General Computing Machines

- Machines so far: DFA, NFA, PDA.
General Computing Machines

- Machines so far: DFA, NFA, PDA.
  - DFA, NFA: Have finite memory.
  - PDA: infinite memory. But they access the memory in a limited way, only via push or pop.
General Computing Machines

- Machines so far: DFA, NFA, PDA.
  - DFA, NFA: Have finite memory.
  - PDA: infinite memory. But they access the memory in a limited way, only via push or pop.
- The complete machine?
General Computing Machines

- Machines so far: DFA, NFA, PDA.
  - DFA, NFA: Have finite memory.
  - PDA: infinite memory. But they access the memory in a limited way, only via push or pop.
- The complete machine?
  - Infinite memory that allows ‘full’ access?
General Computing Machines

- Machines so far: DFA, NFA, PDA.
  - DFA, NFA: Have finite memory.
  - PDA: infinite memory. But they access the memory in a limited way, only via push or pop.

- The complete machine?
  - Infinite memory that allows ‘full’ access? And maybe other ways to use computational resources that we haven’t thought of...
General Computing Machines

- Machines so far: DFA, NFA, PDA.
  - DFA, NFA: Have finite memory.
  - PDA: infinite memory. But they access the memory in a limited way, only via push or pop.

- The complete machine?
  - Infinite memory that allows ‘full’ access? And maybe other ways to use computational resources that we haven’t thought of...

- No limitation on what they can compute?
General Computing Machines

- Machines so far: DFA, NFA, PDA.
  - DFA, NFA: Have finite memory.
  - PDA: infinite memory. But they access the memory in a limited way, only via push or pop.

- The complete machine?
  - Infinite memory that allows ‘full’ access? And maybe other ways to use computational resources that we haven’t thought of...

- No limitation on what they can compute?
  - No! There are far too many languages over \( \{0, 1\} \) than there are “machines” or programs.
General Computing Machines

- Machines so far: DFA, NFA, PDA.
  - DFA, NFA: Have finite memory.
  - PDA: infinite memory. But they access the memory in a limited way, only via push or pop.

- The complete machine?
  - Infinite memory that allows ‘full’ access? And maybe other ways to use computational resources that we haven’t thought of...

- No limitation on what they can compute?
  - No! There are far too many languages over \( \{0, 1\} \) than there are “machines” or programs (as long as machines can be represented digitally)
General Computing Machines

▶ Machines so far: DFA, NFA, PDA.
  ▶ DFA, NFA: Have finite memory.
  ▶ PDA: infinite memory. But they access the memory in a limited way, only via push or pop.
▶ The complete machine?
  ▶ Infinite memory that allows ‘full’ access? And maybe other ways to use computational resources that we haven’t thought of...
    ▶ Come up with a model that describes all “conceivable” computation
▶ No limitation on what they can compute?
  ▶ No! There are far too many languages over \{0, 1\} than there are “machines” or programs (as long as machines can be represented digitally)
General Computing Machines
Alonzo Church, Emil Post, and Alan Turing (1936)

- Church (\(\lambda\)-calculus), Post (Post’s machine), Turing (Turing machine) independently came up with formal definitions of mechanical computation
General Computing Machines
Alonzo Church, Emil Post, and Alan Turing (1936)

▶ Church (\(\lambda\)-calculus), Post (Post’s machine), Turing (Turing machine) independently came up with formal definitions of mechanical computation
▶ All equivalent!
General Computing Machines
Alonzo Church, Emil Post, and Alan Turing (1936)

- Church (λ-calculus), Post (Post’s machine), Turing (Turing machine) independently came up with formal definitions of mechanical computation
- All equivalent!
- In this course: Turing Machines
The ‘aha’ moment
The ‘aha’ moment

And thus the Turing Machine was born.
Turing Machines

- Unrestricted memory: an infinite tape
Turing Machines

- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape
Turing Machines

- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape
  - Can read/write anywhere on the tape
Turing Machines

- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape
  - Can read/write anywhere on the tape
  - Tape is infinite in one direction only (other variants possible)
Turing Machines

- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape
  - Can read/write anywhere on the tape
  - Tape is infinite in one direction only (other variants possible)
- Initially, tape has input and the machine is reading (i.e., tape head is on) the leftmost input symbol.
Turing Machines

Unrestricted memory: an infinite tape
- A finite state machine that reads/writes symbols on the tape
- Can read/write anywhere on the tape
- Tape is infinite in one direction only (other variants possible)

Initially, tape has input and the machine is reading (i.e., tape head is on) the leftmost input symbol.

