CIS 770: Formal Language Theory

Pavithra Prabhakar

Kansas State University

Spring 2016

General Computing Machines

- Machines so far: DFAs, NFAs, GNFAs, PDAs

General Computing Machines

- Machines so far: DFAs, NFAs, GNFAs, PDAs
  - Limitations on how much memory they can use: fixed amount of memory vs infinite memory
  - Limitations on how to access the memory: stack accessed only from the top
  - Limitations on what they can compute/decide: only regular languages or context free languages

General Computing Machines

- Machines so far: DFAs, NFAs, GNFAs, PDAs
  - Limitations on how much memory they can use: fixed amount of memory vs infinite memory
  - Limitations on how to access the memory: stack accessed only from the top
  - Limitations on what they can compute/decide: only regular languages or context free languages
- The complete machine?

General Computing Machines

- Machines so far: DFAs, NFAs, GNFAs, PDAs
  - Limitations on how much memory they can use: fixed amount of memory vs infinite memory
  - Limitations on how to access the memory: stack accessed only from the top
  - Limitations on what they can compute/decide: only regular languages or context free languages
- The complete machine?
  - No limitation on what they can compute?

General Computing Machines

- Machines so far: DFAs, NFAs, GNFAs, PDAs
  - Limitations on how much memory they can use: fixed amount of memory vs infinite memory
  - Limitations on how to access the memory: stack accessed only from the top
  - Limitations on what they can compute/decide: only regular languages or context free languages
- The complete machine?
  - 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: DFAs, NFAs, GNFAs, PDAs
  - Limitations on how much memory they can use: fixed amount of memory vs infinite memory
  - Limitations on how to access the memory: stack accessed only from the top
  - Limitations on what they can compute/decide: only regular languages or context free languages
- The complete machine?
  - 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)
  - Come up with a model that describes all "conceivable" computation

General Computing Machines

Alonzo Church, Emil Post, and Alan Turing (1936)

Alonzo Church

Emil Post

Alan Turing

- Church (λ-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)

Alonzo Church

Emil Post

Alan Turing

- Church (λ-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)

Alonzo Church

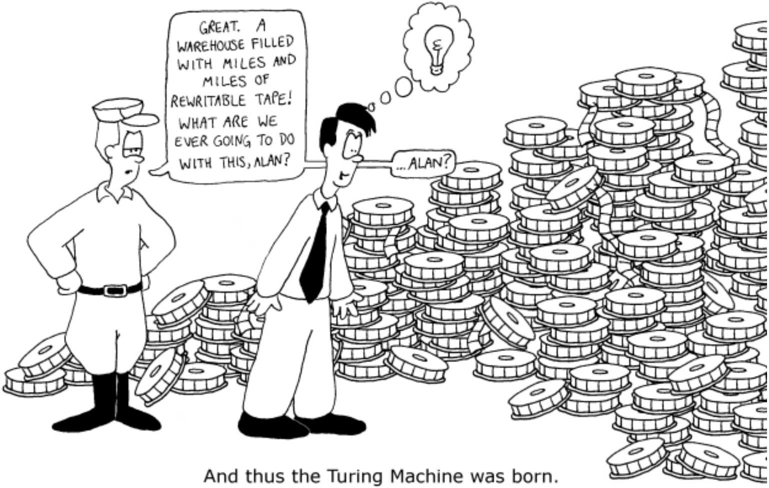
Emil Post

Alan Turing

- 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

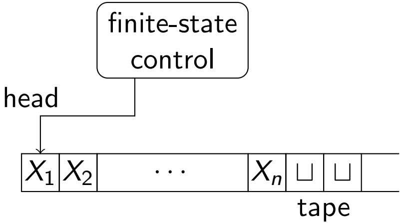
The ‘aha’ moment

The 'aha' moment



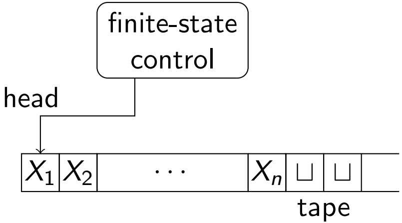
And thus the Turing Machine was born.

Turing Machines



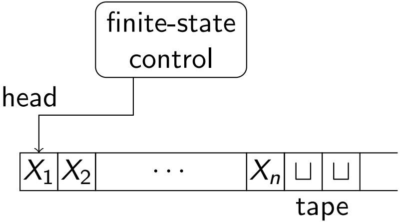
- Unrestricted memory: an infinite tape

Turing Machines



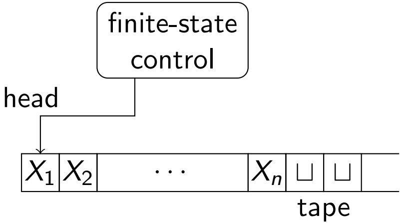
- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape

Turing Machines



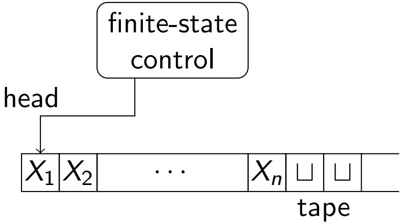
- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape
  - Can read/write anywhere on the tape

