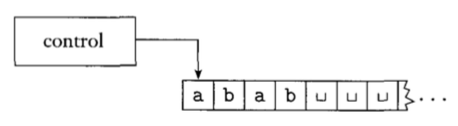
Computability and Complexity
COSC 4200

Turing Machines

Turing Machines (Alan Turing, 1936)

Turing Machines

- simple model of a general purpose computer
- finite automaton with infinite read/write tape



Alan Turing

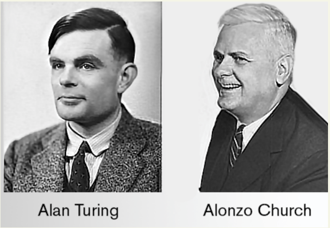
Church-Turing Thesis

Church-Turing Thesis

intuitive notion of algorithms

equals

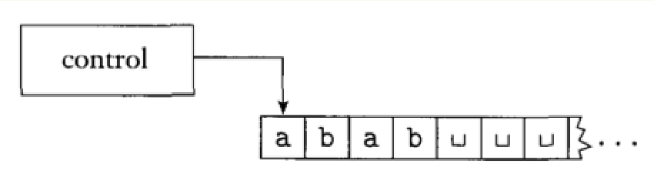
Turing machine algorithms



Turing Machine Operation

Initially, the tape contains the input string on the left and is blank everywhere else. The tape head is on the first cell and the TM is in its initial state.

The TM has a transition function telling what to do based on what the tape head is reading and the current state: the TM can move its tape head left or right, possibly writing information on the the tape.



Informal description of a TM that accepts
$\{w \in \{0,1\}^* \mid w \text{ is a palindrome}\}$.

1. If the current cell is blank, ACCEPT. // even length palindrome

   Remember the symbol in the current cell and erase it with a blank symbol.

Informal description of a TM that accepts
$\{w \in \{0,1\}^* \mid w \text{ is a palindrome}\}$.

1. If the current cell is blank, ACCEPT. // even length palindrome

   Remember the symbol in the current cell and erase it with a blank symbol.

2. Scan to the right until a blank symbol is found.
   Move back to the left one cell.

Informal description of a TM that accepts
$\{w \in \{0,1\}^* \mid w \text{ is a palindrome}\}$.

1. If the current cell is blank, ACCEPT. // even length palindrome

   Remember the symbol in the current cell and erase it with a blank symbol.

2. Scan to the right until a blank symbol is found.
   Move back to the left one cell.

3. If the current cell is blank, ACCEPT. // odd length palindrome

   If the symbol in the current cell matches the remembered symbol, erase it with a blank symbol and go to step 4. If it does not match, REJECT.

Informal description of a TM that accepts
$\{w \in \{0,1\}^* \mid w \text{ is a palindrome}\}$.

1. If the current cell is blank, ACCEPT. // even length palindrome

   Remember the symbol in the current cell and erase it with a blank symbol.

2. Scan to the right until a blank symbol is found.
   Move back to the left one cell.

3. If the current cell is blank, ACCEPT. // odd length palindrome

   If the symbol in the current cell matches the remembered symbol, erase it with a blank symbol and go to step 4. If it does not match, REJECT.

4. Scan to the left until a blank symbol is found.
   Move back to the right one cell.
   Go to step 1.

Formal Definition of a Turing Machine

Definition

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

1. Q is a finite set of *states*
2. Σ is the *input alphabet* not containing the special *blank* symbol $\sqcup$
3. Γ is the *tape alphabet*, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the *transition function*
5. $q_0 \in Q$ is the *initial state*
6. $q_{\mathrm{accept}} \in Q$ is the *accept state*
7. $q_{\mathrm{reject}} \in Q$ is the *reject state*

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the *transition function*

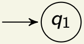
If $\delta(q, a) = (p, b, R)$, then if the TM is in state q and the tape head is reading a, the TM will *write* b on the current tape cell (replacing a), move the tape head to the *right*, and transition to *state* p.

$$q \xrightarrow{a \to b, R} p$$

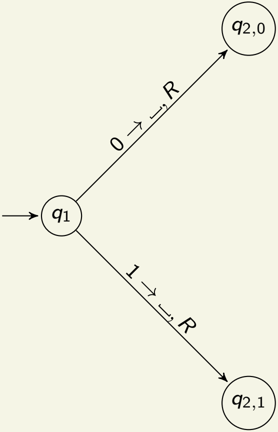
If $\delta(q, a) = (p, b, L)$, then the same things happen, except the head moves to the left.