Transition (based on current state and symbol under head):
- Change control state
- Overwrite a new symbol on the tape cell under the head
- Move the head left, or right.
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\sqcup \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{\text{acc}} \in Q$ is the accept state
- $q_{\text{rej}} \in Q$ is the reject state, where $q_{\text{rej}} \neq q_{\text{acc}}$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.

Given the current state and symbol being read, the transition function describes the next state, symbol to be written and direction (left or right) in which to move the tape head.
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\sqcup \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{\text{acc}} \in Q$ is the accept state
- $q_{\text{rej}} \in Q$ is the reject state, where $q_{\text{rej}} \neq q_{\text{acc}}$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\sqcup \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{\text{acc}} \in Q$ is the accept state
- $q_{\text{rej}} \in Q$ is the reject state, where $q_{\text{rej}} \neq q_{\text{acc}}$.
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.

Given the current state and symbol being read, the transition function describes the next state, symbol to be written and direction (left or right) in which to move the tape head.
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\square \in \Gamma \setminus \Sigma$
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\sqcup \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\square \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{\text{acc}} \in Q$ is the accept state
- $q_{\text{rej}} \in Q$ is the reject state, where $q_{\text{rej}} \neq q_{\text{acc}}$
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\square \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{\text{acc}} \in Q$ is the accept state
- $q_{\text{rej}} \in Q$ is the reject state, where $q_{\text{rej}} \neq q_{\text{acc}}$
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ where

- $Q$ is a finite set of control states
- $\Sigma$ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\square \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{\text{acc}} \in Q$ is the accept state
- $q_{\text{rej}} \in Q$ is the reject state, where $q_{\text{rej}} \neq q_{\text{acc}}$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.
  Given the current state and symbol being read, the transition function describes the next state, symbol to be written and direction (left or right) in which to move the tape head.
Transition Function

$\delta(q_1, X) = (q_2, Y, L)$: Read transition as “the machine when in state $q_1$, and reading symbol $X$ under the tape head, will move to state $q_2$, overwrite $X$ with $Y$, and move its tape head to the left”
Transition Function

\[ \delta(q_1, X) = (q_2, Y, L) : \text{Read transition as “the machine when in state } q_1, \text{ and reading symbol } X \text{ under the tape head, will move to state } q_2, \text{ overwrite } X \text{ with } Y, \text{ and move its tape head to the left”} \]

- In fact \( \delta : (Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \to Q \times \Gamma \times \{L, R\} \).
δ(\(q_1, X\)) = (\(q_2, Y, L\)): Read transition as “the machine when in state \(q_1\), and reading symbol \(X\) under the tape head, will move to state \(q_2\), overwrite \(X\) with \(Y\), and move its tape head to the left”

- In fact \(\delta : (Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\). No transition defined after reaching \(q_{\text{acc}}\) or \(q_{\text{rej}}\)
Transition Function

$\delta(q_1, X) = (q_2, Y, L)$: Read transition as “the machine when in state $q_1$, and reading symbol $X$ under the tape head, will move to state $q_2$, overwrite $X$ with $Y$, and move its tape head to the left”

- In fact $\delta : (Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$. No transition defined after reaching $q_{\text{acc}}$ or $q_{\text{rej}}$
- Transitions are deterministic
Transition Function

\[
\delta(q_1, X) = (q_2, Y, L): \text{Read transition as “the machine when in state } q_1, \text{ and reading symbol } X \text{ under the tape head, will move to state } q_2, \text{ overwrite } X \text{ with } Y, \text{ and move its tape head to the left”}
\]

- In fact \( \delta : (Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \). No transition defined after reaching \( q_{\text{acc}} \) or \( q_{\text{rej}} \)
- Transitions are deterministic
- Convention: if \( \delta(q, X) \) is not explicitly specified, it is taken as leading to \( q_{\text{rej}} \), i.e., say \( \delta(q, X) = (q_{\text{rej}}, \Box, R) \)
Configurations

The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”
The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”

\[ X_1X_2 \cdots X_{i-1}qX_i \cdots X_n \]

