T065001: Introduction to Formal Languages

Lecture 9: Turing machines (1)

Chapter 3.1 in Sipser's textbook

2025-06-16

(Lecture slides by Yih-Kuen Tsay)

Turing Machines

- Finite and pushdown automata are too restricted to serve as models of general-purpose computers.
- A *Turing machine* is similar to a finite automaton but with an unlimited and unrestricted memory—an infinite tape. It has a tape head that can read and write symbols and move around on the tape.
- A Turing machine can do everything that a real computer (as we know it) can do.
- Nonetheless, there are problems that no Turing machines, and hence no real computers, can solve.

Turing Machines (cont.)

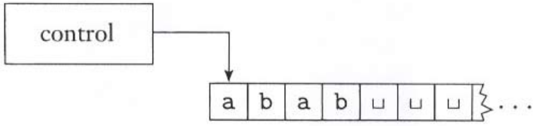


FIGURE 3.1
Schematic of a Turing machine

- The Turing machine model uses an infinite tape as its unlimited memory.
- It has a tape head that can read and write symbols and move around on the tape.
- Initially the tape contains only the input string and is blank everywhere else.
- If the machine needs to store information, it may write this information on the tape.

Turing Machines (cont.)

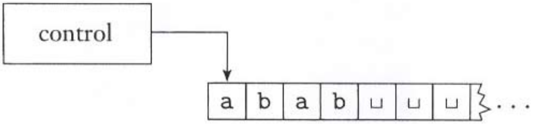


FIGURE 3.1
Schematic of a Turing machine

- To read the information that it has written, the machine can move its head back over it.
- The machine continues computing until it decides to produce an output.
- The outputs accept and reject are obtained by entering designated accepting and rejecting states.
- If it doesn't enter an accepting or a rejecting state, it will go on forever, never halting.

Formal Definition of a TM

Definition (3.3)

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

1. Q is the set of states,
2. Σ is the input alphabet, where the *blank* symbol $\sqcup \notin \Sigma$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Turing Machines (cont.)

A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ computes as follows.

- Initially, M receives input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is blank (i.e., filled with blank symbols).
- The head starts on the leftmost square of the tape.
- Note that Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input.

Turing Machines (cont.)

A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ computes as follows.

- Initially, M receives input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is blank (i.e., filled with blank symbols).
- The head starts on the leftmost square of the tape.
- Note that Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input.
- Once M has started, the computation proceeds according to the rules described by the transition function.
- If M ever tries to move its head to the left off the left-hand end of the tape, the head stays in the same place for that move, even though the transition function indicates L.

Turing Machines (cont.)

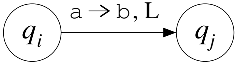
A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ computes as follows.

- Initially, M receives input $w = w_1 w_2 \cdots w_n \in \Sigma^*$ on the leftmost n squares of the tape, and the rest of the tape is blank (i.e., filled with blank symbols).
- The head starts on the leftmost square of the tape.
- Note that Σ does not contain the blank symbol, so the first blank appearing on the tape marks the end of the input.
- Once M has started, the computation proceeds according to the rules described by the transition function.
- If M ever tries to move its head to the left off the left-hand end of the tape, the head stays in the same place for that move, even though the transition function indicates L.
- The computation continues until it enters either the accept or reject states, at which point it halts. If neither occurs, M goes on forever.

$\Rightarrow$ Three possible outcomes: Any computation will accept, reject, or **loop**.

Turing Machines (cont.)

Notation used in state diagrams:


$$\begin{tikzcd} q_i \arrow[r, "\mathtt{a} \rightarrow \mathtt{b}, \mathrm{L}"] & q_j \end{tikzcd}$$

A label of the form "$\mathtt{a} \longrightarrow \mathtt{b}$, L" on a transition from a state q_i to a state q_j means $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{b}, \mathsf{L})$, i.e.:

> *When in state q_i with the tape head reading an* a*, write a* b*, move the tape head one step to the left, and go to state q_j.*

Analogously for "$\mathtt{a} \longrightarrow \mathtt{b}$, R" (move right).

Turing Machines (cont.)

