MODELS OF COMPUTATION

Does a programming language really need if-else, functions, arrays, loops?

Does C++ need types other than bool (int, float, char, etc.)?

Can an abacus multiply?

These are questions about what problems can be solved by different models of computation.

Well before actual computers, people were proving theorems about mathematical models of computation.

Eventually, other people implemented these models in hardware, creating the first computers.

In this section, we will learn about some of these first mathematical models of computation.
STRINGS

An alphabet is a set whose elements are symbols.

Let Σ be an alphabet. A string over Σ is a sequence of symbols in Σ.

Examples: 0110101 is a string over \{0,1\}
            hello is a string over \{a,b,c,...,x,y,z\}
            8675309 is a string over \{0,1,2,...,8,9\}

A binary string is a string over Σ = \{0,1\}.

The length of a string s, also written |s|, is the number of symbols in s.

Examples: |1200021| = 5, |cat| = 3

A substring is a subsequence of another string.

Examples: shall is a substring of shallot, hi is a substring of shin.

Let Σ be an alphabet. Then Σ* = \{strings over Σ\}.

A subset \text{L} \subseteq Σ* is a language over Σ.

Examples: \{00, 11, 000\} is a language over \{0,1\}
            \{cat, dog, hello, goodbye\} is a language over \{a,b,c,...,y,z\}

Let k \in \mathbb{N}. Then Σ^k = \{s \in Σ*: |s| = k\}. (kinda like Cartesian product)
A decision problem is a function \( p : \Sigma^* \rightarrow \{ T, F \} \) for some \( \Sigma \).

Examples:

Problem: NON-ZERO
Input: a binary number \( n \) (a string over \( \{0, 1\} \))
Output: Whether \( n \) is not equal to 0.
\[
p(00010) = F, \ p(1000) = F, \ p(0000) = T, \ p(010101) = F
\]

Problem: POWER-OF-TWO
Input: a binary number \( n \) (a string over \( \{0, 1\} \))
Output: Whether \( n \) is a power of two.
\[
p(00010) = T, \ p(1000) = T, \ p(0000) = T, \ p(010101) = F
\]

Problem: SAT
Input: a Boolean formula \( \Phi \).
Output: Whether \( \Phi \) is satisfiable.
\[
p(x) = T, \ p(\neg x_1 \lor \neg x_2) = T, \ p((\neg x_1 \lor \neg x_2) \land x_1 \land x_2) = F
\]

Every decision problem is a set membership problem: is the input in the language? (of inputs that output \( T \))
BOOLEAN FORMULAS

A Boolean formula consists of:

1. Variables \((x_1, x_2, a, b, y, z, \text{etc.})\) with universe \(\{T, F\}\).
2. Logical operators \((\neg, \land, \lor, \Rightarrow)\) (no quantifiers)
3. Constants, namely \(T\) and \(F\).

Example: \(((x_1 \lor \neg x_2) \land F \land (x_2 \lor x_3)) \lor \neg x_3\)

A literal is a variable or constant and negation (if present).

Examples (from above formula): \(x_1, \neg x_2, F, x_2, x_3, \neg x_3\).

NORMAL FORMS

A formula is in disjunctive normal form provided it looks like:
\[ (l_{1,1} \land l_{1,2} \land \ldots \land l_{1,n_1}) \lor (l_{2,1} \land l_{2,2} \land \ldots \land l_{2,n_2}) \lor \ldots \lor (l_{k,1} \land l_{k,2} \land \ldots \land l_{k,n_k}) \]
clause

A formula is in conjunctive normal form provided it looks like:
\[ (l_{1,1} \lor l_{1,2} \lor \ldots \lor l_{1,n_1}) \land (l_{2,1} \lor l_{2,2} \lor \ldots \lor l_{2,n_2}) \land \ldots \land (l_{k,1} \lor l_{k,2} \lor \ldots \lor l_{k,n_k}) \]
clause
A decision problem on Boolean formulas

A Boolean formula is **satisfiable** if the variables can be set so that the formula outputs $T$.

Example: $(x \land F) \lor (\neg x \land x_2) \lor \neg x_2$ is satisfiable. $(x_1=F, x_2=T)$

$(x, \lor \neg x_2) \land (\neg x, \lor F) \land (x_3 \land T)$ is **not** satisfiable.

**Boolean Satisfiability Problem (SAT):**

Input: a Boolean formula $\Phi$.]

Output: whether $\Phi$ is satisfiable.

SAT can be solved by a C++ program with running time $O(2.1^n)$. SAT is “easy” for formulas in disjunctive normal form:

- There is a C++ program with running time $O(n)$ solving it.

SAT is “hard” for formulas in conjunctive normal form:

- Any C++ program solving it has running time $\Omega(n^{1000})$ (probably).
GATES

A gate takes a set of truth values and outputs T or F.