Turing Machines



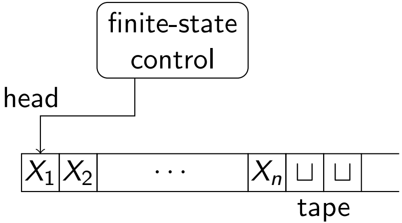
- Unrestricted memory: an infinite tape
  - A finite state machine that reads/writes symbols on the tape
  - Can read/write anywhere on the tape
  - Tape is infinite in one direction only (other variants possible)

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 $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where

A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where

- Q is a finite set of control states

A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where

- Q is a finite set of control states
- Σ is a finite set of input symbols

A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where

- Q is a finite set of control states
- Σ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\sqcup \in \Gamma \setminus \Sigma$

A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where

- Q is a finite set of control states
- Σ 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 $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}})$ where

- Q is a finite set of control states
- Σ 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
- $q_{\mathsf{acc}} \in Q$ is the accept state

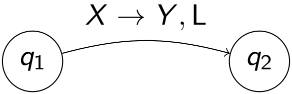
A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where

- Q is a finite set of control states
- Σ 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
- $q_{acc} \in Q$ is the accept state
- $q_{rej} \in Q$ is the reject state, where $q_{rej} \neq q_{acc}$

A Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}})$ where

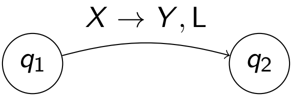
- Q is a finite set of control states
- Σ is a finite set of input symbols
- $\Gamma \supseteq \Sigma$ is a finite set of tape symbols. Also, a blank symbol $\sqcup \in \Gamma \setminus \Sigma$
- $q_0 \in Q$ is the initial state
- $q_{\mathrm{acc}} \in Q$ is the accept state
- $q_{\mathrm{rej}} \in Q$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



$\delta(q_1, X) = (q_2, Y, \mathrm{L})$: 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"

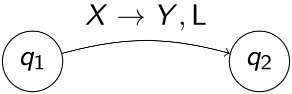
Transition Function



$\delta(q_1, X) = (q_2, Y, \mathsf{L})$: 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 $\delta : (Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}) \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$.

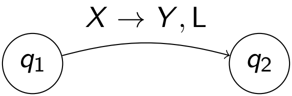
Transition Function



$\delta(q_1, X) = (q_2, Y, \mathsf{L})$: 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 $\delta : (Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}) \times \Gamma \to Q \times \Gamma \times \{\mathsf{L}, \mathsf{R}\}$. No transition defined after reaching q_{acc} or q_{rej}

Transition Function



$\delta(q_1, X) = (q_2, Y, \mathrm{L})$: 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 $\delta : (Q \setminus \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}) \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$. No transition defined after reaching q_{acc} or q_{rej}
- Transitions are deterministic

Transition Function

$$q_1 \xrightarrow{X \to Y, \mathrm{L}} q_2$$

$\delta(q_1, X) = (q_2, Y, \mathrm{L})$: 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 $\delta : (Q \setminus \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}) \times \Gamma \to Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$. No transition defined after reaching q_{acc} or q_{rej}
- Transitions are deterministic
- Convention: if $\delta(q, X)$ is not explicitly specified, it is taken as leading to q_{rej}, i.e., say $\delta(q, X) = (q_{\mathrm{rej}}, \sqcup, \mathrm{R})$

Configurations

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

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

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

Configurations

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: q
- Position of the tape head

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

Configurations

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: 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.

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. Those symbols are $X_1 X_2 \cdots X_n$, where $X_n \neq \sqcup$ 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_{acc} or q_{rej}, respectively

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_{acc} or q_{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 $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathrm{L})$ then

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

Single Step

Definition

We say one configuration (C_1) yields another (C_2), denoted as $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathrm{L})$ 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

Single Step

Definition

We say one configuration (C_1) yields another (C_2), denoted as $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathsf{L})$ 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 $q X_1 X_2 \cdots X_n \vdash p Y X_2 \cdots X_n$

Single Step

Definition

We say one configuration (C_1) yields another (C_2), denoted as $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathrm{L})$ 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 $Y = \sqcup$ then

Single Step

Definition

We say one configuration (C_1) yields another (C_2), denoted as $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathrm{L})$ 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 $Y = \sqcup$ 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 $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathrm{L})$ 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 $Y = \sqcup$ then $X_1 \cdots X_{n-1}qX_n \vdash X_1 \cdots pX_{n-1}$

- If $\delta(q, X_i) = (p, Y, \mathrm{R})$ then

  $$X_1X_2 \cdots X_{i-1}qX_iX_{i+1} \cdots X_n \vdash X_1X_2 \cdots X_{i-1}YpX_{i+1} \cdots X_n$$

Single Step

Definition

We say one configuration (C_1) yields another (C_2), denoted as $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathrm{L})$ 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 $Y = \sqcup$ then $X_1\cdots X_{n-1}qX_n \vdash X_1\cdots pX_{n-1}$

- If $\delta(q, X_i) = (p, Y, \mathrm{R})$ then

  $$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$$

  Boundary Case:

  - If $i = n$ then

Single Step

Definition

We say one configuration (C_1) yields another (C_2), denoted as $\mathrm{C}_1 \vdash \mathrm{C}_2$, if one of the following holds.