- Includes the current state
The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”

\[ X_1 X_2 \cdots X_{i-1} q X_i \cdots X_n \]

- Includes the current state: \( q \)
The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”

\[ X_1 X_2 \cdots X_{i-1} q X_i \cdots X_n \]

- Includes the current state: \( q \)
- Position of the tape head
The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”

\[ X_1 X_2 \cdots X_{i-1} q X_i \cdots X_n \]

- Includes the current state: \( q \)
- Position of the tape head: Scanning \( i^{th} \) symbol \( X_i \)
Configurations

The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”

\[ X_1 X_2 \cdots X_{i-1} q X_i \cdots X_n \]

- Includes the current state: \( q \)
- Position of the tape head: Scanning \( i^{th} \) symbol \( X_i \)
- Contents of all the tape cells till the rightmost nonblank symbol. This is will always be finitely many cells.
The configuration (or “instantaneous description”) contains all the information to exactly capture the “current state of the computation”

\[ X_1 X_2 \cdots X_{i-1} q X_i \cdots X_n \]

- Includes the current state: \( q \)
- Position of the tape head: Scanning \( i^{\text{th}} \) symbol \( X_i \)
- Contents of all the tape cells till the rightmost nonblank symbol. This is will always be finitely many cells. Those symbols are \( X_1 X_2 \cdots X_n \), where \( X_n \neq \square \) unless the tape head is on it.
Special Configurations

- **Start configuration:** $q_0X_1 \cdots X_n$, where the input is $X_1 \cdots X_n$
Special Configurations

- **Start configuration:** $q_0 X_1 \cdots X_n$, where the input is $X_1 \cdots X_n$
- **Accept and reject configurations:** The state $q$ is $q_{\text{acc}}$ or $q_{\text{rej}}$, respectively
Special Configurations

- **Start configuration**: $q_0X_1\cdots X_n$, where the input is $X_1\cdots X_n$
- **Accept and reject configurations**: The state $q$ is $q_{\text{acc}}$ or $q_{\text{rej}}$, respectively. These configurations are **halting configurations**, because there are no transitions possible from them.
Single Step

Definition
We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

- If \(\delta(q, X_i) = (p, Y, L)\) then
  \[
  X_1 X_2 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-2} pX_{i-1} Y X_{i+1} \cdots X_n
  \]
Single Step

Definition
We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

\[\text{▶ If } \delta(q, X_i) = (p, Y, L) \text{ then} \]

\[X_1 X_2 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-2} pX_{i-1} Y X_{i+1} \cdots X_n\]

Boundary Cases:

\[\text{▶ If } i = 1 \text{ then}\]
Single Step

Definition
We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

- If \(\delta(q, X_i) = (p, Y, L)\) then
  \[
  X_1 X_2 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-2} pX_{i-1} Y X_{i+1} \cdots X_n
  \]

Boundary Cases:
- If \(i = 1\) then \(qX_1 X_2 \cdots X_n \vdash p Y X_2 \cdots X_n\)
Single Step

Definition
We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

- If \(\delta(q, X_i) = (p, Y, L)\) then

  \[
  X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash X_1X_2\cdots X_{i-2}pX_{i-1} Y X_{i+1}\cdots X_n
  \]

Boundary Cases:
- If \(i = 1\) then \(qX_1X_2\cdots X_n \vdash pY X_2\cdots X_n\)
- If \(i = n\) and \(Y = \square\) then
Single Step

Definition
We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

- If \(\delta(q, X_i) = (p, Y, L)\) then

\[
X_1X_2 \cdots X_{i-1}qX_iX_{i+1} \cdots X_n \vdash X_1X_2 \cdots X_{i-2}pX_{i-1}YX_{i+1} \cdots X_n
\]

Boundary Cases:
- If \(i = 1\) then \(qX_1X_2 \cdots X_n \vdash pYX_2 \cdots X_n\)
- If \(i = n\) and \(Y = \square\) then \(X_1 \cdots X_{n-1}qX_n \vdash X_1 \cdots pX_{n-1}\)
Single Step

Definition

We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

- If \(\delta(q, X_i) = (p, Y, L)\) then
  \[
  X_1X_2\cdots X_{i-1} qX_iX_{i+1}\cdots X_n \vdash X_1X_2\cdots X_{i-2} pX_{i-1} Y X_{i+1}\cdots X_n
  \]

