CS372
FORMAL LANGUAGES & THE THEORY OF COMPUTATION

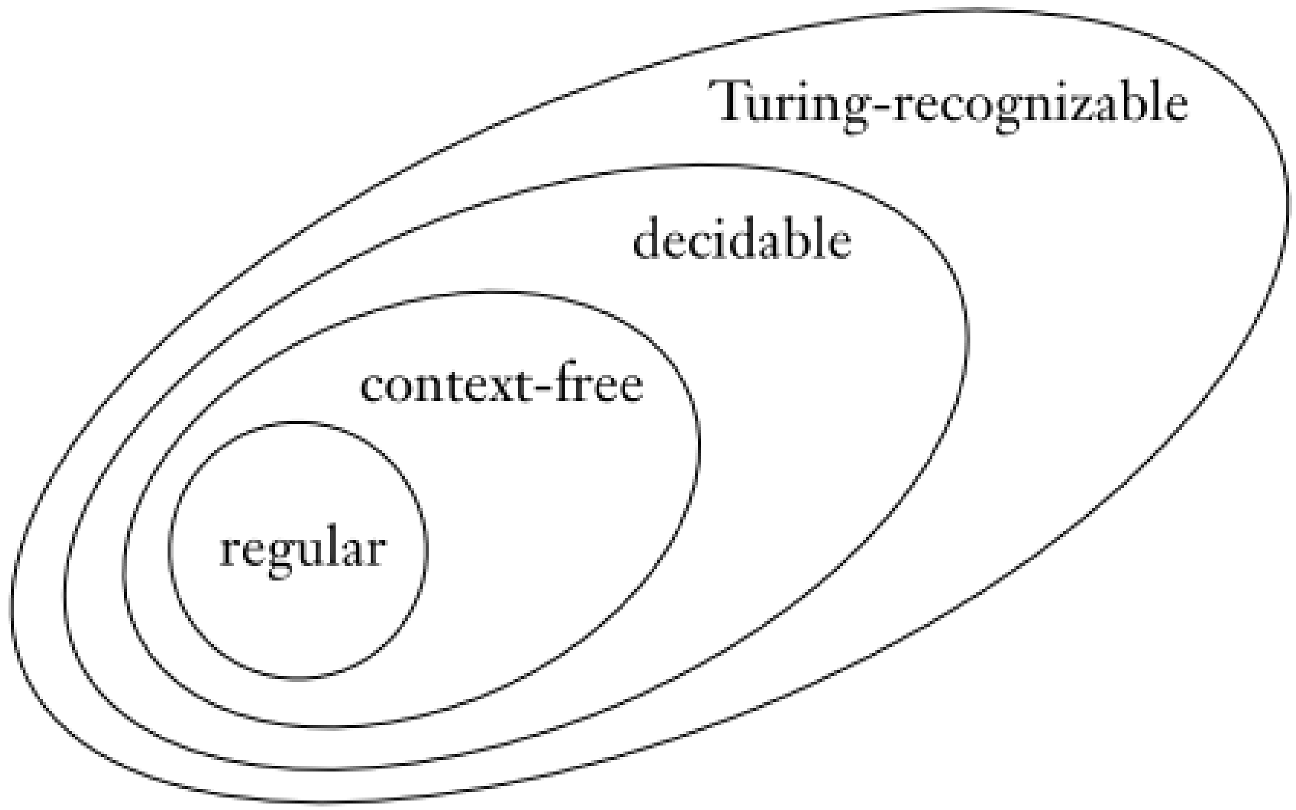
Dr.Nguyen Thi Thu Huong
Phone: +84 24 38696121, Mobi: +84 903253796
Email: huongnt@soict.hust.edu.vn,
huong.nguyenthithu@hust.edu.vn

Unit 9
Undecidability

Contents

- Acceptance Problem for Turing machine
- The diagonalization method
- The halting problem

Relationship among classes of languages



Acceptance Problems for Turing Machines

Theorem 9.1

$A_{TM} = \{(M, w) \mid M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable.