- If $\delta(q, X_i) = (p, Y, \mathrm{L})$ 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 $Y = \sqcup$ then $X_1\cdots X_{n-1}qX_n \vdash X_1\cdots pX_{n-1}$
- If $\delta(q, X_i) = (p, Y, \mathrm{R})$ then

  $$X_1X_2\cdots X_{i-1}qX_iX_{i+1}\cdots X_n \vdash X_1X_2\cdots X_{i-1}YpX_{i+1}\cdots X_n$$

  Boundary Case:
  - If $i = n$ then $X_1\cdots X_{n-1}qX_n \vdash X_1\cdots X_{n-1}Yp\sqcup$

Computations

Definition

We say $\mathrm{C}_1 \vdash^* \mathrm{C}_2$ if the machine can move from C_1 to C_2 in zero or more steps.

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. 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 $q_0 w \vdash^* \alpha_1 q_{\mathsf{acc}} \alpha_2$, where α_1, α_2 are some strings. In other words, the machine M when started in its intial state and with w as input, reaches the accept state.

Acceptance and Recognition

Definition

A Turing machine M accepts w iff $q_0 w \vdash^* \alpha_1 q_{acc} \alpha_2$, where α_1, α_2 are some strings. In other words, the machine M when started in its intial 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 $q_0 w \vdash^* \alpha_1 q_{\mathrm{acc}} \alpha_2$, where α_1, α_2 are some strings. In other words, the machine M when started in its intial 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 $L(M) = \{w \mid M \text{ accepts } w\}$.

Acceptance and Recognition

Definition

A Turing machine M accepts w iff $q_0 w \vdash^* \alpha_1 q_{\mathsf{acc}} \alpha_2$, where α_1, α_2 are some strings. In other words, the machine M when started in its intial 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 $\{0^n 1^n \mid n > 0\}$

Design a TM to accept the language $L = \{0^n 1^n \mid n > 0\}$

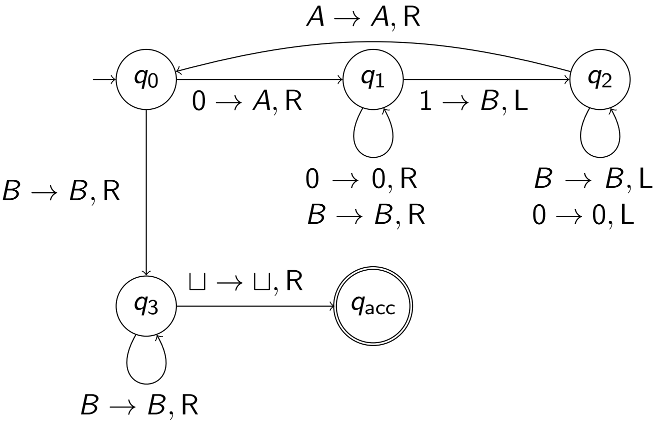
Example 1: TM for $\{0^n1^n \mid n > 0\}$