Notation used in state diagrams:

$$\begin{tikzcd} q_i \arrow[r, "\mathtt{a} \to \mathtt{b}, \mathrm{L}"] & q_j \end{tikzcd}$$

A label of the form "$\mathtt{a} \longrightarrow \mathtt{b}$, L" on a transition from a state q_i to a state q_j means $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{b}, \mathsf{L})$, i.e.:

> *When in state q_i with the tape head reading an* a*, write a* b*, move the tape head one step to the left, and go to state q_j.*

Analogously for "$\mathtt{a} \longrightarrow \mathtt{b}$, R" (move right).

Furthermore, a transition labeled "$\mathtt{a} \longrightarrow$ L" from q_i to q_j means $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{a}, \mathsf{L})$ and "$\mathtt{a} \longrightarrow$ R" means $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{a}, \mathsf{R})$.
Note that the tape contents do not change here.

Turing Machines (cont.)

Notation used in state diagrams:

$$\begin{tikzcd} q_i \arrow[r, "\mathtt{a} \to \mathtt{b}, \mathrm{L}"] & q_j \end{tikzcd}$$

A label of the form "a $\longrightarrow$ b, L" on a transition from a state q_i to a state q_j means $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{b}, \mathsf{L})$, i.e.:

> *When in state q_i with the tape head reading an* a*, write a* b*, move the tape head one step to the left, and go to state q_j.*

Analogously for "a $\longrightarrow$ b, R" (move right).

Furthermore, a transition labeled "a $\longrightarrow$ L" from q_i to q_j means $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{a}, \mathsf{L})$ and "a $\longrightarrow$ R" means $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{a}, \mathsf{R})$. Note that the tape contents do not change here.

Finally, "a, b $\longrightarrow$ L" on a transition from q_i to q_j is a compact way of expressing both $\delta(q_i, \mathtt{a}) = (q_j, \mathtt{a}, \mathsf{L})$ and $\delta(q_i, \mathtt{b}) = (q_j, \mathtt{b}, \mathsf{L})$. Analogously for "a, b $\longrightarrow$ R".

Example 1

Let $B = \{w\#w \mid w \in \{0,1\}^*\}$. A Turing machine M_1 for B may work as follows:

Example 1

Let $B = \{w\#w \mid w \in \{0,1\}^*\}$. A Turing machine M_1 for B may work as follows:

1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, *reject*. Cross off symbols as they are checked to keep track of which symbols correspond.
2. When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right of the #. If any symbols remain, *reject*; otherwise, *accept*."

Example 1 (cont.)

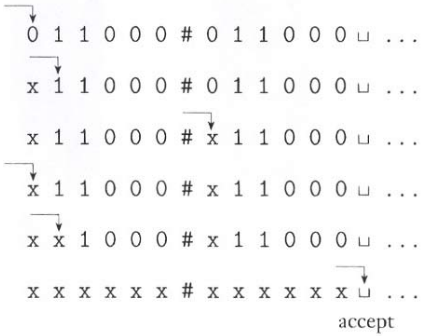
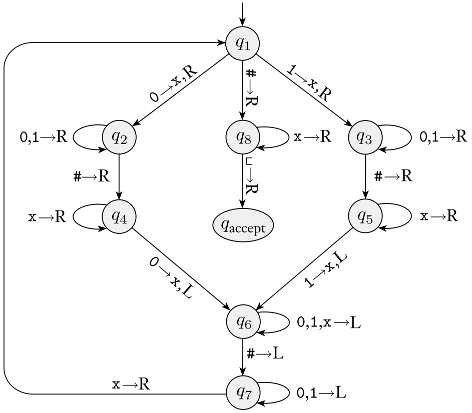

0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .
x 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .
x 1 1 0 0 0 # x 1 1 0 0 0 ␣ . . .
x 1 1 0 0 0 # x 1 1 0 0 0 ␣ . . .
x x 1 0 0 0 # x 1 1 0 0 0 ␣ . . .
x x x x x x # x x x x x x ␣ . . .
accept

FIGURE 3.2
Snapshots of Turing machine M_1 computing on input 011000#011000

Example 1

State diagram of Turing machine M_1 for the language $B = \{w\#w \mid w \in \{0,1\}^*\}$:



Here, $\Sigma = \{0, 1, \#\}$ and $\Gamma = \{0, 1, \#, x, \sqcup\}$. The start state is q_1.
Also: $\delta(q_1, 0) = (q_2, x, R)$, $\delta(q_1, \#) = (q_8, \#, R)$, $\delta(q_2, 0) = (q_2, 0, R)$, etc.
(To simplify the figure, the state q_{reject} and transitions to q_{reject} are not drawn.)

