Computability and Complexity Theory.

- Formalize these notions:

Computational problem. $\equiv$ Language.

Computer. $\equiv$ Turing machine

Algorithm to solve a computational problem $\equiv$ TM program that decides a language.

- Decidable language

Problem that has an algorithm

- TM program halts on every input and gives correct YES/NO answer.

- There are undecidable languages!
  - Halting problem.
  - Hilbert's 10^{th} problem.

- For decidable languages:
  - Running time $\equiv$ # TM steps. (as function of input size)

- Complexity Theory.
  - Classify languages according to their complexity in terms of running time, space, ...
  - P, NP, NP-completeness
  - ......
  - NP-complete problems: Traveling Salesperson,
    $\boxed{\text{Reductions}}$ 3SAT,
    MAX-CUT,
    SUBSET-SUM*, ...

Languages

Def. Alphabet is a finite set of symbols. Eg.

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

$\Sigma = \{a, b, .., z\}$.

$\Gamma = \{a\text{-}z, A\text{-}Z, \#, ?, .., \$\}$

Def. String over an alphabet Σ is a finite sequence of symbols from Σ.

- Length of a string = number of symbols.
- E.g. $\Sigma = \{0,1\}$.

  strings: $0, 1, 00, 11010, \ldots$

Def Empty string denoted by ε, has length zero

Def Σ^* is set of all (finite length) strings.

E.g. $\Sigma = \{0,1\}$.

$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, \ldots\}$.

Def A language L is a subset of Σ^*.

$L \subseteq \Sigma^*$.

Examples

$\Sigma = \{0,1\}$. L_{even} = Set of all even length strings.

$\Sigma = \{0,1,\ldots,9\}$ L_{primes} = Set of all strings that, in decimal, are prime integers.

$\Sigma = \left\{ \begin{matrix} a\text{-}z, A\text{-}Z, \\ \#, \&, \ldots, \$ \end{matrix} \right\}$. $L_{programs}$ = Set of all strings that are valid C programs

Language recognition problem

Def For every language L, the associated recognition problem is ("decision problem")

"Given $\underset{\text{input}}{\underline{\text{string}}}$ $x \in \Sigma^*$, is $x \in L$?"

Examples

$L_{even} \rightarrow$ Given x, is $|x|$ even?

$L_{primes} \rightarrow$ Given x, does x represent a prime integer?

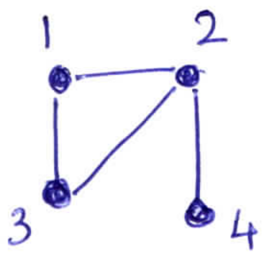
$L_{programs} \rightarrow$ Given x, is x a valid C program (compiler!).

"Fact" "Every" computational problem (decision) can be cast as the recognition problem for an appropriate language.

Encoding ("objects" by strings).

Example Encoding Graphs.

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



G

$$\longrightarrow \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 1 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ \hline \end{array} \longrightarrow 0110\#1011\#1100\#0100$$

Adj. matrix

$\langle G \rangle$ Encoding as a string.

L_{Graphs} = Set of all valid encodings of graphs

$= \{ \langle G \rangle \mid G \text{ is a graph}\} \subseteq \{0,1,\#\}^*$.

$L_{\text{conn}} = \{ \langle G \rangle \mid G \text{ is a connected graph} \} \subseteq L_{\text{Graphs}}$

$L_{\text{bipartite}} = \{ \langle G \rangle \mid G \text{ is a bipartite graph} \}$ "

$L_{\text{Hamiltonian}} = \{ \langle G \rangle \mid G \text{ has a Hamiltonian cycle} \}$ "

Hence the associated recognition problems are

- Is G connected ?
- Is G bipartite ?

} in P (= class of prob./lang. w/ polytime algo).

- Is G Hamiltonian ? } NP-complete.

