* Languages as abstractions of problems

Every computational problem can be thought of as a function $f: \{0,1\}^* \to \{0,1\}^*$

$\{0,1\}^*$ = set of all binary strings of finite length

$L_f = \{ (x, i, b) \mid i^{th} \text{ bit of } f(x) \text{ is } b \}$

$b \in \{0, 1, \perp\}$

↳ denotes the end of the string corresponding to $f(x)$

P1: Given x, compute $f(x)$

P2: Given (x, i, b), check if $(x, i, b) \in L_f$

Theorem: $\exists$ an algorithm for P1 iff $\exists$ an algorithm for P2

Encoding L_f:

(	-	000
)	-	001
0	-	010
1	-	011
,	-	100
$\perp$	-	101

This encoding shows that $L_f \subseteq \{0,1\}^*$

Language: Given an alphabet Σ (say $\{0,1\}$)

a language $L \subseteq \Sigma^*$

Decision problems: For $L \subseteq \Sigma^*$, given $x \in \Sigma^*$ decide if $x \in L$

* Can all computational problems solved by writing a C program?

Theorem: The # of C programs is countable

Proof: Every C program can be encoded as a binary string

$\Rightarrow$ Set of C-programs $\subseteq \{0,1\}^*$

$\exists$ bijection $f: \{0,1\}^* \to \mathbb{N}$

$f(0)=1 \quad f(1)=2 \quad f(00)=3 \quad f(01)=4 \quad f(10)=5$
$f(11)=6 \quad f(000)=7 \ldots\ldots$

Exercise: Show that f is a bijection

$\{0,1\}^* = \epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011 \ldots$
$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$
$0 \; 1 \; 2 \; 3 \; 4$

$\overline{b} = \{0,1\}^* - \epsilon$

$f(\overline{b}) = n(\overline{b}) + 2^{|\overline{b}|-1}$

Theorem: The set of languages $L \subseteq \{0,1\}^*$ is uncountable

The set of all languages $L = \mathcal{P}(\{0,1\}^*)$ is the power set of $\{0,1\}^*$

By contradiction: Let $f: \{0,1\}^* \to \mathcal{P}(\{0,1\}^*)$ be a bijection

f	ϵ	0	1	00	01	10	11	000	001
ϵ	1	0	0	1	0	0	1	1	1
0	0	1	1	0	0	1	0	1	0
1	0	0	0	1	1	0	0	1	0
00	1	1	1	0	0	0	1	1	1
01	0	1	0	1	0	1	0	1	0
10	1	1	1	1	1	1	1	0	0

1 1 0 0 0 1 ... flip

0 0 1 1 1 0 ...

↳ This does not exist as any row in the table given

Cantor's theorem: For any set S, there does not exist a bijection $f: S \to \mathcal{P}(S)$

Proof: Suppose that $\exists f: S \to \mathcal{P}(S)$ that is a bijection

Consider the set $T = \{x \mid x \notin f(x)\}$

Suppose that $T = f(t)$. Then

$$t \in T \iff t \in f(t) \iff t \notin T$$

The set of languages is uncountable, whereas the set of C programs is countable

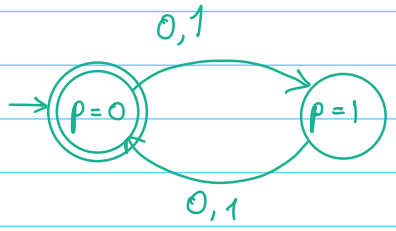
Finite Automata

- Programs that can read the input once
- The values stored in any variable is a constant, independent of the length of the input

Q: Write a program that checks if the length of the input is even

```
p = 0
while inp[i] ≠ ⊥
    p = 1 - p
return p
```

≡



* Captures many simple computational problems
  - Lexical analyzer of a compiler

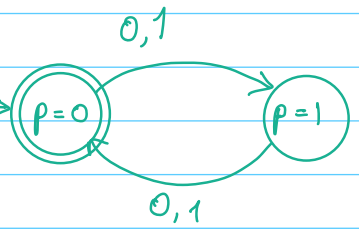
Finite Automata

- Programs that can read the input once
- The values stored in any variable is a constant, independent of the length of the input

