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 8
Decision Problems for Automata and Grammars

Contents

- Decidable Languages
- Decidable problems concerning regular languages
- Decidable problems concerning context free languages

Decidability

- Power of algorithms to solve problems.
- Certain problems can be solved algorithmically and others can not be
- Knowing a problem is unsolvable is useful because
  - it must be simplified or altered before finding an algorithmic solution.
  - we gain a better perspective on computation and its limitations.

Decidable (Turing-decidable) languages

- Turing-decidable languages
  - TM halts in an accepting configuration if *w* is in the language.
  - TM halts in a rejecting configuration if *w* is not in the language.

Input of Turing machines

- The inputs to TMs have to be strings.
- Every object O that enters a computation will be represented with a string $\langle O \rangle$, encoding the object.
- For example if G is a 4 node undirected graph with 4 edges
- $\langle G \rangle = \texttt{(1,2,3,4)((1,2),(2,3),(3,1),(1,4))}$
- Then we can define **problems over graphs**,e.g., as:
  - $A = \{\langle G \rangle \mid G \text{ is a connected undirected graph}\}$

Input of Turing machines (cont'd)

- The graph will be represented as a string over {0,1}
  - Node v_k is represented by 0^k
  - An arc (v_i, v_k) is represented by $en(v_i)1en(v_k)$ *where* $en(v_i)$ *and* $en(v_k)$ are the encoding of nodes v_i and v_k
  - The string 11 is used to separate arcs

Encoding finite automata with strings

- $\langle B \rangle$ represents the encoding of the description of an automaton (DFA/NFA).
- We need to encode Q, Σ, δ and F.
- One possible encoding scheme:
- Encode Q using unary encoding:
  - For $Q = \{q_0, q_1, \ldots q_{n-1}\}$, encode q_i using $i + 1$ 0's, i.e., using the string 0^{i+1}.
  - We assume that q_0 is always the start state.
- Encode Σ using unary encoding:
  - For $\Sigma = \{a_1, a_2, \ldots a_m\}$, encode a_i using i 0's, i.e., using the string 0^i.
- With these conventions, all we need to encode is δ and F! Each entry of δ, e.g., $\delta(q_i, a_j) = q_k$ is encoded as

$$\underbrace{0^{i+1}}_{q_i} 1 \underbrace{0^j}_{a_j} 1 \underbrace{0^{k+1}}_{q_k}$$

Encoding finite automata with strings

- Similar to graphs, the whole function δ can be encoded as
  - $0\underline{0100}0\underline{0100}0\ 1\ 0\underline{00001}0\underline{01000000}0\ \cdots 1\ 0\underline{000001}0\underline{000001}0$

    transition 1   transition 2 ...   transition n

- *F* can be encoded just as a list of the encodings of all the final states. For example, if states 2 and 4 are the final states, *F* could be encoded as

$$000\ 10\underline{0}000$$

$$q_2 \qquad q_4$$

The whole DFA would be encoded by

$$11\ 0\underline{0100010000}0\underline{100000}\ \cdots 0\ 11\ 0\underline{0000000}1\underline{0000000}0\ 11$$

encoding of the transitions   encoding of the final states

Encoding finite automata with strings

$\langle B \rangle$ representing the encoding of the description of an automaton (DFA/NFA) would be something like

$$\langle B \rangle = 11\ 0\underbrace{\underline{010001000}\ 0\underline{100000}\cdots}_{\text{encoding of the transitions}}0\ 11\ \underbrace{0\underline{000000}0\ 1\underline{000000}}_{\text{encoding of the final states}}0\ 11$$

In fact, the description of all DFAs could be described by a regular expression like

$$11(0^+10^+10^+1)^*1(0^+1)^+1$$

Similarly strings over Σ can be encoded with (the same convention)

$$\langle w \rangle = \underbrace{0000}_{a_4}1\underbrace{000000}_{a_6}1 \cdots \underbrace{0}_{a_1}$$

Encoding finite automata with strings

(B, w) represents the encoding of a machine followed by an input string, in the manner above (with a suitable separator between (B) and (w).

Now we can describe our problems over languages and automata as problems over strings (representing automata and languages).

Decision problems for regular languages

- The acceptance problem : Does B accept w?
- The emptiness problem: Is $L(B)$ empty?
- The equivalent problem: Is $L(A) = L(B)$?

The acceptance problem for DFA's

THEOREM 8.1

$A_{DFA} = \{(B, w) \mid B$ is a DFA that accepts input string $w\}$ is a decidable language.