Configurations of a TM

- As a TM computes, changes occur in
  1. the current state,
  2. the current tape contents, and
  3. the current head location.
- A setting of these three items is called a **configuration** of the TM.
- We write uqv to denote the configuration where
  1. the current state is q,
  2. the current tape contents is uv, and
  3. the current head location is the first symbol of v.

Configurations of a TM

- As a TM computes, changes occur in
  1. the current state,
  2. the current tape contents, and
  3. the current head location.
- A setting of these three items is called a **configuration** of the TM.
- We write uqv to denote the configuration where
  1. the current state is q,
  2. the current tape contents is uv, and
  3. the current head location is the first symbol of v.

The tape is always filled with blanks after the last symbol of v.
Therefore, $uqv = uqv_\sqcup = uqv_{\sqcup\sqcup} = \dots$

Configurations of a TM (cont.)

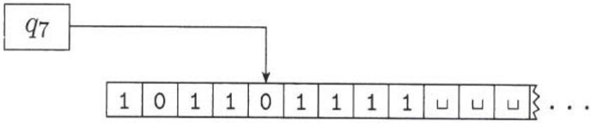


FIGURE 3.4
A Turing machine with configuration $1011q_701111$

Configurations of a TM (cont.)

- $q_0 w$ is the *start configuration* on input w.
- $u q_{\mathrm{accept}} v$ is an *accepting configuration*.
- $u q_{\mathrm{reject}} v$ is a *rejecting configuration*.
- Accepting and rejecting configurations are *halting configurations*.

Computation of a TM

Configuration C_1 *yields* configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step:

1. uaq_ibv yields uq_jacv if $\delta(q_i, b) = (q_j, c, L)$.
2. uaq_ibv yields $uacq_jv$ if $\delta(q_i, b) = (q_j, c, R)$.

Computation of a TM

Configuration C_1 *yields* configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step:

1. uaq_ibv yields uq_jacv if $\delta(q_i, b) = (q_j, c, L)$.
2. uaq_ibv yields $uacq_jv$ if $\delta(q_i, b) = (q_j, c, R)$.

Remark 1: The special case where the head is at the left end of the configuration is covered by 1. and 2. since taking $u = \varepsilon$ and $a = \varepsilon$ gives:

q_ibv yields q_jcv if $\delta(q_i, b) = (q_j, c, L)$

q_ibv yields cq_jv if $\delta(q_i, b) = (q_j, c, R)$

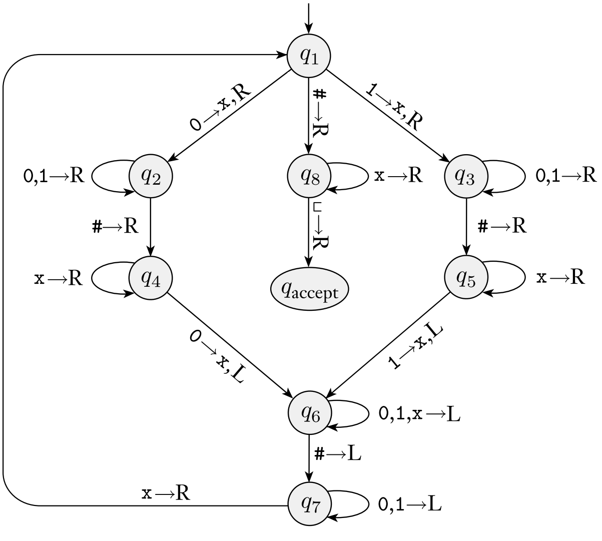
Remark 2: The special case where the head is at the right end of the configuration is also covered by 1. and 2. since we can apply them by using the fact that $uaq_i = uaq_i\sqcup$ and taking $b = \sqcup$ and $v = \varepsilon$.

Computation of a TM (cont.)

- A Turing machine *accepts* input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists where
  1. C_1 the start configuration on w,
  2. 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 $L(M)$.

