

Lecture 4: Turing Machines for Languages

Lecturer: Renjie Yang

4.1 Turing Machines

Definition 4.1 A Turing machine is a 7-tuple $\{Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}\}$, where Q, Σ, Γ are all finite sets and:

- Q is the set of states,
- Σ is the input alphabet not containing the blank symbol \sqcup ,
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ 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, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Definition 4.2 As a Turing machine computes, changes occur in the current state, the current tape contents, and the current head location. A setting of these three items is called a configuration of the Turing machine. For a state q and two strings u and v over the tape alphabet Γ , we write uqv for 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 . The tape contains only blanks following the last symbol of v .

Say that configuration C_1 yields configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step. Suppose that we have a, b, c in Σ , as well as u and v in Σ^* and states q_i and q_j . In that case, $uaq_i bv$ and $uq_i acv$ are two configurations. Say that $uaq_i bv$ yields $uq_j acv$ if in the transition function $\delta(q_i, b) = (q_j, c, L)$. For a rightward move, say that $uaq_i bv$ yields $uacq_j v$ if in the transition function $\delta(q_i, b) = (q_j, c, R)$. For the left-hand end, the configuration $q_i bv$ yields $q_j cv$ if the transition is left-moving, and it yields $cq_j v$ for the right-moving transition.

The start configuration of M on input w is the configuration $q_0 w$, which indicates that the machine is in the start state q_0 with its head at the leftmost position on the tape. In an accepting configuration, the state of the configuration is q_{accept} . In a rejecting configuration, the state of the configuration is q_{reject} . Accepting and rejecting configurations are halting configurations and do not yield further configurations.

A Turing machine M accepts input w if a sequence of configurations C_1, C_2, \dots, C_k exists, where

1. C_1 is the start configuration of M on input w ,
2. each C_i yields C_{i+1} , and
3. C_k is an accepting configuration.

The collection of strings that M accepts is the language of M , or the language recognized by M , denoted $L(M)$. Given an input w , we say that M halts on w if M either accept or reject w . A machine that halts for all inputs is called a decider. A decider that recognizes some language also is said to decide that language.

Definition 4.3 Call a language Turing-recognizable if some Turing machine recognizes it.

Definition 4.4 Call a language Turing-decidable if some Turing machine decides it.

Example A Turing machine M that decides $A = \{0^{2^n}\}$:

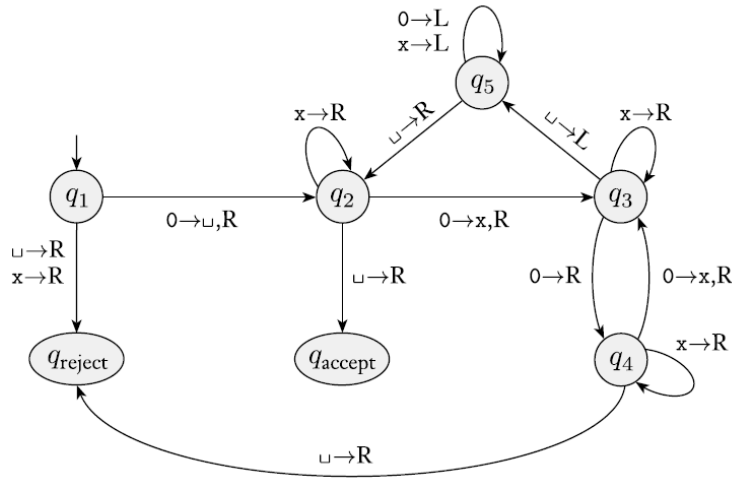
$Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$,

$\Sigma = \{0\}$,

$\Gamma = \{0, x, \sqcup\}$,

The start, accept, and reject states are q_1 , q_{accept} , q_{reject} , respectively.