Q: Write a program that checks if the length of the input is even

```
p = 0
while inp[i] ≠ ⊥
    p = 1 - p
return p
```

≡



* Captures many simple computational problems
  - Lexical analyzer of a compiler

→ As a language L = set of all binary strings of even length

if $x \in L$, accept → after reading the input should be in an accepting state

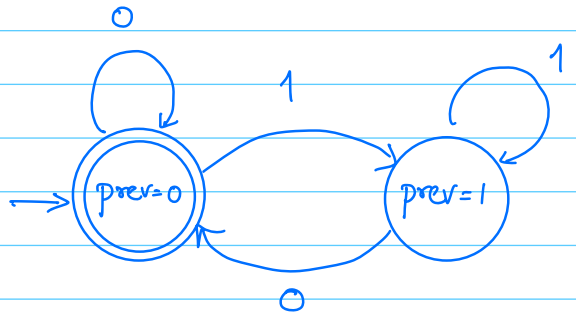
if $x \notin L$, reject

↳ after reading the input should be in a non-accepting state

Examples

① set of strings $w \in \{0,1\}^*$ that correspond to even integers

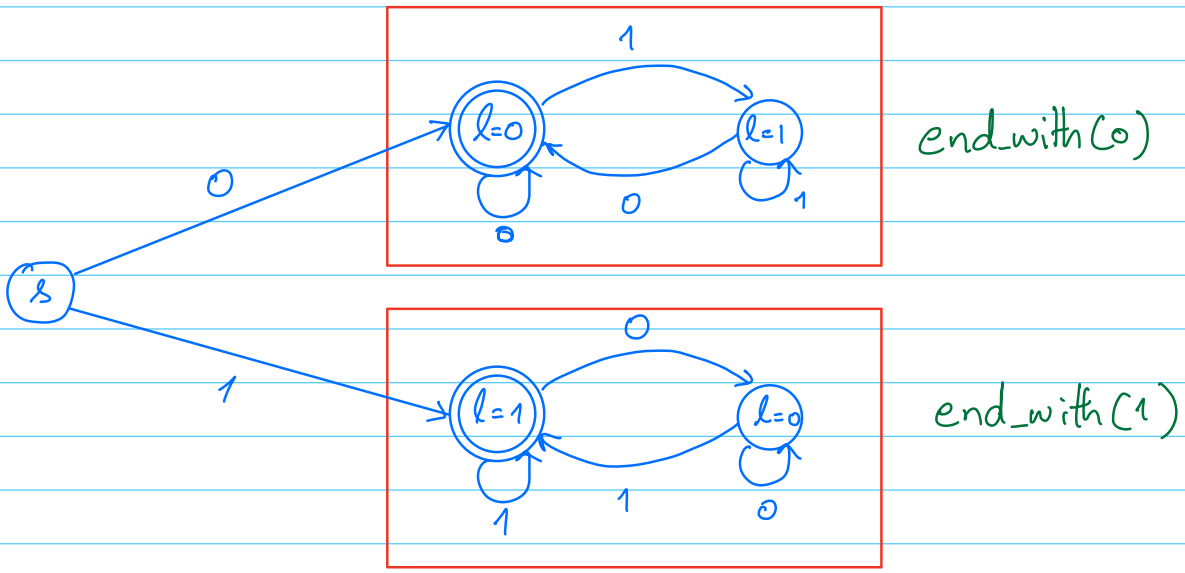
```
prev ← 0
while a[i] ≠ ⊥
    prev = a[i]
return prev
```



② strings that start and end with the same bit

```
end_with(b)
    last = b
    while a[i] ≠ ⊥
        last = a[i]
    if last = b
        return 1
    else 0
```

```
main()
    first ← a[1]
    end_with(first)
```



Definition: A Deterministic Finite Automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

