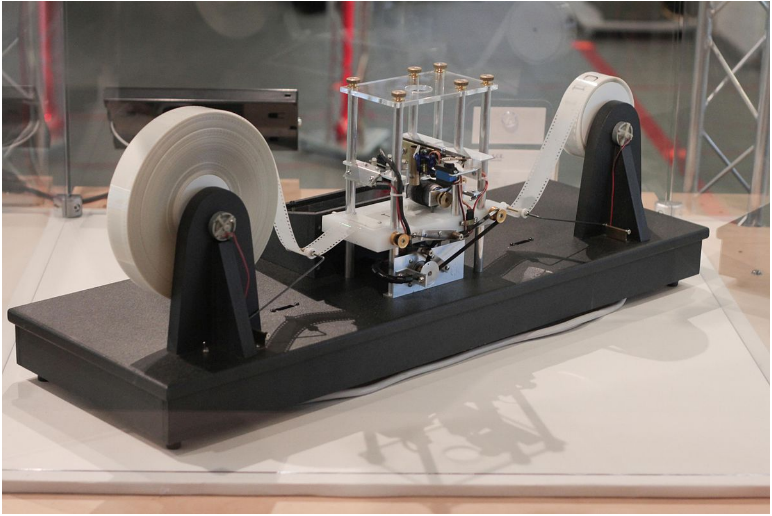
Week 1: Introduction to Introduction to the Theory of Computation



Textbook: Chapter 0 / Introduction

Motivation

- **What are the limits of what computers can do?**
- When solving a problem, we need to know how good a solution can ever be
- Some problems cannot be solved
- Must prove our programs halt in finite time on all input cases
- **Automata and language theory**: What even *are* "problems" and "computation"?
- **Computability**: What problems can be solved?
- **Complexity theory**: How well have we solved a problem?

Mathematical Prereqs

This class requires:

- **Discrete structures** (mathematical terminology and proofs)
- **CS3** (algorithms, complexity)

The remainder of this slideshow goes over specific prerequisite topics.

Sets

- Zero or more objects grouped together
- Sets can contain anything **except themselves**
- Usually enclosed by curly braces
- Can use `:` to mean "where"
  - Ex: $\{x : x \texttt{ is odd}\}$ = "the set of all x where x is odd"
- Ex: $\{1, 2, 3\}, \{5, \texttt{banana}, \pi\}, \{\}$
- The **empty set** $\{\}$ is also notated $\emptyset$

Set math

- If item a is a member of set A, we say $a \in A$. If it is not, we say $a \notin A$
- If every item in the set A is a member of the set B, we say that A is a **subset** of B and $A \subseteq B$. We also say that B is a **superset** of A and $B \supseteq A$. If $A \subseteq B$ and there is at least one element of $B \notin A$, A is said to be a **proper subset** of B and $A \subset B$
- The **union** of sets A and B is given by $A \cup B$ and is the set of all items either $\in A$, $\in B$, **or both**
- The **intersection** of A and B is $A \cap B$, and is the set of all items $\in A$ **and** $\in B$
- The **complement** of A, notated $\bar{A}$, is the set of all items $\notin A$. **Note:** This only works if you have a set of all items, called the **universe** U
- The **power set** of A, notated $\mathcal{P}(A)$, is the set of all subsets of A, including $\emptyset$.
  - Ex:
$$\mathcal{P}(\{0,1\}) = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$$

Set math examples

- A set A intersected with its complement is the empty set: $A \cap \bar{A} = \emptyset$
- A set A unioned with its complement is the universal set U: $A \cup \bar{A} = U$
- The complement of the empty set is the universal set and vice versa: $\bar{\emptyset} = U$ and $\bar{U} = \emptyset$

Why sets cannot contain themselves: Russell's paradox

Assume that sets may contain themselves. Following from this assumption is the set of all sets which *do not* contain themselves. Let this set be called S and be defined $S = \{x : x \notin x\}$.

Does S contain itself? That is to say, is $S \in S$? There are two cases: $S \in S$ and $S \notin S$.

Case 1: If $S \in S$, then by definition $S \notin S$. Therefore, $(S \in S) \rightarrow (S \notin S)$. Case 1 causes a contradiction.