The transition function δ is described by the following state diagram:



Example A Turing machine M that decides $B = \{w\#w \mid w \in \{0, 1\}^*\}$:

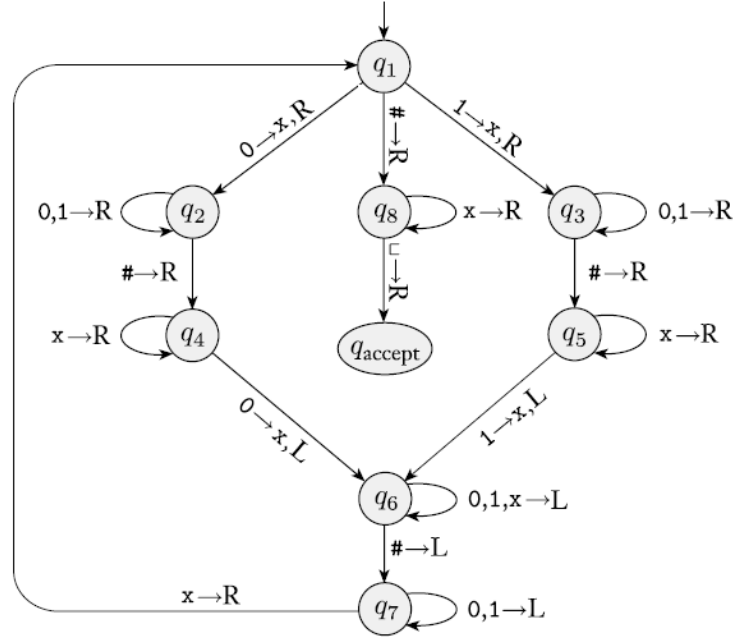
$Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{accept}, q_{reject}\}$,

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

$\Gamma = \{0, 1, \#, x, \sqcup\}$,

The start, accept, and reject states are q_1 , q_{accept} , q_{reject} , respectively.

The transition function δ is described by the following state diagram:



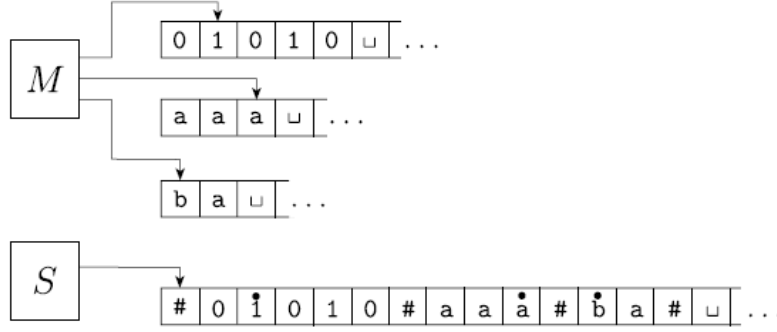
4.2 Turing Machines Variants

Definition 4.5 A multitape Turing machine is a 7-tuple $\{Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}\}$, where Q, Σ, Γ are all finite sets and:

- Q is the set of states,
- Σ is the input alphabet not containing the blank symbol \sqcup ,
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$ 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, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Theorem 4.6 Every multitape Turing machine has an equivalent single-tape Turing machine.

Proof: We can convert a multitape Turing machine M to an equivalent single tape Turing machine S . The key idea is to show how to simulate M with S .



S = “ On input $w = w_1 \cdots w_n$:

1. First S puts its tape into the format that represents all k tapes of M . The formatted tape contains $\# \overset{\bullet}{w_1} \overset{\bullet}{w_2} \cdots \overset{\bullet}{w_n} \# \sqcup \# \sqcup \# \cdots \#$
2. To simulate a single move, S scan its tape from the first $\#$, which marks the left-hand end, to the $(k+1)$ th $\#$, which marks the right-hand end, in order to determine the symbols under the virtual heads. Then S makes a second pass to update the tapes according to the way that M 's transition function dictates.
3. If at any point S moves one of the virtual heads to the right onto a $\#$, this action signifies that M has moved the corresponding head onto the previously unread blank portion of that tape. So S writes a blank symbol on this tape cell and shifts the tape contents, from this cell until the rightmost $\#$, one unit to the right. Then it continues the simulation as before. ”

■

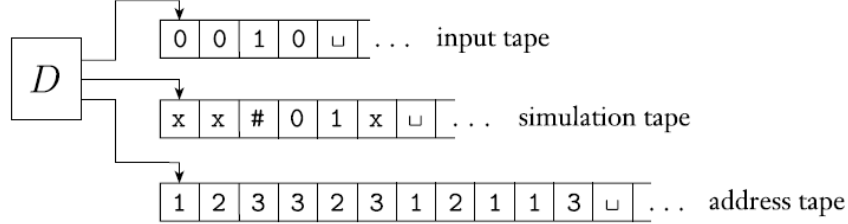
Corollary 4.7 *A language is Turing-recognizable if and only if some multitape Turing machine recognizes it.*

