INT201 Decision, Computation and Language

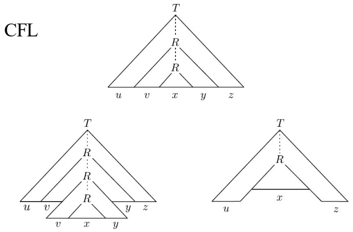
Lecture 9 – Turing Machine and Variants

Dr Yushi Li & Dr Chunchuan Lyu

Xi'an Jiaotong-Liverpool University
西交利物浦大學

Recap

- Equivalence between PDA and CFL
- Pumping Lemma for CFL



Today

- Turing Machine
- Turing-recognizable and Turing-decidable languages
- Multi-tape TM and Nondeterministic TM

Alan Turing 1912-1954

English mathematician, computer scientist, logician, cryptanalyst, philosopher and theoretical biologist

Run marathon in 2:46:03
while the 1948 Olympic Marathon winner was 2:34:52

- Computability and λ-Definability (1936)
- On computable numbers, with an application to the Entscheidungsproblem (1937)
- Computing Machinery and Intelligence (1950)

What is the limit of computation?

Can machines think?

DFA, NFA and PDA

DFA

- $M = (Q, \Sigma, \delta, q, F)$
- $\delta : Q \times \Sigma \rightarrow Q$

Finite control (δ) based on

- State
- Input symbol

NFA

- $M = (Q, \Sigma, \delta, q, F)$
- $\delta : Q \times \Sigma_\epsilon \rightarrow P(Q)$

PDA

- $M = (Q, \Sigma, \Gamma, \delta, q, F)$:
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow Q \times \Gamma_\varepsilon{}^*$

Finite control (δ) based on

- State
- Input symbol
- Variable popped from stack

Turing Machine

Finite Automata	Pushdown Automata	Turing Machine
Regular	Context-free	Regular, context-free, context-sensitive, recursively enumerable.

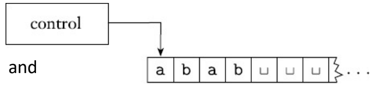
Previous machines can be used to accept or generate regular and context-free languages. However, they are not powerful enough to accept simple language such as

$$A = \{a^m b^n c^{mn} : m \geq 0, n \geq 0\}.$$

Turing machine is a simple model of real computer.

Turing Machine

- infinitely long tape, divided into cells. Each cell stores a symbol belonging to Γ (tape alphabet).



- Tape head (↓) can move both right and left, one cell per move. It read from or write to a tape
- State control can be in any one of a finite number of states Q. It is based on: state and symbol read from tape
- Machine has one start state, one accept state and one reject state.
- Machine can run forever: infinite loop.

Properties of Turing Machine

- Turing machine can both read from tape and write on it.
- Tape head can move both right and left.
- Tape is infinite and can be used for storage.
- Accept and reject states take immediate effect.

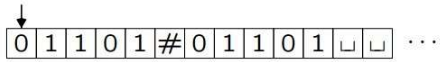
Turing Machine

Example

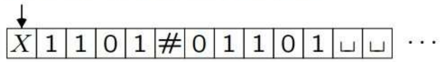
Machine for language $A = \{\, s\#s \mid s \in \{0, 1\}^* \,\}$, input string is 01101#01101 ∈ A.

Idea: Zig-zag across tape, crossing off matching symbols.

- Consider string $01101\#01101 \in A$.
- Tape head starts over leftmost symbol



- Record symbol in control and overwrite it with X



- Scan right: *reject* if blank "␣" encountered before #

Turing Machine

Example

Machine for language $A = \{\, s\#s \mid s \in \{0, 1\}^* \,\}$, input string is 01101#01101 $\in$ A.

- When $\#$ encountered, move right one cell.

$$|X|1|1|0|1|\#|0|1|1|0|1|\sqcup|\sqcup|\ \cdots$$

- If current symbol doesn't match previously recorded symbol, *reject*.
- Overwrite current symbol with X

$$|X|1|1|0|1|\#|X|1|1|0|1|\sqcup|\sqcup|\ \cdots$$

- Scan left, past $\#$ to X
- Move one cell right
- Record symbol and overwrite it with X

$$|X|X|1|0|1|\#|X|1|1|0|1|\sqcup|\sqcup|\ \cdots$$

- Scan right past $\#$ to (last) X and move one cell to right ...

