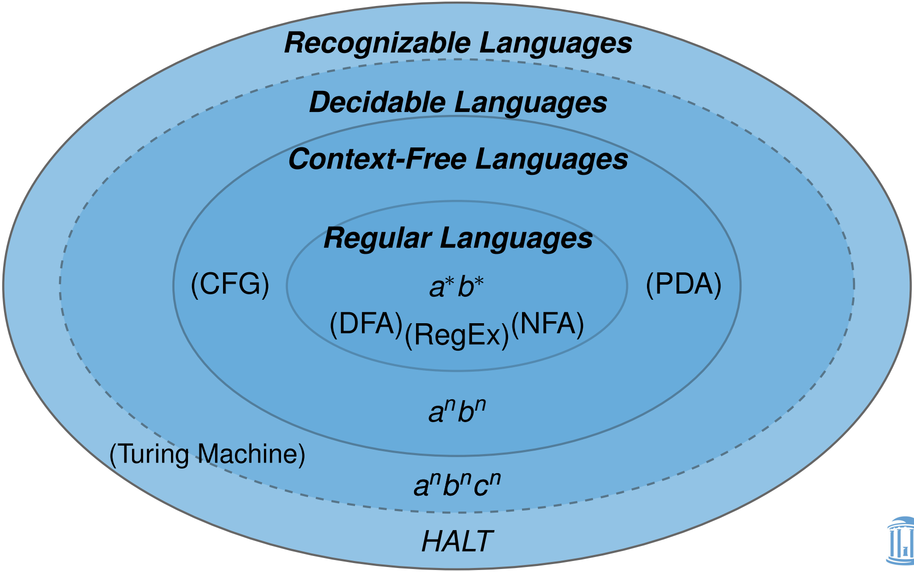
(Un)decidability, (Un)recognizability, and Reductions

Models of Languages and Computation

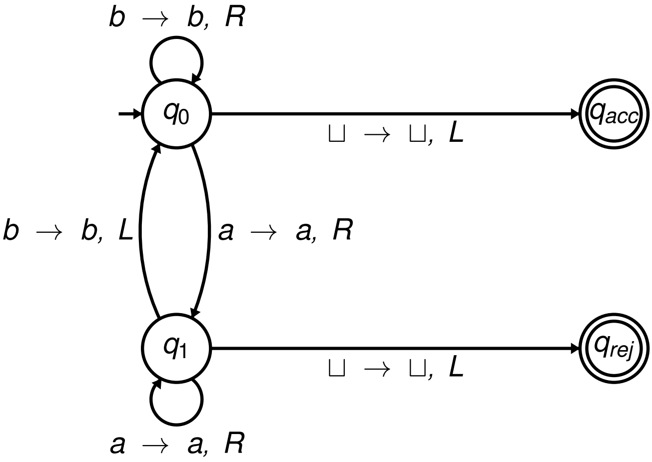
Overview

1. **Clarifying Terms**
   - Decides vs Recognizes (Machines)
   - Decidable vs Recognizable (Languages)
   - Mapping Reduction
2. **Example Proofs**
   - Reduce One Decidable Language to Another
   - $A_{TM} \leq_m REGULAR_{TM}$
   - $HALT \leq_m VST$
   - $A_{TM} \leq_m NE_{TM}$
   - L_d is not recognizable
   - $L_d \leq_m E_{TM}$

Languages So Far



Decides vs Recognizes (Machines)



Decides vs Recognizes (Machines)

Definition (Decides)

A Turing machine M ***decides*** a language L if

- For every string $w \in L$, M accepts w, and
- For every string $w \notin L$, M rejects w.

