Turing Machines
Recursive/Recursively Enumerable Languages

Schematic of Turing Machines

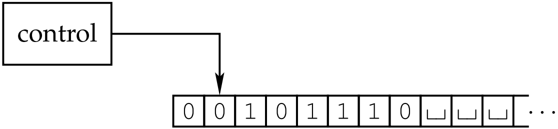


Figure: Schematic of Turing Machines

- A Turing machine has a finite set of control states.
- A Turing machine reads and writes symbols on an infinite tape.
- A Turing machine starts with an input on the left end of the tape.
- A Turing machine moves its read-write head in both directions.
- A Turing machine outputs accept or reject by entering its accepting or rejecting states respectively.
    - A Turing machine need not read all input symbols.
    - A Turing machine may not accept nor reject an input.

Turing Machines

- Consider $B = \{w\#w : w \in \{\texttt{0}, \texttt{1}\}^*\}$.
- $M_1 =$ "On input string w:
  1. Record the first uncrossed symbol from the left and cross it. If the first uncrossed symbol is $\#$, go to step 6.
  2. Move the read-write head to the symbol $\#$. If there is no such symbol, reject.
  3. Move to the first uncrossed symbol to the right.
  4. Compare with the symbol recorded at step 1. If they are not equal, reject.
  5. Cross the current symbol and go to step 1.
  6. Check if all symbols to the right of $\#$ are crossed. If so, accept; otherwise, reject."

Turing Machines – Formal Definition

Definition 1

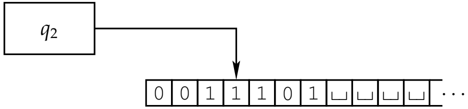
A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where

- Q is the finite set of states;
- Σ is the finite input alphabet not containing the blank symbol $\sqcup$;
- Γ is the finite tape alphabet with $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function;
- $q_0 \in Q$ is the start state;
- $q_{\text{accept}} \in Q$ is the accept state; and
- $q_{\text{reject}} \in Q$ is the reject state with $q_{\text{reject}} \neq q_{\text{accept}}$.

- We only consider deterministic Turing machines.
- Initially, a Turing machine receives its input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ on the leftmost n cells of the tape.
- Other cells on the tape contain the blank symbol $\sqcup$.

Computation of Turing Machines

- A configuration of a Turing machine contains its current states, current tape contents, and current head location.
- Let $q \in Q$ and $u, v \in \Gamma$. We write uqv to denote the configuration where the current state is q, the current tape contents is uv, and the current head location is the first symbol of v.
  - When we say "the current tape contents is uv," we mean an infinite tape contains $uv\sqcup\sqcup\cdots\sqcup\cdots$.
- Consider the configuration $\texttt{001}q_2\texttt{1101}$. The Turing machine
  - is at the state q_2;
  - has the tape contents `0011101`; and
  - has its head location at the second `1` from the left.



Computation of Turing Machines

- Let C_1 and C_2 be configurations. We say C_1 yields C_2 if the Turing machine can go from C_1 to C_2 in one step.
- Formally, let $a, b, c \in \Gamma$, $u, v \in \Gamma^*$, and $q_i, q_j \in Q$.

$$\begin{array}{rl} uaq_ibv \text{ \underline{yields} } uq_jacv & \text{if } \gamma(q_i, b) = (q_j, c, L) \\ q_ibv \text{ \underline{yields} } q_jcv & \text{if } \gamma(q_i, b) = (q_j, c, L) \\ uaq_ibv \text{ \underline{yields} } uacq_jv & \text{if } \gamma(q_i, b) = (q_j, c, R) \end{array}$$

- Note the special case when the current head location is the leftmost cell of the tape.
  - A Turing machine updates the leftmost cell without moving its head.
- Recall that uaq_i is in fact $uaq_i\sqcup$.

Accept, Reject, and Halting

- The start configuration of M on input w is $q_0 w$.
- An accepting configuration of M is a configuration whose state is q_{accept}.
- A rejecting configuration of M is a configuration whose state is q_{reject}.
- Accepting and rejecting configurations are halting configurations and do not yield further configurations.
  - That is, a Turing machine accepts or rejects as soon as it reaches an accepting or rejecting configuration.

Recognizable Languages

- A Turing machine M accepts an input w if there is a sequence of configurations $C_1, C_2, \ldots, C_k$ such that
  - C_1 is the start configuration of M on input w;
  - each C_i yields C_{i+1}; and
  - C_k is an accepting configuration.
- The language of M or the language recognized by M (written $L(M)$) is thus

$$L(M) = \{w : M \text{ accepts } w\}.$$

Definition 2

A language is Turing-recognizable or recursively enumerable if some Turing machine recognizes it.