Boundary Cases:
- If \(i = 1\) then \(qX_1X_2\cdots X_n \vdash pY X_2\cdots X_n\)
- If \(i = n\) and \(Y = \Box\) then \(X_1\cdots X_{n-1} qX_n \vdash X_1\cdots pX_{n-1}\)
- If \(\delta(q, X_i) = (p, Y, R)\) then
  \[
  X_1X_2\cdots X_{i-1} qX_iX_{i+1}\cdots X_n \vdash X_1X_2\cdots X_{i-1} Y pX_{i+1}\cdots X_n
  \]
Single Step

Definition

We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

- If \(\delta(q, X_i) = (p, Y, L)\) then
  \[
  X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-2} p X_{i-1} Y X_{i+1} \cdots X_n
  \]

  Boundary Cases:
  - If \(i = 1\) then
    \[
    q X_1 X_2 \cdots X_n \vdash p Y X_2 \cdots X_n
    \]
  - If \(i = n\) and \(Y = \square\) then
    \[
    X_1 \cdots X_{n-1} q X_n \vdash X_1 \cdots p X_{n-1}
    \]

- If \(\delta(q, X_i) = (p, Y, R)\) then
  \[
  X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-1} Y p X_{i+1} \cdots X_n
  \]

  Boundary Case:
  - If \(i = n\) then
Single Step

Definition
We say one configuration \((c_1)\) yields another \((c_2)\), denoted as \(c_1 \vdash c_2\), if one of the following holds.

- If \(\delta(q, X_i) = (p, Y, L)\) then
  \[
  X_1 X_2 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-2} pX_{i-1} Y X_{i+1} \cdots X_n
  \]

  **Boundary Cases:**
  - If \(i = 1\) then \(qX_1 X_2 \cdots X_n \vdash pY X_2 \cdots X_n\)
  - If \(i = n\) and \(Y = \Box\) then \(X_1 \cdots X_{n-1} qX_n \vdash X_1 \cdots pX_{n-1}\)

- If \(\delta(q, X_i) = (p, Y, R)\) then
  \[
  X_1 X_2 \cdots X_{i-1} qX_i X_{i+1} \cdots X_n \vdash X_1 X_2 \cdots X_{i-1} Y pX_{i+1} \cdots X_n
  \]

  **Boundary Case:**
  - If \(i = n\) then \(X_1 \cdots X_{n-1} qX_n \vdash X_1 \cdots X_{n-1} Y p\Box\)
Computations

**Definition**
We say $c_1 \vdash^* c_2$ if the machine can move from $c_1$ to $c_2$ in zero or more steps.
Definition
We say $c_1 \vdash^* c_2$ if the machine can move from $c_1$ to $c_2$ in zero or more steps. i.e., $c_1 = c_2$ or there exist $c_1', \ldots, c_n'$ such that $c_1 = c_1'$, $c_2 = c_n'$ and $c_i' \vdash c_{i+1}'$.
Acceptance and Recognition

Definition
A Turing machine $M$ accepts $w$ iff $q_0w \vdash^* \alpha_1 q_{acc} \alpha_2$, where $\alpha_1, \alpha_2$ are some strings. In other words, the machine $M$ when started in its initial state and with $w$ as input, reaches the accept state.
Acceptance and Recognition

Definition
A Turing machine $M$ accepts $w$ iff $q_0w\vdash^*\alpha_1q_{acc}\alpha_2$, where $\alpha_1, \alpha_2$ are some strings. In other words, the machine $M$ when started in its initial state and with $w$ as input, reaches the accept state.

Note: The machine may not read all the symbols in $w$. It may pass back and forth over some symbols of $w$ several times. Finally, $w$ may have been completely overwritten.
Acceptance and Recognition

Definition
A Turing machine $M$ accepts $w$ iff $q_0 w \vdash^* \alpha_1 q_{\text{acc}} \alpha_2$, where $\alpha_1, \alpha_2$ are some strings. In other words, the machine $M$ when started in its initial state and with $w$ as input, reaches the accept state.

Note: The machine may not read all the symbols in $w$. It may pass back and forth over some symbols of $w$ several times. Finally, $w$ may have been completely overwritten.