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

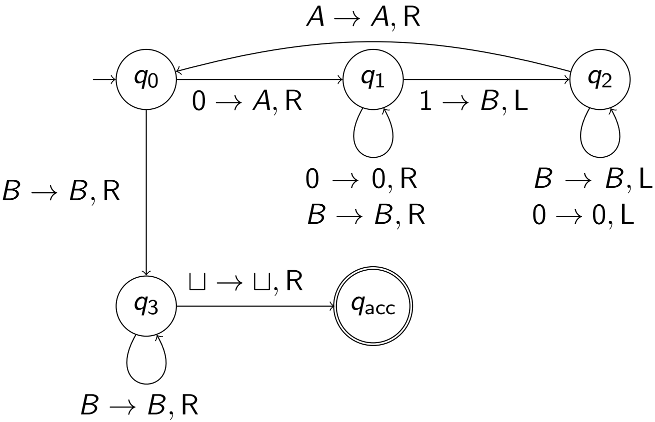
High level description

```
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 $\{0^n1^n \mid n > 0\}$

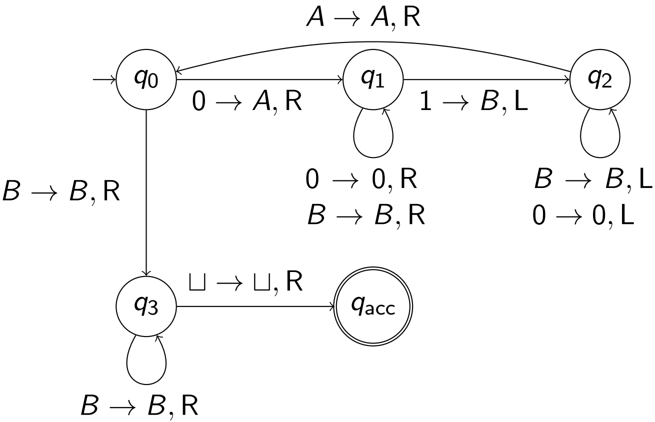


Example 1: TM for $\{0^n1^n \mid n > 0\}$



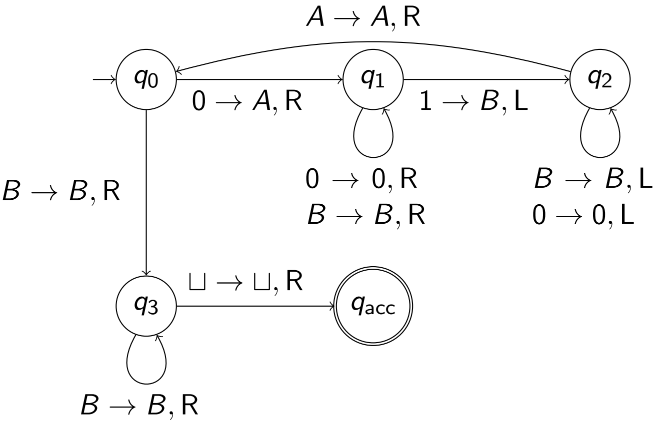
- Accepts input 0011: $q_0 0011 \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



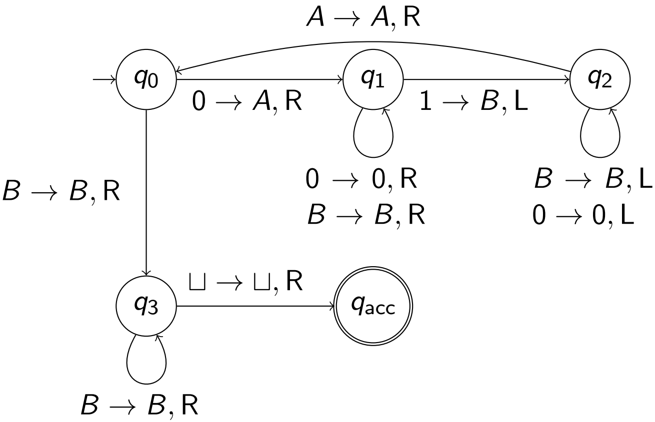
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



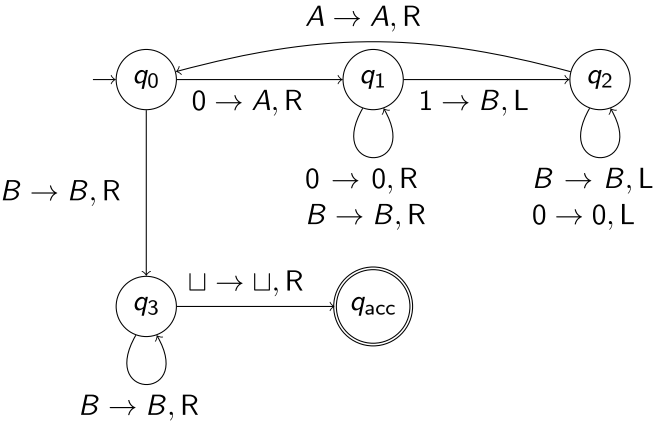
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



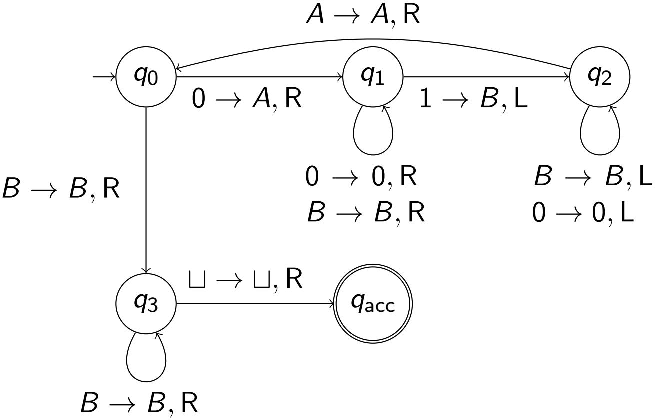
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



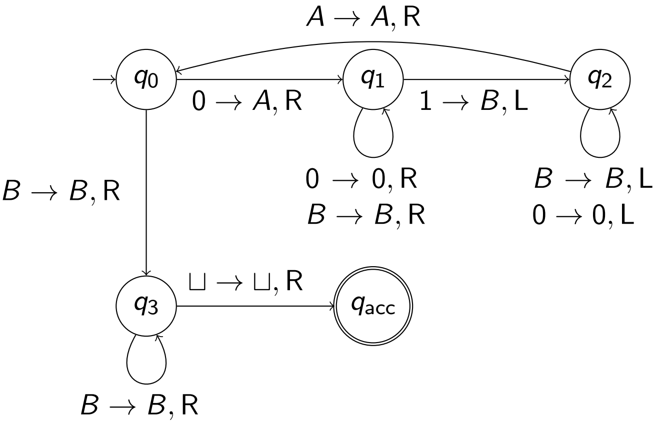
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



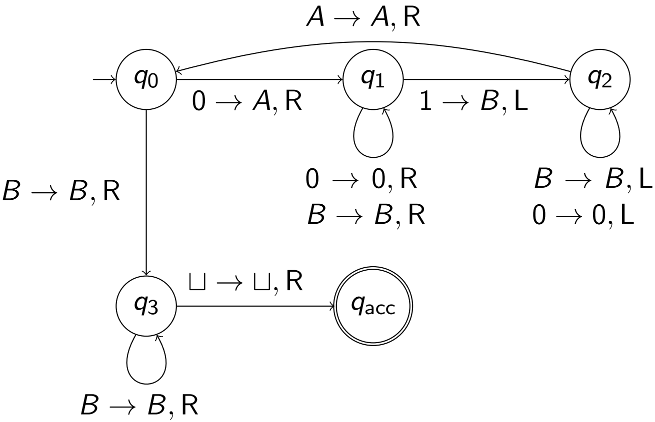
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



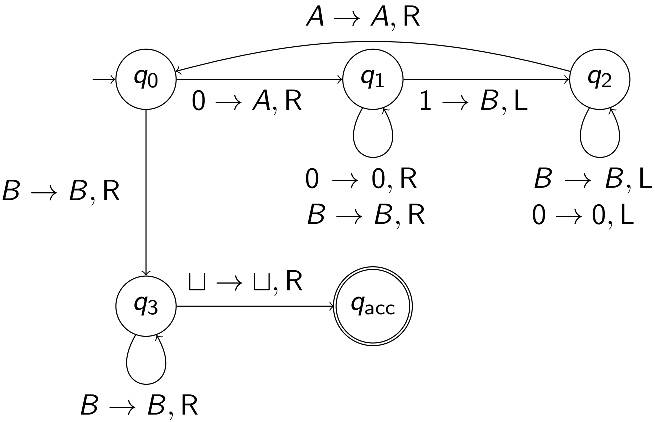
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash$

Example 1: TM for $\{0^n 1^n \mid n > 0\}$



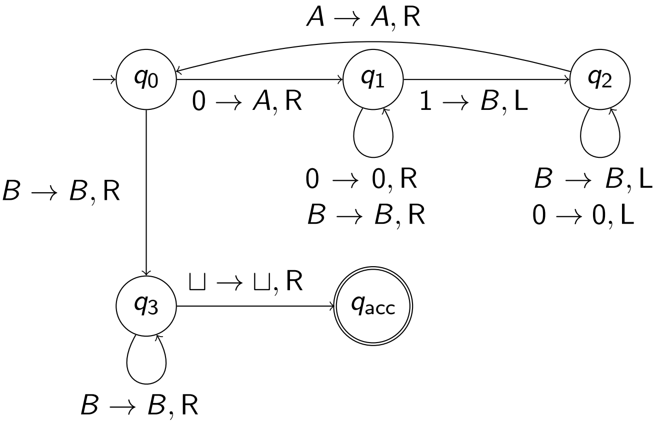
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



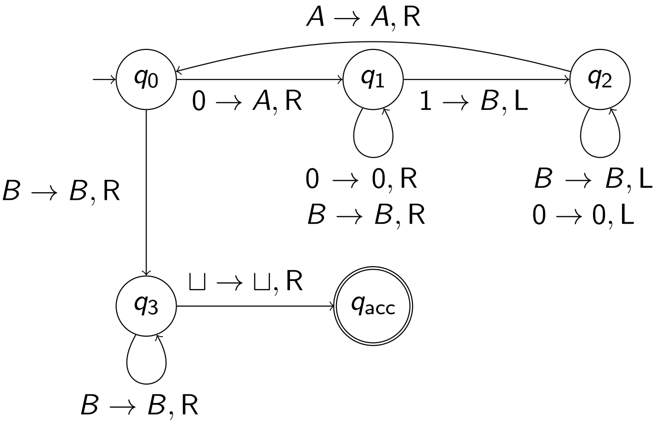