* Q - set of states
* Σ - alphabet
* δ - transition function
  $\delta: Q \times \Sigma \to Q$
* q_0 - start state
* $F \subseteq Q$ - set of final/accepting states

Computation by a DFA: $M = (Q, \Sigma, \delta, q_0, F)$

$w \in \Sigma^* \longrightarrow$ input : $w = \sigma_0 \sigma_1 \sigma_2 \dots \sigma_n$

- $q_0 \to q_1 = \delta(q_0, \sigma_0) \to q_2 = \delta(q_1, \sigma_1) \to q_3 = \delta(q_2, \sigma_2)$
  $\dots \to q_{n+1} = \delta(q_n, \sigma_n)$

  $f(w) = 1$ iff $q_{n+1} \in F$

  $\equiv$ $w \in L(M)$ iff $q_{n+1} \in F$

- $q_{n+1} = \delta(\delta(\dots \delta(\delta(\delta(q_0, \sigma_0), \sigma_1) \dots, \sigma_n)$

extended transition function: $\hat{\delta} : Q \times \Sigma^* \to Q$

Base case: $\hat{\delta}(q, \sigma) = \delta(q, \sigma)$ when $|\sigma| = 1$

Recursive step: $\hat{\delta}(q, w\sigma) = \delta(\hat{\delta}(q, w), \sigma)$

Language of a DFA M: For a DFA $M = (Q, \Sigma, \delta, q_0, F)$

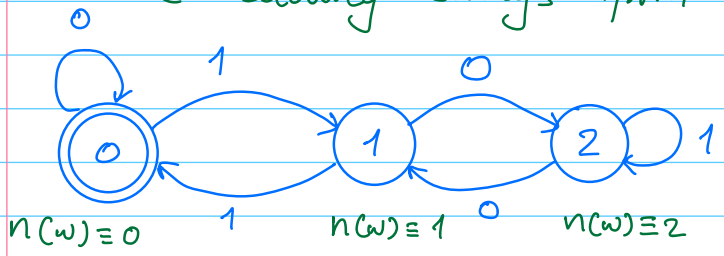
$$L(M) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \}$$

A language $L \subseteq \Sigma^*$ is said to be regular if $\exists$ DFA M s.t. $L(M) = L$.

Set of binary strings that are the binary representations of numbers divisible by 3

$000 \in L \quad 001 \notin L \quad 00011 \in L \quad \ldots$

(Reading strings from MSB)



$n(w)$ = # corr. to w

$n(w0) = 2 \cdot n(w)$

$n(w1) = 2 \cdot n(w) + 1$

$\delta(q, \sigma) = 2q + \sigma \pmod 3$ → stronger induction hypothesis

$\hat{\delta}(0, w) \in L \iff \hat{\delta}(\hat{\delta}(0, w'), \sigma) \in L$

$\hat{\delta}(0, w) = \delta(\hat{\delta}(0, w'), \sigma)$ where $w = w'\sigma$

$= \delta(n(w') \bmod 3, \sigma)$

$= 2(n(w') \bmod 3) + \sigma \pmod 3$

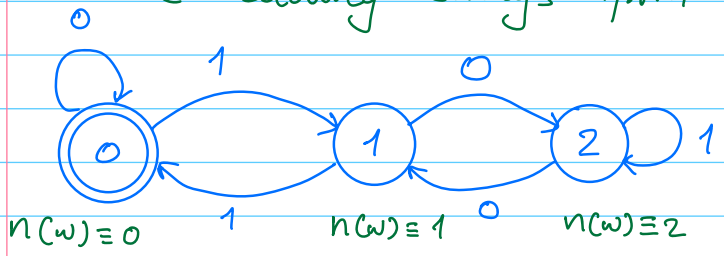
$= 2\, n(w') + \sigma \pmod 3$

$= n(w) \pmod 3$

Set of binary strings that are the binary representations of numbers divisible by 3

$000 \in L \quad 001 \notin L \quad 00011 \in L \quad \ldots$

(Reading strings from MSB)