Definition
For a Turing machine $M$, define $L(M) = \{w \mid M$ accepts $w\}$. 
Acceptance and Recognition

Definition
A Turing machine $M$ accepts $w$ iff $q_0w \vdash^* \alpha_1 q_{\text{acc}} \alpha_2$, where $\alpha_1, \alpha_2$ are some strings. In other words, the machine $M$ when started in its initial state and with $w$ as input, reaches the accept state.

Note: The machine may not read all the symbols in $w$. It may pass back and forth over some symbols of $w$ several times. Finally, $w$ may have been completely overwritten.

Definition
For a Turing machine $M$, define $L(M) = \{w \mid M \text{ accepts } w\}$. $M$ is said to accept or recognize a language $L$ if $L = L(M)$. 
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

Design a TM to accept the language \( L = \{0^n1^n \mid n > 0\} \)
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

Design a TM to accept the language \( L = \{0^n1^n \mid n > 0\} \)

High level description

On input string \( w \)

while there are unmarked 0s, do

Mark the left most 0

Scan right till the leftmost unmarked 1;

if there is no such 1 then crash

Mark the leftmost 1

done

Check to see that there are no unmarked 1s;

if there are then crash

accept
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

- **q<sub>0</sub>**
  - 0 → A, R
- **q<sub>1</sub>**
  - 1 → B, L
- **q<sub>2</sub>**
- **q<sub>3</sub>**
  - ⊥ → ⊥, R
- **q<sub>acc</sub>**
  - B → B, R

- A → A, R
- B → B, R
- 0 → 0, R
- B → B, R
- B → B, L
- 0 → 0, L

- Accepts input 0011: q<sub>0</sub>0011 ⊢
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

Accepts input 0011: \( q_00011 \vdash Aq_1011 \vdash \)
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

- \(q_0\) to \(q_1\): 0 → A, R
- \(q_1\) to \(q_2\): 1 → B, L
- \(q_2\) to \(q_3\): B → B, R
- \(q_3\) to itself: ⊥ → ⊥, R
- \(q_3\) to \(q_{acc}\): ⊥ → ⊥, R
- \(q_{acc}\) to itself: A → A, R

- \(q_0\) to itself: A → A, R
- \(q_1\) to itself: B → B, L
- \(q_2\) to itself: 0 → 0, L

Accepts input 0011: \(q_0\)0011 ⊢ A\(q_1\)011 ⊢ A0\(q_1\)11 ⊢
Example 1: TM for $\{0^n1^n \mid n > 0\}$

- Accepts input 0011: $q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash$
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

- **Accepts input 0011:** 
  \[ q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash \]

- **Transition rules:**
  - \( A \rightarrow A, R \)
  - \( 0 \rightarrow A, R \)
  - \( 1 \rightarrow B, L \)
  - \( B \rightarrow B, R \)
  - \( B \rightarrow B, L \)
  - \( 0 \rightarrow 0, R \)
  - \( 0 \rightarrow 0, L \)
  - \( \square \rightarrow \square, R \)
  - \( B \rightarrow B, R \)
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

Accepts input 0011: \(q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash\)
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

Accepts input 0011: \( q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash \ldots \)
Example 1: TM for $\{0^n1^n \mid n > 0\}$

Accepts input $0011$: $q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash AABq_11 \vdash$
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

- Accepts input 0011: \( q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash AABq_11 \vdash AAq_2BB \vdash \)

- \( R \)
Example 1: TM for \( \{0^n1^n \mid n > 0 \} \)

Accepts input 0011: \( q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash AABq_11 \vdash AAq_2BB \vdash Aq_2ABB \vdash \)
Example 1: TM for $\{0^n1^n \mid n > 0\}$

Accepts input $0011$: $q_00011 \mid Aq_1011 \mid A0q_111 \mid Aq_20B1 \mid q_2A0B1 \mid Aq_00B1 \mid AAq_1B1 \mid AABq_11 \mid AAq_2BB \mid Aq_2ABB \mid AAq_0BB \mid$
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

- **States:**
  - \(q_0\)
  - \(q_1\)
  - \(q_2\)
  - \(q_3\)
  - \(q_{\text{acc}}\)