$$q \xrightarrow{a \to b, L} p$$

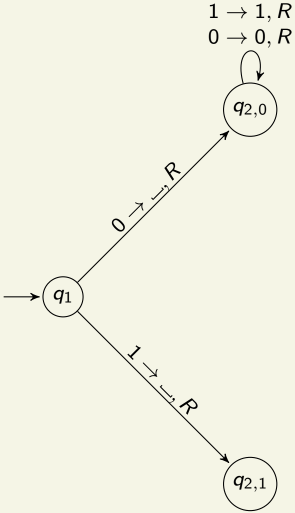
TM for Palindromes



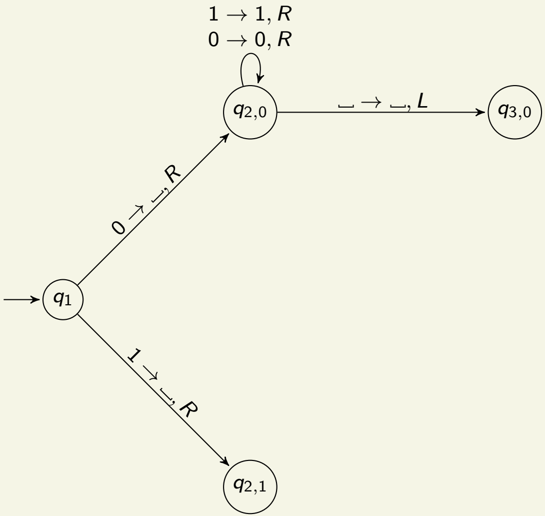
TM for Palindromes



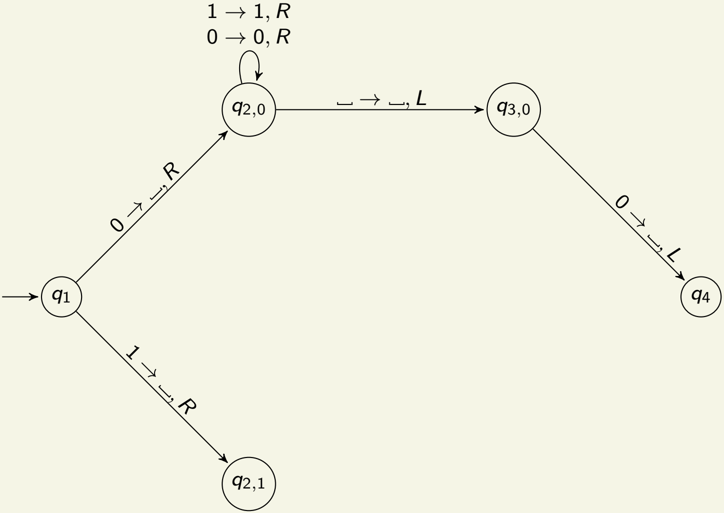
TM for Palindromes



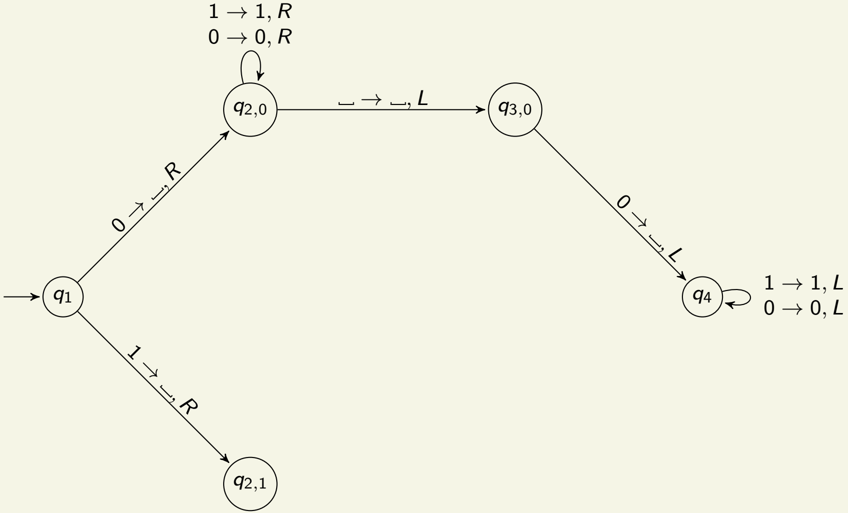
TM for Palindromes



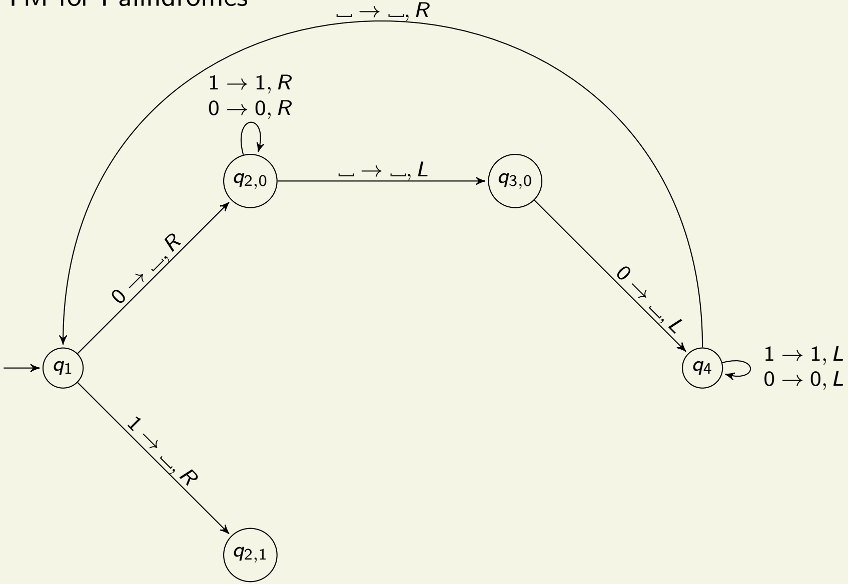
TM for Palindromes



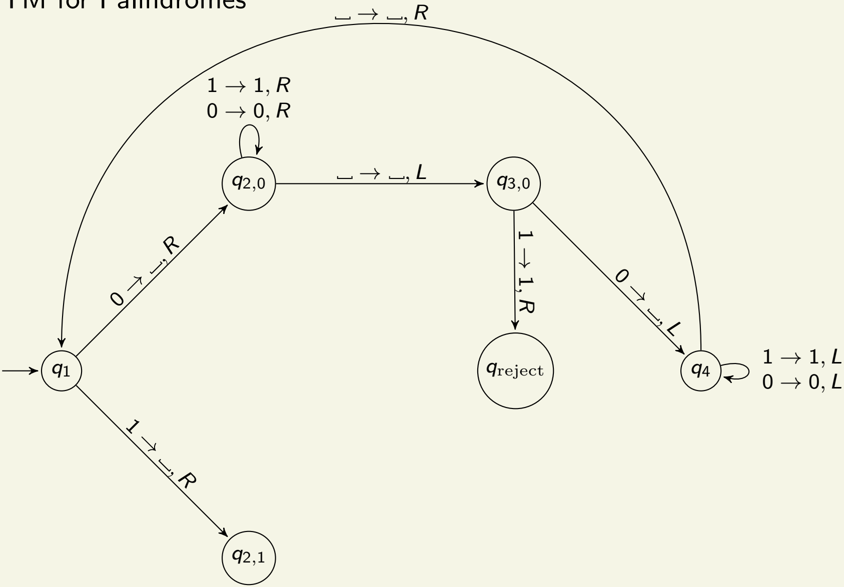
TM for Palindromes



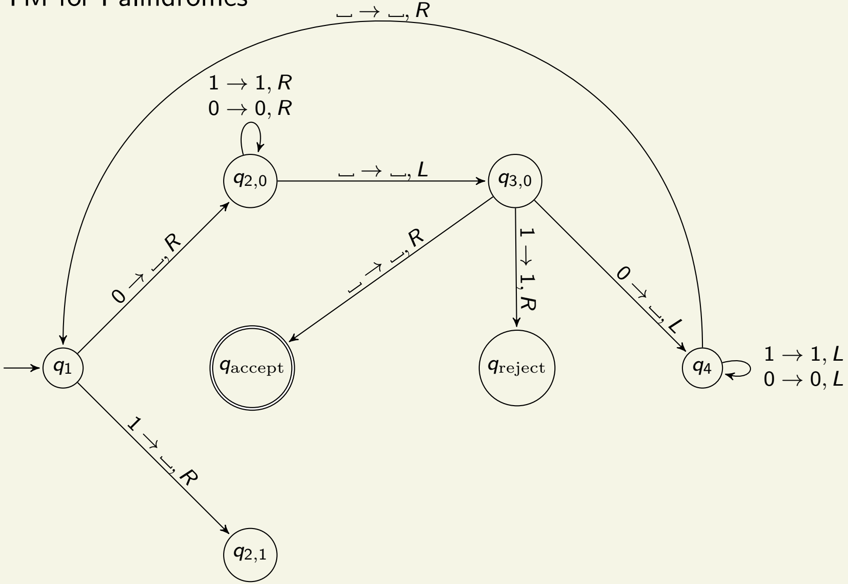
TM for Palindromes



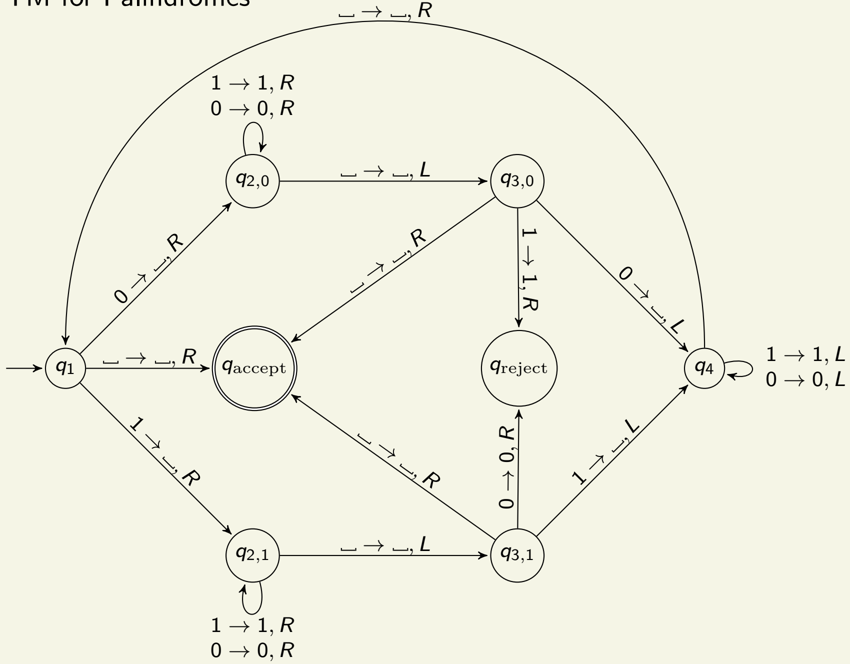
TM for Palindromes



TM for Palindromes

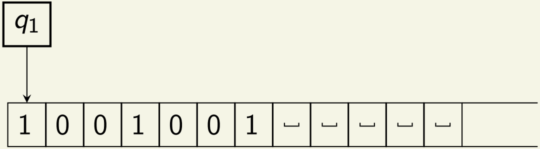


TM for Palindromes



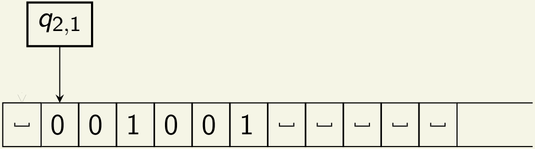
TM for Palindromes

input: 1001001

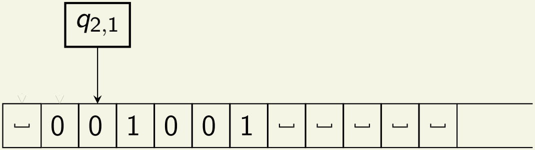


TM for Palindromes

input: 1001001

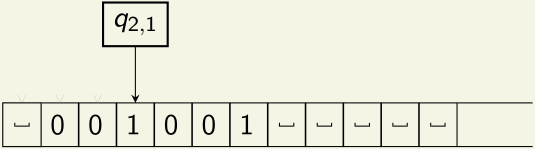


TM for Palindromes
input: 1001001

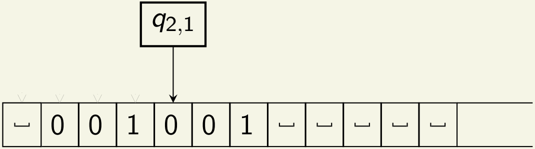


TM for Palindromes

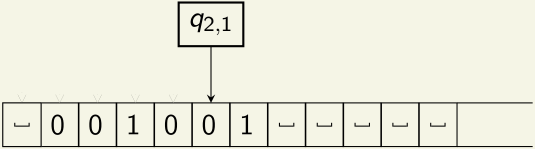
input: 1001001



TM for Palindromes
input: 1001001

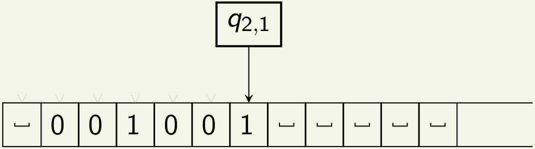


TM for Palindromes
input: 1001001



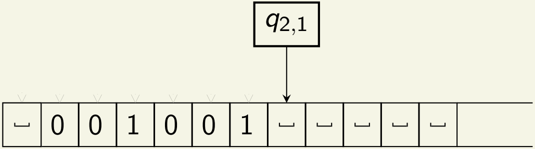
TM for Palindromes

input: 1001001

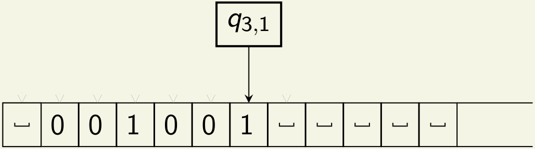


TM for Palindromes

input: 1001001

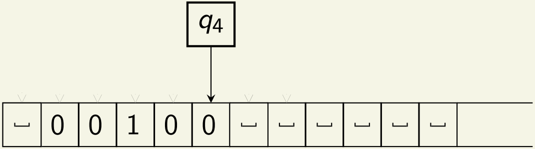


TM for Palindromes
input: 1001001



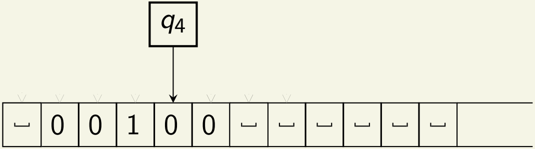
TM for Palindromes

input: 1001001



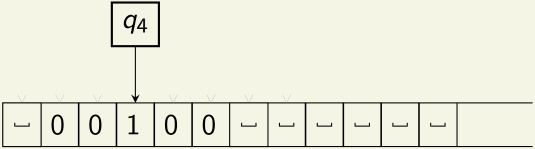
TM for Palindromes

input: 1001001

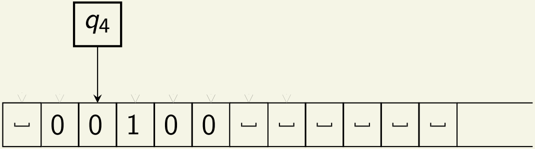


TM for Palindromes

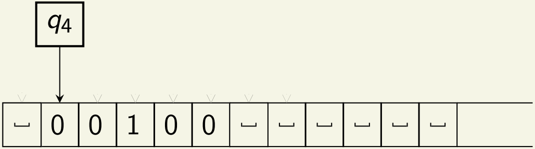
input: 1001001



TM for Palindromes
input: 1001001

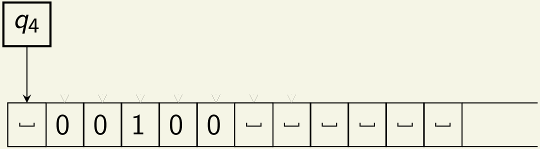


TM for Palindromes
input: 1001001



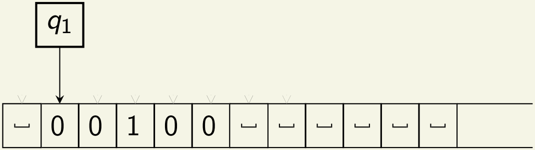
TM for Palindromes

input: 1001001

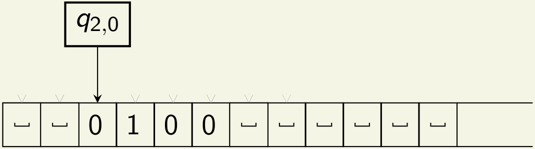


TM for Palindromes

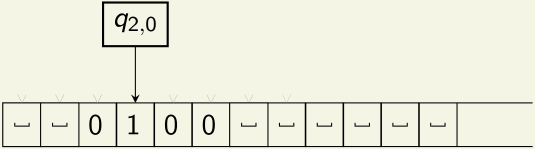
input: 1001001



TM for Palindromes
input: 1001001

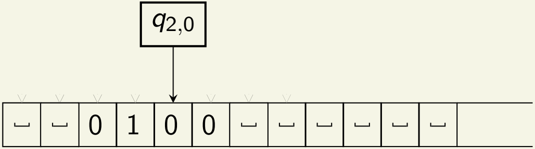


TM for Palindromes
input: 1001001



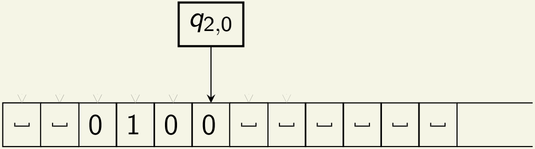
TM for Palindromes

input: 1001001



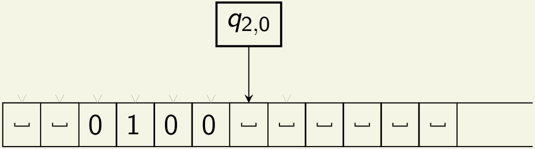
TM for Palindromes

input: 1001001



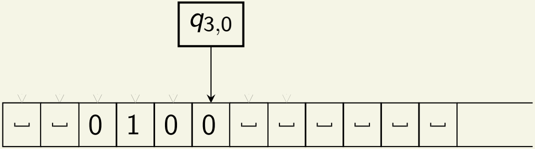
TM for Palindromes

input: 1001001



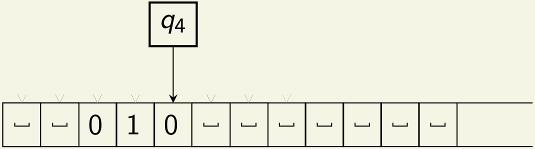
TM for Palindromes

input: 1001001



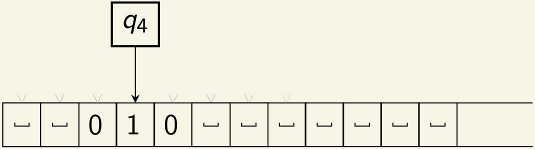
TM for Palindromes

input: 1001001

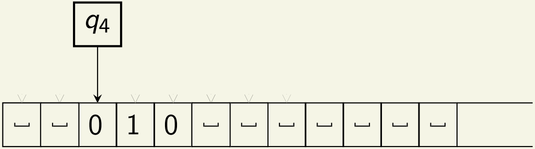


TM for Palindromes

input: 1001001

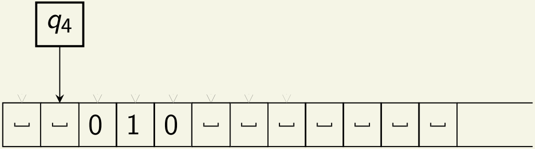


TM for Palindromes
input: 1001001



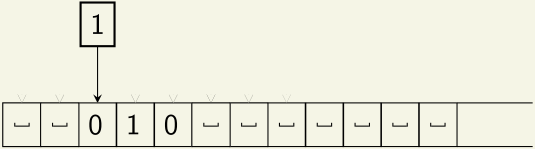
TM for Palindromes

input: 1001001

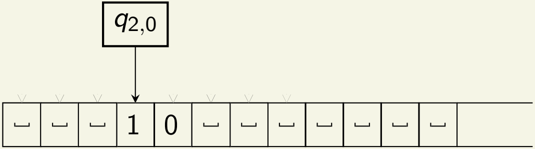


TM for Palindromes

input: 1001001

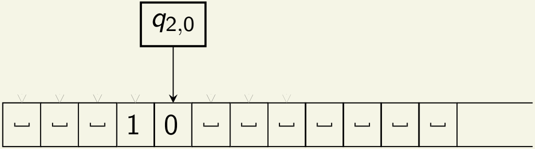


TM for Palindromes
input: 1001001



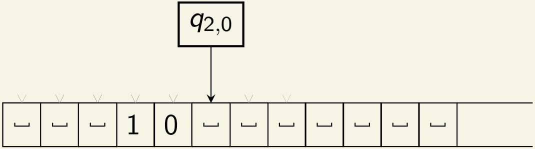
TM for Palindromes

input: 1001001



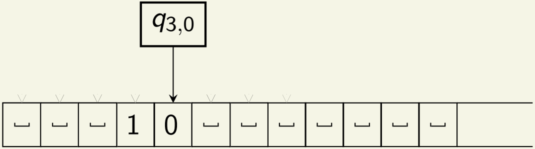
TM for Palindromes

input: 1001001

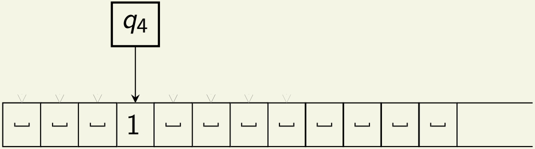


TM for Palindromes

input: 1001001

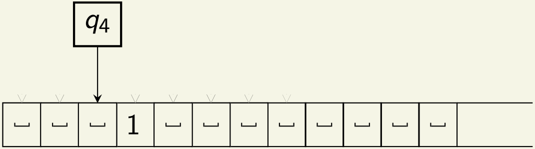


TM for Palindromes
input: 1001001

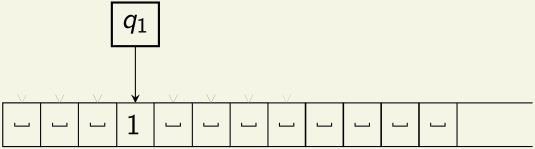


TM for Palindromes

input: 1001001

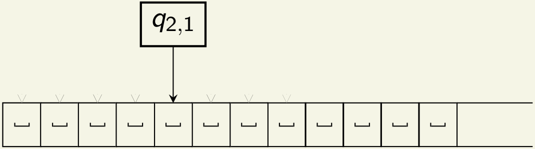


TM for Palindromes
input: 1001001



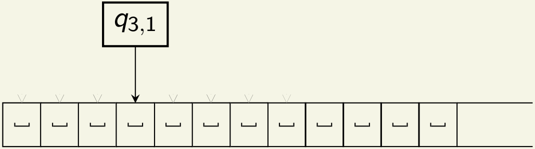
TM for Palindromes

input: 1001001

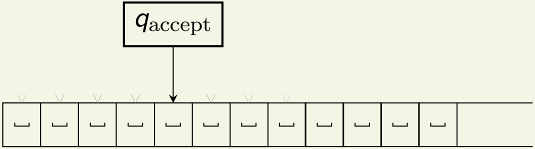


TM for Palindromes

input: 1001001

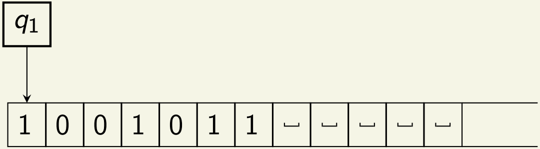


TM for Palindromes
input: 1001001



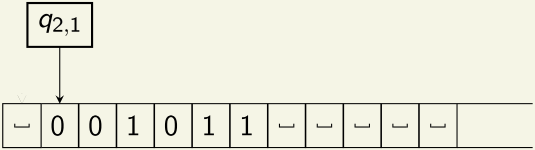
TM for Palindromes

input: 1001011



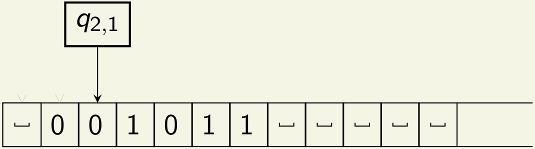
TM for Palindromes

input: 1001011



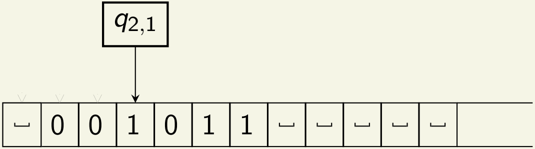
TM for Palindromes

input: 1001011

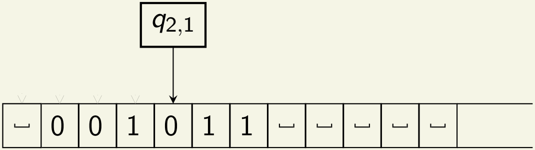


TM for Palindromes

input: 1001011

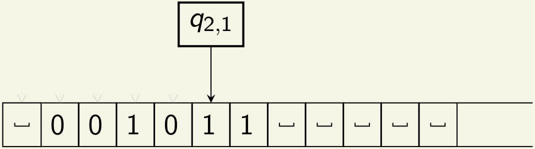