Decidable Languages

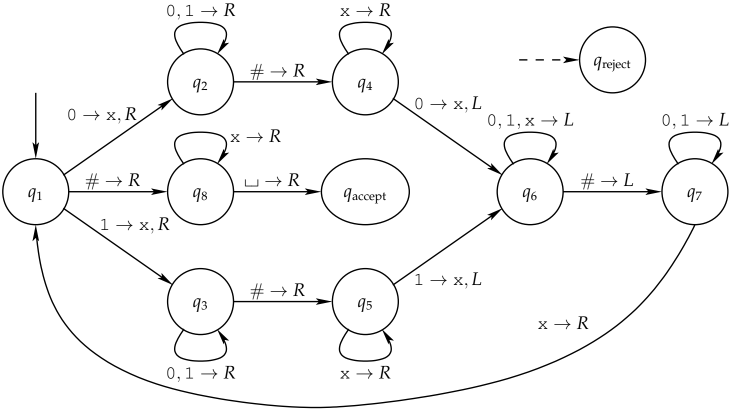
- When a Turing machine is processing an input, there are three outcomes: accept, reject, or loop.
  - "Loop" means it never enters a halting configuration.
- A deterministic finite automaton or deterministic pushdown automaton have only two outcomes: accept or reject.
- For a nondeterministic finite automaton or nondeterminsitic pushdown automaton, it can also loop.
  - "Loop" means it does not finish reading the input (ϵ-transitions).
- A Turing machine that halts on all inputs is called a decider.
- When a decider recognizes a language, we say it decides the language.

Definition 3

A language is Turing-decidable (decidable, or recursive) if some Turing machine decides it.

Turing Machines – Example

- We now formally define $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{accept}}, q_{\text{reject}})$ which decides $B = \{w\#w : w \in \{0,1\}^*\}$.
- $Q = \{q_1, \ldots, q_{14}, q_{\text{accept}}, q_{\text{reject}}\}$;
- $\Sigma = \{0, 1, \#\}$ and $\Gamma = \{0, 1, \#, \text{x}, \sqcup\}$.



Turing Machines whose Heads can Stay

- Recall that the transition function of a Turing machine indicate whether its read-write head moves left or right.
- Consider a new Turing machine whose head can stay.
- Hence we have $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$.
- Is the new Turing machine more powerful?
- Of course not, we can always simulate S by an R and then an L.

Multitape Turing Machines

- A multitape Turing machine has several tapes.
- Initially, the input appears on the tape 1.
- If a multitape Turing machine has k tapes, its transition function now becomes
$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$$
- $\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, d_1, \ldots, d_k)$ means that if the machine is in state q_i and reads a_i from tape i for $1 \leq i \leq k$, it goes to state q_j, writes b_i to tape i for $1 \leq i \leq k$, and moves the tape head i towards the direction d_i for $1 \leq i \leq k$.
- Are multitape Turing machines more powerful than signel-tape Turing machines?

Multitape Turing Machines

Theorem 4

Every multitape Turing machine has an equivalent single-tape Turing machine.

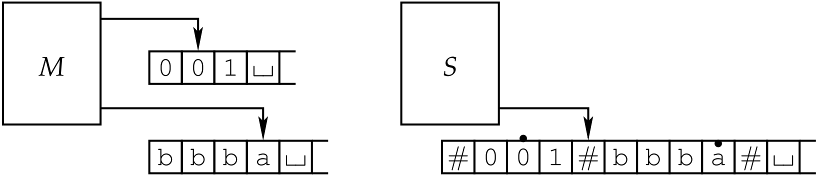
Proof.

We use a special new symbol # to separate contents of k tapes. Moreover, k marks are used to record locations of the k virtual heads.

$S =$ "On input $w = w_1 w_2 \cdots w_n$:

1. Write w in the correct format: $\#\dot{w}_1 w_2 \cdots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \cdots \#$.
2. Scan the tape and record all symbols under virtual heads. Then update the symbols and virtual heads by the transition function of the k-tape Turing machine.
3. If S moves a virtual head to the right onto a #, S writes a blank symbol and shifts the tape contents from this cell to the rightmost # one cell to the right. Then S resumes simulation." □

Multitape Turing Machines



- A "mark" is in fact a different tape symbol.
  - Say the tape alphabet of the multitape TM M is $\{0, 1, a, b, \sqcup\}$.
  - Then S has the tape alphabet $\{\#, 0, 1, a, b, \sqcup, \dot{0}, \dot{1}, \dot{a}, \dot{b}, \dot{\sqcup}\}$.