- Accepts input 0011: $q_00011 \vdash Aq_1011 \vdash A0q_111 \vdash Aq_20B1 \vdash q_2A0B1 \vdash Aq_00B1 \vdash AAq_1B1 \vdash AABq_11 \vdash AAq_2BB \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



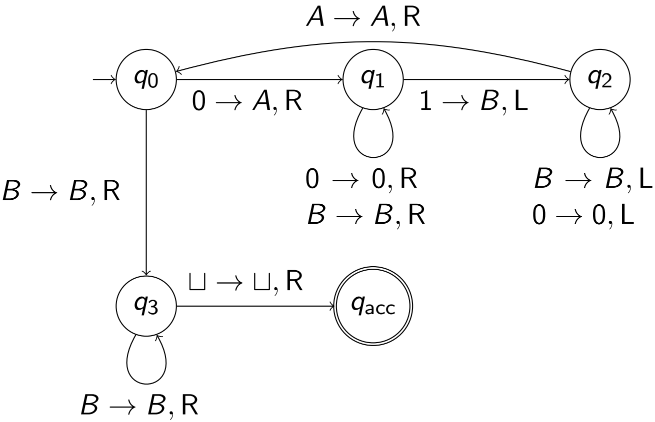
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash AAq_2 BB \vdash Aq_2 ABB \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



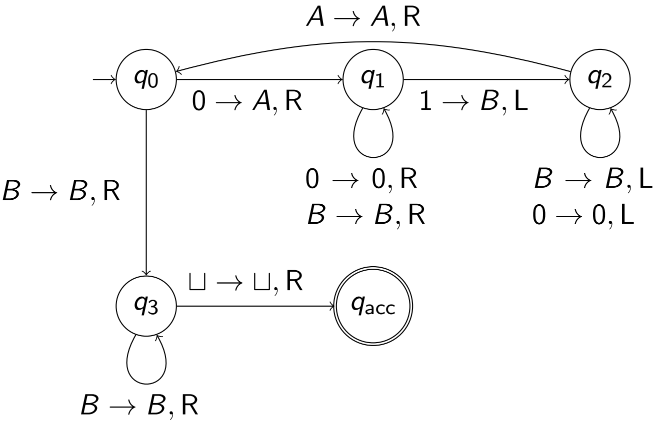
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash AAq_2 BB \vdash Aq_2 ABB \vdash AAq_0 BB \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



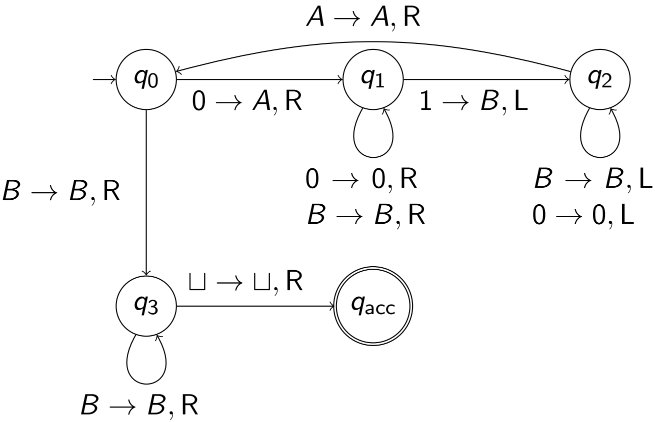
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash AAq_2 BB \vdash Aq_2 ABB \vdash AAq_0 BB \vdash AABq_3 B \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



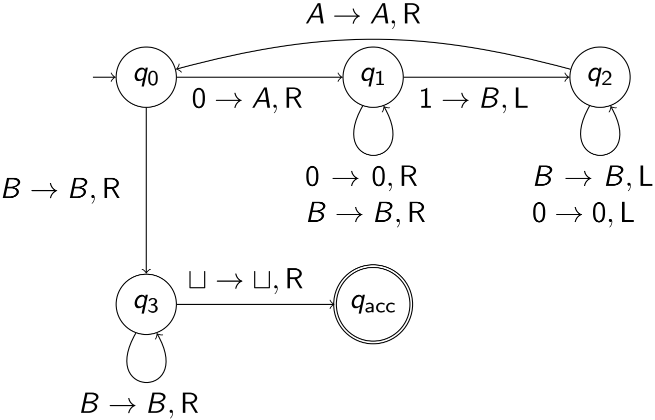
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash AAq_2 BB \vdash Aq_2 ABB \vdash AAq_0 BB \vdash AABq_3 B \vdash AABBq_3 \sqcup \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



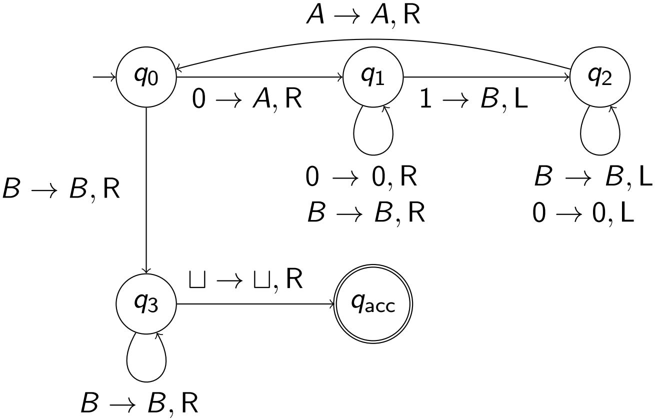
- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash AAq_2 BB \vdash Aq_2 ABB \vdash AAq_0 BB \vdash AABq_3 B \vdash AABBq_3 \sqcup \vdash AABB\sqcup q_{\mathrm{acc}} \sqcup$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash AAq_2 BB \vdash Aq_2 ABB \vdash AAq_0 BB \vdash AABq_3 B \vdash AABBq_3 \sqcup \vdash AABB\sqcup q_{acc} \sqcup$
- Rejects input 00: $q_0 00 \vdash Aq_1 0 \vdash A0q_1 \sqcup \vdash$

Example 1: TM for $\{0^n1^n \mid n > 0\}$



- Accepts input 0011: $q_0 0011 \vdash Aq_1 011 \vdash A0q_1 11 \vdash Aq_2 0B1 \vdash q_2 A0B1 \vdash Aq_0 0B1 \vdash AAq_1 B1 \vdash AABq_1 1 \vdash AAq_2 BB \vdash Aq_2 ABB \vdash AAq_0 BB \vdash AABq_3 B \vdash AABBq_3 \sqcup \vdash AABB\sqcup q_{acc} \sqcup$
- Rejects input 00: $q_0 00 \vdash Aq_1 0 \vdash A0q_1 \sqcup \vdash A0 \sqcup q_{rej} \sqcup$

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

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

- $Q = \{q_0, q_1, q_2, q_3, q_{\text{acc}}, q_{\text{rej}}\}$

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

Example: $\{0^n 1^n \mid n > 0\}$

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 $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$ where

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

$$
\begin{array}{ll}