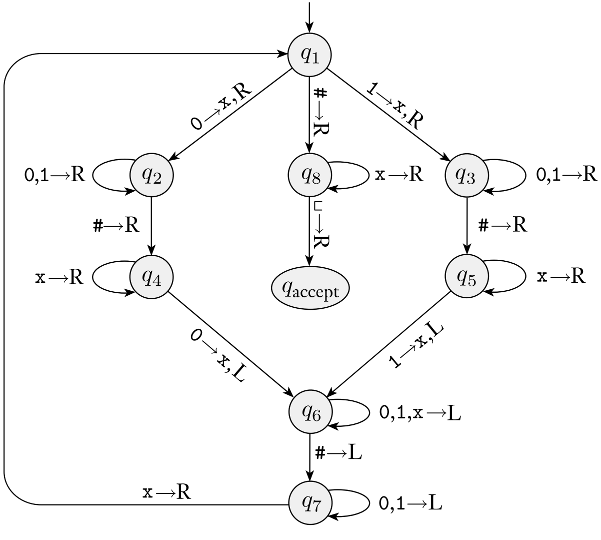
Exercise: What is the sequence of configurations entered by M_1 on input 1#1?

$q_1$1#1,



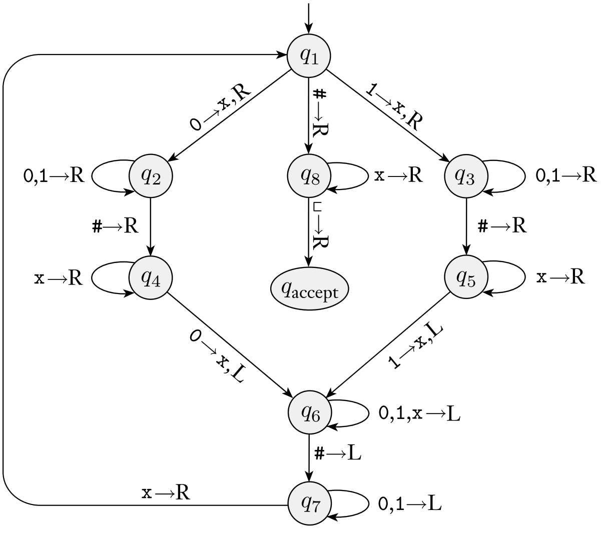
Exercise: What is the sequence of configurations entered by M_1 on input `1#1`?

q_1`1#1`, `x`q_3`#1`,



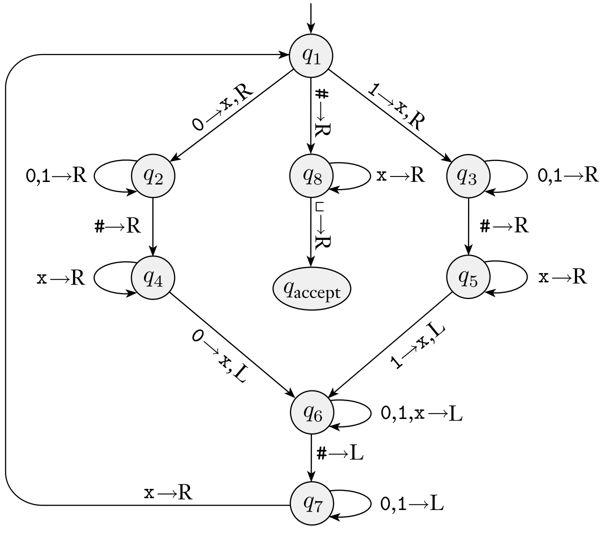
Exercise: What is the sequence of configurations entered by M_1 on input 1#1?

$q_1$1#1, xq_3#1, x#$q_5$1,



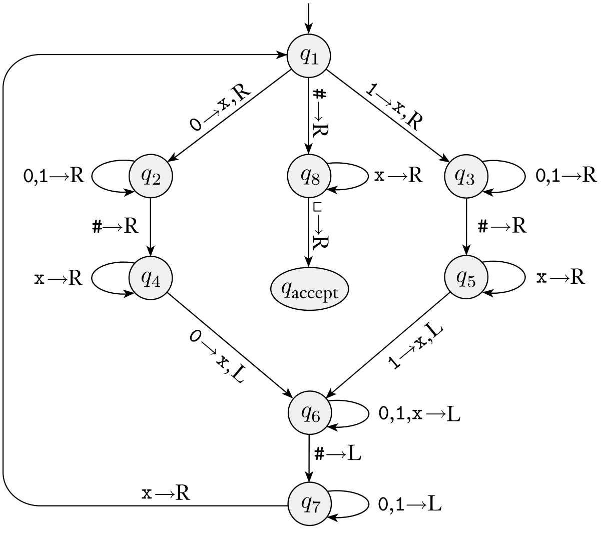
Exercise: What is the sequence of configurations entered by M_1 on input 1#1?