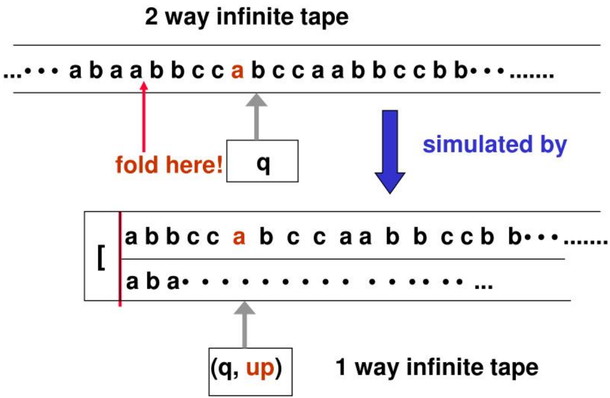
Corollary 5

A language is Turing-Recognizable if and only if some multitape Turing machine recognizes it.

Turing Machines with 2-way Infinite Tape

Theorem 6

A TM with a 2-way infinite tape can be simulated by one with a 1-way infinite tape.



Nondeterministic Turing Machines

- A nondeterministic Turing machine has its transition function of type $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$.
- Is nondeterministic Turing machines more powerful than deterministic Turing machines?
  - Recall that nondeterminism does not increase the expressive power in finite automata.
  - Yet nondeterminism does increase the expressive power in pushdown automata.

Nondeterministic Turing Machines

Theorem 7

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof.

Nondeterministic computation can be seen as a tree. The root is the start configuration. The children of a tree node are all possible configurations yielded by the node. By ordering children of a node, we associate an address with each node. For instance, ϵ is the root; $\texttt{1}$ is the first child of the root; $\texttt{21}$ is the first child of the second child of the root.
We simulate an NTM N with a 3-tape DTM D. Tape 1 contains the input; tape 2 is the working space; and tape 3 records the address of the current configuration.
Let b be the maximal number of choices allowed in N. Define $\Sigma_b = \{\texttt{1}, \texttt{2}, \ldots, \texttt{b}\}$. We now describe the Turing machine D.

Nondeterministic Turing Machines

Proof.

1. Initially, tape 1 contains the input w; tape 2 and 3 are empty.
2. Copy tape 1 to tape 2.
3. Simulate N from the start state on tape 2 according to the address on tape 3.
   - When compute the next configuration, choose the transition by the next symbol on tape 3.
   - If no more symbol is on tape 3, the choice is invalid, or a rejecting configuration is yielded, go to step 4.
   - If an accepting configuration is yielded, accept the input.
4. Replace the string on tape 3 with the next string lexicographically and go to step 2. □

- Observe the D simulates N by breadth.
  - Can we simulate by depth?

Nondeterministic Turing Machines

Corollary 8

A language is Turing-recognizable if and only if some nondeterministic Turing machine recognizes it.

- A nondeterministic Turing machine is a decider if all branches halt on all inputs.
- If the NTM N is a decider, a slight modification of the proof makes D always halt. (How?)

Corollary 9

A language is decidable if and only if some nondeterministic Turing machine decides it.

Schematic of Enumerators

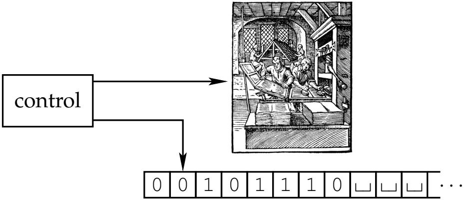


Figure: Schematic of Enumerators

- An enumerator is a Turing machine with a printer.
- An enumerator starts with a blank input tape.
- An enumerator outputs a string by sending it to the printer.
- The language enumerated by an enumerator is the set of strings printed by the enumerator.
  - Since an enumerator may not halt, it may output an infinite number of strings.
  - An enumerator may output the same string several times.

Enumerators

Theorem 10

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

Let E be an enumerator. Consider the following TM M:

$M =$ "On input w :

1. Run E and compare any output string with w.
2. Accept if E ever outputs w."

Conversely, let M be a TM recognizing A. Consider

$E =$ "Ignore the input.