TM for Palindromes
input: 1001011



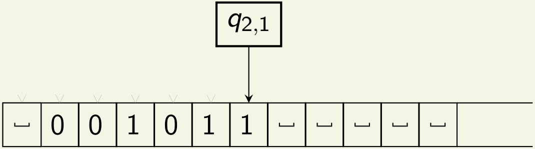
TM for Palindromes

input: 1001011



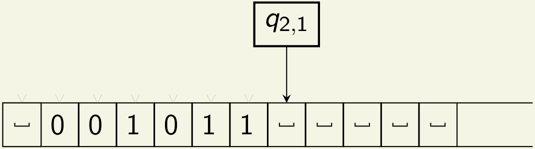
TM for Palindromes

input: 1001011

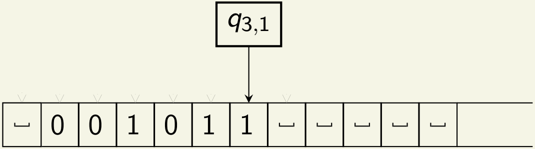


TM for Palindromes

input: 1001011

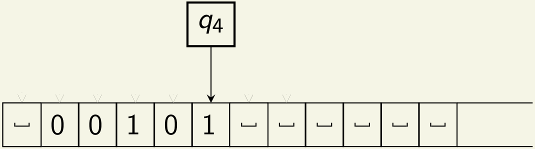


TM for Palindromes
input: 1001011



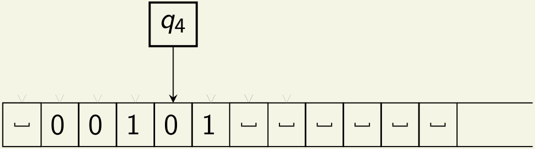
TM for Palindromes

input: 1001011



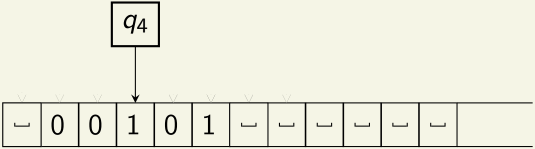
TM for Palindromes

input: 1001011



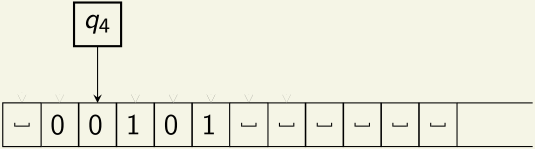
TM for Palindromes

input: 1001011



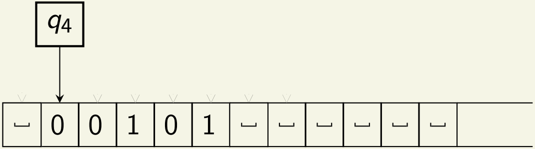
TM for Palindromes

input: 1001011

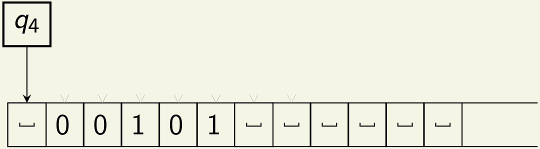


TM for Palindromes

input: 1001011

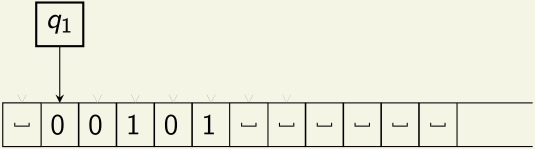


TM for Palindromes
input: 1001011



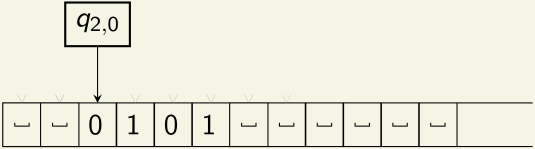
TM for Palindromes

input: 1001011



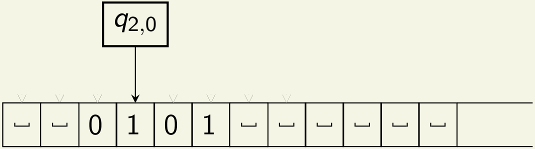
TM for Palindromes

input: 1001011



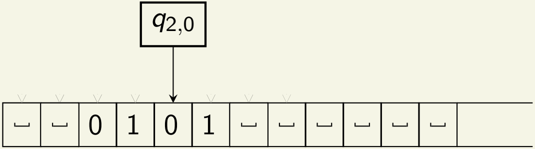
TM for Palindromes

input: 1001011



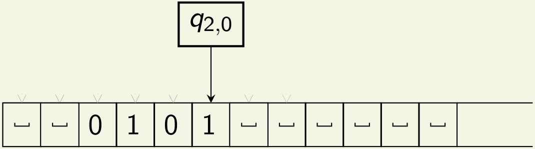
TM for Palindromes

input: 1001011



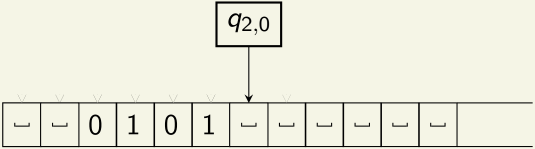
TM for Palindromes

input: 1001011

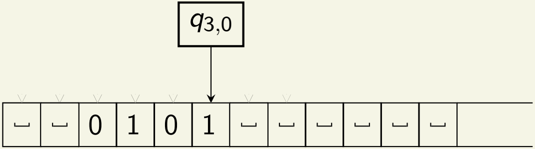


TM for Palindromes

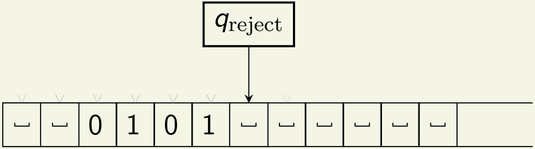
input: 1001011



TM for Palindromes
input: 1001011

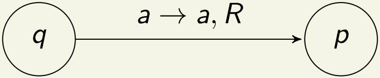


TM for Palindromes
input: 1001011

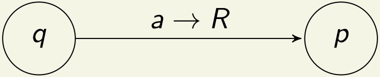


Simplified Turing Machine Diagrams

Consider a transition that doesn't change the symbol in the current cell:



We can use the following shorthand for the above transition.



Simplified Turing Machine Diagrams

Consider a transition that doesn't change the symbol in the current cell:

$$q \xrightarrow{\quad a \to a, R \quad} p$$

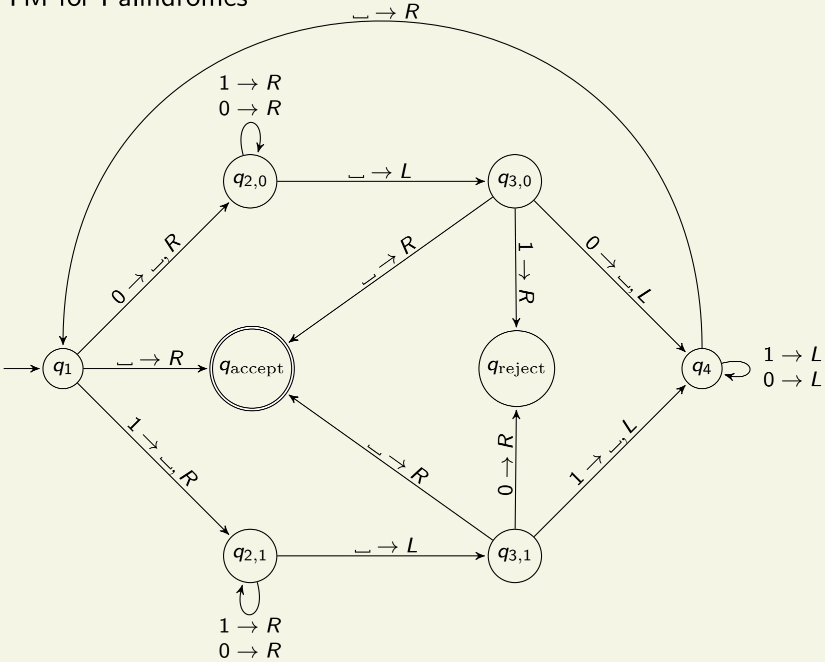
We can use the following shorthand for the above transition.

$$q \xrightarrow{\quad a \to R \quad} p$$

Analogously, for going to the left:

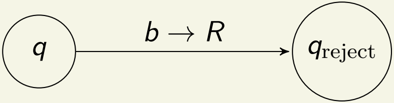
$$q \xrightarrow{\quad a \to L \quad} p$$

TM for Palindromes



Simplified Turing Machine Diagrams

If the diagram shows no transition for a state q and symbol b, there is an implicit transition to the reject state.



Recall the language $POWERS = \{0^{2^n} \mid n \geq 0\}$ is not context-free.

Informal description of a TM that accepts $POWERS$

1. Sweep left to right across the tape, crossing off every other 0.

Recall the language $POWERS = \{0^{2^n} \mid n \geq 0\}$ is not context-free.

Informal description of a TM that accepts *POWERS*

1. Sweep left to right across the tape, crossing off every other 0.
   - If the tape contained no 0s, REJECT.

Recall the language $POWERS = \{0^{2^n} \mid n \geq 0\}$ is not context-free.

Informal description of a TM that accepts *POWERS*

1. Sweep left to right across the tape, crossing off every other 0.
   - If the tape contained no 0s, REJECT.
   - If the tape contained a single 0, ACCEPT.

Recall the language $POWERS = \{0^{2^n} \mid n \geq 0\}$ is not context-free.

Informal description of a TM that accepts *POWERS*

1. Sweep left to right across the tape, crossing off every other 0.
   - If the tape contained no 0s, REJECT.
   - If the tape contained a single 0, ACCEPT.
   - If the tape contained more than one 0 and the number of 0s was odd, REJECT.

Recall the language $POWERS = \{0^{2^n} \mid n \geq 0\}$ is not context-free.

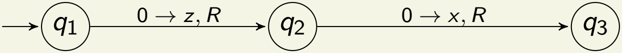
Informal description of a TM that accepts *POWERS*

1. Sweep left to right across the tape, crossing off every other 0.
   - If the tape contained no 0s, REJECT.
   - If the tape contained a single 0, ACCEPT.
   - If the tape contained more than one 0 and the number of 0s was odd, REJECT.
2. Return the head to the left-hand end of the tape.
3. Go to step ??.