**AND gate:** $x_1 \land x_2$  
**NOT gate:** $x_i \rightarrow \neg x_i$

**OR gate:** $x_1 \lor x_2$  
**XOR gate:** $x_1 \oplus x_2$

CIRCUITS

A circuit is a group of gates combined to make a "supergate":

For each combination of input truth values, a circuit outputs T or F.
A DECISION PROBLEM ON CIRCUITS

A circuit is satisfiable if the inputs can be set so the output is $T$?

Circuit Satisfiability Problem (Circuit SAT):

Input: a circuit $C$.
Output: whether $C$ is satisfiable.

Circuit SAT $\approx$ SAT because every Boolean formula is equivalent to a circuit and vice versa:

$$(x_1 \land \neg x_2 \lor \neg x_3) \land x_4$$

WIRE SPLITTING

How to turn $(x_1 \land \neg x_2) \lor (x_2 \land x_3)$ into a circuit?

Use wire splitting:
CIRCUITS ARE A MODEL OF COMPUTATION

Circuits are a model of computation.

A circuit takes as input a string over $\Sigma = \{T, F\}$ of length $k \in \mathbb{N}$ and gives as output $T$ or $F$.

So circuits solve problems that are functions $f : \{T, F\}^k \rightarrow \{T, F\}$, not decision problems.

In fact, circuits solve all such problems.

Just like we can ask about restrictions of C++ programs (no if-else?),
we can ask about restrictions of circuits (no OR gates).

UNIVERSAL GATES

A gate is universal provided every Boolean formula is the output of some circuit consisting of only that gate.

AND gate is not universal (cannot output $x, v x_a$).

NAND gate: $\overline{x_1 \land x_2} = \overline{x_1} \lor \overline{x_2}$

NAND gate is universal:

\[
\begin{align*}
&x_1 \rightarrow \overline{x_1} \\
&x_1 \land x_2 \rightarrow \overline{x_1 \land x_2} = x_1 \lor x_2
\end{align*}
\]
FINITE STATE MACHINES (FSMs)

FSMs are a model of computation that solves some decision problems.

Roughly:

- Machine consists of states and transitions.

- Input string determines a path through several states, beginning at the unique start state. → 0

- Output is determined by last state on path:
  - T if an accept state 0
  - F otherwise 0

- Moving from state to state follows transitions labelled with input symbols from $\Sigma$.

- Every state has a transition leaving it labelled $x$ for each $x \in \Sigma$.

\[ \begin{eqnarray*}
A \text{ FSM over } \Sigma = \{0, 1\} & & \\
\text{The path determined by input 10011} & & \text{Output is } T. \\
\text{A FSM over } \Sigma = \{a, b, c\} & & \\
\text{The path determined by input aabc} & & \text{output is } F.
\end{eqnarray*} \]
The language of a FSM $M$, also written $L(M)$, is the set of input strings for which $M$ outputs $T$.

Examples:

$M_1$

$L(M_1) = \{0, 1\}$

$M_2$

$L(M_2) = \{s \in \{0, 1\}^*: s \text{ has at least 2 } 1s\}$

$M_3$

$L(M_3) = \{s \in \{0, 1\}^*: s \text{ has exactly one } 0\}$

$M_4$

$L(M_4) = \{s \in \{0, 1\}^*: |s| \text{ is even}\}$

A decision problem $p: \Sigma^* \rightarrow \{T, F\}$ is solved by a FSM $M$ provided $L(M) = \{s \in \Sigma^*: p(s) = T\}$

Is every decision problem solved by a FSM?

• What about $\{s \in \{0, 1\}^*: s \text{ has more } 1s \text{ than } 0s\}$?
COUNTABILITY

A set $S$ is countable provided $|S| = |\mathbb{N}|$.

Examples: $\{2, 4, 6, 8, \ldots\}$, $\mathbb{Z}$. (proofs in lecFuncs)

Theorem: $\mathbb{Q}$ is countable.

Proof: Use the bijection $f: \mathbb{N} \to \mathbb{Q}$ defined as:

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & P \\
1 & \frac{1}{1} & \frac{3}{1} & \frac{5}{1} & \frac{7}{1} & \frac{9}{1} & \frac{11}{1} & \frac{13}{1} & \frac{15}{1} & \frac{17}{1} & \frac{19}{1} & \frac{21}{1} & \frac{23}{1} & \frac{25}{1} & \frac{27}{1} & \frac{29}{1} & \frac{31}{1} \\
2 & \frac{1}{2} & \frac{2}{2} & \frac{3}{2} & \frac{4}{2} & \frac{5}{2} & \frac{6}{2} & \frac{7}{2} & \frac{8}{2} & \frac{9}{2} & \frac{10}{2} & \frac{11}{2} & \frac{12}{2} & \frac{13}{2} & \frac{14}{2} & \frac{15}{2} & \frac{16}{2} & \frac{17}{2} & \frac{18}{2} \\
3 & \frac{1}{3} & \frac{2}{3} & \frac{3}{3} & \frac{4}{3} & \frac{5}{3} & \frac{6}{3} & \frac{7}{3} & \frac{8}{3} & \frac{9}{3} & \frac{10}{3} & \frac{11}{3} & \frac{12}{3} & \frac{13}{3} & \frac{14}{3} & \frac{15}{3} & \frac{16}{3} & \frac{17}{3} & \frac{18}{3} \\
4 & \frac{1}{4} & \frac{2}{4} & \frac{3}{4} & \frac{4}{4} & \frac{5}{4} & \frac{6}{4} & \frac{7}{4} & \frac{8}{4} & \frac{9}{4} & \frac{10}{4} & \frac{11}{4} & \frac{12}{4} & \frac{13}{4} & \frac{14}{4} & \frac{15}{4} & \frac{16}{4} & \frac{17}{4} & \frac{18}{4} \\
5 & \frac{1}{5} & \frac{2}{5} & \frac{3}{5} & \frac{4}{5} & \frac{5}{5} & \frac{6}{5} & \frac{7}{5} & \frac{8}{5} & \frac{9}{5} & \frac{10}{5} & \frac{11}{5} & \frac{12}{5} & \frac{13}{5} & \frac{14}{5} & \frac{15}{5} & \frac{16}{5} & \frac{17}{5} & \frac{18}{5} \\
6 & \frac{1}{6} & \frac{2}{6} & \frac{3}{6} & \frac{4}{6} & \frac{5}{6} & \frac{6}{6} & \frac{7}{6} & \frac{8}{6} & \frac{9}{6} & \frac{10}{6} & \frac{11}{6} & \frac{12}{6} & \frac{13}{6} & \frac{14}{6} & \frac{15}{6} & \frac{16}{6} & \frac{17}{6} & \frac{18}{6} \\
7 & \frac{1}{7} & \frac{2}{7} & \frac{3}{7} & \frac{4}{7} & \frac{5}{7} & \frac{6}{7} & \frac{7}{7} & \frac{8}{7} & \frac{9}{7} & \frac{10}{7} & \frac{11}{7} & \frac{12}{7} & \frac{13}{7} & \frac{14}{7} & \frac{15}{7} & \frac{16}{7} & \frac{17}{7} & \frac{18}{7} \\
8 & \frac{1}{8} & \frac{2}{8} & \frac{3}{8} & \frac{4}{8} & \frac{5}{8} & \frac{6}{8} & \frac{7}{8} & \frac{8}{8} & \frac{9}{8} & \frac{10}{8} & \frac{11}{8} & \frac{12}{8} & \frac{13}{8} & \frac{14}{8} & \frac{15}{8} & \frac{16}{8} & \frac{17}{8} & \frac{18}{8} \\
9 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
10 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
11 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
12 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
13 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
14 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
15 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
16 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
17 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
18 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

 noticed if $\frac{p}{q}$ is reducible.

\[
\begin{array}{c|c}
\text{x} & \text{f(x)} \\
1 & 0 \\
2 & 1 \\
3 & -1 \\
4 & \frac{1}{2} \\
5 & -\frac{1}{2} \\
6 & 2 \\
7 & -2 \\
8 & -3 \\
9 & -3 \\
10 & \frac{1}{3} \\
11 & -\frac{1}{3} \\
12 & \frac{1}{4} \\
13 & -\frac{1}{4} \\
14 & \frac{2}{3} \\
15 & -\frac{2}{3} \\
16 & \frac{3}{2} \\
17 & -4 \\
\end{array}
\]
Theorem: \( \mathbb{R} \) is not countable.

Proof: By contradiction.
Suppose \( \mathbb{R} \) is countable.
So \( \exists \) a bijection \( f: \mathbb{N} \to \mathbb{R} \).
So every real number has a natural number that \( f \) maps to it.

Consider the real number \( r = 0.d_1d_2d_3d_4d_5d_6\ldots \)
where \( d_i = 0 \) if the \( i \)th digit of \( f(i) \neq 0 \)
\( d_i = 1 \) if the \( i \)th digit of \( f(i) = 0 \)

So \( \forall i \in \mathbb{N}, f(i) \neq r \), since the \( i \)th decimal of \( r \) and \( f(i) \) are not equal.
Contradiction!

Theorem: There is a decision problem not solved by any FSM.

Idea: \( |\{FSM\ s\}| = |\mathbb{N}| \), \( |\{\text{decision problems}\}| = |\mathbb{R}| \), \( |\mathbb{N}| \neq |\mathbb{R}| \)

Proof: By contradiction.
So every decision problem has a FSM that solves it.
So every language over \( \{0,1\} \) has a FSM.
Number the machines \( M_1, M_2, M_3, \ldots \)

Consider the problem \( P: \{0,1\}^* \to \{T,F\} \)
where \( P(00\ldots 0) = T \) if \( M_i \) outputs \( T \) on \( 00\ldots 0 \)
\( P(00\ldots 0) = F \) if \( M_i \) outputs \( F \) on \( 00\ldots 0 \)

So \( \forall i \in \mathbb{N}, P \) is not solved by \( M_i \),
since \( P(00\ldots 0) \) and output of \( M_i \) are not equal. Contradiction!