<u>Encoding</u> Similarly one can encode

- numbers, graphs, polynomials, ...
- (TM) programs! (e.g. a C-program is already a string)

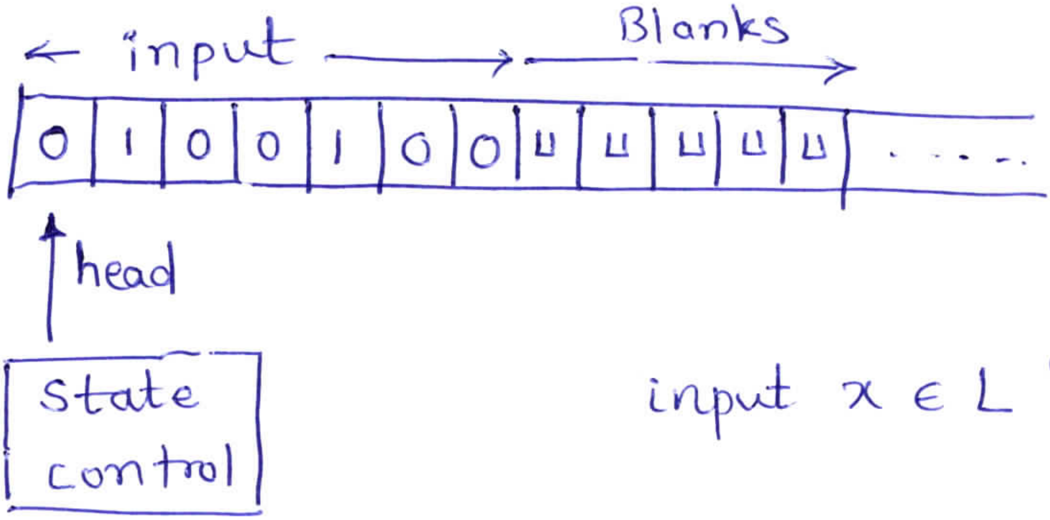
and cast computational problems as language recognition problems!

<u>Computational model</u> ("device", "computer")

- "weak" models.
  - Finite automata — Regular langs.
  - Push-down automata — Context free langs
- "Definitive" model.
  - Turing machine. — Decidable langs
    $\equiv$ "real" computer.

Turing Machine

- Alan Turing 1936.
- Model for "general purpose" computer.
- "TM can do everything that a real computer can do."



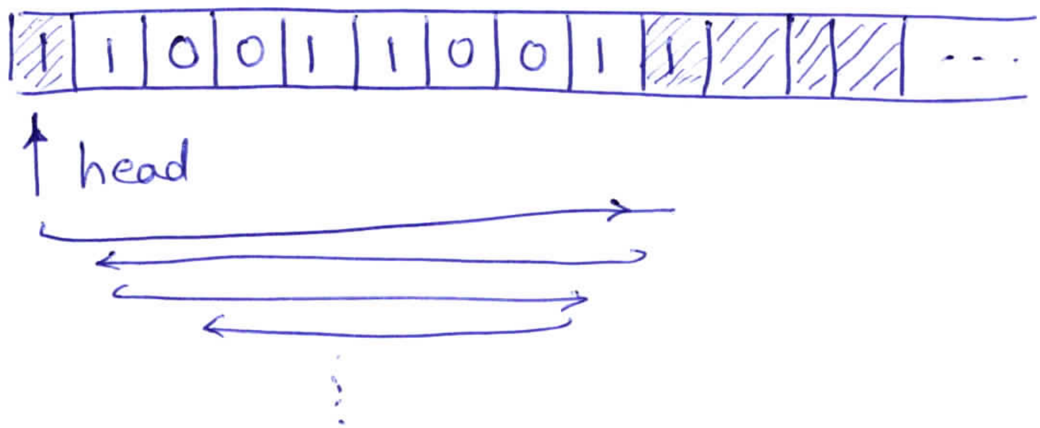
input $x \in L$?

- A $\frac{\text{move}}{\text{step}}$:
  - Read symbol scanned by the head.
  - (over-) write, (possibly) change state, move head Left or right.

- Computation :
  - Start w/ initial configuration.
  - Make moves.
  - $\frac{\text{Accept / Reject}}{\text{YES / NO}}$ after entering appropriate states.

