Church–Turing Thesis
Number of Turing Machines

How Many Turing Machines?

• Turing Machine ↔ Finite List of 5-tuples
  -- \( f : S \times \Sigma \rightarrow S \times \Sigma \times \{R, L\} \)
  -- \((s, x, s^*, x^*, L / R)\)

• Number of finite lists of 5-tuples in countable
  -- # Turing machines with 1 state is countable
  -- \# TM with 1 state = # 5-tuples = 2|\Sigma|^2
  -- # Turing machines with \(n\) states is countable
  -- #TM with \(n\) states = # 5-tuples = \(2n^2|\Sigma|^2\)

• Countable union of countable sets is countable
• Number of Turing Machines is countable
Non Computable Functions

Existence of Non computable Functions

• There are only countably many Turing machines

• There are uncountably many functions $f : N \rightarrow \{0,1\}$
Church–Turing Thesis

Observations

- These changes do not increase the power of the Turing machine
  - more tapes
  - nondeterminism

Conjecture

- Any problem that can be solved by an effective algorithm can be solved by a Turing machine.

- There is no computational model more powerful than a Turing machine.
Two Unsolvable Problems

Problems

1. Find a set of axioms (for arithmetic) from which all true statements can be derived.

2. Given a set of axioms (for arithmetic), find an algorithm to determine if an arbitrary statement \( w \) is a consequence of these axioms.

Godel’s Theorem

1. There is no set of axioms (for arithmetic) from which all true statements can be derived.

Turing’s Theorem

2. There exist problems that cannot be solved by a Turing machine (and hence cannot be solved by any algorithm).
Godel’s Theorem

There is no set of axioms (for arithmetic) from which all true statements can be derived.

Proof: Encode the following statement in the language of arithmetic

This statement is True, but is not Provable from the Axioms. (S)

Case 1: S is Provable.

S is False

Axioms not Consistent because a Provable statement is False

Case 2: S is Not Provable.

S is True

Axioms not Complete because a True statement is Not Provable
**Equivalent Models of Computation**

Turing Machines

Digital Computers with Unbounded Memory

Lambda Calculus

Partial Recursive Functions

Tag Systems = Finite State Automata with a Queue

Unrestricted Grammars = Grammars with Arbitrary Left Hand Sides for Rules

Post Production Systems = Grammars with Variables

Markov Algorithms

Conway’s Game of Life

One Dimensional Cellular Automata -- See S. Wolfram *A New Kind of Science*

Lindemeyer Systems
Lambda Calculus

Programming Language

- Lisp
- Scheme
- ML

Style

- Functional Programming
- Functions as Arguments to Other Functions
- Recursive Functions
Tag Systems = Post Production Systems

Description

• Finite State Automata with a Queue

• Push Down Automata with Queue in Place of Stack

Simulating a Turing Machine

• Queue = Treat as a Loop
  -- Turing Tape = Queues in Reverse Order (First In First Out)

• Move Right = Move Head of Queue to Tail of Queue

• Move Left = Move All but the First Symbol from the Head to the Tail

• If Blank, Insert an Additional Symbol onto Tail and Move All Other Symbols From Front to Back of Queue
Post Systems and Unrestricted Grammars

Post Systems

• Grammars whose Rules have Variables
• Same Power as Turing Machines

Unrestricted Grammars

• Grammars whose Rule have Arbitrary Left Hand Sides
• Same Power as Turing Machines
Grammars and Languages

Grammar = Rewrite System

\[ G = \{ N, \Sigma, R, S \} \]

• \( N \) = non-terminal symbols (upper case)
• \( \Sigma \) = terminal symbols (lower case)
• \( R \) = rewrite rules
• \( S \) = start symbol (non-terminal)

Language

\( L(G) = \) Collection of strings \( w \) in \( \Sigma^* \) derivable from the start symbol \( S \) using the rewrite rules \( R -- S \Rightarrow_{G^*} w \)
Regular Grammars and Context Free Grammars

Regular Grammars

- \( T \rightarrow a \) (non-terminal goes to terminal)
- \( T_1 \rightarrow aT_2 \) (non-terminal goes to terminal followed by non-terminal)
- Example: Regular Expressions
- Machine: Finite State Automata

Context Free Grammars

- \( T \rightarrow \textit{any string of terminal and nonterminal symbols} \)
  -- Independent of the context (what symbols surround) \( T \)
- Example: Programming Languages
- Machine: Push Down Automata
Unrestricted Grammar

Unrestricted (Context Sensitive) Grammars

• \( \alpha X \delta \rightarrow \alpha Y \delta \)

  -- \( X \) = Any String of Terminals and Non-Terminals
      Containing at Least One Non-Terminal

  -- \( Y \) = Any String of Terminals and Non-Terminals

  -- Depends on the context (what symbols surround) \( X \)

• Example: Semi-Decidable Languages

• Machine: Turing Machine
Example

Language

- \( L = \{a^n b^n c^n \mid n \geq 0\} \)

Grammar

- \( S \rightarrow aBSc \) \{Equal Number of a’s, B’s, c’s\}
- \( S \rightarrow \varepsilon \) \{Eliminate S\}
- \( Ba \rightarrow aB \) \{Move a’s to Right of B’s\}
- \( Bc \rightarrow bc \) \{Reduce B before first c to b\}
- \( Bb \rightarrow bb \) \{Reduce all remaining B’s to b\}
Markov Algorithms

Markov Grammar

- $\Sigma =$ Input Symbols
- $V =$ Non Terminal and Output Symbol
  -- Special $Accept$ and $Reject$ Symbols
- $R =$ Ordered Set of Rules
  -- Maps Strings in $V$ to Strings in $V$
  -- Continuing and Terminating Rules
- Analogous to Prolog

Operate on Input

- Apply the First Possible Rule to Current String Starting with Input
- $Halt$ when a Terminating Rule is Applied
- Accept if Final String Contains $Accept$
- Reject if Final String Contains $Reject$
- Same Power as Turing Machine
Conway’s Game of Life

Setup

• White and Black Cells on a Rectangular Grid
• Web Page: http://www.bitstorm.org/gameoflife/

Rules

• Birth: White Cells become Black if Exactly 3 Neighbors are Black
• Survival: Black Cells stay Black if 2 or 3 Neighbors are Black
• Death: Otherwise White Cells remain White and Black Cells become White

Possible Outcomes

• All Cells become White
• Configuration Becomes Stable
• Configuration Loops
• Can Encode Action of a Turing Machine on an Input String
One Dimensional Cellular Automata

Setup
• Linear Array of White and Black Cells

Rules
• New Color Determined by Current Color and Color of 2 Neighboring Cells

Possible Rules
• $2^8$ Possible Rules
• One of these Rules can Encode Turing Machines
• Wolfram’s Book: http://www.wolframscience.com/nksonline/page-23
• Web Page: http://www.hostsrv.com/webmaa/app1/MSP/webm1010/onedca
Open Question

Turtle Graphics

Is Turtle Graphics Equivalent to a Turing Machine?