$q_1$1#1, xq_3#1, x#$q_5$1, xq_6#x,



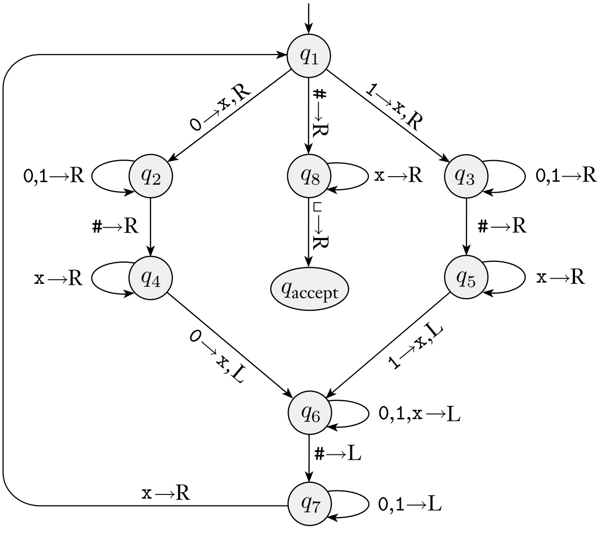
Exercise: What is the sequence of configurations entered by M_1 on input 1#1?

$q_1$1#1, xq_3#1, x#$q_5$1, xq_6#x, q_7x#x,



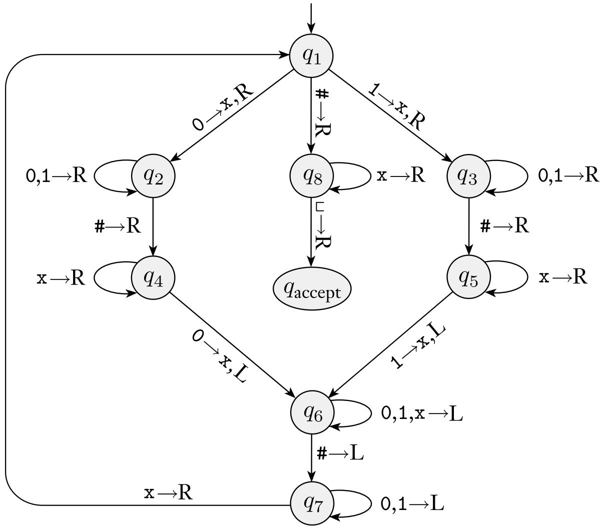
Exercise: What is the sequence of configurations entered by M_1 on input 1#1?

$q_1$1#1, xq_3#1, x#$q_5$1, xq_6#x, q_7x#x, xq_1#x,



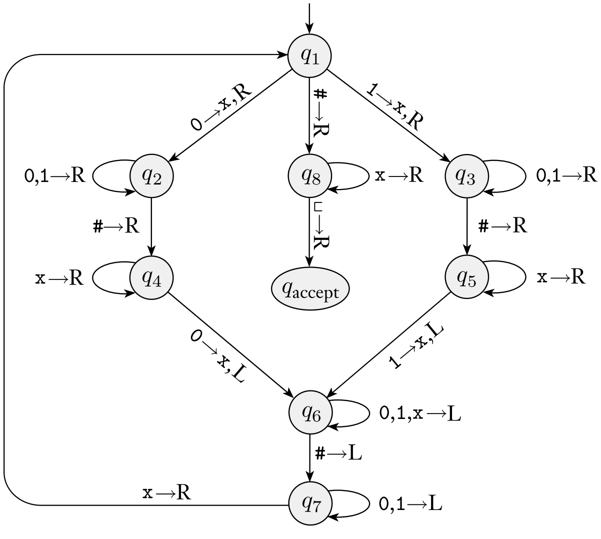
Exercise: What is the sequence of configurations entered by M_1 on input `1#1`?