$n(w)$ = # corr. to w

$n(w0) = 2 \cdot n(w)$

$n(w1) = 2 \cdot n(w) + 1$

$\delta(q, \sigma) = 2q + \sigma \pmod 3$ → stronger induction hypothesis

$\hat{\delta}(0, w) \in L \iff \hat{\delta}(\hat{\delta}(0, w'), \sigma) \in L$

$\hat{\delta}(0, w) = \delta(\hat{\delta}(0, w'), \sigma)$ where $w = w'\sigma$

$= \delta(n(w') \bmod 3, \sigma)$

$= 2(n(w') \bmod 3) + \sigma \pmod 3$

$= 2\, n(w') + \sigma \pmod 3$

$= n(w) \pmod 3$

Closure properties of languages

- $L_1 \cup L_2$ - set union
- $L_1 \cap L_2$ - set intersection
- $L_1 \cdot L_2$ - concatenation

$= \{ w \cdot w' \mid w \in L_1, w' \in L_2 \}$

Eg ① $L_1, L_2 \subseteq \{0,1,2,..,9\}^*$

$L_1 = \{ n \in \mathbb{N} \mid n \text{ is even} \}$

$L_2 = \{ n \in \mathbb{N} \mid n \text{ is odd} \}$

Qn: $L_1 \cdot L_2 = \{ n \in \mathbb{N} \mid n$ contains 0,2,4,6,8 & ends in 1,3,5,7,9 $\}$