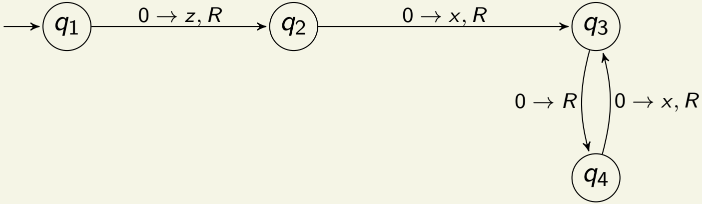
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$

$$\rightarrow q_1 \xrightarrow{0 \to z, R} q_2$$

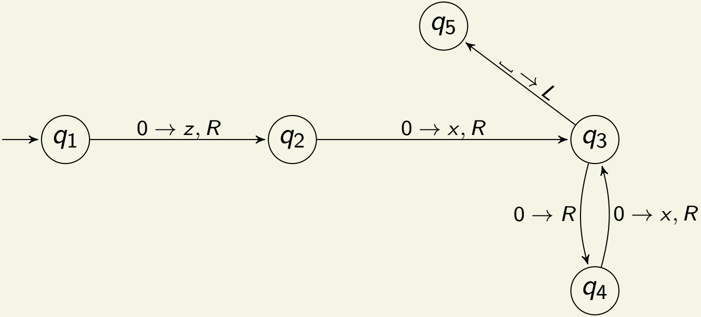
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



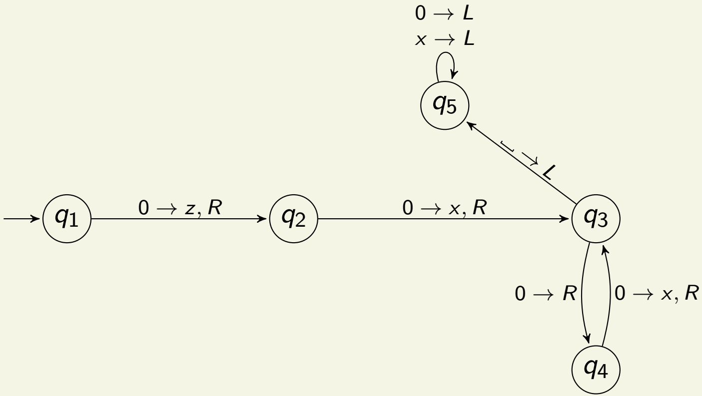
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



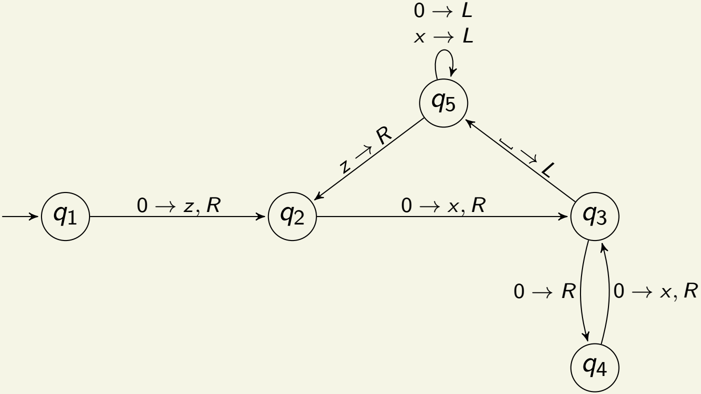
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



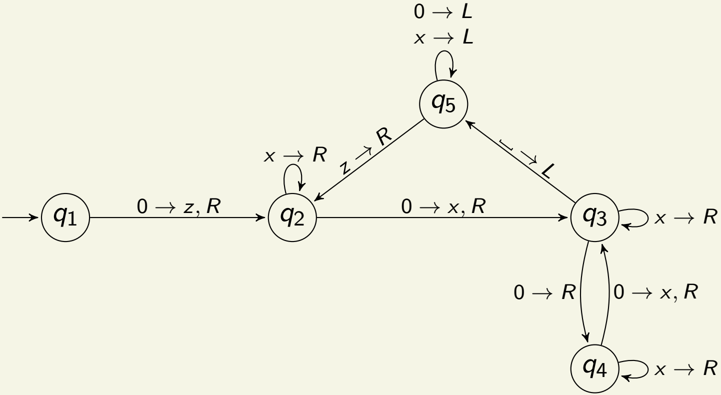
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



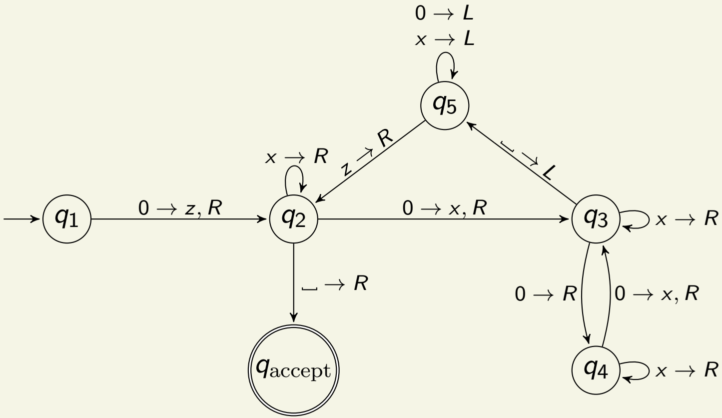
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



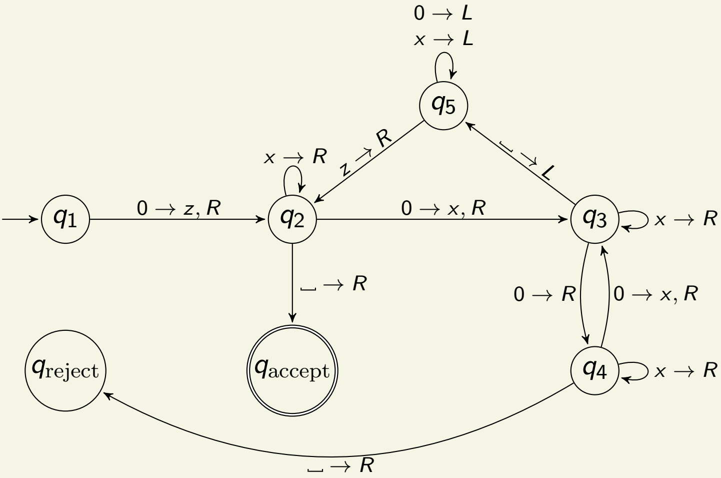
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



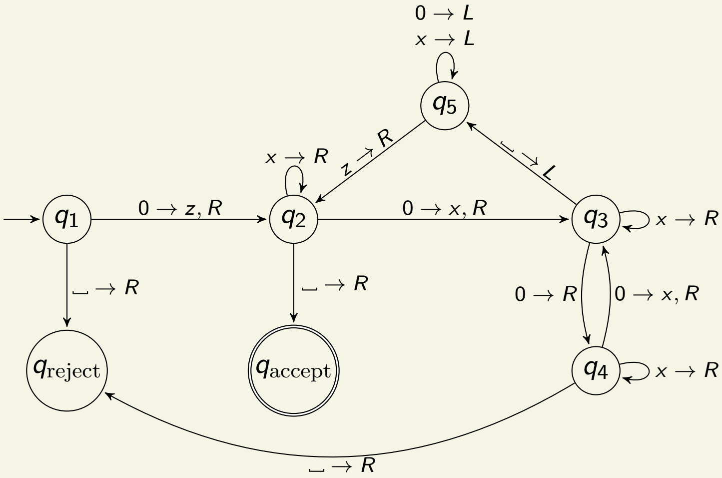
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



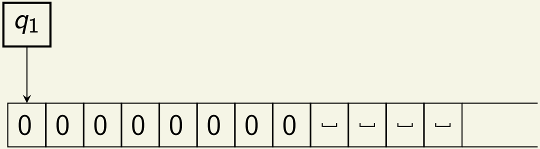
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



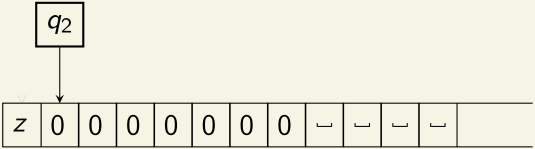
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$



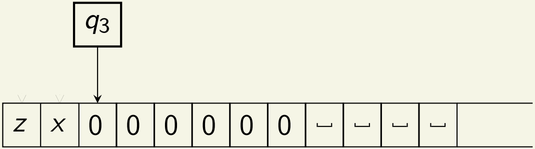
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

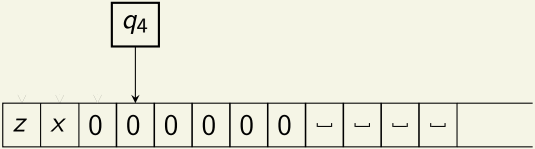


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



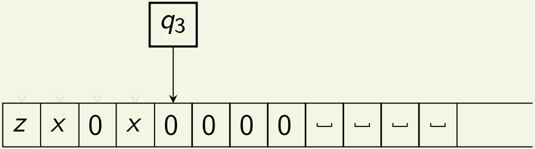
TM for *POWERS* $= \{0^{2^n} \mid n \geq 0\}$

input: $0^8 = 00000000$

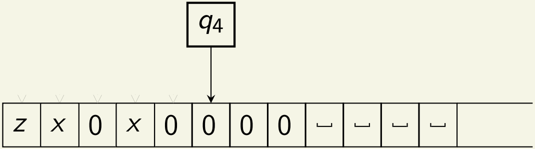


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$

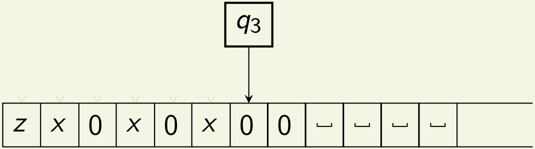
input: $0^8 = 00000000$



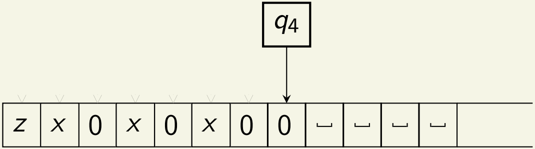
TM for $\mathit{POWERS} = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



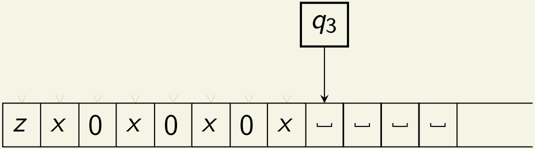
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



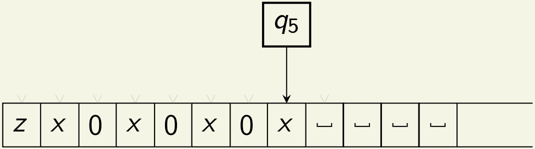
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



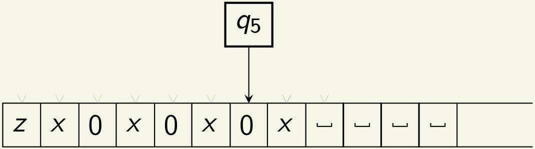
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



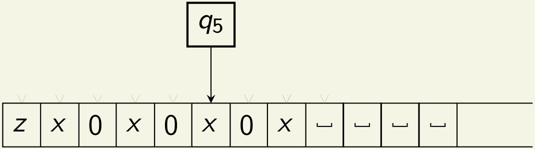
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



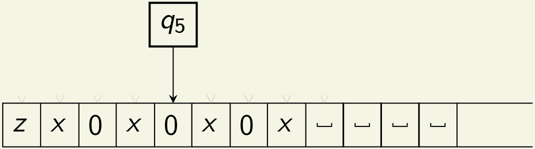
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



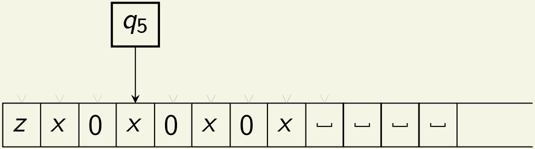
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



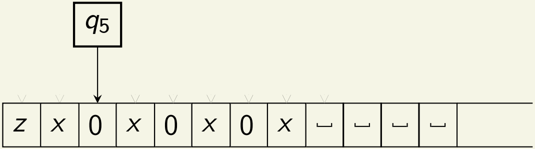
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



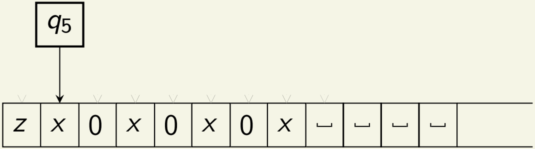
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



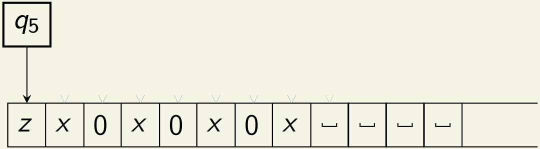
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



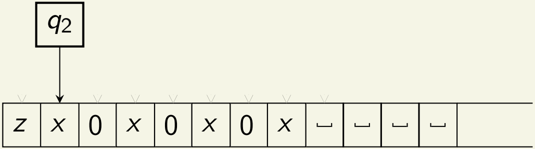
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



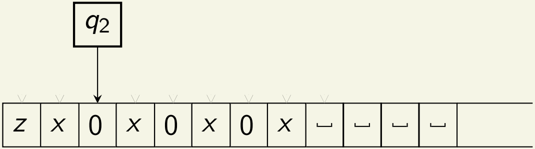
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



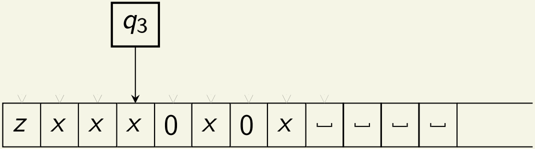
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



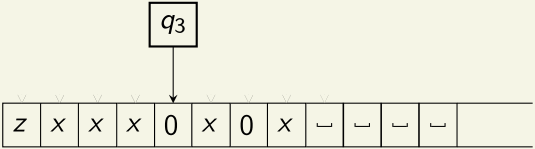
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