Definition 4.8 *A nondeterministic Turing machine is a 7-tuple $\{Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}\}$, where Q, Σ, Γ are all finite sets and:*

- Q is the set of states,
- Σ is the input alphabet not containing the blank symbol \sqcup ,
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \rightarrow \wp(Q \times \Gamma \times \{L, R\}^k)$ 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, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Theorem 4.9 *Every nondeterministic Turing machine has an equivalent deterministic Turing machine.*

Proof: The simulating deterministic Turing machine D has three tapes. By Theorem 4.6, this arrangement is equivalent to having a single tape. The machine D uses its three tapes in a particular way, as illustrated in the following figure. Tape 1 always contains the input string and is never altered. Tape 2 maintains a copy of N 's tape on some branch of its nondeterministic computation. Tape 3 keeps track of D 's location in N 's nondeterministic computation tree.



Describe D as the following:

1. Initially, tape 1 contains the input w , and tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2 and initialize the string on tape 3 to be ϵ
3. Use tape 2 to simulate N with input w on one branch of its nondeterministic computation. Before each step of N , consult the next symbol on tape 3 to determine which choice to make among those allowed by N 's transition function. If no more symbols remain on tape 3 or if this nondeterministic choice is invalid, abort this branch by going to stage 4. Also go to stage 4 if a rejecting configuration is encountered. If an accepting configuration is encountered, accept the input.
4. Replace the string on tape 3 with the next string in the string ordering. Simulate the next branch of N 's computation by going to stage 2.

■

Corollary 4.10 *A language is Turing-recognizable if and only if some nondeterministic Turing machine recognizes it.*

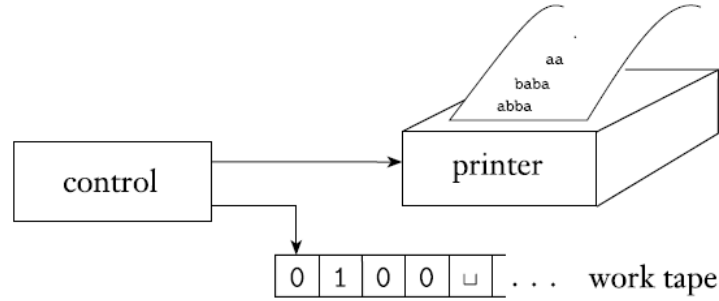
Corollary 4.11 *A language is decidable if and only if some nondeterministic Turing machine decides it.*

Definition 4.12 *An enumerator is a 7-tuple $\{Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}\}$, where Q, Σ, Γ are all finite sets and:*

- Q is the set of states,
- Σ is the input alphabet not containing the blank symbol \sqcup ,
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \times \Sigma_\epsilon$ 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, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Definition 4.13 The computation of an enumerator E is defined as in an ordinary Turing machine, except that it has two tapes, a working tape and a printing tape, both initially blank. At each step, the machine may write a symbol from Σ on the output tape, or nothing, as determined by δ . If $\delta(q, a) = (r, b, L, c)$, it means that in state q , reading a , enumerator e enters state r , write b on the work tape, moves the work tape head left, writes c on the output tape, and moves the output tape head to the right if $c \neq \epsilon$. Whenever state q_{print} is entered, the output tape is reset to blank and the head returns to the left-hand end. The machine halts when q_{accept} is entered. $L(E) = \{w \in \Sigma^* \mid w \text{ appears on the work tape if } q_{\text{print}} \text{ is entered.}\}$



Theorem 4.14 A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof: First we show that if we have an enumerator E that enumerates a language A , a Turing machine M recognizes A . The Turing machine M works in the following way.

M = “On input w :

1. Run E every time that E outputs a string, compare it with w .
2. If w ever appears in the output of E , accept.”

Clearly, M accepts those strings that appear on E ’s list. Now consider the other direction. If M recognizes a language A , we can construct the following enumerator E for A . Say that s_1, s_2, s_3, \dots is a list of all possible strings in Σ^* .

E = “Ignore the input.

1. Repeat the following for $i = 1, 2, 3, \dots$
2. Run M for i steps on each input, s_1, s_2, \dots, s_i .
3. If any computations accept, print out the corresponding s_j .”