- **Transitions:**
  - \(A \rightarrow A, R\)
  - \(0 \rightarrow A, R\)
  - \(1 \rightarrow B, L\)
  - \(B \rightarrow B, R\)
  - \(0 \rightarrow 0, R\)
  - \(B \rightarrow B, R\)
  - \(B \rightarrow B, L\)
  - \(0 \rightarrow 0, L\)

- **Diagram:**
  - From \(q_0\) to \(q_1\) on input 0
  - From \(q_1\) to \(q_2\) on input 1
  - From \(q_2\) to \(q_{\text{acc}}\) on input 0
  - From \(q_{\text{acc}}\) to \(q_{\text{rej}}\) on input 0

- **Accepts input 0011:**
  - \(q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash AABq_11 \vdash AAq_2BB \vdash Aq_2ABB \vdash AAq_0BB \vdash AABq_3B \vdash \)
Example 1: TM for \{0^n1^n \mid n > 0\}

- Accepts input 0011: \( q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash AABq_11 \vdash AAq_2BB \vdash Aq_2ABB \vdash AAq_0BB \vdash AABq_3B \vdash AABBq_3\)
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

- Accepts input 0011: \(q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash AABq_11 \vdash AAq_2BB \vdash Aq_2ABB \vdash AAq_0BB \vdash AABq_3B \vdash AABBq_3 \sqcup \vdash AABB\sqcup q_{acc}\sqcup \)

- Rejects input 00: \(q_000 \vdash Aq_10 \vdash A0q_1 \vdash Aq_2B \vdash q_2B \vdash Aq_0B \vdash AAq_1 \vdash AABq_1 \vdash AAq_2BB \vdash Aq_2ABB \vdash AAq_0BB \vdash AABq_3B \vdash AABBq_3 \sqcup \vdash AABBB\sqcup q_{acc}\sqcup \)
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

Accepts input 0011: 
\[ q_00011 \rightarrow Aq_1011 \rightarrow A0q_111 \rightarrow Aq_20B1 \rightarrow q_2A0B1 \rightarrow Aq_00B1 \rightarrow AAq_1B1 \rightarrow AABq_11 \rightarrow AAq_2BB \rightarrow Aq_2ABB \rightarrow AAq_0BB \rightarrow AABq_3B \rightarrow AABBBq_3\square \rightarrow AABB\square q_{\text{acc}}\square \]

Rejects input 00: 
\[ q_000 \rightarrow Aq_10 \rightarrow A0q_1\square \rightarrow \]
Example 1: TM for \( \{0^n1^n \mid n > 0\} \)

- Accepts input 0011: \( q_00011 \vdash A q_1011 \vdash A 0 q_111 \vdash A q_20B1 \vdash q_2 A 0 B1 \vdash A q_00B1 \vdash A A q_1 B1 \vdash A A B q_1 1 \vdash A A q_2 B B \vdash A q_2 A B B \vdash A A q_0 B B \vdash A A B q_3 B \vdash A A B B q_3 \sqcup \vdash A A B B \sqcup q_{acc} \sqcup \)

- Rejects input 00: \( q_000 \vdash A q_10 \vdash A 0 q_1 \sqcup \vdash A 0 \sqcup q_{rej} \sqcup \)
Example: \( \{0^n1^n \mid n > 0\} \)

**Formal Definition**

The machine is \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}) \) where

\[
\delta(q_0, 0) = (q_1, A, R) \\
\delta(q_0, B) = (q_3, B, R) \\
\delta(q_1, 0) = (q_1, 0, R) \\
\delta(q_1, B) = (q_1, B, R) \\
\delta(q_1, 1) = (q_2, B, L) \\
\delta(q_2, B) = (q_2, B, L) \\
\delta(q_2, 0) = (q_2, 0, L) \\
\delta(q_2, A) = (q_0, A, R) \\
\delta(q_3, B) = (q_3, B, R) \\
\delta(q_3, \sqcup) = (q_{\text{acc}}, \sqcup, R) \\
\delta(q_{\text{rej}}, \sqcup) = (q_{\text{rej}}, \sqcup, R).
\]
Example: \( \{0^n1^n \mid n > 0\} \)

Formal Definition

The machine is \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}) \) where

- \( Q = \{q_0, q_1, q_2, q_3, q_{\text{acc}}, q_{\text{rej}}\} \)
Example: \( \{0^n1^n \mid n > 0\} \)