Decides vs Recognizes (Machines)

Definition (Decides)

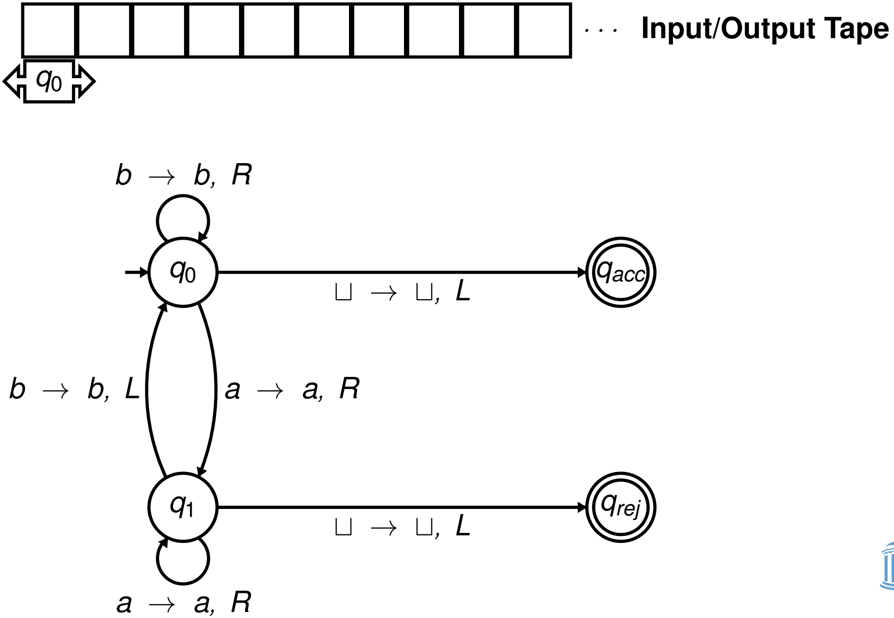
A Turing machine M ***decides*** a language L if

- For every string $w \in L$, M accepts w, and
- For every string $w \notin L$, M rejects w.

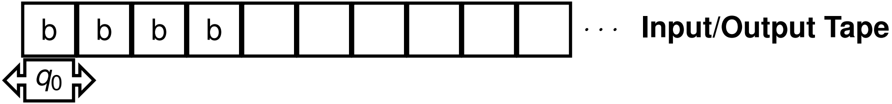
Definition (Recognizes)

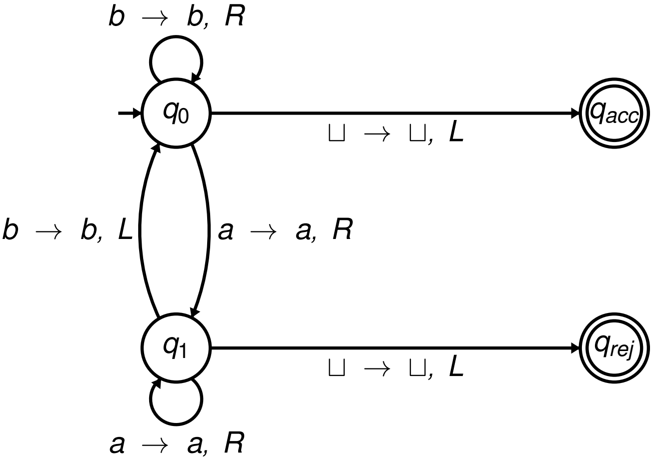
A Turing machine M ***recognizes*** a language L if

- For every string $w \in L$, M accepts w, and
- For every string $w \notin L$, M either rejects or goes into an infinite loop on w.

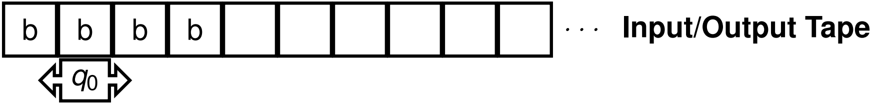

Input/Output Tape
q0
b → b, R
q0
⊔ → ⊔, L
qacc
b → b, L
a → a, R
q1
⊔ → ⊔, L
qrej
a → a, R