Case 2: If $S \notin S$, then by definition $S \in S$. Therefore, $(S \notin S) \rightarrow (S \in S)$. Case 2 causes a contradiction.

Since both case 1 and case 2 cause contradictions, the truth value of $S \in S$ can be neither true nor false and is thus a contradiction. Thus, a "properly-formed" set must not contain itself, lest a contradiction arise.

Sequences / tuples

- A sequence is an ordered series of values, usually in parenthesis
- Ex: $(1, 1, 2, 3, 5, 8)$
- A sequence of length k is also known as a k-**tuple**

Set Cartesian / cross products

- For sets A and B, the Cartesian or cross product of A with B, notated $A \times B$, is the set of all 2-tuples $(a, b) : a \in A, b \in B$
- Ex:

$$\{0, 1\} \times \{a, b\} = \{(0, a), (0, b), (1, a), (1, b)\}$$

(Mathematical) functions

- Results are always deterministic given inputs (unlike programming "functions")
- Also commonly called mappings
- For a function f mapping A to B, we say $f : A \to B$
- Ex: "g maps even numbers to real numbers" is

$$g : \{x \in \mathbb{Z} : x \texttt{ is even}\} \to \mathbb{R}$$

“Arity”

- A function taking k arguments is called k-ary
  - Ex: f takes 12 arguments, so f is 12-ary. The “arity” of f is 12.
  - “Unary”, “binary”, and “ternary” are common special cases for 1, 2, and 3-argument functions

Some functions have special forms

- Prefix: The operator goes before the arguments (EG function calls, negation operator)
- Infix: The operator goes between the arguments (EG addition, most common in binary operators)
  - The C ternary operator is an interesting special case: `a ? b : c` has operators between each of the three arguments
- Postfix: The operator goes after the argument (EG reverse polish notation, where `1 2 +` is 3)

Graphs

- A 2-tuple $G = (V, E)$, where V is the set of vertices and E is the set of edges $E \subseteq (V \times V)$
- Digraphs (directed graphs)
  - A graph with directed edges (arrows) instead of undirected ones (lines)
  - In an undirected graph, the order of the edge between source and target $\{s, t\}$ (note the curly braces) doesn't matter
  - In a digraph, the order of the edge between source and target (s, t) (not the parenthesis) does matter
- Labeled graphs
  - Where some function λ adds labels to nodes and/or vertices

Graph Properties

- Paths / Cycles
  - A path through a graph is a series of edges where the source of the next edge is the target of the previous
  - A cycle through a graph is a path where the ending node is the beginning node
- (Strongly) connected graphs
  - A connected graph has paths of any length from every node to every other node
  - A strongly connected graph has paths of length 1 from every node to every other node

Trees

- A tree is an acyclic graph where a node has either zero or one "parent", which is a node pointing to them and zero or more "children", which are nodes pointed to by them
- A node with no parent is a "root"
  - A graph with several trees is called a "forest"
- A node with no children is a "leaf"

Alphabets, strings, and languages

- An **alphabet** is a set of characters, say $\Sigma = \{a, b, c\}$
- A **string** over an alphabet is a finite concatenation of zero or more characters from that alphabet
- The empty string is ϵ (or λ in some texts)
- The length of a string w is given by $|w|$ and is equal to the number of characters in that string
- A **language** is a set of strings

Basic boolean logic

- Boolean values are 0 (false) or 1 (true)
- `A AND B` (`A && B`): $A \wedge B$
- `A OR B` (`A || B`): $A \vee B$
- `NOT A` (`!A`): $\neg A$
- "A, but not (B or not C)": $A \wedge \neg(B \vee \neg C)$
- Identities
  - $\neg\neg A = A$
  - $\neg(A \wedge B) = \neg A \vee \neg B$
  - $\neg(A \vee B) = \neg A \wedge \neg B$
- `A IMPLIES B`: $A \rightarrow B$
  - If A, takes the value of B
  - Else, is true
  - Equivalent to $\neg A \vee B$
  - In C: `A ? B : true`
- `A IF AND ONLY IF B`: $A \iff B$
  - Commonly shortened to **iff**
  - Equivalent to $(A \rightarrow B) \wedge (B \rightarrow A)$