Turing Machine

Example

Machine for language $A = \{\, s\#s \mid s \in \{0, 1\}^* \,\}$, input string is 01101#01101 $\in$ A.

- After several more iterations of zigzagging, we have

$$|X|X|X|X|X|\#|X|X|X|X|X|\sqcup|\sqcup|\ \cdots$$

- After all symbols left of $\#$ have been matched to symbols right of $\#$, check for any remaining symbols to the right of $\#$.
  - If blank $\sqcup$ encountered, *accept*.
  - If 0 or 1 encountered, *reject*.

$$|X|X|X|X|X|\#|X|X|X|X|X|\sqcup|\sqcup|\ \cdots$$

- The string that is accepted or not by our machine is the original input string 01101#01101.

Turing Machine

Definition

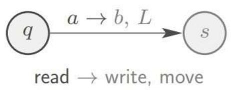
A Turing machine (TM) is a 7-tuple $M = (\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject})$, where

- Σ is a finite set, called the input alphabet; the blank symbol ␣ is not contained in Σ,
- Γ is a finite set, called the tape alphabet; this alphabet contains the blank symbol ␣, and $\Sigma \subseteq \Gamma$,
- Q is a finite set, whose elements are called states,
- q is an element of Q; it is called the start state,
- q_{accept} is an element of Q; it is called the accept state,
- q_{reject} is an element of Q; it is called the reject state, $q_{reject} \neq q_{accept}$
- δ is called the transition function, which is a function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$.
  L: move to left, R: move to right, N: no move.

Turing Machine

Transition function

$\delta(q, a) = (s, b, L)$



If TM

- in state $q \in Q$,
- tape head reads tape symbol $a \in \Gamma$

Then TM

- moves to state $s \in Q$
- overwrites a with $b \in \Gamma$
- moves head left (i.e., $L \in \{L, R, N\}$)

Before $|a|b|a|a|\sqcup|\sqcup|$

After $|a|b|b|a|\sqcup|\sqcup|$

Is no move necessary ?

Turing Machine

Computation steps

- Before the computation step, the Turing machine is in a state $r \in Q$, and the tape head is on a certain cell.
- TM M proceeds according to transition function:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$$

- Depending on r and k symbols read from tape:
  (a) switches to a state $r' \in Q$;
  (b) tape head writes a symbol of Γ in the cell it is currently scanning;
  (c) tape head moves one cell to the left or right or stay at the current cell.
- Computation continues until q_{reject} or q_{accept} is entered. (stopped once entered)
- Otherwise, M will run forever (input string is neither accepted nor rejected)

Turing Machine

Example

TM M for language

$$A = \{0^{2^n} \mid n \geq 0\},$$

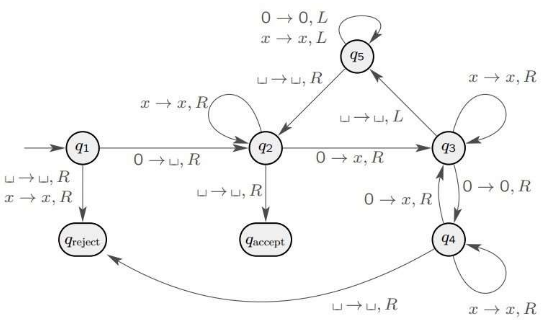
which consists of strings of 0s whose length is a power of 2.

On input string w:

- Sweep left to right across the tape, crossing off every other 0.
- If in stage 1 the tape contained a single 0, accept.
- If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, reject.
- Return the head to the left end of the tape.
- Go to stage 1.