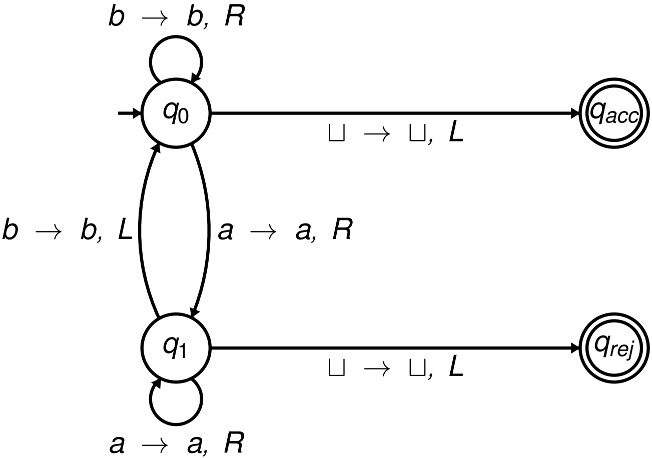
- What happens to the input bbbb?



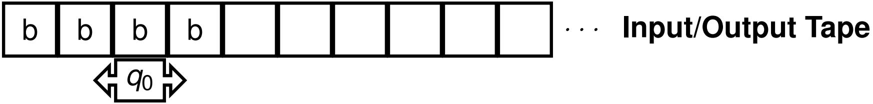


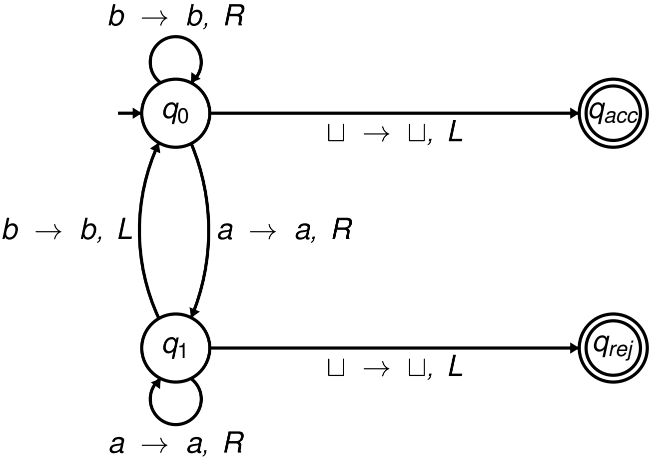
- What happens to the input bbbb?



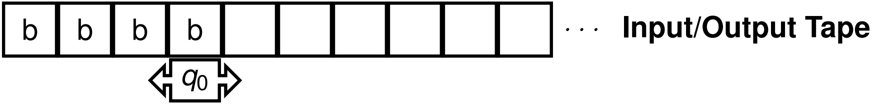


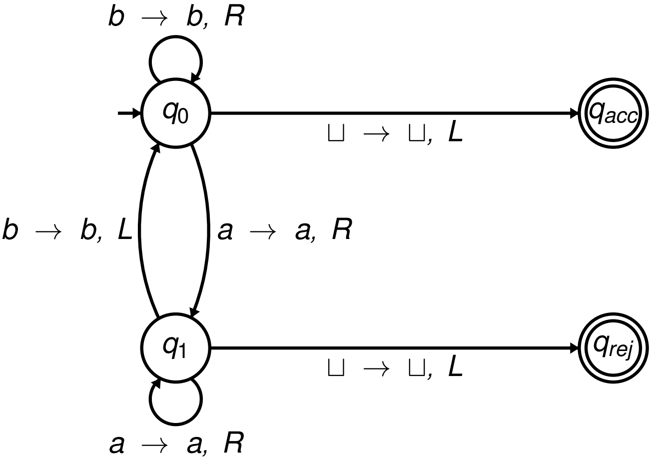
- What happens to the input bbbb?



