How do we define a model of computation?

Input: finite sequence of symbols

alphabet Σ: finite set of symbols $\{0,1\}$, $\{a,b,c,\ldots z\}$ (binary alphabet)

string (or word) over Σ: finite sequence of symbols from Σ | 010 hello

$|w|$ = length of w

w_i = i^{th} element of w so if $|w|=n$ then $w = w_1 w_2 \ldots w_n$

Σ^k: strings over Σ of length k (k nonnegative integer) ($k \geq 0$)

$\Sigma^0 = \{\epsilon\}$ ← empty string

Σ^* = all strings over Σ $= \bigcup_{k \geq 0} \Sigma^k$ (union)

$\{0,1\}^2 = \{00, 01, 10, 11\}$

$\{0,1\}^3 = \{000, 001, \ldots\}$

$\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

What about graphs or other kinds of input? Use binary encodings.

Outputs: for every possible input, YES or NO

decision problem: computational problem with a yes-or-no answer

e.g. does a binary string contain an even # of 1s?

011 → yes

0100 → no

ϵ → yes (exactly 0 ones)

$|0| = 1$

$|00| = 2$

$|\epsilon| = 0$

Note: to specify a decision problem, it's enough to identify the strings with answer YES — it's "language"

a language over alphabet Σ is a subset $L \subseteq \Sigma^*$

the decision problem for L: given $w \in \Sigma^*$, decide whether $w \in L$

e.g. L_{103} = "all strings containing the substring 103 (over English alphabet)"

CSE103 $\in L_{103}$

CSE10 $\notin L_{103}$

310303 $\in L_{103}$

$\mathcal{L}_{PRIME}$ = "binary strings encoding prime numbers"

$10 \in \mathcal{L}_{PRIME}$ "2"

$101 \in \mathcal{L}_{PRIME}$ "5"

$1001 \notin \mathcal{L}_{PRIME}$ "9"

$\in$ "is a member of"

$S = \{1, 4, 9\}$

$1 \in S$

$3 \notin S$

$\notin$ "is not a member of"

Procedure: mapping from inputs to outputs

In general, a function $f : \Sigma^* \to \{0, 1\}$

NO YES

$$\mathcal{L} = \{x \in \Sigma^* \mid f(x) = 1\}$$

x: variable

$\Leftarrow$ all $x \in \Sigma^*$ such that $f(x) = 1$

However:

you can have a well-defined function f without an "effective procedure" to evaluate it

$$f(x) = \begin{cases} 1 & \text{if } x \text{ encodes a Python program that terminates} \\ 0 & \text{otherwise} \end{cases}$$

Later we'll see there is no effective procedure to compute f

$\{x \in \mathbb{Z} \mid \exists y \in \mathbb{Z}.\ x = y^2\}$

$\mathbb{Z}$: all integers $= \{1, 4, 9, 16, \ldots\}$

$\{\} = \text{empty set} = \emptyset$

$\emptyset \overset{?}{\notin} \Sigma^*$ $\quad \in \in \Sigma^*$

$\{1, 2\} \notin \Sigma^*$ $\quad$ $\in$: empty string

"Effective procedure" = a function that can be computed

Define "procedure" by way of an abstract machine that maps inputs to outputs

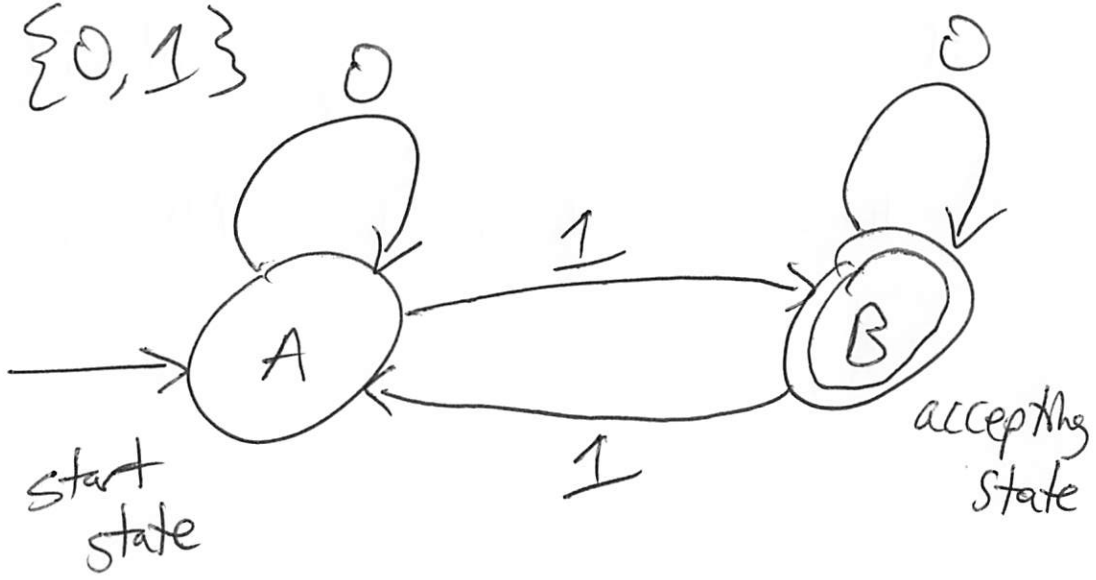
Today: Finite Automaton

Finite Automata: "computers with limited memory"

- machine M that reads an input string 1 symbol at a time
- has a finite number of modes or states (like the state of memory)
- transitions to the next state based on the input symbol and the current state
- after reading ~~every~~ all symbols accepts the input if you end up in an accepting state; otherwise it rejects the input
- language of M = $L(M)$ = set of accepted strings

Draw as directed graph: vertex for each state
edge for each transition

$\Sigma = \{0, 1\}$



state q	input s	next state $\delta(q, s)$
A	0	A
A	1	B
B	0	B
B	1	A

$w =$ 0 0 1

states A A A B ✓ accepts 0 0 1

$w =$ 1 1 0

states A B A A ✗ rejects 1 1 0