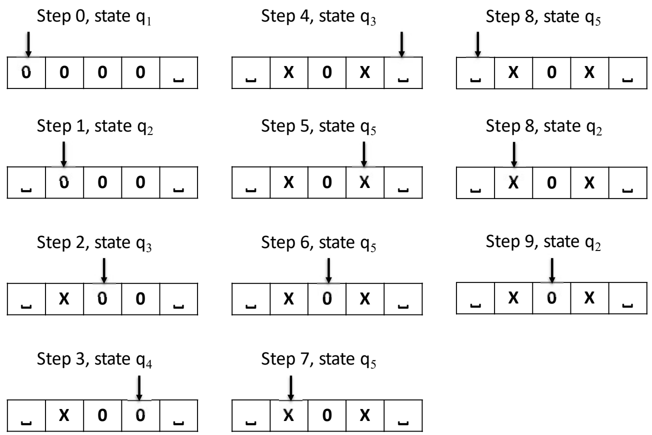
Turing Machine

Example



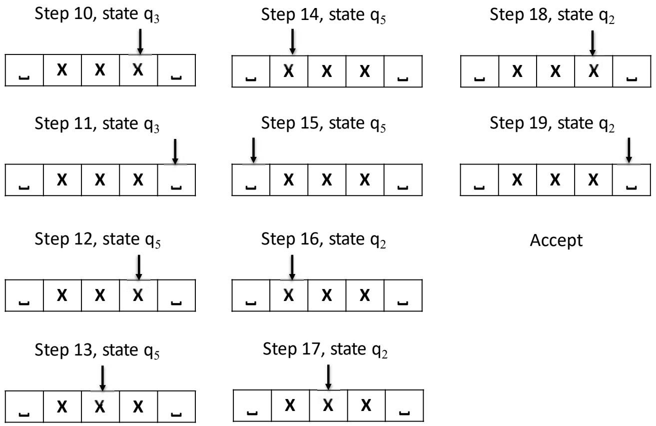
Example

Run M when input w = 0000



Example

Run M when input $w = 0000$



Turing Machine

- **Start configuration**. The input is a string over the input alphabet Σ. Initially, this input string is stored on the first tape, and the head of this tape is on the leftmost symbol of the input string.
- **Computation and termination**. Starting in the start configuration, the Turing machine performs a sequence of computation steps. The computation terminates at the moment when the Turing machine enters the accept state q_{accept} or the reject state q_{reject}. (If the machine never enters q_{accept} and q_{reject} the computation does not terminate.)
- **Acceptance**. The Turing machine M accepts the input string $w \in \Sigma^*$, if the computation on this input terminates in the state q_{accept}.

TM Configuration

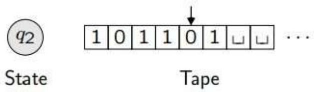
Provides a "snapshot" of TM at any point during computation:

- state
- tape contents
- head location

Example

Configuration 1011q01:

- current state is q
- LHS of tape is 1011
- RHS of tape is 01
- head is on RHS 0



TM Configuration

Definition

Configuration of a TM $M = (Q, \Sigma, \Gamma, \delta, q, q_{accept}, q_{reject})$ is a string uqv with $u, v \in \Gamma^*$ and $q \in Q$, and specifies that currently

- M is in state q
- tape contains uv
- tape head is pointing to the cell containing the first symbol in v.

TM Transitions

Definition

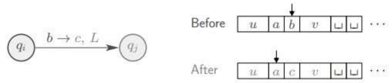
Configuration C1 yields configuration C2 if the Turing machine can legally go from C1 to C2 in a single step. For TM $M = (Q, \Sigma, \Gamma, \delta, q, q_{accept}, q_{reject})$, suppose

- $u, v \in \Gamma^*$
- $a, b, c \in \Gamma$
- $q_{ii}, q_{jj} \in Q$
- transition function $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$.

Example

configuration $uaqiibv$ yields configuration $uq_{jj}acv$

if $\delta(q_i, b) = (q_j, c, L)$.



TM Computation

Definition

Given a TM $M = (Q, \Sigma, \Gamma, \delta, q, q_{accept}, q_{reject})$ and input string $w \in \Sigma^*$. M accepts input w if there is a finite sequence of configurations $C_1, C_2, \ldots, C_k$ for some $k \geq 1$ with

- C_1 is the starting configuration q0w
- C_i yields C_{i+1} for all $i = 1, \ldots, k-1$ (sequence of configurations obeys transition function δ)
- C_k is an accepting configuration $uq_{accept}v$ for some $u, v \in \Gamma^*$.

Language accepted by TM

Definition

The language L (M) accepted by the Turing machine M is the set of all strings in Σ^* that are accepted by M.

Language A is **Turing-recognizable** if there is a TM M such that A = L(M)

- On an input $w \notin L(M)$, the machine M can either halt in a rejecting state, or it can loop indefinitely.
- Turing-recognizable not practical because never know if TM will halt.

Decider

Definition

A **decider** is TM that halts on all inputs, i.e., never loops.