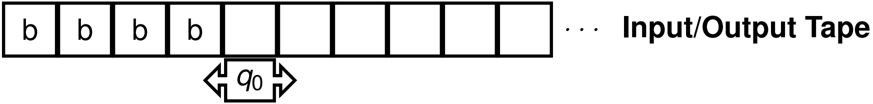


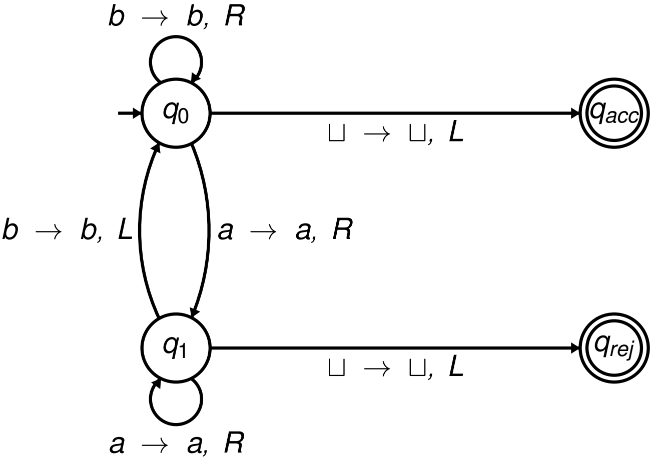
- What happens to the input bbbb?



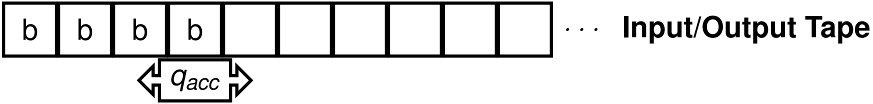


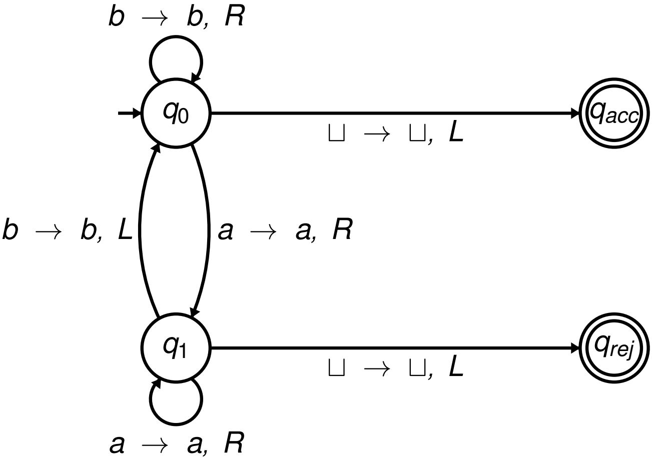
- What happens to the input bbbb?



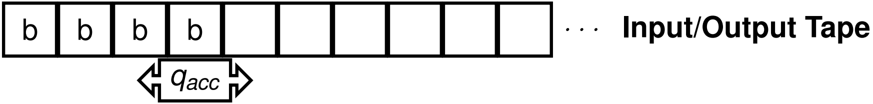


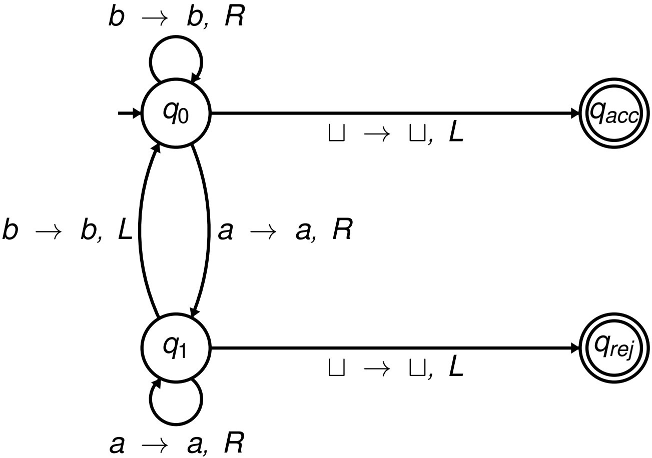
- What happens to the input bbbb?