1. Repeat for $i = 1, 2, \ldots$
   1. Let $s_1, s_2, \ldots, s_i$ be the first i strings in Σ^* (say, lexicographically).
   2. Run M for i steps on each of $s_1, s_2, \ldots, s_i$.
   3. If M accepts s_j for $1 \leq j \leq i$, output s_j. □

Enumerators

Theorem 11

A language is Turing-decidable if and only if some enumerator enumerates it in lexicographical order.

Proof.

Let E be an enumerator. Consider the following TM M:
$M =$ "On input w :

1. Run E and compare each generated output string with w.
2. Accept if E ever outputs w; reject if E outputs a w' with $w < w'$"

Conversely, let M be a TM deciding A, and assume that $\Sigma = \{0, 1\}$.
$E =$ "Ignore the input.

1. Repeat for $w = \epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots$
   1. Run M on w;
   2. If M accepts w, output s_j;
   3. If M rejects w, exit

□

Algorithms

- Let us suppose we lived before the invention of computers.
  - say, circa 300 BC, around the time of Euclid.
- Consider the following problem:
  Given two positive integers a and b, find the largest integer r such that r divides a and r divides b.
- How do we "find" such an integer?
- Euclid's method is in fact an algorithm.
- Keep in mind that the concept of algorithms has been in mathematics long before the advent of computer science.

Hilbert's Problems

- Mathematician David Hilbert listed 23 problems in 1900.
  - These problems are challenges for mathematicians in 20th century.
- His 10th problem is to devise "a process according to which it can be determined by a finite number of operations," that tests whether a polynomial has an integral root.
  - In other words, Hilbert wants to find an algorithm to test whether a polynomial has an integral root.
- If such an algorithm exists, we just need to invent it.
- What if there is no such algorithm?
  - How can we argue Hilbert's 10th problem has no solution?
- We need a precise definition of algorithms!

Church-Turing Thesis

- In 1936, two papers came up with definitions of algorithms.
- Alonzo Church used λ-calculus to define algorithms.
  - If you don't know λ-calculus, take Programming Languages.
- Alan Turing used Turing machines to define algorithms.
  - If you don't know TM now, please consider dropping this course.
- It turns out that both definitions are equivalent!
- The connection between the informal concept of algorithms and the formal definitions is called the Church-Turing thesis.

Hilbert's 10th Problem

- In 1970, Yuri Matijasevič showed that Hilbert's 10th problem is not solvable.
  - That is, there is no algorithm for testing whether a polynomial has an integral root.
- Define $D = \{p : p \text{ is a polynomial with an integral root}\}$.
- Consider the following TM:
  $M =$ "The input is a polynomial p over variables $x_1, x_2, \ldots, x_k$
  1. Evaluate p on an enumeration of k-tuple of integers.
  2. If p ever evaluates to 0, accept."
- M recognizes D but does not decide D.

Encodings of Turing Machines

To represent a Turing machine

$$M = (Q, \{0,1\}, \Gamma, \delta, q_1, B, F)$$

as a binary string, we must first assign integers to the states, tape symbols, and directions L and R:

- Assume the states are $q_1, q_2, ..., q_r$ for some r. The start state is q_1, and the only accepting state is q_2.
- Assume the tape symbols are $X_1, X_2, ..., X_s$ for some s. Then: $0 = X_1, 1 = X_2$, and $B = X_3$.
- $L = D_1$ and $R = D_2$.
- Encode the transition rule $\delta(q_i, X_j) = (q_k, X_l, D_m)$ by $0^i 1 0^j 1 0^k 1 0^l 1 0^m$. Note that there are no two consecutive 1s.
- Encode an entire Turing machine by concatenating, in any order, the codes Ci of its transition rules, separated by $11 : C_1 11 C_2 11 \cdots C_{n-1} 11 C_n$.

Example

$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q2\})$ where δ is defined by: $\delta(q_1, 1) = (q_3, 0, R)$, $\delta(q_3, 0) = (q_1, 1, R)$, $\delta(q_3, 1) = (q_2, 0, R)$, and $\delta(q_3, B) = (q_3, 1, L)$.

- Codes for the transition rules:
  0100100010100   0001010100100
  00010010010100   0001000100010010
- Code for M: 01001000101001<u>1</u>000101010010011
  00010010010100110001000100010010

Given a Turing machine M with code w_i , we can now associate an integer to it: M is the ith Turing machine, referred to as M_i. Many integers do no correspond to any Turing machine at all. Examples: 11001 and 001110.