Language A = L(M) is decided by TM M if on each possible input $w \in \Sigma^*$, the TM finishes in a halting configuration, i.e.,

- M ends in q_{accept} for each $w \in A$
- M ends in q_{reject} for each $w \notin A$.

Is decidable language nicer?

A is **Turing-decidable** if ∃ TM M that decides A

- Differences to Turing-recognizable language:

  (a) Turing-decidable language has TM that halts on every string $w \in \Sigma^*$

  (b) TM for Turing-recognizable language may loop on strings $w \notin$ this language

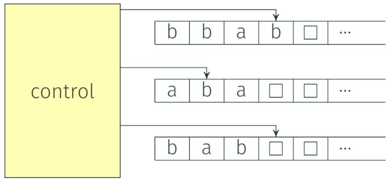
Multi-tape TM

Multi-tape TM has multiple tapes

- Each tape has its own head
- Transition determined by

  (1) state

  (2) the content read by all heads
- Reading and writing of each head are independent of others



Multi-tape TM

Definition

A k-tape Turing machine (TM) is a 7-tuple $M = (\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject})$ has k different tapes and k different read/write heads, where

- Σ is a finite set, called the input alphabet; the blank symbol ␣ is not contained in Σ,
- Γ is a finite set, called the tape alphabet; this alphabet contains the blank symbol ␣, and $\Sigma \subseteq \Gamma$,
- Q is a finite set, whose elements are called states,
- q is an element of Q; it is called the start state,
- q_{accept} is an element of Q; it is called the accept state,
- q_{reject} is an element of Q; it is called the reject state
- δ is called the transition function, which is a function $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, N\}^k$.
  $\Gamma^k = \Gamma \times \Gamma \times \ldots \times \Gamma$

Multi-tape TM

Transition

Transition function:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, N\}^k$$

Given $\delta(q_i, a_1, a_2, \ldots, a_k) = (q_j, b_1, b_2, \ldots, b_k, L, R, \ldots, L)$

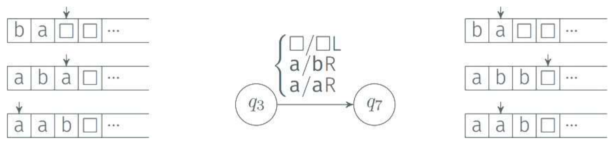
- TM is in state q_i
- heads 1-k read $a_1, a_2, \ldots, a_k$

Then

- TM moves to q_j
- heads 1-k write $b_1, b_2, \ldots, b_k$
- Heads move (left or right) or don't move as specified (L, R, N).

Multi-tape TM

Example



- Multiple tapes are convenient
- Some tapes can serve as temporary storage

Multi-tape TM equivalent to 1-tape TM

Let $k \geq 1$ be an integer. Any k-tape Turing machine can be converted to an equivalent 1-tape Turing machine.

For every multi-tape TM M, there is a single-tape TM M' such that $L(M) = L(M')$.

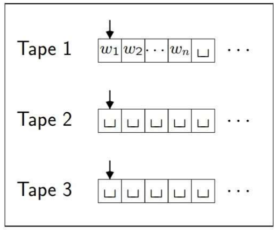
Proof

Basic idea: simulate k-tape TM using 1-tape TM.

Multi-tape TM equivalent to 1-tape TM

Proof

Let TM $M = (\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject})$ be a k-tape TM.



M has:

- input $w = w_1, w_2, \ldots, w_k$ on tape 1
- other tapes contain only blanks ␣
- each head points to first cell.

Construct 1-tape TM M' by extending tape alphabet

$$\Gamma' = \Gamma \cup \dot{\Gamma} \cup \{\#\}$$

$$\# \; \dot{w}_1 \; w_2 \cdots w_n \; \# \; \dot{␣} \; \# \; \dot{␣} \; \# \; ␣ \; ␣ \; \cdots$$

Note: head positions of different tapes are marked by dotted symbol

Multi-tape TM equivalent to 1-tape TM

Proof

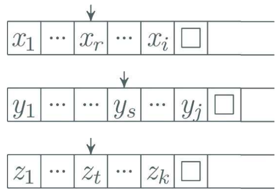
For each step of k-tape TM M, 1-tape M' operates its tape as:

- At the start of the simulation, the tape head of M' is on the leftmost $\#$
- Scans the tape from first $\#$ to $(k+1)$st $\#$ to read symbols under heads.
- Rescans to write new symbols and move heads.