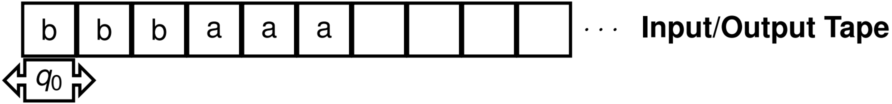


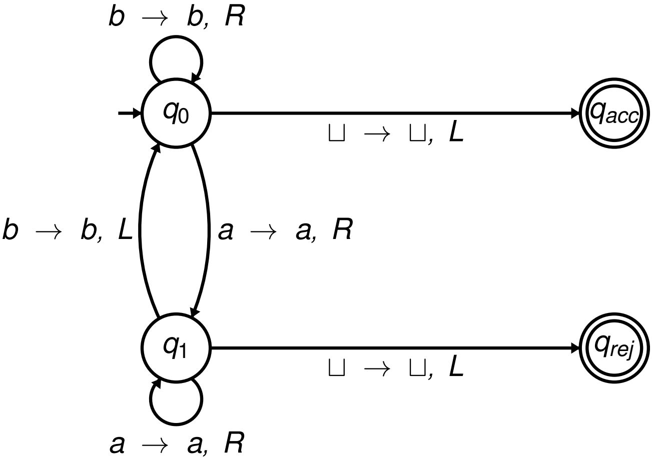
- What happens to the input bbbb? We accept it



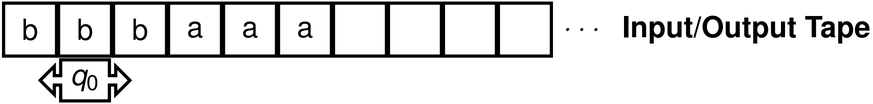


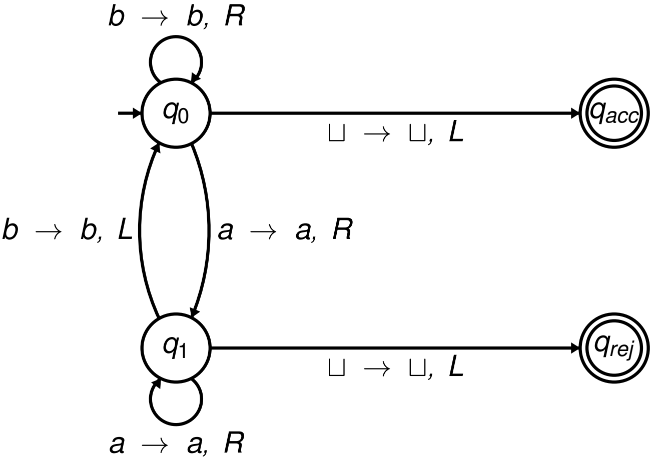
- What happens to the input bbbaaa?



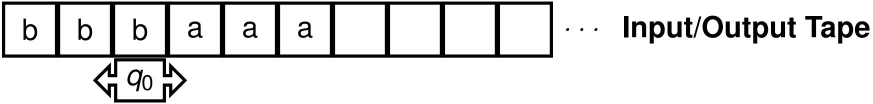


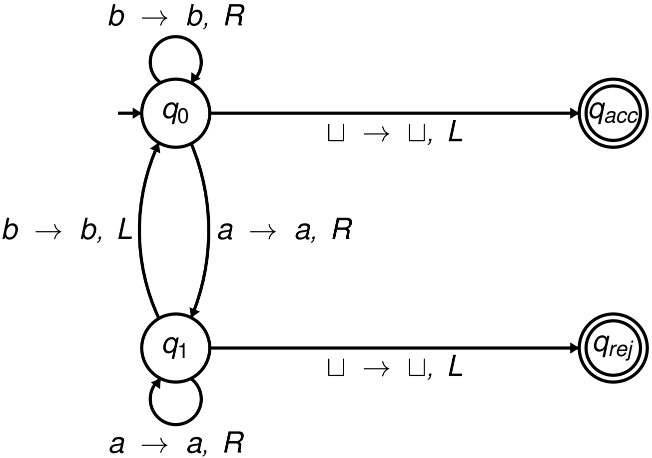
- What happens to the input bbbaaa?