$\exists$ and $\forall$

- $\exists a$: "There exists some a"
  - Ex: $\exists a[b = a]$ means "there exists some a such that $b = a$"
- $\forall b$: "For all b"
  - Ex: $\forall b[b \neq a]$ means "for all b, $b \neq a$". Note that this is the negation of the previous example: $\forall b[b \neq a] \to \neg\exists a[b = a]$

Proof by construction

- Used when a theorem says that something must exist
- Simply find that thing or provide an algorithm for finding it

Example:

Thm: There is an integer larger than nine.

Pf: By construction. The number ten is an integer, and ten is larger than nine. Therefore, at least one integer is larger than nine. Therefore, the theorem holds. End of proof.

Proof by contradiction

- Assume that the theorem is false, then arrive at a contradiction
- Assume the negation of the theorem
- Follow with logical steps making no further assumptions
- Arrive at a paradox (for instance $1 = 0$)
- Since you have derived a contradiction using only known truths and one assumption, that assumption must be false

Proof by contradiction example: Irrationality of $\sqrt{2}$

A number a is rational iff there exist some integers b and c such that $a = \frac{b}{c}$. **Thm:** $\sqrt{2}$ is not rational.

Pf: By contradiction. Assume that $\sqrt{2}$ is rational. Then there must exist two integers b, c such that $\sqrt{2} = \frac{b}{c}$. Without loss of generality, choose b and c to be such that they share no divisors. Since they share no divisors, they cannot both be divisible by 2 and thus at least one of them must be odd. Then, we apply the following steps.

Multiply both sides by n: $n\sqrt{2} = m$. Square both sides: $2n^2 = m^2$.

Since n^2 is the square of an integer, m^2 is equal to 2 times some integer, making it even. Since the square of an odd is always odd (pf. excluded), m is even and thus there exists some integer k such that $m = 2k$. Substituting: $2n^2 = (2k)^2$. Simplifying: $n^2 = 2k^2$, implying that n^2 and thus n is even.

We have now derived both that n and m are even and that at least one of them is odd, causing a contradiction. End of proof.

Proof by induction

- Used if the theorem makes a claim about a countably infinite set (like the positive integers, **NOT** like the reals)
- Similar to a recursive function: Prove some easy "base case", then use an "inductive step" to prove it generally
- For example: Want to prove "$F(n)$ is true for any non-negative integer n"
  - The non-negative integers are countably infinite! We could never write out and prove all the cases in finite time
  1) Start by proving that $F(0)$ is true (base case)
  2) Then show that, for some arbitrary integer i, $F(i)$ being true implies that $F(i+1)$ is true
  - By (1), the theorem holds for $n = 0$. By (2), this implies it holds for 1, implying it holds for 2, ad infinitum
- This proves a statement over a (countably) infinite space

Proof by induction example: Sum of powers of 2

Thm: For all nonnegative integers n, $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Pf: By induction. We will first prove the base case $n = 0$, then show that, for arbitrary integer j, the theorem holding for j implies that it holds for $j + 1$ (the inductive step).

Base case: $n = 0$. When $n = 0$, $\sum_{i=0}^{0} 2^i = 2^0 = 1 = 2^1 - 1 = 2 - 1 = 1$. Thus, the theorem holds for the base case.

Inductive step: Let j be an integer for which the theorem holds. We want to show that the theorem holds for $j + 1$.

The theorem states: $\sum_{i=0}^{j} 2^i = 2^{j+1} - 1$. Adding 2^{j+1} to both sides: $\sum_{i=0}^{j} 2^i + 2^{j+1} = 2^{j+1} - 1 + 2^{j+1}$. Simplifying: $\sum i = 0^{j+1} 2^i = 2^{j+2} - 1$. This is the theorem for $j + 1$. Therefore, the theorem holding for integer i implies that it holds for $j + 1$. Since we know that it holds for 0, this means that the theorem holds for all nonnegative integers. End of proof.

End of math prereqs. Next time: Automata and languages