\delta(q_0, 0) = (q_1, A, R) & \delta(q_0, B) = (q_3, B, R) \\
\delta(q_1, 0) = (q_1, 0, R) & \delta(q_1, B) = (q_1, B, R) \\
\delta(q_1, 1) = (q_2, B, L) & \delta(q_2, B) = (q_2, B, L) \\
\delta(q_2, 0) = (q_2, 0, L) & \delta(q_2, A) = (q_0, A, R) \\
\delta(q_3, B) = (q_3, B, R) & \delta(q_3, \sqcup) = (q_{acc}, \sqcup, R)
\end{array}
$$

In all other cases, $\delta(q, X) = (q_{rej}, \sqcup, R)$.

Example: $\{0^n 1^n \mid n > 0\}$

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 $\Gamma = \{0, 1, A, B, \sqcup\}$
- δ is given as follows

$$
\begin{array}{ll}
\delta(q_0, 0) = (q_1, A, R) & \delta(q_0, B) = (q_3, B, R) \\
\delta(q_1, 0) = (q_1, 0, R) & \delta(q_1, B) = (q_1, B, R) \\
\delta(q_1, 1) = (q_2, B, L) & \delta(q_2, B) = (q_2, B, L) \\
\delta(q_2, 0) = (q_2, 0, L) & \delta(q_2, A) = (q_0, A, R) \\
\delta(q_3, B) = (q_3, B, R) & \delta(q_3, \sqcup) = (q_{\text{acc}}, \sqcup, R)
\end{array}
$$

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 $L = \{0^n1^n2^n \mid n > 0\}$

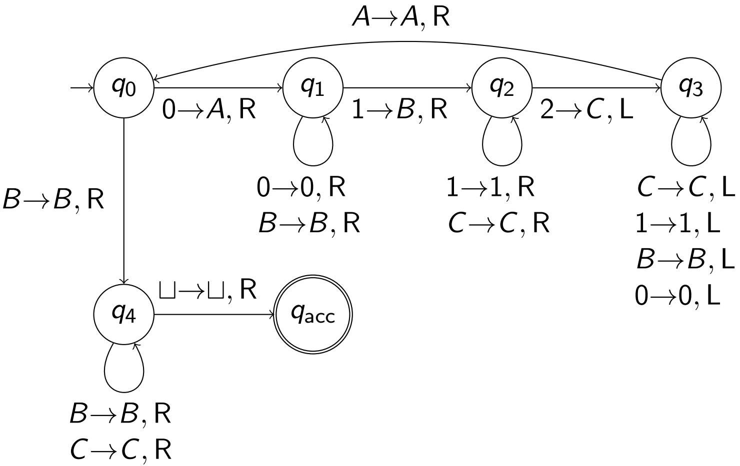
Example 2: TM for $\{0^n1^n2^n \mid n > 0\}$

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

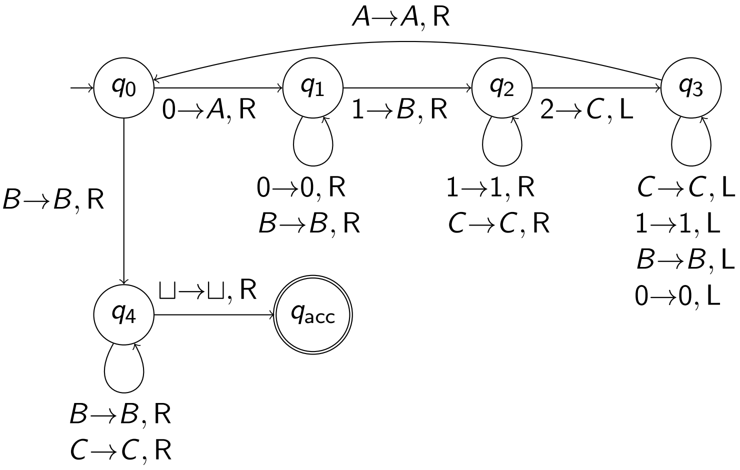
High level description