If w_i is not a valid TM code, then we shall take Mi to be the Turing machine (with one state and no transitions) that immediately halts on any input. Hence $L(M_i) = \emptyset$ if w_i is not a valid TM code.

Relationship among Languages

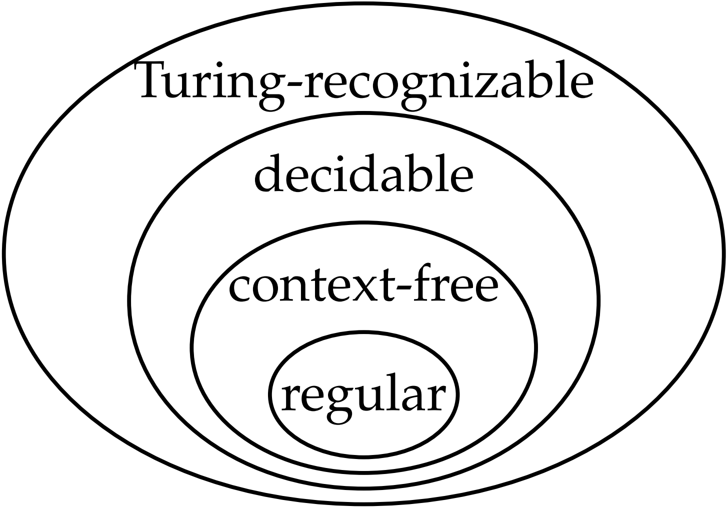


Figure: Relationship among Different Languages

Acceptance Problem for TM's

- Consider

$$A_{\text{TM}} = \{\langle M, w \rangle : M \text{ is a TM and } M \text{ accepts } w\}$$

- Consider the following TM:
  $U =$ "On input $\langle M, w \rangle$ where M is a TM and w is a string:
  1. Simulate M on the input w.
  2. If M enters its accept state, accept; if M enters its reject state, reject."
- Does U decide A_{TM}? Why not?
- The TM U is called the universal Turing machine.

Counting Arguments

- Recall that $|\mathbb{N}| = |\mathbb{Z}| = |\Sigma^*| = \aleph_0$ (Σ is finite).
- Also recall that $|\mathcal{P}(\Sigma^*)| > \aleph_0$.
  - Consult your textbook or my notes on discrete mathematics if you are not sure.

Corollary 12

Some languages are not Turing-recognizable.

Proof.

The set of all Turing machines is countable since each TM M has an encoding $\langle M \rangle$ in Σ^*.
The set of all languages over Σ is $\mathcal{P}(\Sigma^*)$ and hence is uncountable.
Hence some languages are not Turing-recognizable. □

- There are in fact uncountably many languages that cannot be recognized by Turing machines.
- Can we find a concrete example?

Undecidability of the Acceptance Problem for TM's

Theorem 13

$A_{TM} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ accepts } w\}$ *is not a decidable language.*

Proof.

Suppose there is a TM H deciding A_{TM}. That is,

$$H(\langle M, w\rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

Consider the following TM:
$D =$ "On input $\langle M\rangle$ where M is a TM:

1. Run H on the input $\langle M, \langle M\rangle\rangle$.
2. If H accepts, reject. If H rejects, accept."

Consider

$$D(\langle D\rangle) = \begin{cases} \text{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \text{reject} & \text{if } D \text{ accepts } \langle D\rangle \end{cases}$$

A contradiction. □

A Turing-unrecognizable Language

- A language is co-Turing-recognizable if it is the complement of a Turing-recognizable language.

Theorem 14

A language is decidable if and only if it is Turing-recognizable and co-Turing-recognizable.

Proof.

If A is decidable, then A and $\overline{A}$ are both recognizable. Since $\overline{\overline{A}} = A$, A is Turing-recognizable and co-Turing-recognizable.
Now suppose A and $\overline{A}$ are Turing-recognizable by M_1 and M_2 respectively. Consider
$M =$ "On input w:

1. Run both M_1 and M_2 on the input w in parallel.
2. If M_1 accepts, accept; if M_2 accepts; reject."

□

A Turing-unrecognizable Language

Corollary 15

$\overline{A_{\text{TM}}}$ is not Turing-recognizable.

Proof.

A_{TM} is Turing-recognizable. If $\overline{A_{\text{TM}}}$ is Turing-recognizable, A_{TM} is both Turing-recognizable and co-Turing-recognizable. By Theorem 14, A_{TM} is decidable. A contradiction. □