q_1`1#1`, `x`q_3`#1`, `x#`q_5`1`, `x`q_6`#x`, q_7`x#x`, `x`q_1`#x`, `x#`q_8`x`,



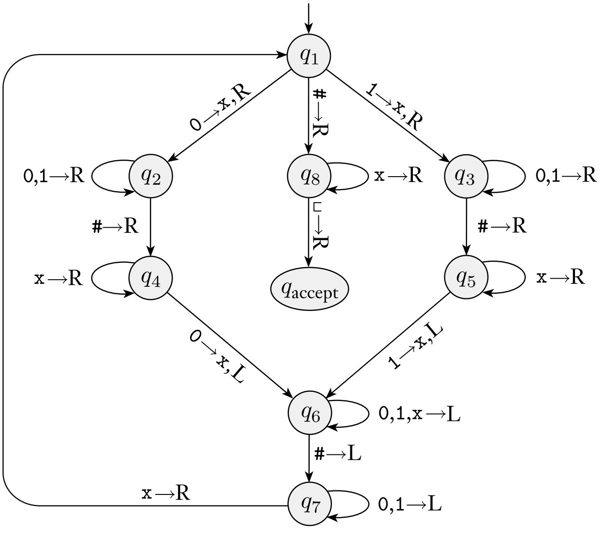
Exercise: What is the sequence of configurations entered by M_1 on input 1#1?

$q_1$1#1, xq_3#1, x#$q_5$1, xq_6#x, q_7x#x, xq_1#x, x#q_8x, x#x$q_{8\sqcup}$,



Exercise: What is the sequence of configurations entered by M_1 on input `1#1`?

q_1`1#1`, `x`q_3`#1`, `x#`q_5`1`, `x`q_6`#x`, q_7`x#x`, `x`q_1`#x`, `x#`q_8`x`, `x#x`q_8`␣`, `x#x␣`q_{accept}



Example 2

Define $A = \{0^{2^n} \mid n \geq 0\}$, i.e., A consists of all nonempty strings of 0s whose length is a power of 2. A Turing machine M_2 for A:

Example 2

Define $A = \{0^{2^n} \mid n \geq 0\}$, i.e., A consists of all nonempty strings of 0s whose length is a power of 2. A Turing machine M_2 for A:

1. Sweep left to right across the tape, crossing off every second 0.
2. If in stage 1 the tape contained a single 0, *accept*.
3. If in stage 1 the tape contained more than one 0 and the number of 0s was odd, *reject*.
4. Return head to the left-hand end of the tape.
5. Go to stage 1.

Example 2

Define $A = \{0^{2^n} \mid n \geq 0\}$, i.e., A consists of all nonempty strings of 0s whose length is a power of 2. A Turing machine M_2 for A:

1. Sweep left to right across the tape, crossing off every second 0.
2. If in stage 1 the tape contained a single 0, *accept*.
3. If in stage 1 the tape contained more than one 0 and the number of 0s was odd, *reject*.
4. Return head to the left-hand end of the tape.
5. Go to stage 1.

Remarks:
To implement step 1., let M_2 cross off 0s by replacing them by $\mathtt{x}$-symbols.

Example 2

Define $A = \{0^{2^n} \mid n \geq 0\}$, i.e., A consists of all nonempty strings of 0s whose length is a power of 2. A Turing machine M_2 for A:

1. Sweep left to right across the tape, crossing off every second 0.
2. If in stage 1 the tape contained a single 0, *accept*.
3. If in stage 1 the tape contained more than one 0 and the number of 0s was odd, *reject*.
4. Return head to the left-hand end of the tape.
5. Go to stage 1.

Remarks:

To implement step 1., let M_2 cross off 0s by replacing them by $\mathtt{x}$-symbols.

Also, replace the very first 0 by a $\sqcup$-symbol in a preprocessing step before step 1 so that M_2 can find the left end of the tape in step 4. (When counting 0s, interpret that particular $\sqcup$-symbol as a non-crossed out 0.)

Example 2 (cont.)