```
On input string w
    while there are unmarked 0s, do
        Mark the left most 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 $\{0^n1^n2^n \mid n > 0\}$

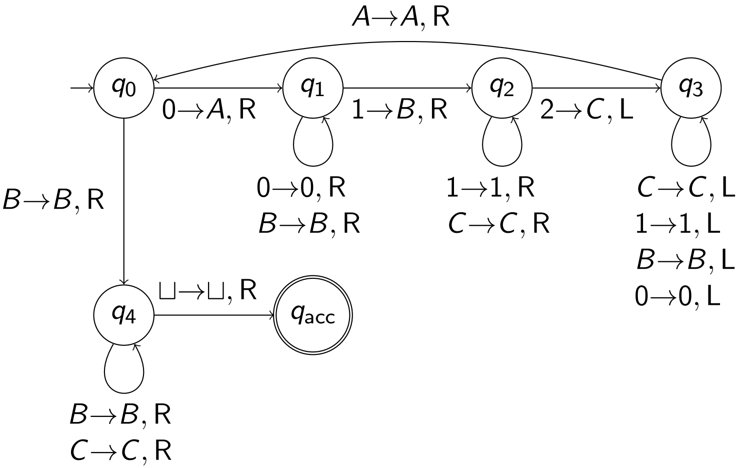


Example 2: TM for $\{0^n 1^n 2^n \mid n > 0\}$



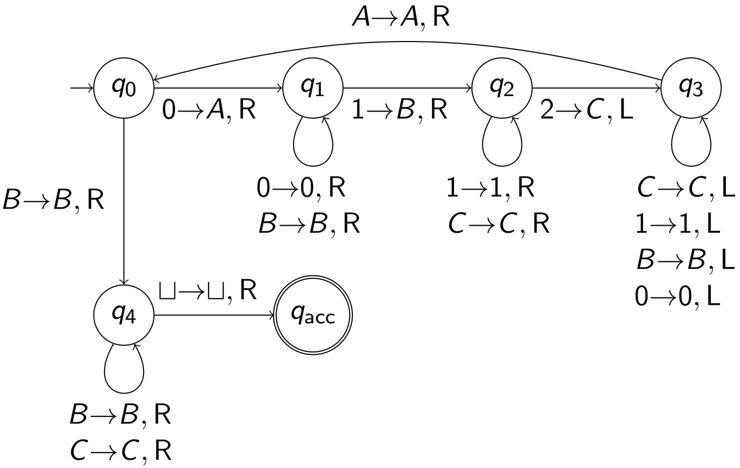
e.g.: $q_0 001122 \vdash^* A0Bq_3 1C2$

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



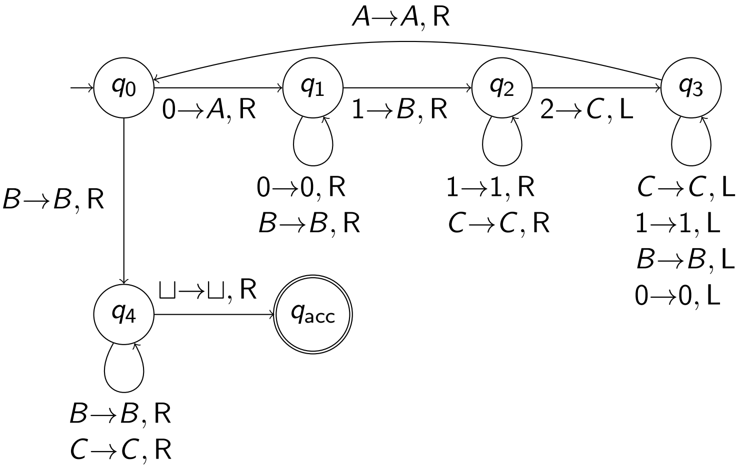
e.g.: $q_0 001122 \vdash^* A0Bq_3 1C2 \vdash^* q_3 A0B1C2$

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



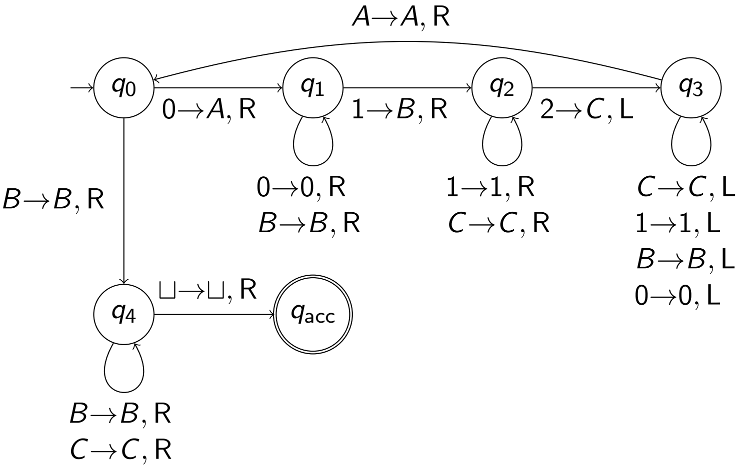
e.g.: $q_0 001122 \vdash^* A0Bq_3 1C2 \vdash^* q_3 A0B1C2 \vdash Aq_0 0B1C2$

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



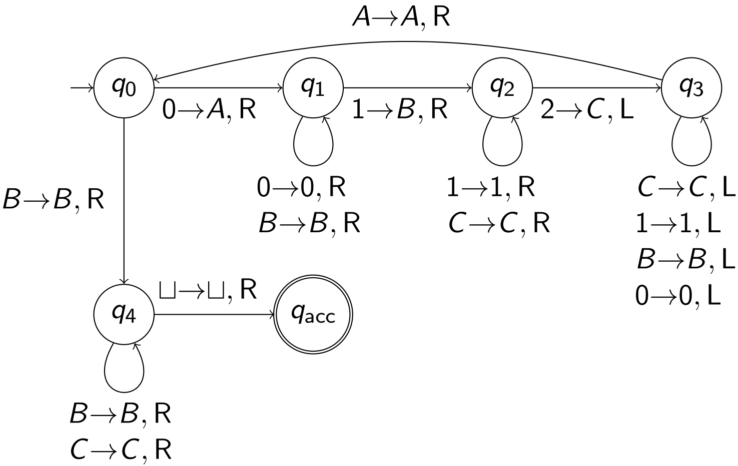
e.g.: $q_0001122 \vdash^* A0Bq_31C2 \vdash^* q_3A0B1C2 \vdash Aq_00B1C2$
$\vdash^* AAq_0BBCC$

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



e.g.: $q_0 001122 \vdash^* A0Bq_3 1C2 \vdash^* q_3 A0B1C2 \vdash Aq_0 0B1C2$
$\vdash^* AAq_0 BBCC \vdash^* AABBCCq_4 \sqcup$

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



e.g.: $q_0 001122 \vdash^* A0Bq_3 1C2 \vdash^* q_3 A0B1C2 \vdash Aq_0 0B1C2$
$\vdash^* AAq_0 BBCC \vdash^* AABBCCq_4 \sqcup \vdash AABBCC\sqcup q_{\mathrm{acc}} \sqcup$

Deciding a Language

Deciding a Language

- Only halting configurations are those with state q_{acc} or q_{rej}

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

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
- Then the machine does not accept that input

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
- Then the machine does not accept that input
- So two ways to not accept: reject or never halt

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
- Then the machine does not accept that input
- So two ways to not accept: reject or never halt

Definition

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

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
- Then the machine does not accept that input
- So two ways to not accept: reject or never halt

Definition

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

Deciding a language is more than recognizing it.

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
- Then the machine does not accept that input
- So two ways to not accept: reject or never halt

Definition

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

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