Multi-tape TM equivalent to 1-tape TM

Example

Simulate a 3-tape TM

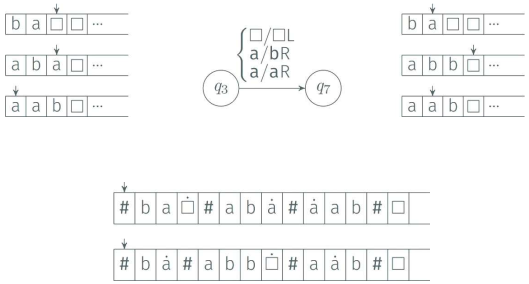


$$\# x_1 x_2 \ldots \dot{x}_r \ldots x_i \# y_1 y_2 \ldots \dot{y}_s \ldots y_j \# z_1 z_2 \ldots \dot{z}_t \ldots z_k \#$$

Multi-tape TM equivalent to 1-tape TM

Example

Suppose the given TM M moves like this:



Multi-tape TM equivalent to 1-tape TM

Key points of simulation

To simulate a model M by another model N:

- Say how the state and storage of N is used to represent the state and storage of M
- Say what should be initially done to convert the input of N
- Say how each transition of M can be implemented by a sequence of transitions of N

Turing-recognizable ⟷ Multiple-tape Turing-recognizable

Language L is TM-recognizable if and only if some multi-tape TM recognizes L.

Nondeterministic TM

A **nondeterministic Turing machine** (NTM) M can have several options at every step. It is defined by the 7-tuple $M = (\Sigma, \Gamma, Q, \delta, q, q_{accept}, q_{reject})$, where

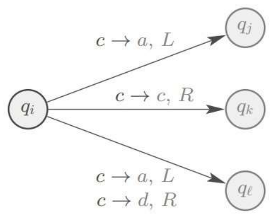
- Σ is input alphabet (without blank ␣)
- Γ is tape alphabet with $\{ ␣ \} \cup \Sigma \subseteq \Gamma$
- Q is a finite set, whose elements are called states δ is transition function $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$
- q is start state $\in Q$
- q_{accept} is accept state $\in Q$
- q_{reject} is reject state $\in Q$

Nondeterministic TM

Transition

Transition function

$$\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$$

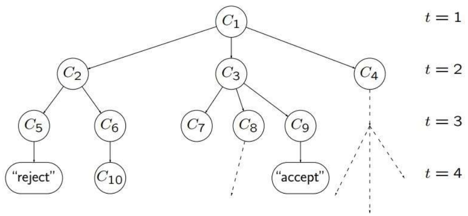


$$\delta(q_i, c) = \{ (q_j, a, L), (q_k, c, R), (q_l, a, L), (q_l, d, R) \}$$

Nondeterministic TM (NTM)

Computation

With any input w, computation of NTM is represented by a configuration tree.



If ∃ (at least) one accepting leaf, then NTM accepts.

NTM equivalent to TM

Every nondeterministic TM has an equivalent deterministic TM.

Proof

- Build TM D to simulate NTM N on each input w. D tries all possible branches of N's tree of configurations.
- If D finds any accepting configuration, then it accepts input w.
- If all branches reject, then D rejects input w.
- If no branch accepts and at least one loops, then D loops on w.

Address

- Every node in the tree **has at most b children**.
- b is size of largest set of possible choices for N's transition function.
- Every node in tree has an address that is a string over the alphabet $\Gamma_b = \{1, 2, \ldots, b\}$

  To get to node with address 231:

  (1) start at root

  (2) take second branch

  (3) then take third branch

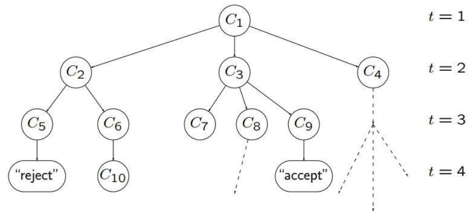
  (4) then take first branch
- Ignore meaningless addresses.
- Visit nodes in breadth-first search order by listing addresses in Γ_b^* in string order:

$$\varepsilon, 1, 2, \ldots, b, 11, 12, \ldots, 1b, 21, 22, \ldots$$

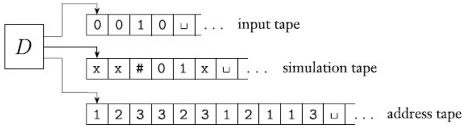
Address

Example

- "accept" configuration has address 231
- Configuration C_6 has address 12.
- Configuration C_1 has address ε.
- Address 132 is meaningless.