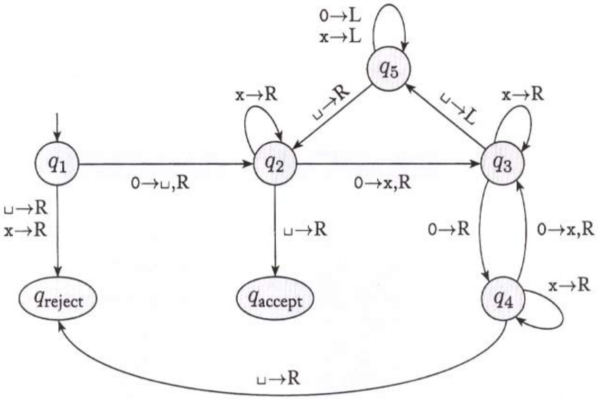


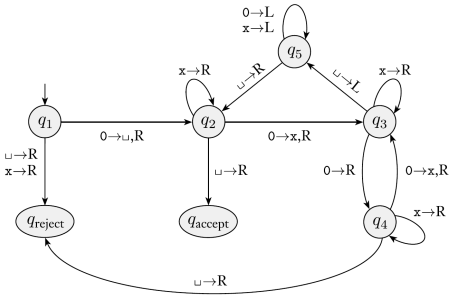
FIGURE 3.8
State diagram for Turing machine M_2

Example 2

$q_1\mathtt{0000}$
$\sqcup q_2\mathtt{000}$
$\sqcup\mathtt{x}q_3\mathtt{00}$
$\sqcup\mathtt{x0}q_4\mathtt{0}$
$\sqcup\mathtt{x0x}q_3\sqcup$
$\sqcup\mathtt{x0}q_5\mathtt{x}\sqcup$
$\sqcup\mathtt{x}q_5\mathtt{0x}\sqcup$

$\sqcup q_5\mathtt{x0x}\sqcup$
$q_5\sqcup\mathtt{x0x}\sqcup$
$\sqcup q_2\mathtt{x0x}\sqcup$
$\sqcup\mathtt{x}q_2\mathtt{0x}\sqcup$
$\sqcup\mathtt{xx}q_3\mathtt{x}\sqcup$
$\sqcup\mathtt{xxx}q_3\sqcup$
$\sqcup\mathtt{xx}q_5\mathtt{x}\sqcup$

$\sqcup\mathtt{x}q_5\mathtt{xx}\sqcup$
$\sqcup q_5\mathtt{xxx}\sqcup$
$q_5\sqcup\mathtt{xxx}\sqcup$
$\sqcup q_2\mathtt{xxx}\sqcup$
$\sqcup\mathtt{x}q_2\mathtt{xx}\sqcup$
$\sqcup\mathtt{xx}q_2\mathtt{x}\sqcup$
$\sqcup\mathtt{xxx}q_2\sqcup$
$\sqcup\mathtt{xxx}\sqcup q_{\mathrm{accept}}$


q_1
q_2
q_3
q_4
q_5
q_{reject}
q_{accept}
$\mathtt{0}\to\sqcup,\mathrm{R}$
$\mathtt{0}\to\mathtt{x},\mathrm{R}$
$\sqcup\to\mathrm{R}$
$\mathtt{x}\to\mathrm{R}$
$\sqcup\to\mathrm{R}$
$\mathtt{x}\to\mathrm{R}$
$\sqcup\to\mathrm{R}$
$\mathtt{0}\to\mathrm{R}$
$\mathtt{0}\to\mathtt{x},\mathrm{R}$
$\mathtt{x}\to\mathrm{R}$
$\mathtt{x}\to\mathrm{R}$
$\sqcup\to\mathrm{R}$
$\sqcup\to\mathrm{L}$
$\mathtt{0}\to\mathrm{L}$
$\mathtt{x}\to\mathrm{L}$
$\sqcup\to\mathrm{R}$

Turing Machines (cont.)

- We can formally describe a particular Turing machine by specifying each of its seven parts.
- However, going to that level of detail can be cumbersome for all but the tiniest Turing machines.
- Accordingly, we won't spend much time giving such descriptions.
- Mostly we will give only higher level descriptions because they are precise enough for our purposes and are much easier to understand.
- Nevertheless, it is important to remember that every higher level description is actually just shorthand for its formal counterpart.
- With patience and care we could describe any of the Turing machines in our class in complete formal detail.

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
a a a b b b b b c c c c c c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a b b b b b c c c c c c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a y b b b b c c c c c c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a y b b b b x c c c c c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a y y b b b x c c c c c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a y y b b b x x c c c c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a y y y b b x x c c c c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a y y y y y x x x x x c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x a a b b b b b x x x x x c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

```
x x a b b b b b x x x x x c c c c c c c c c c
```

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol $\mathtt{x}$ to cross out a's and c's, and symbol $\mathtt{y}$ to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

. . .