Example Deciding Palindromes.

$$L = \{ w \mid w \in \{0,1\}^*, \; w = w^R \}.$$



- "Remember" 1. Write .
- Traverse right till you hit , turn left.
- "match" 1. write
- Move left till you hit , turn right.

⋮

- Accept if all symbols "matched".
  Reject if an "unmatch" is detected.
- Running time $O(n^2)$. if $n = |w|$.

Formal Definition of TM

A (finite) set of instructions of type:

(current state, symbol read) $\rightarrow$

(new state, symbol written, Left/Right)

Def. A TM M is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_{start}, q_{accept}, q_{reject})$$

where
- Q is a set of states.
- Σ is input alphabet.
- Γ is tape alphabet, $\Sigma \subseteq \Gamma$
  $\sqcup \in \Gamma \setminus \Sigma$.
- $q_{start}, q_{accept}, q_{reject} \in Q$. are three special states.
- Transition function.

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}.$$

__Example__ $L = \{ w \mid |w| \text{ is even} \}, \; \Sigma = \{0\}$

0	0	0	...	0	0	⊔	⊔	⊔	...

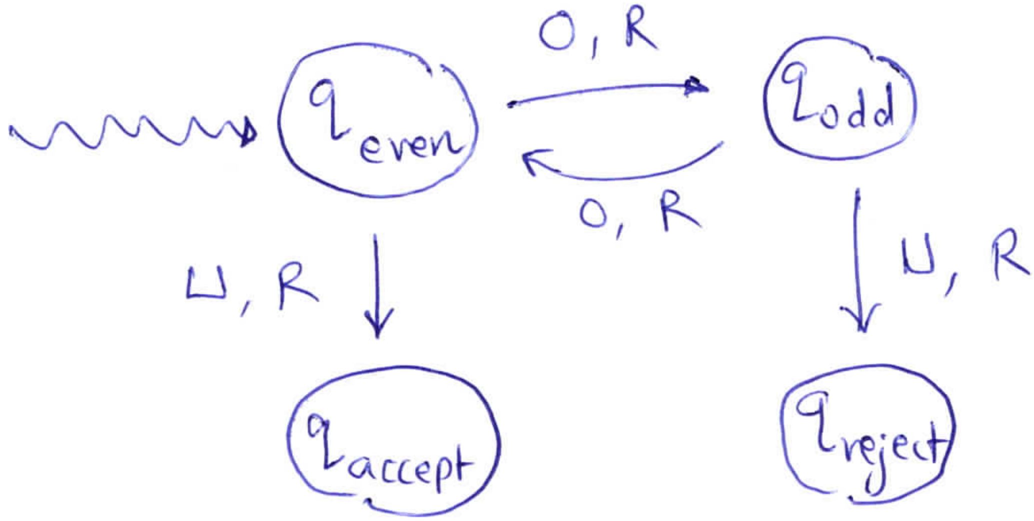
- $\Sigma = \{0\}$. $\Gamma = \{0, \sqcup\}$.
- $Q = \{ q_{start} = q_{even}, q_{odd}, q_{accept}, q_{reject} \}$
- $\delta(q_{even}, 0) = (q_{odd}, 0, R)$

  $\delta(q_{odd}, 0) = (q_{even}, 0, R)$

  $\delta(q_{even}, \sqcup) = (q_{accept}, \sqcup, R)$

  $\delta(q_{odd}, \sqcup) = (q_{reject}, \sqcup, R)$.

__State diagram__



Def A configuration of a TM consists of (contents of tape, state, head position).

0	1	#	1	0	1	0	⊔	⊔	⊔	⊔	...
				↑ q_{17}							

Def Initial configuration

$\boxed{\text{Is } x \in L \text{ ?}}$

← input →

x_1	x_2	x_3	...	x_n	⊔	⊔	⊔	...
↑ q_{start}								

input $x = x_1 x_2 .. x_n \in \Sigma^*$.

Computation - Start in the initial config.

- Machine keeps making moves until the state reaches q_{accept} or q_{reject}.
- If so, machine stops and accepts or rejects the input respectively.

$x \notin L$. $x \in L$.