Formal Definition

The machine is \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}) \) where

- \( Q = \{q_0, q_1, q_2, q_3, q_{\text{acc}}, q_{\text{rej}}\} \)
- \( \Sigma = \{0, 1\} \), and

\( \delta \) is given as follows:

- \( \delta(q_0, 0) = (q_1, A, R) \)
- \( \delta(q_0, B) = (q_3, B, R) \)
- \( \delta(q_1, 0) = (q_1, 0, R) \)
- \( \delta(q_1, B) = (q_1, B, R) \)
- \( \delta(q_2, B) = (q_2, B, L) \)
- \( \delta(q_2, A) = (q_0, A, R) \)
- \( \delta(q_3, B) = (q_3, B, R) \)
- \( \delta(q_3, \_\_) = (q_{\text{acc}}, \_\_, R) \)

In all other cases, \( \delta(q, X) = (q_{\text{rej}}, \_\_, R) \).
Example: \( \{0^n1^n \mid n > 0\} \)

Formal Definition

The machine is \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}) \) where

- \( Q = \{q_0, q_1, q_2, q_3, q_{\text{acc}}, q_{\text{rej}}\} \)
- \( \Sigma = \{0, 1\} \), and \( \Gamma = \{0, 1, A, B, \sqcup\} \)
Example: \( \{0^n1^n \mid n > 0\} \)

Formal Definition

The machine is \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}) \) where

- \( Q = \{ q_0, q_1, q_2, q_3, q_{\text{acc}}, q_{\text{rej}} \} \)
- \( \Sigma = \{ 0, 1 \} \), and \( \Gamma = \{ 0, 1, A, B, \square \} \)
- \( \delta \) is given as follows

\[
\begin{align*}
\delta(q_0, 0) &= (q_1, A, R) & \delta(q_0, B) &= (q_3, B, R) \\
\delta(q_1, 0) &= (q_1, 0, R) & \delta(q_1, B) &= (q_1, B, R) \\
\delta(q_1, 1) &= (q_2, B, L) & \delta(q_2, B) &= (q_2, B, L) \\
\delta(q_2, 0) &= (q_2, 0, L) & \delta(q_2, A) &= (q_0, A, R) \\
\delta(q_3, B) &= (q_3, B, R) & \delta(q_3, \square) &= (q_{\text{acc}}, \square, R)
\end{align*}
\]

In all other cases, \( \delta(q, X) = (q_{\text{rej}}, \square, R) \).
Example: \( \{0^n1^n \mid n > 0 \} \)

Formal Definition

The machine is \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}) \) where

- \( Q = \{ q_0, q_1, q_2, q_3, q_{\text{acc}}, q_{\text{rej}} \} \)
- \( \Sigma = \{ 0, 1 \} \), and \( \Gamma = \{ 0, 1, A, B, \sqcup \} \)
- \( \delta \) is given as follows

\[
\begin{align*}
\delta(q_0, 0) &= (q_1, A, R) & \delta(q_0, B) &= (q_3, B, R) \\
\delta(q_1, 0) &= (q_1, 0, R) & \delta(q_1, B) &= (q_1, B, R) \\
\delta(q_1, 1) &= (q_2, B, L) & \delta(q_2, B) &= (q_2, B, L) \\
\delta(q_2, 0) &= (q_2, 0, L) & \delta(q_2, A) &= (q_0, A, R) \\
\delta(q_3, B) &= (q_3, B, R) & \delta(q_3, \sqcup) &= (q_{\text{acc}}, \sqcup, R)
\end{align*}
\]

In all other cases, \( \delta(q, X) = (q_{\text{rej}}, \sqcup, R) \). So for example, \( \delta(q_0, 1) = (q_{\text{rej}}, \sqcup, R) \).
Example 2: TM for \( \{0^n1^n2^n \mid n > 0\} \)

Design a TM to accept the language \( L = \{0^n1^n2^n \mid n > 0\} \)
Example 2: TM for \( \{0^n1^n2^n \mid n > 0\} \)

Design a TM to accept the language \( L = \{0^n1^n2^n \mid n > 0\} \)