Example 3

$C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$. A Turing machine M_3 for C:

1. Scan the input to be sure that it is a member of $aa^*bb^*cc^*$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*. Use symbol x to cross out a's and c's, and symbol y to cross out b's.
4. Restore the crossed off b's and repeat Stage 3 if there is another a to cross off.
5. If all a's and c's are crossed off, *accept*; otherwise, *reject*.

`x x x b b b b b x x x x x x x x x x x x x x x` **ACCEPT**

Example 3 (cont.)

Technical issue: In step 2, how can M_3 see when it's at the left end of the tape?

Two alternative approaches:

Example 3 (cont.)

Technical issue: In step 2, how can M_3 see when it's at the left end of the tape?

Two alternative approaches:

1. Mark the leftmost symbol in some way when the machine starts with its head on that symbol (like we did in Example 2 earlier today).

Example 3 (cont.)

Technical issue: In step 2, how can M_3 see when it's at the left end of the tape?

Two alternative approaches:

1. Mark the leftmost symbol in some way when the machine starts with its head on that symbol (like we did in Example 2 earlier today).

OR:

2. By definition, if a TM tries to move its head beyond the left-hand end of the tape, it stays in the same place. Take advantage of this fact as follows:
    i. First write a special symbol over the current position while remembering the symbol that it replaced, implemented by using some additional states.
    ii. Then, try to move the head to the left.
    The head is still over the special symbol if and only if the head was at the left-hand end of the tape.
    iii. Restore the changed symbol to the original symbol and continue.

(From earlier today)

- A Turing machine *accepts* input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists where
  1. C_1 the start configuration on w,
  2. 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 $L(M)$.

Decidable Languages

Definition (3.5)

A language is **Turing-recognizable** (also called *recursively enumerable*) if some Turing machine recognizes it.

Decidable Languages

Definition (3.5)

A language is **Turing-recognizable** (also called *recursively enumerable*) if some Turing machine recognizes it.

- A Turing machine can fail to accept an input by entering the q_{reject} state and rejecting, or by looping (not halting).
- A machine is called a *decider* if it halts on all inputs. A decider that recognizes some language is said to *decide* the language.

Definition (3.6)

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

Remark: The TMs M_1, M_2, and M_3 in today's lecture were all deciders.

Decidable Languages

Comparison between the two definitions:

- Language L is decidable: There exists a TM M such that for each string $w \in L$, M accepts w, while for each $w \notin L$, M rejects w. M does not get stuck in an infinite loop for any w.
- Language L is Turing-recognizable: There exists a TM M such that for each string $w \in L$, M accepts w. For each $w \notin L$, M either rejects or loops.

	$w \in L$	$w \notin L$
Decidable	accept	reject
Turing-recognizable	accept	reject or loop

Decidable Languages

Comparison between the two definitions:

- Language L is decidable: There exists a TM M such that for each string $w \in L$, M accepts w, while for each $w \notin L$, M rejects w.
  M does not get stuck in an infinite loop for any w.

- Language L is Turing-recognizable: There exists a TM M such that for each string $w \in L$, M accepts w.
  For each $w \notin L$, M either rejects or loops.

	$w \in L$	$w \notin L$
Decidable	accept	reject
Turing-recognizable	accept	reject or loop

By definition, "decidable" implies "Turing-recognizable".
In Lecture 11, we'll prove that "decidable" $\neq$ "Turing-recognizable".

Classes of Languages

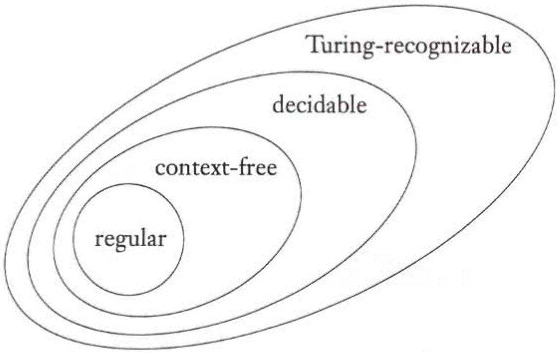


FIGURE **4.10**
The relationship among classes of languages