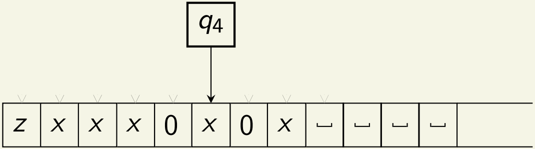
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



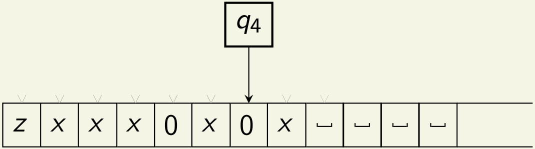
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

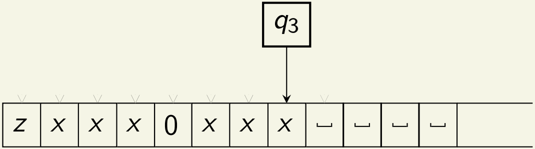


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

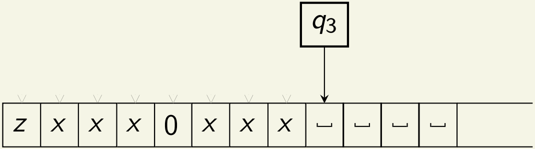


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$

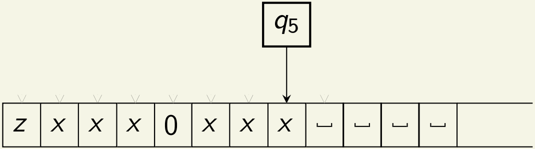
input: $0^8 = 00000000$



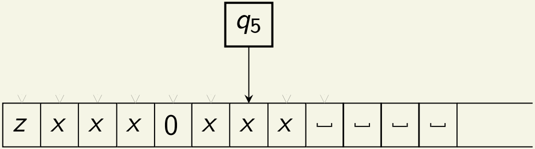
TM for *POWERS* $= \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



TM for $\mathit{POWERS} = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

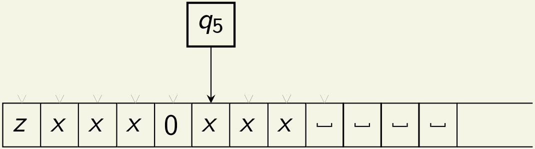


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

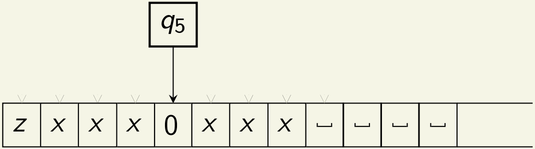


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$

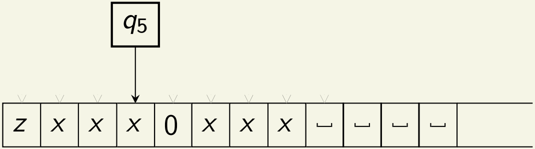
input: $0^8 = 00000000$



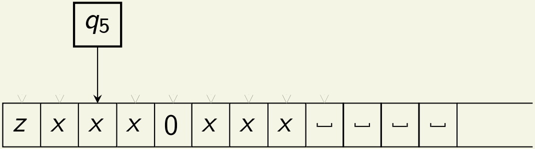
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



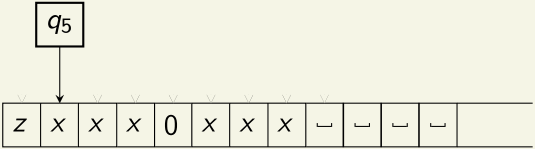
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



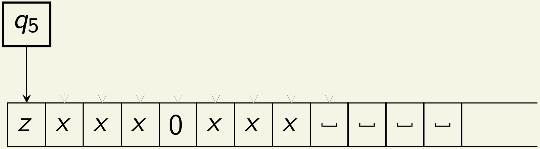
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



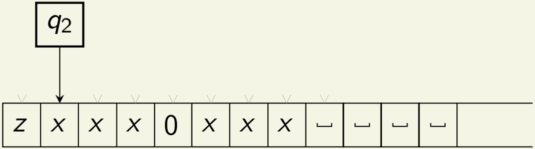
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



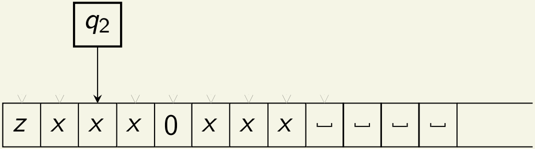
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



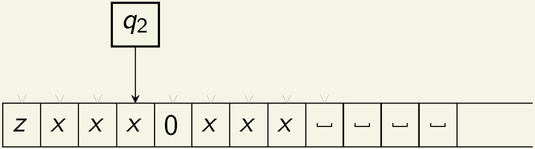
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



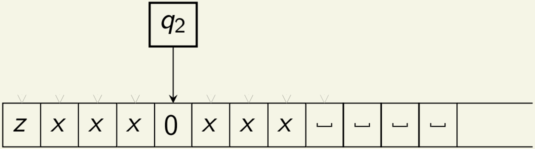
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



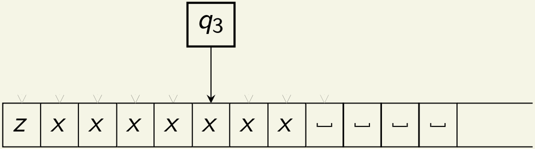
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



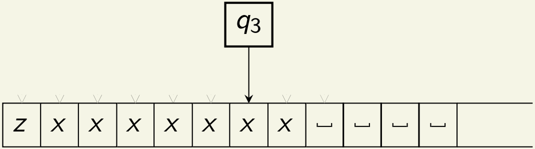
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



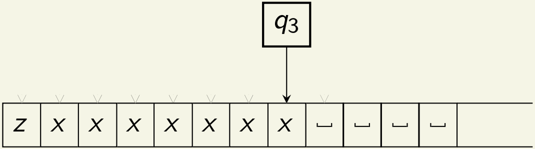
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



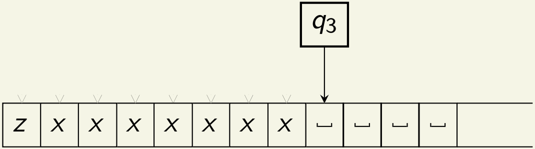
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



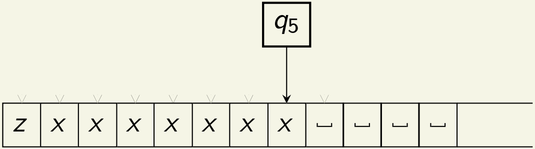
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



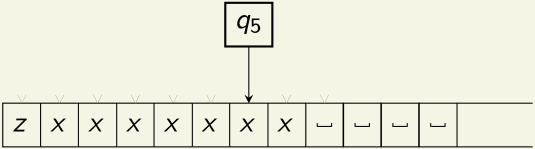
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



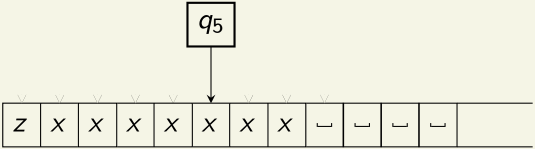
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



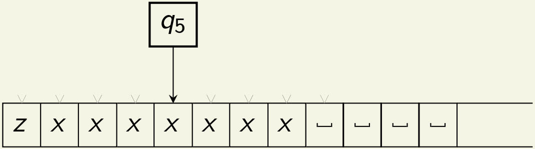
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



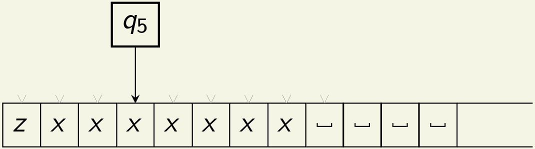
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

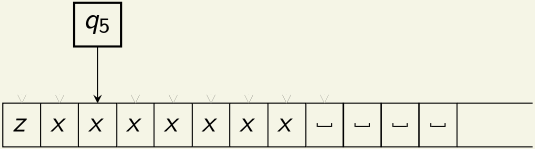


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

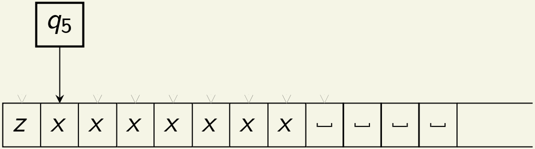


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$

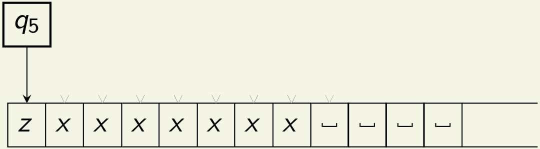
input: $0^8 = 00000000$



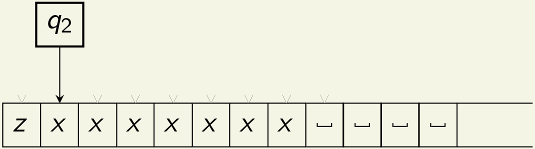
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



TM for *POWERS* $= \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

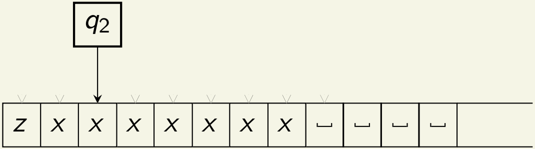


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

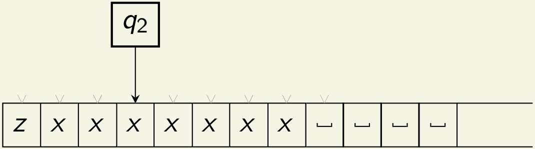


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$

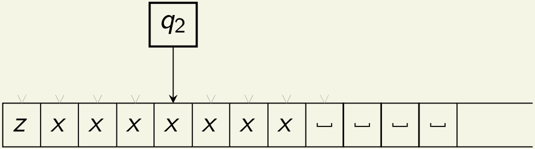
input: $0^8 = 00000000$



TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

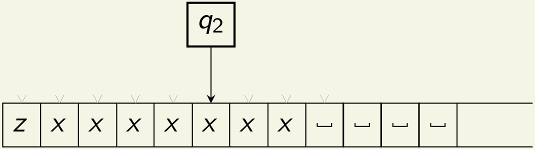


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$

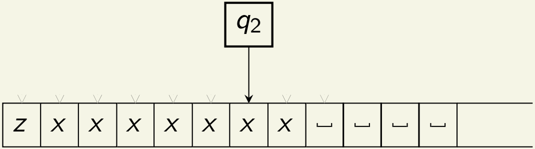


TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$

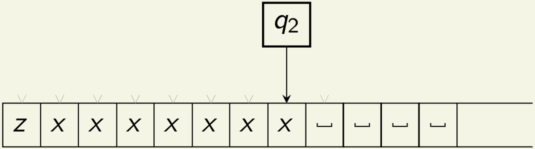
input: $0^8 = 00000000$



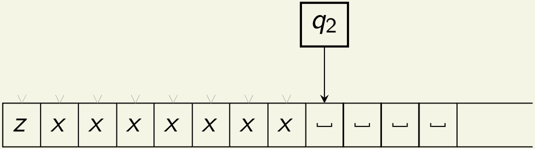
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



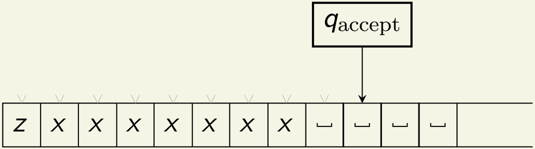
TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



TM for *POWERS* $= \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



TM for $POWERS = \{0^{2^n} \mid n \geq 0\}$
input: $0^8 = 00000000$



Recall the language $\{ww \mid w \in \{0,1\}^*\}$ is not context-free.
The language $\{w\#w \mid w \in \{0,1\}^*\}$ is also not context-free.

Informal description of a TM that accepts $\{w\#w \mid w \in \{0,1\}^*\}$

Recall the language $\{ww \mid w \in \{0,1\}^*\}$ is not context-free.
The language $\{w\#w \mid w \in \{0,1\}^*\}$ is also not context-free.

Informal description of a TM that accepts $\{w\#w \mid w \in \{0,1\}^*\}$

1. If the current symbol is $\#$, move to the right.
   While the current symbol is an x, scan to the right.
   If the current symbol is $\sqcup$, accept.

Recall the language $\{ww \mid w \in \{0,1\}^*\}$ is not context-free.
The language $\{w\#w \mid w \in \{0,1\}^*\}$ is also not context-free.

Informal description of a TM that accepts $\{w\#w \mid w \in \{0,1\}^*\}$

1. If the current symbol is $\#$, move to the right.
   While the current symbol is an x, scan to the right.
   If the current symbol is $\sqcup$, accept.

2. Remember the symbol in the current cell and replace it with an x.

Recall the language $\{ww \mid w \in \{0,1\}^*\}$ is not context-free.
The language $\{w\#w \mid w \in \{0,1\}^*\}$ is also not context-free.

Informal description of a TM that accepts $\{w\#w \mid w \in \{0,1\}^*\}$

1. If the current symbol is $\#$, move to the right.
   While the current symbol is an x, scan to the right.
   If the current symbol is $\sqcup$, accept.
2. Remember the symbol in the current cell and replace it with an x.
3. Scan to the right until the first 0 or 1 after the $\#$.

Recall the language $\{ww \mid w \in \{0,1\}^*\}$ is not context-free.
The language $\{w\#w \mid w \in \{0,1\}^*\}$ is also not context-free.

Informal description of a TM that accepts $\{w\#w \mid w \in \{0,1\}^*\}$

1. If the current symbol is $\#$, move to the right.
   While the current symbol is an x, scan to the right.
   If the current symbol is $\sqcup$, accept.