1. Note that A_{TM} is Turing-recognizable. This theorem when proved, shows that recognizers are more powerful than deciders.
2. We can encode TMs with strings just like we did for DFA's

Encoding aTuring Machine

- Encode Q using unary encoding:
  - For $Q = \{q_0, q_1, \dots q_{n-1}\}$, encode q_i using $i + 1$ 0's, i.e., using the string 0^{i+1}. q_{accept} is encoded with 0^{n+1}, q_{reject} is encoded with 0^{n+2},
  - We assume that q_0 is always the start state.
- Encode Γ using unaryencoding:$\Gamma = \{a_1, a_2, a_3\}$, where $a_1 = 0$, $a_2 = 1$, $a_3 = \square$. Encode a_1 using 0, a_2 using 00, a_3 using 000.
- Encode $\Delta = \{\Delta_1, \Delta_2\} = \{L, R\}$ using 0^t (t = 1 or t = 2)
- Encode δ. Each entry of δ, e.g., $\delta(q_i, a_j) = (q_k, a_m, \Delta_t)$ is encoded as

$$0^{i+1} 1\, 0^{j+1} 1\, 0^{k+1} 1\, 0^{m+1} 1 0^t$$

- Separate entries of δ with 11, the code for the whole machine begins with 111 and ends with 111 .

Example for TM Encoding

- TM for language $L = \{0^{2n} \mid n \geq 0\}$ with the following state diagram is

111

0101001010011

01001000010010011

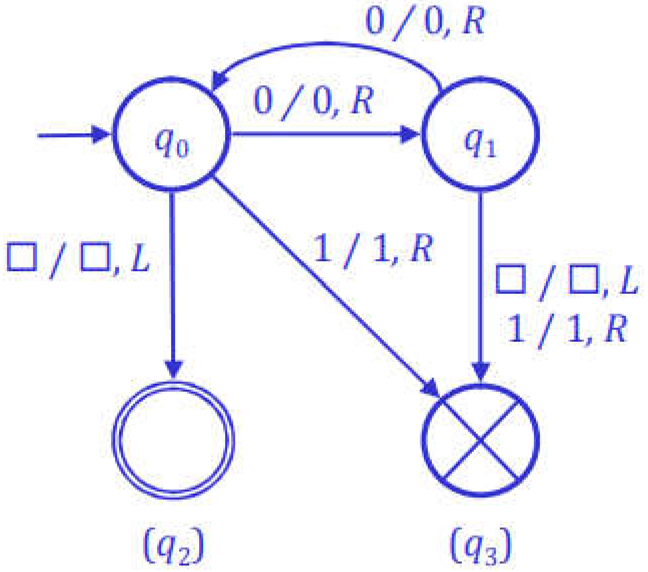
01000100010001011

0010101010011

001001000010010011

0010001000010010

111



Acceptance Problems for Turing Machines

- The TM U recognizes A_{TM}
- $U =$ "On input (M, w) where M is a TM and w is a string:
  1. Simulate M on w
  2. If M ever enters its accepts state, *accept*; if M ever enters its reject state, *reject*.
- Note that if M loops on w, then U loops on (M, w), which is why it is NOT a decider!
- U can not detect that M halts on w .
- In some documents A_{TM} is also known as the Halting Problem
- U is known as the Universal Turing Machine because it can simulate every TM (including itself!)

The diagonalization method

- Let A and B be any two sets (not necessarily finite) and f be a function from A to B.
- f is one-to-one if $f(a) \neq f(b)$ whenever $a \neq b$.
- f is onto if for every $b \in B$ there is an $a \in A$ such that $f(a) = b$.
- We say A and B are the same size if there is a one-to-one and onto function $f : A \longrightarrow B$.
- Such a function is called a correspondence for pairing A and B.
  - Every element of A maps to a unique element of B
  - Each element of B has a unique element of A mapping to it.

The diagonalization method

- Let $\mathbb{N}$ be the set of natural numbers $\{1, 2, \ldots\}$ and let E be the set of even numbers $\{2, 4, \ldots\}$.
- $f(n) = 2n$ is a correspondence between $\mathbb{N}$ and E.
- Hence, $\mathbb{N}$ and E have the same size (though $E \subset \mathbb{N}$).
- A set A is countable if it is either finite or has the same size as $\mathbb{N}$.
- $\mathbb{Q} = \{ \frac{m}{n} \mid m, n \in \mathbb{N} \}$ is countable!
- Z the set of integers is countable:

$$f(n) = \begin{cases} \frac{n}{2} & n \text{ even} \\ -\frac{n+1}{2} & n \text{ odd} \end{cases}$$

The diagonalization method

THEOREM

$\mathbb{R}$ *is uncountable*

PROOF.