② $L_1 = \{ w \in \{0,1\}^* \mid w$ contains ≥ 2 zeroes $\}$

$L_2 = \{ w \in \{0,1\}^* \mid w$ contains ≥ 3 zeroes $\}$

$L_1 \cdot L_2 = \{ w \in \{0,1\}^* \mid w$ contains ≥ 5 zeroes $\}$

③ $L_1 = \{ w \in \{0,1\}^* \mid w$ contains ≥ 2 zeroes $\}$

$L_2 = \{ w \in \{0,1\}^* \mid w$ contains ≤ 3 zeroes $\}$

$L_1 \cdot L_2 = L_1$

$$L^i = L \cdot L^{i-1} \quad ; \quad L^0 = \{\epsilon\}$$

- Asterate: $L^* = \bigcup_{i \geq 0} L^i$

eg: $L_{odd} = \{ w \in \{0,1\}^* \mid |w| \equiv 1 \pmod 2 \}$

$L_{even} = \{ w \in \{0,1\}^* \mid |w| \equiv 0 \pmod 2 \}$

$L_{even}^* = L_{even}$

$L_{odd}^* = \{0,1\}^*$

Product construction

Theorem: If $L_1, L_2 \subseteq \Sigma^*$ are regular, Then

(i) $L_1 \cup L_2$ is regular

(ii) $\bar{L_1}$ is regular

(iii) $L_1 \cap L_2$ is regular

Proof: (i) $\exists\ M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ &

$M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ s.t

$L(M_1) = L_1$ & $L(M_2) = L_2$

Construct M s.t $L(M) = L_1 \cup L_2$

- Simulate M_1 & M_2 simultaneously

$Q = Q_1 \times Q_2$ → to keep track of M_1 & M_2

$\delta : Q_1 \times Q_2 \times \Sigma \to Q_1 \times Q_2$

$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$

↓ simulation of 1 step of M_1 & M_2

$s = (s_1, s_2)$ → both M_1 & M_2 start from their respective start states

$F = (Q_1 \times F_2) \cup (F_1 \times Q_2)$

↳ at least one of M_1 & M_2 should land in an accepting state

Product construction

Theorem: If $L_1, L_2 \subseteq \Sigma^*$ are regular, then

(i) $L_1 \cup L_2$ is regular

(ii) $\overline{L_1}$ is regular

(iii) $L_1 \cap L_2$ is regular

Proof: (i) $\exists\ M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ &

$M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ s.t

$L(M_1) = L_1$ & $L(M_2) = L_2$

Construct M s.t $L(M) = L_1 \cup L_2$

- Simulate M_1 & M_2 simultaneously

$Q = Q_1 \times Q_2$ → to keep track of M_1 & M_2

$\delta : Q_1 \times Q_2 \times \Sigma \to Q_1 \times Q_2$

$\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$

↓ simulation of 1 step of M_1 & M_2

$s = (s_1, s_2)$ → both M_1 & M_2 start from their respective start states

$F = (Q_1 \times F_2) \cup (F_1 \times Q_2)$

↳ at least one of M_1 & M_2 should land in an accepting state

Claim: $L(M) = L_1 \cup L_2$

① $\hat{\delta}((q_1, q_2), w) = (\hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w))$

Base case: By definition

Induction step: $w = w'\sigma$

$$\hat{\delta}((q_1, q_2), w\sigma) = \delta(\hat{\delta}((q_1, q_2), w), \sigma)$$
$$= \delta((\hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w)), \sigma)$$
$$= (\delta_1(\hat{\delta}_1(q_1, w), \sigma), \delta_2(\hat{\delta}_2(q_2, w), \sigma))$$
$$= (\hat{\delta}_1(q_1, w\sigma), \hat{\delta}_2(q_2, w\sigma))$$

② $\hat{\delta}((s_1, s_2), w) \in (Q_1 \times F_2) \cup (F_1 \times Q_2)$

iff $\hat{\delta}_1(s_1, w) \in F_1$ or $\hat{\delta}_2(s_2, w) \in F_2$

$$\hat{\delta}((s_1, s_2), w) = (\hat{\delta}_1(s_1, w), \hat{\delta}_2(s_2, w))$$

$$\downarrow$$

$$(Q_1 \times F_2) \cup (F_1 \times Q_2)$$

$$\iff \hat{\delta}_1(s_1, w) \in F_1 \text{ or } \hat{\delta}_2(s_2, w) \in F_2$$

(ii) $\forall w \in \Sigma^*$ $\hat{\delta}(s,w) \in F$ or $\hat{\delta}(s,w) \in Q \setminus F$

$w \notin L \iff \hat{\delta}(s,w) \in Q \setminus F$

$M' = (Q, \Sigma, \delta, s, Q \setminus F)$

$L(M') = \overline{L(M)}$ for $M = (Q, \Sigma, \delta, s, F)$

(iii) $L_1 \cap L_2 = \overline{\overline{L_1 \cap L_2}} = \overline{\overline{L_1} \cup \overline{L_2}}$

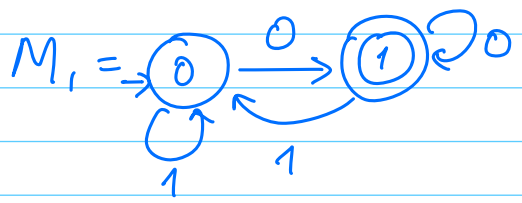
Exercise: Try using the product construction

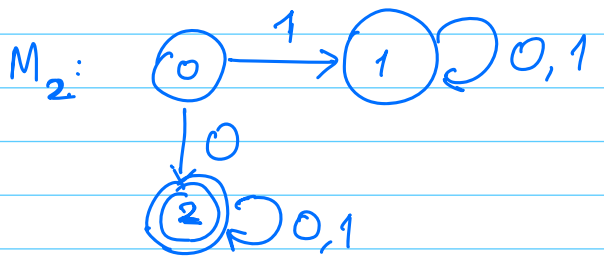
Qn: If L_1 & L_2 are regular, is $L_1 \cdot L_2$ regular?

$L_1 = \{ w \in \{0,1\}^* \mid w \text{ ends with a } 0 \}$

$L_2 = \{ w \in \{0,1\}^* \mid w \text{ starts with a } 0 \}$

$L = L_1 \cdot L_2$





Can you patch up M_1 and M_2?

- The new DFA should guess when it should stop the simulation of M_1 and start simulating M_2

- This is captured by the idea of non-determinism