2. Remember the symbol in the current cell and replace it with an x.

3. Scan to the right until the first 0 or 1 after the $\#$.

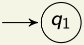
4. If the symbol in the current cell matches the remembered symbol, replace it with an x.
   If it does not match, REJECT.

Recall the language $\{ww \mid w \in \{0,1\}^*\}$ is not context-free.
The language $\{w\#w \mid w \in \{0,1\}^*\}$ is also not context-free.

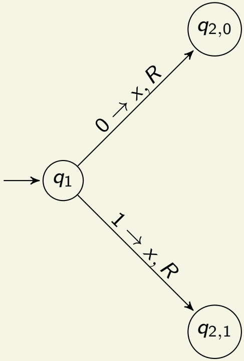
Informal description of a TM that accepts $\{w\#w \mid w \in \{0,1\}^*\}$

1. If the current symbol is $\#$, move to the right.
   While the current symbol is an x, scan to the right.
   If the current symbol is $\sqcup$, accept.
2. Remember the symbol in the current cell and replace it with an x.
3. Scan to the right until the first 0 or 1 after the $\#$.
4. If the symbol in the current cell matches the remembered symbol, replace it with an x.
   If it does not match, REJECT.
5. Scan to the left until the first x to the left of the $\#$.
   Move back to the right one cell. Go to step ??.

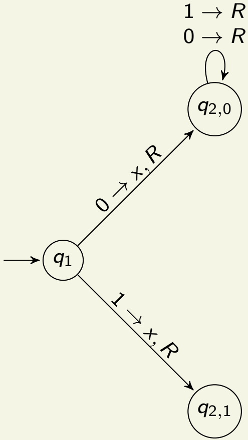
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



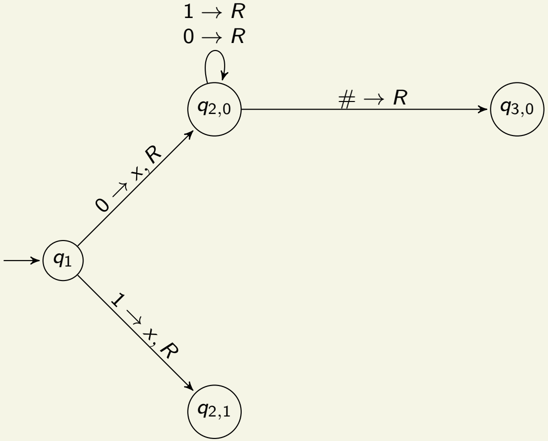
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



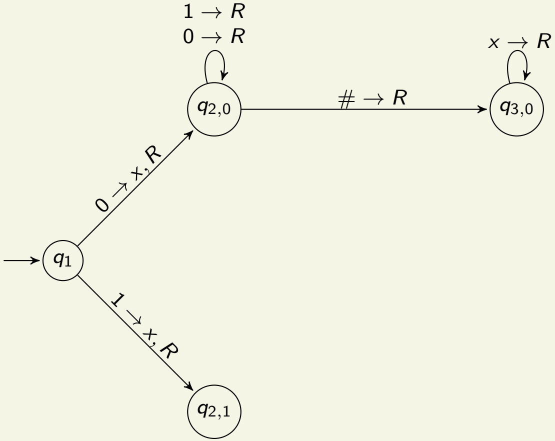
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



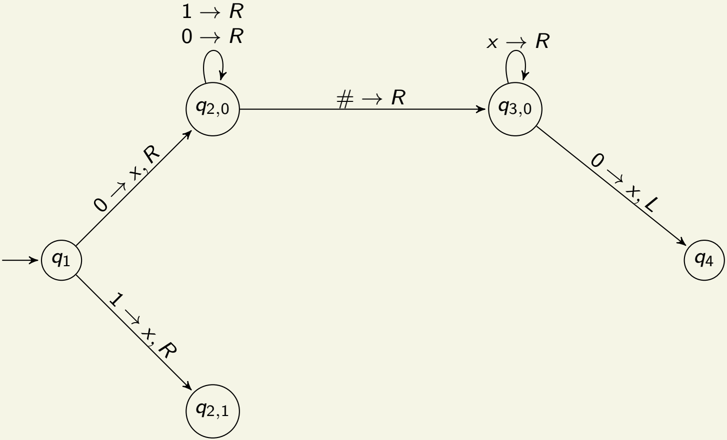
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



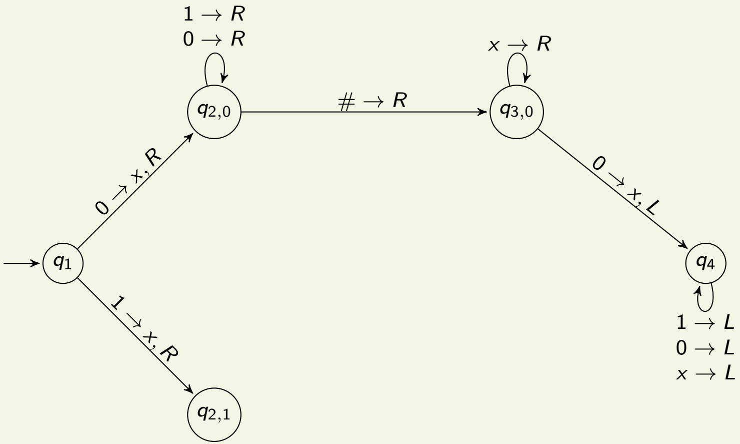
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



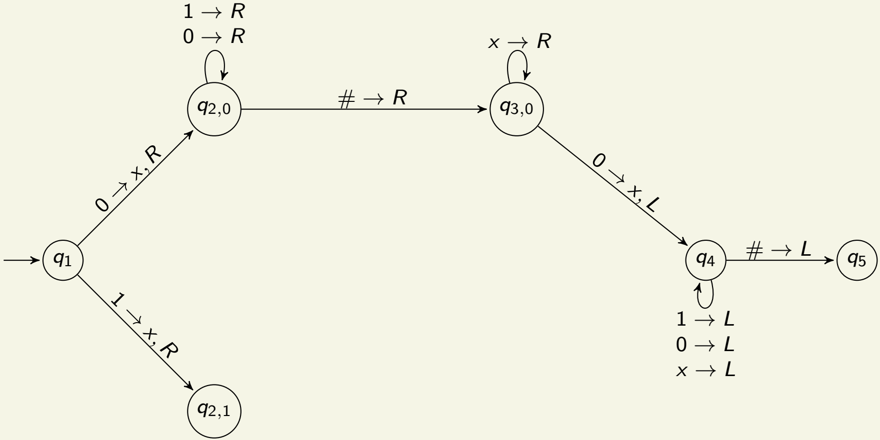
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



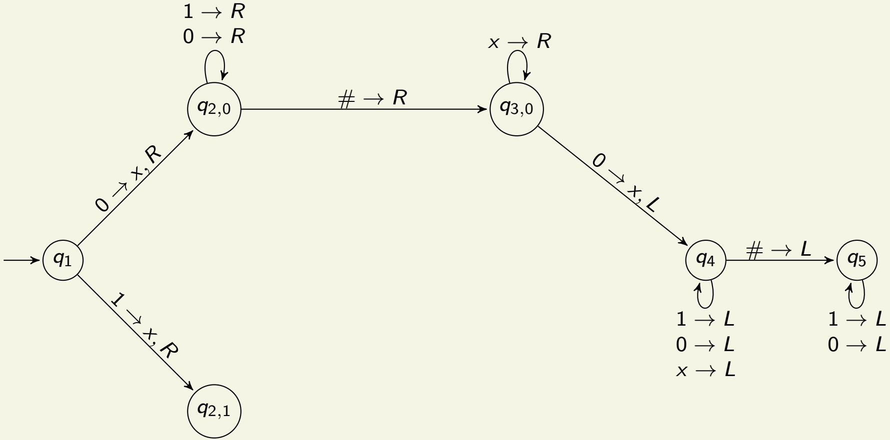
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



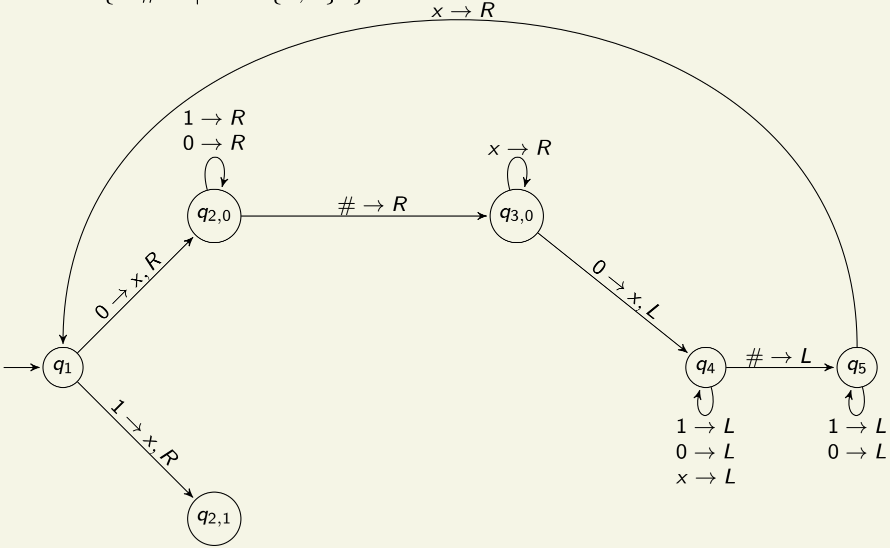
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



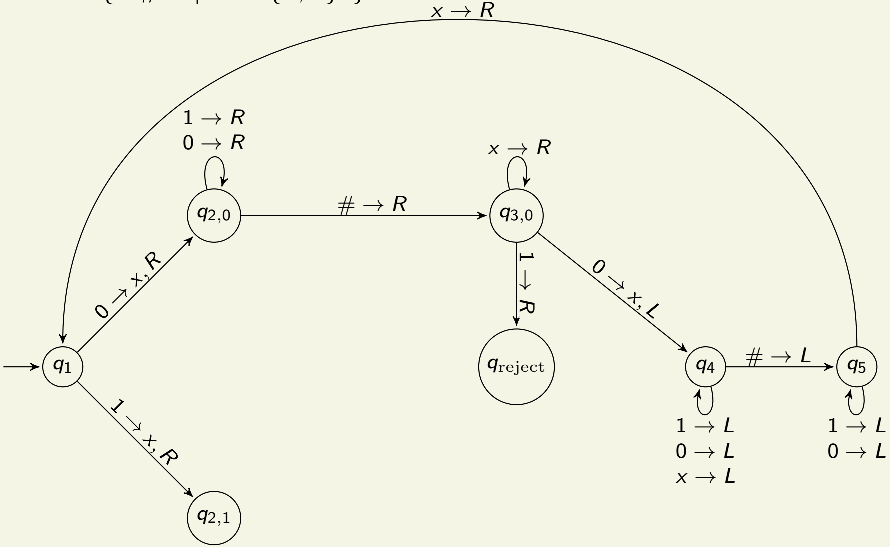
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



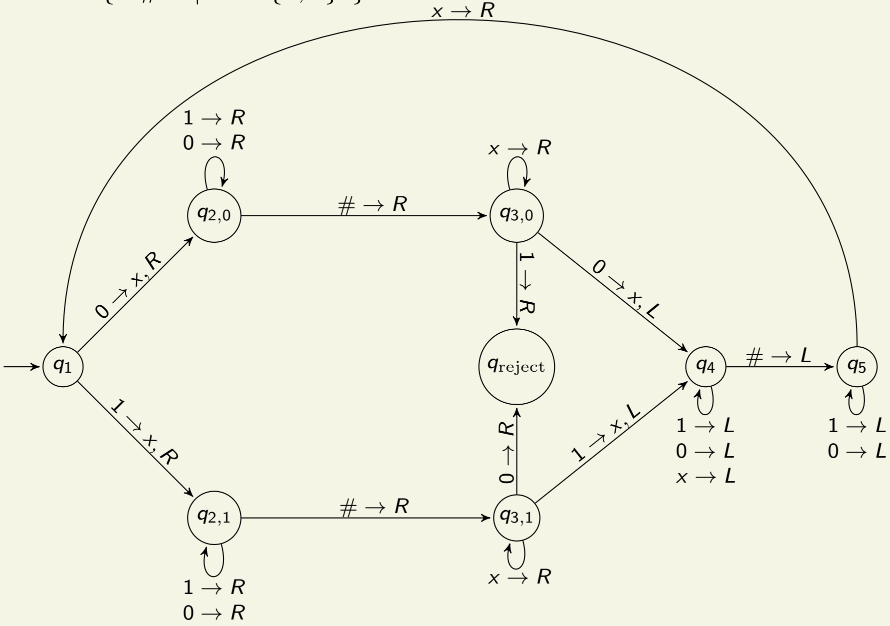
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