Simulating NTM by DTM



1.Initially, input tape contains input string w. Simulation and address tapes are initially empty.
2. Copy input tape to simulation tape.
3.Use simulation tape to simulate NTM N on input w on path in tree from root to the address on address tape.
- At each node, consult next symbol on address tape to determine which branch to take.
- Accept if accepting configuration reached.
- Skip to next step if
  a. symbols on address tape exhausted
  b. nondeterministic choice invalid
  c. rejecting configuration reached

4. Replace string on address tape with next string in Γ_b^* in string order, and go to Stage 2

Encoding

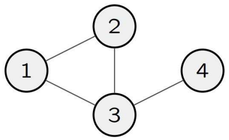
Input to a Turing machine is a string of symbols over an alphabet.

When we want TMs to work on different objects such as:
- Polynomials
- Graphs
- Grammars
- etc

We need to encode this object as a string of symbols over an alphabet.

Encoding of Graph

Given an undirected graph G



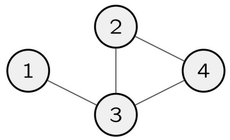
One possible encoding

$$\langle G\rangle = (1, 2, 3, 4)\ (\ (1, 2), (1, 3), (2, 3), (3, 4)\ )$$

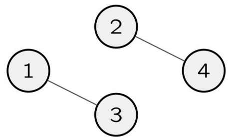
$\langle G\rangle$ of graph G is string of symbols over some alphabet Σ, where the string starts with list of nodes and followed by list of edges.

Encoding of Graph

An undirected graph is **connected** if every node can be reached from any other node by travelling along edge



Connected graph G_1



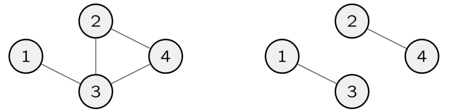
Unconnected graph G_2

Let A be the language consisting of strings representing connected undirected graph

$$A = \{\ \langle G\rangle \mid G \text{ is a connected undirected graph}\ \}$$

So $\langle G_1\rangle \in$ A and $\langle G_2\rangle \notin$ A.

TM to decide the connectedness of a Graph



Connected graph G_1 Unconnected graph G_2

A = { ⟨G⟩ | G is a connected undirected graph }

On input ⟨G⟩ ∈ Ω, where G is an undirected graph:

1. Check if G is a valid graph encoding. If not, reject.
2. Select first node of G and mark it.
3. Repeat until no new nodes marked.
4. For each node in G, mark it if it's attached by an edge to a node already marked
5. Scan all nodes of G to see whether they all are marked. If they are, accept; otherwise, reject."

Ω denotes the **universe** of a decision problem, comprising all instances.

TM to decide the connectedness of a Graph

For TM M that decides A = { ⟨G⟩ | G is a connected undirected graph }

Ω = {⟨G⟩ | G is an undirected graph }

Step 1 checks that input ⟨G⟩ ∈ Ω is valid encoding:

- Two list

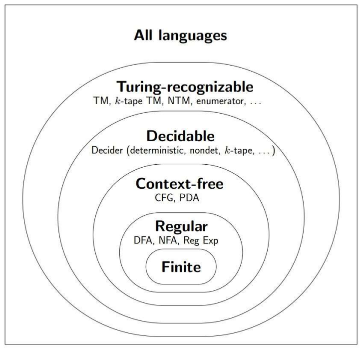
  (a) first is a list of numbers

  (b) second is a list of pairs of numbers
- First list contains no duplicate

•Every node in second list appears in first list

Step 2-5 check if G is connected.

Language Hierarchy



Quick review

- Turing machine is DFA with a Tape
- A is Turing-recognizable if A=L(M) for some TM M
- A is Turing-decidable if A=L(M) for some TM M (decider)that halts on all inputs
- Multi-tape TM and Nondeterministic TM are equivalent to TM


THANK YOU
Xi'an Jiaotong-Liverpool University
西交利物浦大学
XJTLU
SCHOOL OF FILM AND TV ARTS