PROOF

- Simulate with a two-tape TM.
  - One tape has (B, w)
  - The other tape is a work tape that keeps track of which state of B the simulation is in.
- $M =$ "On input (B, w)

  Simulate B on input w

  If the simulation ends in an accept state of B, *accept*; if it ends in a nonaccepting state, *reject*."

The acceptance problem for NFA's

THEOREM 8.2

$A_{NFA} = \{(B, w) \mid B$ is a NFA that accepts input string $w\}$ is a decidable language.

PROOF

- Convert NFA to DFA and use Theorem 1
- $N =$ "On input (B, w) where B is an NFA

  Convert NFA B to an equivalent DFA C, using the determinization procedure.
  Run TM M in Thm 4.1 on input (C, w)
  If M accepts *accept*; otherwise *reject*."

The generation problem for regular expressions

THEOREM 8.3

$A_{REX} = \{(R, w) \mid R$ is a regular exp. that generates string $w\}$ is a decidable language.

PROOF

- Convert R to an NFA and use Theorem 8.2
- $P =$ "On input (R, w) where R is a regular expression

  1. Convert R to an equivalent NFA A, using the Regular Expression-to-NFA procedure
  2. Run TM N in Thm 4.2 on input (A, w)
  3. If N accepts *accept*; otherwise *reject*."

The emptiness problem for DFAs

THEOREM 8.4

$E_{DFA} = \{\langle A \rangle \mid A$ is a DFA and $L(A) = \Phi\}$ is a decidable language.

PROOF

Use the DFS algorithm to mark the states of DFA

$T =$ "On input $\langle A \rangle$ where A is a DFA.

Mark the start state of A

Repeat until no new states get marked.

Mark any state that has a transition coming into it from any state already marked.

If no final state is marked, *accept*; otherwise *reject*."

The equivalence problem for DFA’s

THEOREM 4.5

$EQ_{DFA} = \{(A,\ B) \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ is a decidable language.

PROOF

- Construct the machine for $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$ and check if $L(C) = \Phi$.
- $T =$ “On input $(A,\ B)$ where A and B are DFAs.
  1. Construct the DFA for $L(C)$ as described above.
  2. Run TM T of Theorem 8.4 on input (C).
  3. If T accepts, *accept*; otherwise *reject*.”

Decision problems for context free languages

- The generation problem : Does grammar G generate w?
- The emptiness problem: Is $L(G)$ empty?

The generation problem for CFL

THEOREM 8.7

$A_{CFG} = \{(G, w) \mid G$ is a CFG that generates string $w\}$ is a decidable language.

PROOF

- Convert G to Chomsky Normal Form and use the CYK algorithm.
- C = "On input (G, w) where G is a CFG

  Convert G to an equivalent grammar in CNF

  Run CYK algorithm on w of length n

  If $S \in V_{i,n}$ *accept*; otherwise *reject*."

The generation problem for CFL

- Convert G to Chomsky Normal Form and check all derivations up to a certain length (Why!)
- $S =$ "On input (G, w) where G is a CFG

  1. Convert G to an equivalent grammar in CNF
  2. List all derivations with $2n - 1$ steps where n is the length of w . If $n = 0$ list all derivations of length 1.
  3. If any of these strings generated is equal to w , *accept*; otherwise *reject*."

This works because every derivation using a CFG in CNF either increase the length of the sentential form by 1 (using a rule like $A \rightarrow BC$ or leaves it the same (using a rule like $A \rightarrow a$)

Obviously this is not very efficient as there may be exponentially many strings of length up to $2n - 1$.

The emptiness problem for CFL

THEOREM 8.8

$E_{CFG} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Phi\}$ is a decidable language.

PROOF

Mark variables of G systematically if they can generate terminal strings, and check if S is unmarked.

R = "On input $\langle G\rangle$ where G is a CFG.

1) Mark all terminal symbols G
2) Repeat until no new variable get marked.

   Mark any variable A such that G has a rule $A \rightarrow U_1U_2 \cdots U_k$ and $U_1, U_2, \ldots U_k$ are already marked.

3) If start symbol is NOT marked, *accept*; otherwise *reject*."

The equivalence problem for CFGs

$$EQ_{CFG} = \{(G,\ H) \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$$

- It turns out that EQ_{DFA} is not a decidable language.
- The construction for DFAs does not work because CFLs are NOT closed under intersection and complementation.

Decidability of CFLs

THEOREM 8.9

Every context free language is decidable.

PROOF

Design a TM M_G that has G built into it and use the result of A_{CFG}.

M_G = "On input w

Run TM S (from Theorem 8.7) on input (G, w)

If S accepts, *accept*, otherwise *reject*.

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