Question: how many barbers are in this club?***

…

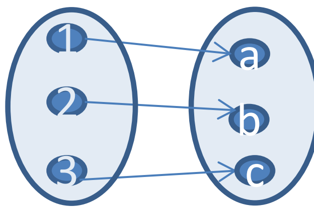
Infinitely many!

Barber 0 grows a beard.

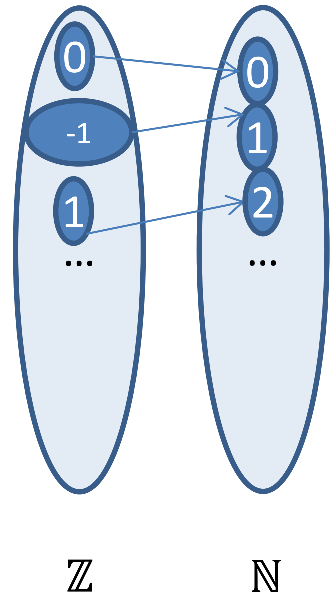
For all $n \in \mathbb{N}$, barber n shaves barber n+1

Cardinalities of infinite sets

- Two finite sets A and B have the same cardinality if they have the same number of elements
  - That is, for each element of A there is exactly one matching element of B.



- For infinite A and B, define |A|=|B| iff there exists a bijection between A and B.
  - If there is both a one-to-one function from A to B, and an onto function from A to B.
- A set A is countable iff |A| = |ℕ|.
  - ℤ is countable: take $f: \mathbb{Z} \to \mathbb{N}$, $f(x) = 2x$ if $x \geq 0$, else $f(x) = -(1+2x)$
  - Set of all finite strings over {0,1}, denoted $\{0,1\}^*$, is countable.
    - Empty string, 0, 1, 00, 01,10,11,000,001,…
  - An infinite subset of a countable language is countable. A Cartesian product of countable languages is countable:
    - $\mathbb{N} \times \mathbb{N}$: (0,0), (0,1), (1,0), (2,0), (1,1), (0,2), (3,0), (2,1), (1,2),…
  - $\mathbb{Q}$ is countable: $\mathbb{Q} \subset \mathbb{Z} \times \mathbb{Z}$



Diagonalization: $\mathbb{R}$

- Is there a bigger infinity?
  - Yes! In particular, $\mathbb{R}$ is uncountable. Even [0,1) interval of the real line is uncountable!
    - Reals may have infinite strings of digits after the decimal point.
    - Imagine if there were a numbered list of all reals in [0,1)
      - $a_0, a_1, a_2, a_3, \ldots$
    - For example:
      - a_1 = 0.23145…
      - a_2 = 0.30000…
      - …
  - Let number d be:
    - d[i]=$(a_i[i] + 1)\ mod\ 10$
    - Here, $[i]$ is i^{th} digit.
    - This d is a valid real number!

0.	r[1]	r[2]	r[3]	r[4]	r[5]	…	r[k]		
a_0	2	3	1	4	5	…			
1	3	0	0	0	0	…			
2	9	9	9	9	9	…			
…									
k	2	1	3	4	3	…	5	…	
…									
d	3	1	0	…	…	…	6	…	

  - But if number d were in the list, e.g. k^{th}, a contradiction
    - It would have to differ from itself in k^{th} place.

Diagonalization: languages

- An **alphabet** is a finite set of symbols.
  - For example, {0,1} is the binary alphabet.
- A **language** is a set of finite strings over a given alphabet.
  - For example, $\{0,1\}^*$ is the set of all finite binary strings.
  - PRIMES $\subset \{0,1\}^*$ is all strings coding prime numbers in binary.
  - PYTHON $\subset \{0,1\}^*$ is all strings coding valid Python programs in binary.
- Every language is countable.
  - $\{0,1\}^*$, PRIMES, PYTHON are countable
- Set of all languages is uncountable.
  - Put "yes" if $s \in L$, "no" if $s \notin L$
  - Let language D be:
    - s ∈ D iff $s \notin L_s$
  - If D were in the list, e.g. as L_k, a contradiction
    - It would have to differ from itself in k^{th} place.

		0	1	00	01	…	s_k		
L_0	yes	yes	no	yes	yes	…			
L_1	yes	no	yes	no	yes	…			
	no	no	no	no	no	…			
…									
L_k	no	yes	yes	no	yes	…	yes	…	
…									
D	no	yes	yes	…	…	…	no	…	

- So there is a language for which there is no Python program which would correctly print "yes" on strings in the language, and "no" otherwise.
- In general, for any set A, finite or infinite, its powerset $P(A)$ is larger than A: that is, |A| < $P(A)$

Halting problem

- A specific example of a problem not solvable by any program: the **Halting problem,** invented by Alan Turing:
  - Input:
    - Prog: A program as piece of code (e.g., in Python):
    - x: Input to that program.
  - Output:
    - “yes” if this Prog(x) stops (that is, program Prog stops on input x).
    - “no” if Prog goes into an infinite loop on input x.
  - Suppose there is a program Halt(Prog, x) which always stops and prints “yes” or “no” correctly.
    - Nothing wrong with giving a piece of code as an input to another program.
  - Then there is a program HaltOnItself(Prog) = Halt(Prog,Prog)
  - And a program Diag(Prog):
    - if Halt(Prog, Prog) says “yes”, go into infinite loop (e.g. add “while 0 <1: “ to Halt’s code).
    - if Halt(Prog, Prog) says “no”, stop.
  - Now, what should Diag(Diag) do?...
    - Paradox! It is like a barber who shaves everybody who does not shave himself.
    - So the program Diag does not exist… Thus the program Halt does not exist!
- So there is no program that would always stop and give the right answer for the Halting problem.