High level description

On input string \( w \)

while there are unmarked 0s, do

Mark the leftmost 0
Scan right to reach the leftmost unmarked 1;
if there is no such 1 then crash
Mark the leftmost 1
Scan right to reach the leftmost unmarked 2;
if there is no such 2 then crash
Mark the leftmost 2

done

Check to see that there are no unmarked 1s or 2s;
if there are then crash

accept
Example 2: TM for $\{0^n 1^n 2^n \mid n > 0\}$
Example 2: TM for \( \{0^n1^n2^n \mid n > 0\} \)

e.g.: \( q_0001122 \vdash A0Bq_31C2 \)
Example 2: TM for $\{0^n1^n2^n \mid n > 0\}$

e.g.: $q_0001122 \vdash^* A0Bq_31C2 \vdash^* q_3A0B1C2$
Example 2: TM for \( \{0^n1^n2^n \mid n > 0\} \)

e.g.: \( q_0001122 \vdash^* A0Bq_31C2 \vdash^* q_3A0B1C2 \vdash Aq_00B1C2 \)
Example 2: TM for \( \{0^n1^n2^n \mid n > 0\} \)

e.g.: \( q_0001122 \vdash^* A0Bq_31C2 \vdash^* q_3A0B1C2 \vdash Aq_00B1C2 \vdash^* AAq_0BBCC \)
Example 2: TM for \( \{0^n1^n2^n \mid n > 0\} \)

e.g.: \( q_0001122 \vdash^* A0Bq_31C2 \vdash^* q_3A0B1C2 \vdash Aq_00B1C2 \vdash^* AAq_0BBCC \vdash^* AABBCq_4 \)
Example 2: TM for $\{0^n1^n2^n \mid n > 0\}$

e.g.: $q_0001122 \vdash *A0Bq_31C2 \vdash *q_3A0B1C2 \vdash Aq_00B1C2 \vdash *AAq_0BBCC \vdash *AABBCCq_4\sqcup \vdash AABBCC\sqcup q_{\text{acc}}\sqcup$
Recognizing/Deciding a Language

- Only halting configurations are those with state $q_{\text{acc}}$ or $q_{\text{rej}}$.
- A Turing machine may keep running forever on some input.
- Then the machine does not accept that input.
- So two ways to not accept: reject or never halt (loop).
- Three possible outcomes: acept, reject, or loop.
Recognizing/Deciding a Language

- Only halting configurations are those with state $q_{acc}$ or $q_{rej}$
Recognizing/Deciding a Language

- Only halting configurations are those with state $q_{acc}$ or $q_{rej}$
- A Turing machine may keep running forever on some input
Recognizing/Deciding a Language

- Only halting configurations are those with state $q_{\text{acc}}$ or $q_{\text{rej}}$
- A Turing machine may keep running forever on some input
- Then the machine does not accept that input
Recognizing/Deciding a Language

- Only halting configurations are those with state $q_{\text{acc}}$ or $q_{\text{rej}}$
- A Turing machine may keep running forever on some input
- Then the machine does not accept that input
- So two ways to not accept: reject or never halt (loop)
Recognizing/Deciding a Language

- Only halting configurations are those with state \( q_{\text{acc}} \) or \( q_{\text{rej}} \)
- A Turing machine may keep running forever on some input
- Then the machine does not accept that input
- So two ways to not accept: reject or never halt (loop)
  Three possible outcomes: accept, reject, or loop
Recognizing a Language

Definition
A language $L$ is said to be Turing-recognizable if there is some Turing machine $M$ that recognizes it.
That is, for every $w \in L$, $M$ must accept $w$. Also for every $w \in L$, $M$ must not accept $w$ (either reject it or does not halt).
Deciding a Language

Definition
A Turing machine $M$ is said to decide a language $L$ if $L = L(M)$ and $M$ halts on every input.

Definition
A language $L$ is said to be Turing-decidable (or simply decidable) if there is some Turing machine $M$ that decides it.
Deciding a Language

Definition
A Turing machine $M$ is said to decide a language $L$ if $L = L(M)$ and $M$ halts on every input.

Definition
A language $L$ is said to be Turing-decidable (or simply decidable) if there is some Turing machine $M$ that decides it.
Deciding a language is more than recognizing it!
Deciding a Language

Definition
A Turing machine $M$ is said to decide a language $L$ if $L = L(M)$ and $M$ halts on every input.

Definition
A language $L$ is said to be Turing-decidable (or simply decidable) if there is some Turing machine $M$ that decides it.

Deciding a language is more than recognizing it! There are languages which are recognizable, but not decidable.