Assume f exists and every number in $\mathbb{R}$ is listed.

Assume $x \in \mathbb{R}$ is a real number such that x differs from the j^{th} number in the j^{th} decimal digit.

If x is listed at some position k, then it differs from itself at k^{th} position; otherwise the premise does not hold

f does not exist

n	$f(n)$
1	3.14159. . .
2	55.77777. . .
3	0.12345. . .
4	0.50000. . .
.	.

$x = .4527\ldots$ defined as such, can not be on this list.

Diagonalization over languages

Corollary

Some languages are not Turing-recognizable.

Proof

For any alphabet Σ, Σ^* is countable. Order strings in Σ^* by length and then alphanumerically, so $\Sigma^* = \{s_1, s_2, \ldots, s_i, \ldots\}$

The set of all TMs is a countable language.

Each TM M corresponds to a string $\langle M \rangle$.

Generate a list of strings and remove any strings that do not represent a TM to get a list of TMs.

Diagonalization over languages

Proof (continued)

- The set of infinite binary sequences, $\mathcal{B}$, is uncountable. (Exactly the same proof we gave for uncountability of $\mathcal{R}$)
- Let $\mathcal{L}$ be the set of all languages over Σ.
- For each language $A \in \mathcal{L}$ there is unique infinite binary sequence X_A
    - The i^{th} bit in X_A is 1 if $s_i \in A$, 0 otherwise.

$\Sigma^*=\{$	ε,	0,	1,	00,	01,	10,	11,	000,	001,	$\cdots$	$\}$
$A=\{$		0,		00,	01,			000,	001,	$\cdots$	$\}$
$X_A=\{$	0	1	0	1	1	0	0	1	1	$\cdots$	$\}$

Diagonalization over languages

Proof (continued)

- The function $f : \mathcal{L} \longrightarrow \mathcal{B}$ is a correspondence. Thus $\mathcal{L}$ is uncountable.
- So, there are languages that can not be recognized by some TM. There are not enough TMs to go around.

The Acceptance problem is undecidable

Theorem

- $A_{TM} = \{(M, w) \mid M \text{ is a TM and } M \text{ accepts } w\}$, *is undecidable.*

- We assume A_{TM} is decidable and obtain a contradiction.
  - Suppose H decides A_{TM}

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

The Acceptance problem is undecidable

PROOF (CONTINUED)

- We now construct a new TM D
  $D =$ "On input (M), where M is a TM
  1. Run H on input $(M, (M))$.
  2. If H accepts, D *rejects*, if H rejects, D *accepts*"
- So

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

- When D runs on itself we get

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

- Neither D nor H can exist.

Consider the behaviour of all possible deciders:

$$
\begin{array}{r|cccccccc}
 & & & & & & (D) & \\
 & (M_1) & (M_2) & (M_3) & (M_4) & \cdots & (M_j) & \cdots \\
\hline
M_1 & \underline{\textit{accept}} & \textit{reject} & \textit{accept} & \textit{reject} & \cdots & \textit{accept} & \cdots \\
M_2 & \textit{accept} & \underline{\textit{accept}} & \textit{accept} & \textit{accept} & \cdots & \textit{accept} & \cdots \\
M_3 & \textit{reject} & \textit{reject} & \underline{\textit{reject}} & \textit{reject} & \cdots & \textit{reject} & \cdots \\
M_4 & \textit{accept} & \textit{accept} & \textit{reject} & \underline{\textit{reject}} & \cdots & \textit{accept} & \cdots \\
\vdots & & \vdots & & & \ddots & & \\
D = M_j & \textit{reject} & \textit{reject} & \textit{accept} & \textit{accept} & \cdots & \underline{?} & \cdots \\
\vdots & & \vdots & & & & & \ddots
\end{array}
$$

- D computes the opposite of the diagonal entries!

A Turing unrecognizable language

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

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

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

- We know A_{TM} is Turing-recognizable.
- If A_{TM} were also co- Turing-recognizable, A_{TM} would have to be decidable.
- We know A_{TM} is not decidable.
- A_{TM} must not be co -Turing-recognizable.