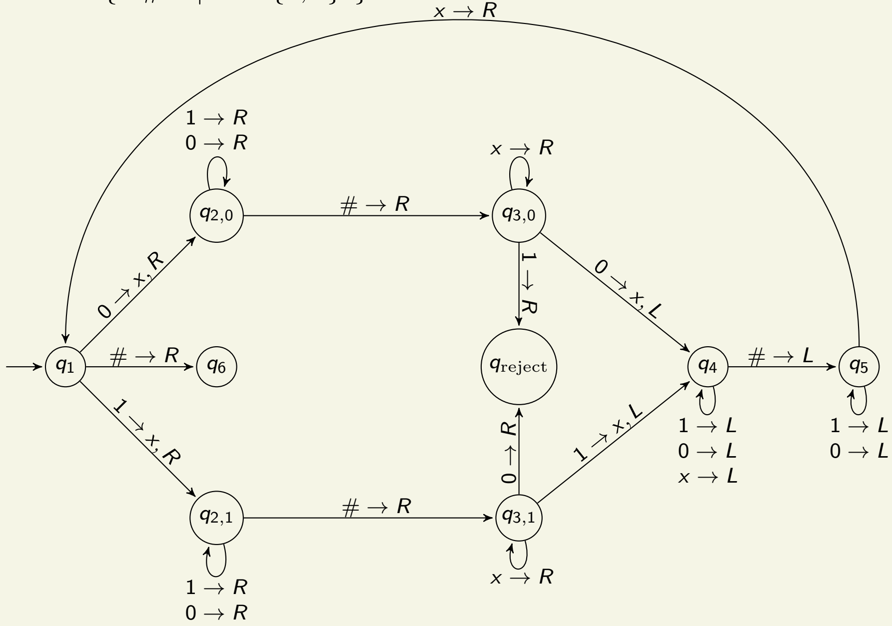
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



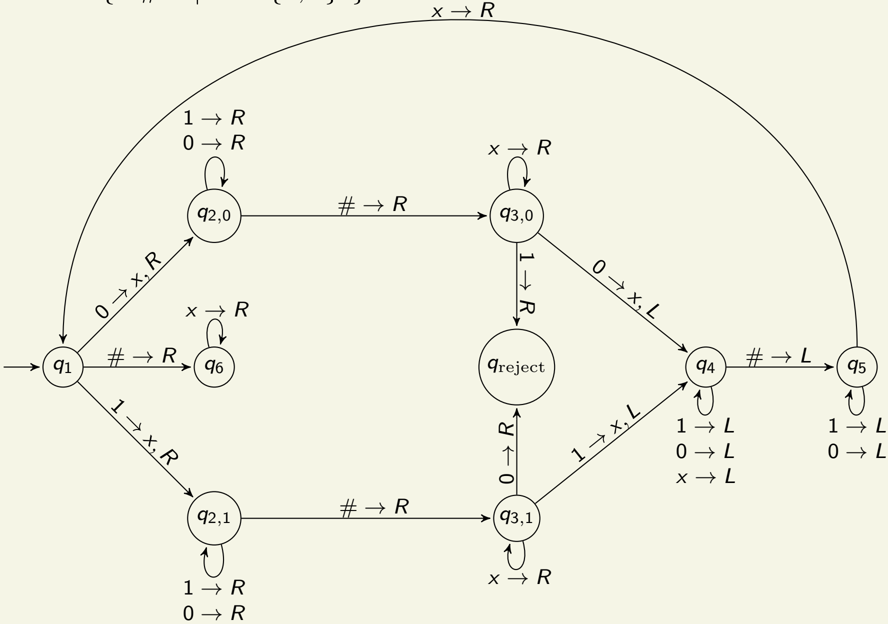
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



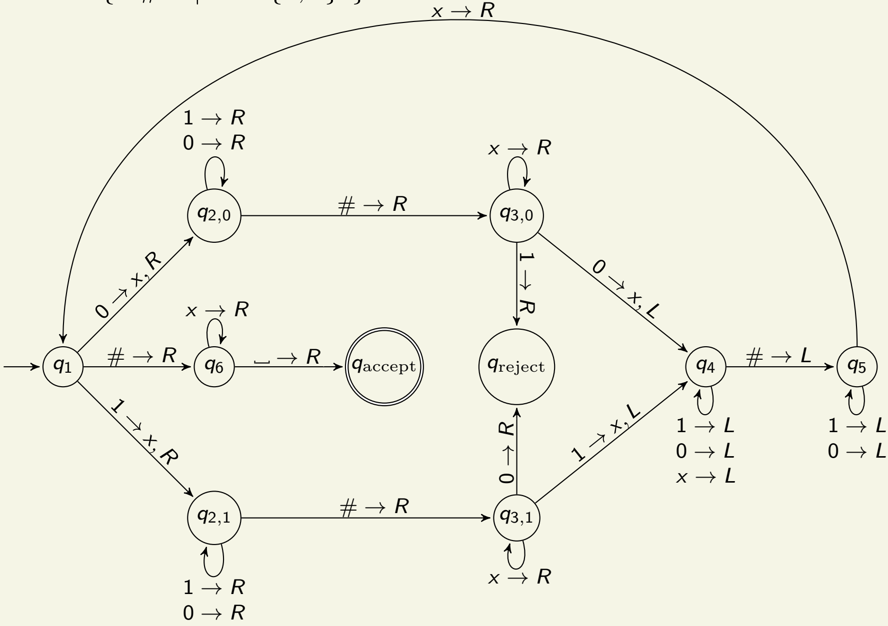
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



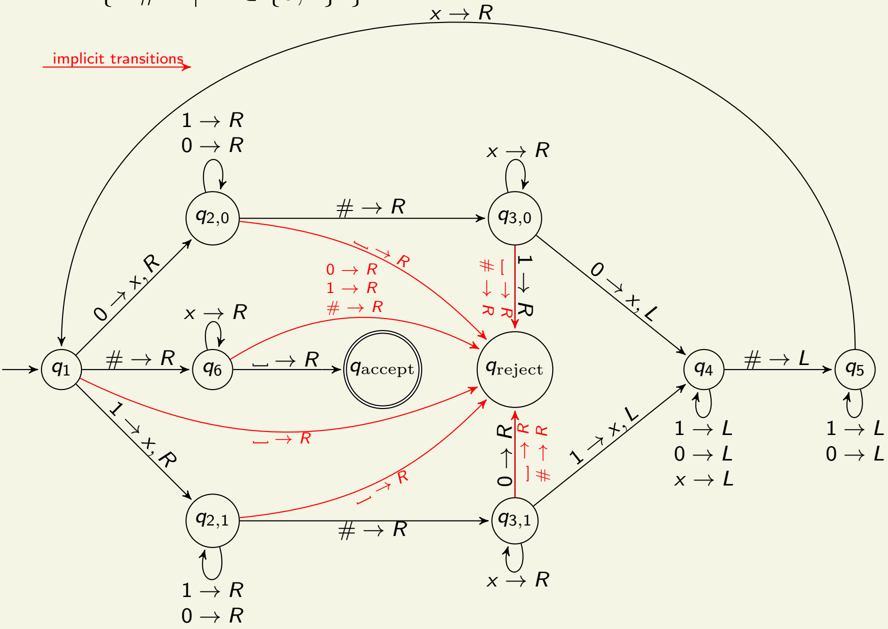
TM for $\{w\#w \mid w \in \{0,1\}^*\}$



TM for $\{w\#w \mid w \in \{0,1\}^*\}$



TM for $\{w\#w \mid w \in \{0,1\}^*\}$



Computation of a TM

Initially the input $w \in \Sigma^*$ is written on the tape, one symbol per tape cell, followed by blanks. The tape head points to the first symbol of w and the TM is in its initial state.

The computation proceeds according to the transition function. If the TM ever tries to move its head off the left end of the tape, the tape head stays in place for that step.

The computation continues until the TM enters the accept state or the reject state, at which point it halts. Otherwise, the TM goes on forever.

A *configuration* of a TM consists of the current state, the current tape contents, and the current tape head location. A configuration is often represented in the form

$$uqv$$

where $u, v \in \Gamma^*$ and $q \in Q$, meaning that

- the current state is q
- uv is the tape contents
- the tape head is pointing to the first symbol of v

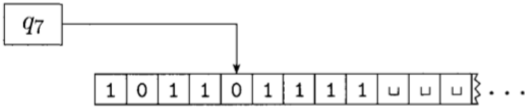


FIGURE 3.4
A Turing machine with configuration $1011q_701111$

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{accept}}, q_{\mathrm{reject}})$ be a TM.

- The *start configuration* of M on input $w \in \Sigma^*$ is the configuration $q_0 w$.

- In an *accepting configuration* the state is q_{accept}.

- In a *rejecting configuration* the state is q_{reject}.

- Accepting and rejecting configurations are called *halting configurations*.

We say that M *accepts* w if there exist a sequence of configurations $C_1, \ldots, C_k$ where

1. C_1 is the initial configuration of M on w.
2. each C_i yields C_{i+1} via the transition function δ in one step.
3. C_k is an accepting configuration.

We say that M *accepts* w if there exist a sequence of configurations $C_1, \ldots, C_k$ where

1. C_1 is the initial configuration of M on w.
2. each C_i yields C_{i+1} via the transition function δ in one step.
3. C_k is an accepting configuration.

The *language of M* is

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}.$$

We say that $L(M)$ is the language *recognized* by M.

Definition

A language is *Turing-recognizable* if some Turing machine recognizes it. That is, $B \subseteq \Sigma^*$ is Turing-recognizable if $B = L(M)$ for some Turing machine M.

Definition

A language is *Turing-recognizable* if some Turing machine recognizes it. That is, $B \subseteq \Sigma^*$ is Turing-recognizable if $B = L(M)$ for some Turing machine M.

Note: outside of our textbook and this course, the terms *recursively enumerable (r.e.)* and *computably enumerable (c.e.)* are typically used for this concept.

On an input w, a TM M has three possible outcomes:

- accept
- reject
- never halt

On an input w, a TM M has three possible outcomes:

- accept
- reject
- never halt

A Turing machine that always halts (i.e. accepts or rejects every input) is called a *decider*. A decider that recognizes some language is said to *decide* that language.

On an input w, a TM M has three possible outcomes:

- accept
- reject
- never halt

A Turing machine that always halts (i.e. accepts or rejects every input) is called a *decider*. A decider that recognizes some language is said to *decide* that language.

Definition

A language is *Turing-decidable* (or simply *decidable*) if some Turing machine decides it. That is, $B \subseteq \Sigma^*$ is decidable if $B = L(M)$ for some always-halting Turing machine M (a decider M).

On an input w, a TM M has three possible outcomes:

- accept
- reject
- never halt

A Turing machine that always halts (i.e. accepts or rejects every input) is called a *decider*. A decider that recognizes some language is said to *decide* that language.

Definition

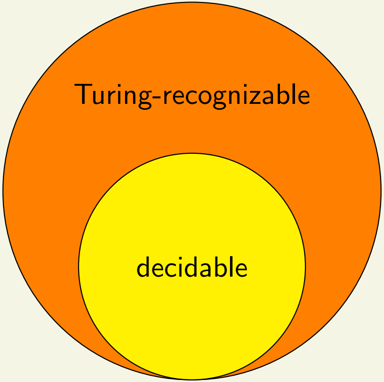
A language is *Turing-decidable* (or simply *decidable*) if some Turing machine decides it. That is, $B \subseteq \Sigma^*$ is decidable if $B = L(M)$ for some always-halting Turing machine M (a decider M).

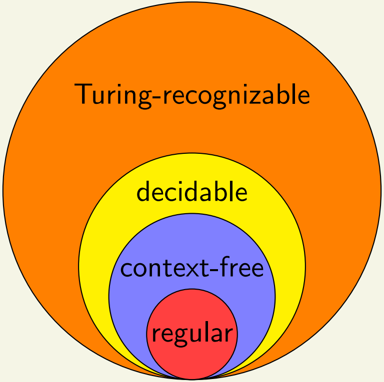
Note: the term *recursive* is also used for this concept.

Proposition

Every decidable language is also Turing-recognizable.

This is simply because every always-halting Turing machine is a Turing machine.




Turing-recognizable
decidable
context-free
regular

Describing Turing machines.

- *formal description*: low-level programming, complete detail, state-transition diagram
- *implementation description*: in English, how the TM moves its head and uses its tapes to store data
- *high-level description*: description of algorithm, ignoring model

Other Turing Machine Models

1. Multitape Turing Machines
2. Nondeterministic Turing Machines
3. Enumerators

Multitape Turing Machines

Multitape Turing Machines

- k tapes, for some $k \geq 2$
- transition function:

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k$$

If $\delta(q, a_1, \ldots, a_k) = (p, b_1, \ldots, b_k, D_1, \ldots, D_k)$, then when the TM is in state a and the k tape heads are reading $a_1, \ldots, a_k$, the symbols $b_1, \ldots, b_k$ are written on each of the tapes and the heads are moved in directions $D_1, \ldots, D_k$.

Multitape vs Single-Tape

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

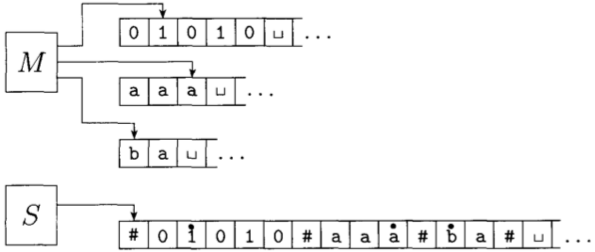


FIGURE 3.14
Representing three tapes with one

Nondeterministic Turing Machines

Nondeterministic Turing Machines (NTMs)

- transition function

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

The computation of an NTM is a tree of configurations. If some branch leads to an accepting configuration, then the input is accepted.

NTMs vs TMs

Theorem

Every NTM has an equivalent deterministic TM.

NTMs vs TMs

Theorem

Every NTM has an equivalent deterministic TM.

Proof. Let N be an NTM. The idea is to do a breadth-first search on N's computation tree. We will design a 3-tape deterministic TM D for this.

