Lecture 2 CSE 421 Intro to Theory of Computation

My Office Hour MW 12:20-12:50 W 3:00-3:50

Volunteer note-taker for student with disabilities? Please email me because @cs.uw.edu

New room next week

Review: Prehistory of Computation: reasoning & infinity

...

Cantor (1875)

$|\text{integers}| = |\text{rationals}| \neq |\text{reals}|$

↑ dovetailing diagonalization

Frege (1879) Predicate logic

Peano (1889) Axioms for number theory (natural #'s)

Frege (1893) Foundations of Arithmetic based on set theory & formal proofs

Russell (1902) Paradox in Frege's Set Theory

Let S be "the set of all sets that aren't members of themselves"

Is $S \in S$?

Hilbert's 23 Problems (1900) Math Congress

- formalizing mechanizing classifying mathematics

2, 10

$x^2y + 2xy - 3yz = 0$

Given polynomial P in many variables integer coeffs. determine if $P(\vec{x}) = 0$ has a soln

Russell & Whitehead (1913)

Principia Mathematica

Formal system avoids Russell's Paradox

"type theory"

very slow to derive

1+1=2

Gödel (1925)

primitive recursive functions

like a PL with restricted loops

have to know # of times a loop is executed before you enter it.

loop N times
{
}

Ackermann (1927) Function

$A_1(x) = f_x(x)$ x integer

$f_0(x) = x+1$

$f_{k+1}(x) = f_k \circ \cdots \circ f_k(x)$ (x times)

$f_1(x) = f_0 \circ \cdots \circ f_0(x) = 2x$ (x)

$f_2(x) = 2^x \cdot x$

$f_3(x) > 2^{2^{2^{2^{\cdots}}}}$ height x

f_k requires loop nesting k in pure form

Union-find $O(n\alpha(n))$ inverse $\alpha(n) = A^{-1}(n)$ Ackermann

Entscheidungsproblem (Hilbert-Ackermann)
(1928)

Given a logical system and a set of axioms & symbols
find a mechanistic procedure to determine "truth"

Gödel (1929) Completeness Theorem
If a logical statement is valid
then it has a proof

Gödel (1931) Incompleteness Theorem
No consistent logical system
(given by a primitive recursive
set of axioms can prove
all true statements of
number theory

Peano's Axioms

logical statements about number theory
that can't be proven true or false
(used diagonally & diagonalization)

Turing, Church, Post, Kleene (1936)
Characterize computation
- Turing Machines (Turing-Post)
- λ-calculus (Church)
LISP "LAMBDA" keyword

- μ-recursive functions (Kleene)

All equivalent: Why Turing?

Turing did this

- strong intuitive justification that this is everything
- Universal TM
  programs as data
- Undecidability of Halting Problem
- Entscheidungsproblem is not solvable
- proved equivalence of his and λ-calculus

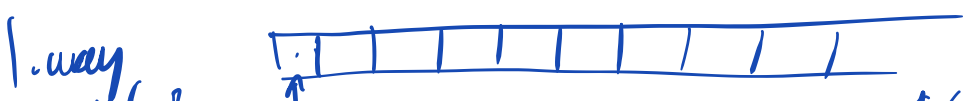
Key properties:

Finite set of states

Finite set of symbols

Tape divided into cells / squares

unbounded



all but a finite portion blank

each operation: . read a cell
. change state
. overwrite cell
. move L or R one cell

Input Alphabet Σ on tape

Finite set of symbols, not blank

Tape Alphabet $\Gamma \supseteq \Sigma$ $\sqcup \in \Gamma \setminus \Sigma$ blank

Finite set of states Q

Start state $q_0 \in Q$

Accept state $q_{acc} \in Q$

Reject state $q_{rej} \in Q$

transition function $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

$Q \setminus \{q_{acc}, q_{rej}\}$

old state — scanned symbol — new state — new symbol — left / right

$(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$