If M accepts a particular string s , eventually it will appear on the list generated by E . In fact, it will appear on the list infinitely many times because M runs from the beginning on each string for each repetition of step 1. This procedure gives the effect of running M in parallel on all possible input string. ■

4.3 The Pumping Lemma

Theorem 4.15 *Pumping Lemma for regular languages: if A is a regular language, then there is a number p , which is called the pumping length, where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:*

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

Proof: Let $M = \{Q, \Sigma, \delta, q_1, F\}$ be a DFA recognizing A and p be the number of the states of M . Let $s = s_1s_2 \dots s_n$ be a string in A of length n , where $n \geq p$. Let r_1, \dots, r_{n+1} be the sequence of states that M enters while processing s , so $r_{i+1} = \delta(r_i, s_i)$ for $1 \leq i \leq n$. This sequence has length $n+1$, which is at least $p+1$. Among the first $p+1$ elements in the sequence, two must be the same state, by the pigeonhole principle. We call the first of these r_j and the second r_l . Because r_l occurs among the first $p+1$ places in a sequence starting at r_1 , we have $l \leq p+1$. Now let $x = s_1 \dots s_{j-1}$, $y = s_j \dots s_{l-1}$, and $z = s_l \dots s_n$.

As x takes M from r_1 to r_j , y takes M from r_j to r_j , and z takes M from r_j to r_{n+1} , which is an accept state, M must accept xy^iz for $i \geq 0$. We know that $j \neq l$, so $|y| > 0$; and $l \leq p+1$, so $|xy| \leq p$. Thus all three conditions of the pumping lemma are satisfied. ■

Example Show that the language $C = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Proof: Assume to the contrary that C is regular. Let p be the pumping length given by the pumping lemma. Choose s to be the string $0^p 1^p$. Because s is a member of C and s has length more than p , the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string xy^iz is in C . We consider three cases to show that this result is impossible.

1. The string y consists only of 0's. In this case, the string $xyyz$ has more 0's than 1's and so is not a member of C , violating condition 1 of the pumping lemma. Contradiction.
2. The string y consists only of 1's. This case also gives a contradiction.
3. The string y consists of both 0's and 1's. In this case, the string $xyyz$ may have the same number of 0's and 1's, but they will be out of order with some 1's before 0's. hence it is not a member of C . Another Contradiction.

Thus a contradiction is unavoidable if we make the assumption that C is regular, so C is not regular. ■

Example Let $D = \{w \mid w \text{ has an equal number of 0's and 1's}\}$. Show that D is not regular.

Proof: Assume to the contrary that D is regular. Let p be the pumping length given by the pumping lemma. Let s be the string $0^p 1^p$. Since s is a member of D and having length more than p , the pumping lemma guarantees that s can be split into three pieces $s = xyz$, where for any $i \geq 0$ the string xy^iz is in D . By condition 3 of the pumping lemma, $|xy| \leq p$, y must consist only of 0's, so $xyyz \notin D$, therefore s cannot be pumped. Contradiction. ■

Theorem 4.16 *Pumping lemma for context-free languages: If A is a context-free language, then there is a number p , the pumping length, such that if s is any string in A of length at least p , then s may be divided into five pieces $s = uvxyz$ satisfying the conditions*

1. for each $i \geq 0$, $uv^i xy^i z \in A$
2. $|vy| > 0$
3. $|vxy| \leq p$

See the theorem 2.34 for the proof details.

Example Show that the language $E = \{a^n b^n c^n\}$ is not context-free.

Proof: Assume that E is a CFL and construct a contradiction. Let p be the pumping length for E that is guaranteed to exist by the pumping lemma. Select the string $s = a^p b^p c^p$. Clearly s is a member of E and of length at least p . The pumping lemma states that s can be pumped, but we show that it cannot. In other words, we show that no matter how we divide s into $uvxyz$, one of the three conditions of the lemma is violated.

First, condition 2 stipulates that either v or y is nonempty. Then we consider one of two cases, depending on whether substrings v and y contain more than one type of alphabet symbol.

1. When both v and y contain only one type of alphabet symbol, v does not contain both a 's and b 's or both b 's and c 's, and the same holds for y . In this case, the string $uv^2 xy^2 z$ cannot contain equal numbers of a 's, b 's, and c 's. Therefore, it cannot be a member of E . That violates condition 1 of the lemma and is thus a contradiction.
2. When either v or y contains more than one type of symbol, $uv^2 xy^2 z$ may contain equal numbers of the three alphabet symbols but not in the correct order. Hence it cannot be a member of E and a contradiction occurs.

One of these cases must occur, a contradiction is unavoidable. Therefore E is not a CFL. ■