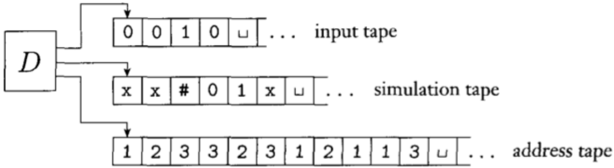


FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

- input tape: contains the input; is never altered

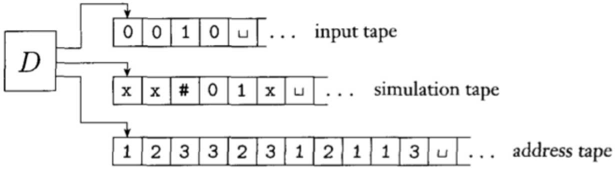


FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

- input tape: contains the input; is never altered
- simulation tape: copy of N's tape on some branch of its computation tree

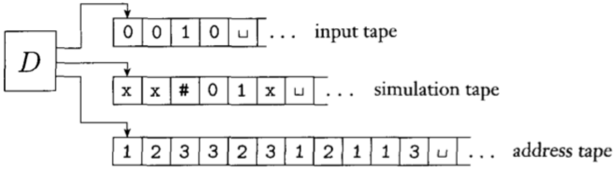


FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

- input tape: contains the input; is never altered
- simulation tape: copy of N's tape on some branch of its computation tree
- address tape: used to keep track of D's location in N's computation tree

- On tape 3 (the address tape), we use the alphabet $\Sigma_b = \{1, \ldots, b\}$, where b is the size of the largest set of possible choices in N's transition function.

- Every node in N's computation tree is given an address that is a string in Σ_b^*.

- For example, address 231 identifies the node we get by taking the 2^{nd} nondeterministic choice, then the 3^{rd} choice, and lastly the 1^{st} choice.

- Some addresses may not correspond to nodes.

Description of deterministic TM D to simulate N

1. Initially, tape 1 contains the input w. Tapes 2 and 3 are empty.

Description of deterministic TM D to simulate N

1. Initially, tape 1 contains the input w. Tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.

Description of deterministic TM D to simulate N

1. Initially, tape 1 contains the input w. Tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
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 nondeterministic choice to make.

Description of deterministic TM D to simulate N

1. Initially, tape 1 contains the input w. Tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
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 nondeterministic choice to make.
    - If no more symbols remain on tape 3 or this is not a valid choice, abort this branch and go to step 4.

Description of deterministic TM D to simulate N

1. Initially, tape 1 contains the input w. Tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
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 nondeterministic choice to make.
    - If no more symbols remain on tape 3 or this is not a valid choice, abort this branch and go to step 4.
    - Also go to step 4 if this choice leads to a rejecting configuration.

Description of deterministic TM D to simulate N

1. Initially, tape 1 contains the input w. Tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
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 nondeterministic choice to make.
    - If no more symbols remain on tape 3 or this is not a valid choice, abort this branch and go to step 4.
    - Also go to step 4 if this choice leads to a rejecting configuration.
    - If this choice leads to an accepting configuration, ACCEPT

Description of deterministic TM D to simulate N

1. Initially, tape 1 contains the input w. Tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
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 nondeterministic choice to make.
   - If no more symbols remain on tape 3 or this is not a valid choice, abort this branch and go to step 4.
   - Also go to step 4 if this choice leads to a rejecting configuration.
   - If this choice leads to an accepting configuration, ACCEPT
4. Replace the string on tape 3 with lexicographically next string. Go to step 2.

Suppose N accepts w. Then there is some sequence of nondeterministic choices that leads N to an accepting state. Consider the lexicographically least sequence of such choices and let $a \in \Sigma_b^*$ be its address. Eventually a will be on tape 3 and D will accept.

Suppose N accepts w. Then there is some sequence of nondeterministic choices that leads N to an accepting state. Consider the lexicographically least sequence of such choices and let $a \in \Sigma_b^*$ be its address. Eventually a will be on tape 3 and D will accept.

Now suppose that N does not accept w. Then no sequence of nondeterministic choices leads to an accepting configuration. This means D will never accept (in fact, D will run forever).

Suppose N accepts w. Then there is some sequence of nondeterministic choices that leads N to an accepting state. Consider the lexicographically least sequence of such choices and let $a \in \Sigma_b^*$ be its address. Eventually a will be on tape 3 and D will accept.

Now suppose that N does not accept w. Then no sequence of nondeterministic choices leads to an accepting configuration. This means D will never accept (in fact, D will run forever).

Therefore $L(D) = L(N)$. $\square$

Theorem

Every NTM has an equivalent deterministic TM.

Corollary. A language is Turing-recognizable if only if some NTM recognizes it.

Enumerators

Three names for the same concept:

$$\begin{aligned} \text{Turing-recognizable} &\equiv \text{recursively enumerable} \\ &\equiv \text{computably enumerable} \end{aligned}$$

What is meant by enumerable?

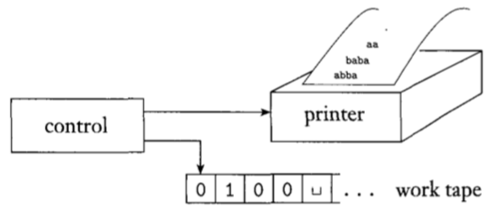
Enumerators

Three names for the same concept:

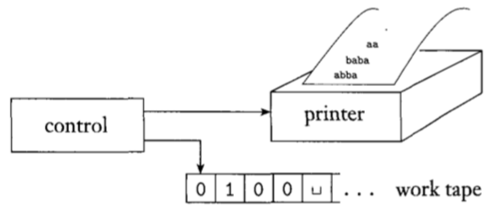
$$\begin{aligned} \text{Turing-recognizable} &\equiv \text{recursively enumerable} \\ &\equiv \text{computably enumerable} \end{aligned}$$

What is meant by enumerable?

An *enumerator* is a TM with a second output tape, which we may conceptualize as a printer.



Enumerators



An enumerator E starts out with an empty tape. As it computes, it may print a string from time to time. If it runs forever, it may print an infinite list of strings.

The *language enumerated by E* is the collection of strings that it prints. The strings may be printed in any order, possibly with repetitions.

Recognition vs Enumeration

Theorem

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

Recognition vs Enumeration

Theorem

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

Proof. ($\Leftarrow$) Let A be a language. Suppose E is an enumerator that enumerates A. We use E to design a TM M:

Recognition vs Enumeration

Theorem

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

Proof. ($\Leftarrow$) Let A be a language. Suppose E is an enumerator that enumerates A. We use E to design a TM M:

> M: On input w:
> Run E. Every time E outputs a string, compare it with w.
> If w ever appears in the output of E, accept.

The M recognizes A, so A is Turing-recognizable.

($\Rightarrow$) Suppose we have a TM M that recognizes A. We use a technique called *dovetailing* to design an enumerator for A. Let $s_1, s_2, s_3, \ldots$ be a list of all strings in Σ^* in lexicographic order.

($\Rightarrow$) Suppose we have a TM M that recognizes A. We use a technique called *dovetailing* to design an enumerator for A. Let $s_1, s_2, s_3, \ldots$ be a list of all strings in Σ^* in lexicographic order.

E: If any computations accept, print out the corresponding s_j's.

($\Rightarrow$) Suppose we have a TM M that recognizes A. We use a technique called *dovetailing* to design an enumerator for A. Let $s_1, s_2, s_3, \ldots$ be a list of all strings in Σ^* in lexicographic order.

E: If any computations accept, print out the corresponding s_j's.

If M accepts some string $x = s_j$ in t computation steps, then it will appear in E's output on iteration $\max(j, t)$ of the for loop. Also, E only outputs strings that M accepts. Therefore E enumerates A. $\square$

Quantifiers, Predicatesm, and Recognizability

Another way to define Turing-recognizability is in terms of existential quantifiers and decidable predicates.

Theorem

A language A is Turing-recognizable if and only if there is a decidable language D such that for all $w \in \Sigma^*$,

$$x \in A \Leftrightarrow (\exists w \in \Sigma^*)\ \langle x, w \rangle \in D.$$

- A string w with $\langle x, w \rangle \in D$ is a *witness* that $x \in A$.
- We will use quantifiers, predicates, and witnesses again later when we study the complexity class NP.

Proof. ($\Leftarrow$) We are given a decidable language D such that for all $w \in \Sigma^*$,

$$x \in A \Leftrightarrow (\exists w \in \Sigma^*)\ \langle x, w \rangle \in D.$$

Proof. $(\Leftarrow)$ We are given a decidable language D such that for all $w \in \Sigma^*$,

$$x \in A \Leftrightarrow (\exists w \in \Sigma^*)\ \langle x, w \rangle \in D.$$

Let M be a decider for D. Let $s_1, s_2, \ldots$ be a list of all strings in Σ^* in lexicographic order.

Proof. ($\Leftarrow$) We are given a decidable language D such that for all $w \in \Sigma^*$,

$$x \in A \Leftrightarrow (\exists w \in \Sigma^*)\ \langle x, w \rangle \in D.$$

Let M be a decider for D. Let $s_1, s_2, \ldots$ be a list of all strings in Σ^* in lexicographic order.

Consider the following algorithm N.

N: On input x:
    For $i = 1, 2, 3, \ldots$
        For $j = 1$ to i
            Run M on input $\langle x, s_j \rangle$ for i steps
            If M accepts, accept

Proof. ($\Leftarrow$) We are given a decidable language D such that for all $w \in \Sigma^*$,

$$x \in A \Leftrightarrow (\exists w \in \Sigma^*)\ \langle x, w \rangle \in D.$$

Let M be a decider for D. Let $s_1, s_2, \ldots$ be a list of all strings in Σ^* in lexicographic order.

Consider the following algorithm N.

```
N: On input x:
    For i = 1, 2, 3, ...
        For j = 1 to i
            Run M on input ⟨x, s_j⟩ for i steps
            If M accepts, accept
```

- If M accepts $\langle x, w \rangle$ for some w, then it accepts in some number of steps, so N will accept x.

Proof. ($\Leftarrow$) We are given a decidable language D such that for all $w \in \Sigma^*$,

$$x \in A \Leftrightarrow (\exists w \in \Sigma^*)\ \langle x, w \rangle \in D.$$

Let M be a decider for D. Let $s_1, s_2, \ldots$ be a list of all strings in Σ^* in lexicographic order.

Consider the following algorithm N.

> N: On input x:
>   For $i = 1, 2, 3, \ldots$
>   For $j = 1$ to i
>   Run M on input $\langle x, s_j \rangle$ for i steps
>   If M accepts, accept

- If M accepts $\langle x, w \rangle$ for some w, then it accepts in some number of steps, so N will accept x.
- If M does not accept $\langle x, w \rangle$ for any w, then N does not accept x.

Proof. ($\Leftarrow$) We are given a decidable language D such that for all $w \in \Sigma^*$,

$$x \in A \Leftrightarrow (\exists w \in \Sigma^*)\ \langle x, w\rangle \in D.$$

Let M be a decider for D. Let $s_1, s_2, \ldots$ be a list of all strings in Σ^* in lexicographic order.

Consider the following algorithm N.

```
N: On input x:
    For i = 1, 2, 3, ...
        For j = 1 to i
            Run M on input ⟨x, s_j⟩ for i steps
            If M accepts, accept
```

- If M accepts $\langle x, w\rangle$ for some w, then it accepts in some number of steps, so N will accept x.
- If M does not accept $\langle x, w\rangle$ for any w, then N does not accept x.

Thus N recognizes A, so A is Turing-recognizable.

($\Rightarrow$) Assume that A is Turing-recognizable and let M be a TM that recognizes A.

($\Rightarrow$) Assume that A is Turing-recognizable and let M be a TM that recognizes A.
Define

$$D = \{\langle x, t\rangle \mid M \text{ accepts } x \text{ in at most } t \text{ steps}\}.$$

Then

$$\begin{aligned} x \in A &\iff x \in L(M) \\ &\iff M \text{ accepts } x \end{aligned}$$

($\Rightarrow$) Assume that A is Turing-recognizable and let M be a TM that recognizes A.
Define

$$D = \{\langle x, t\rangle \mid M \text{ accepts } x \text{ in at most } t \text{ steps}\}.$$

Then

$$\begin{aligned} x \in A &\iff x \in L(M) \\ &\iff M \text{ accepts } x \\ &\iff (\exists t)\ M \text{ accepts } x \text{ in at most } t \text{ steps} \end{aligned}$$

($\Rightarrow$) Assume that A is Turing-recognizable and let M be a TM that recognizes A.
Define

$$D = \{\langle x, t\rangle \mid M \text{ accepts } x \text{ in at most } t \text{ steps}\}.$$

Then

$$\begin{aligned}
x \in A &\iff x \in L(M) \\
&\iff M \text{ accepts } x \\
&\iff (\exists t)\ M \text{ accepts } x \text{ in at most } t \text{ steps} \\
&\iff (\exists t)\ \langle x, t\rangle \in D.
\end{aligned}$$

$\square$

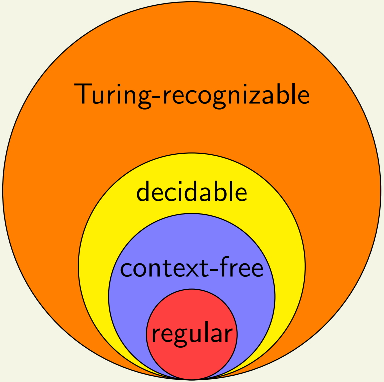
Summary: Turing-recognizability

The following names for Turing-recognizability are equivalent:

- A is Turing-recognizable
- A is computably enumerable (c.e.)
- A is recursively enumerable (r.e.)

The following definitions for Turing-recognizability are equivalent:

- A is recognized by a TM
- A is recognized by a multitape TM
- A is recognized by an NTM
- A is enumerated by an enumerator
- A is definable with an existential quantifer and a decidable predicate


Turing-recognizable
decidable
context-free
regular