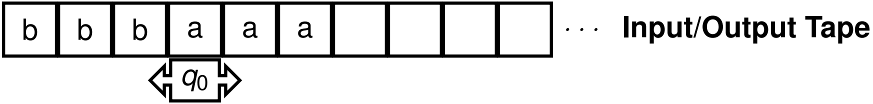


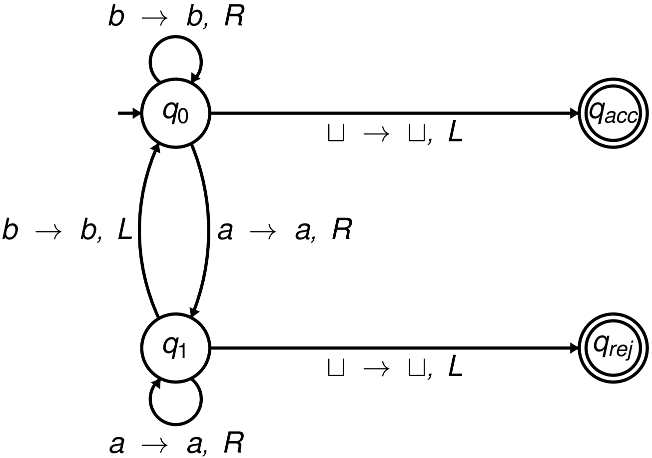
- What happens to the input bbbaaa?



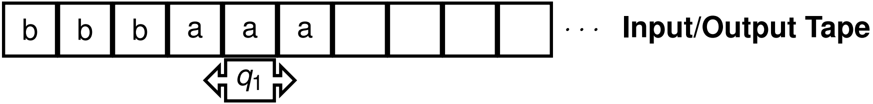


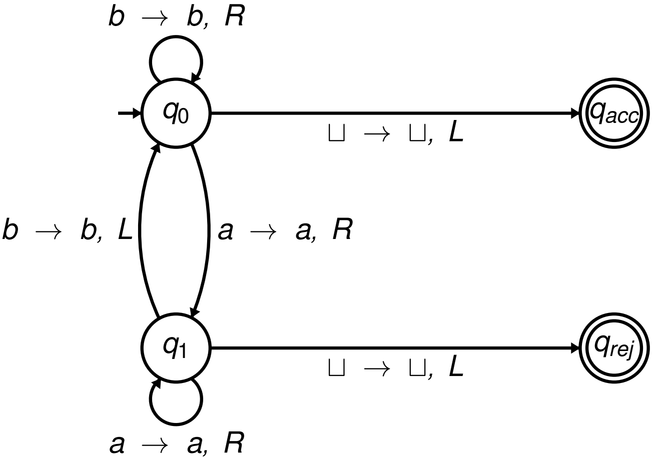
- What happens to the input bbbaaa?



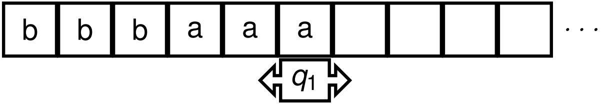


- What happens to the input bbbaaa?

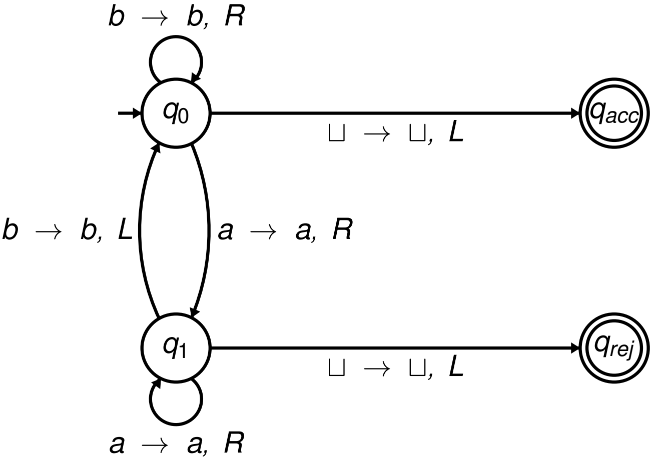




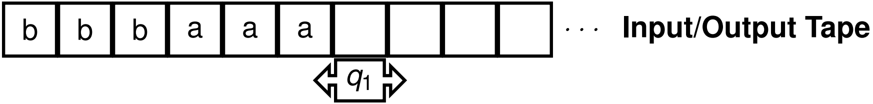
- What happens to the input bbbaaa?

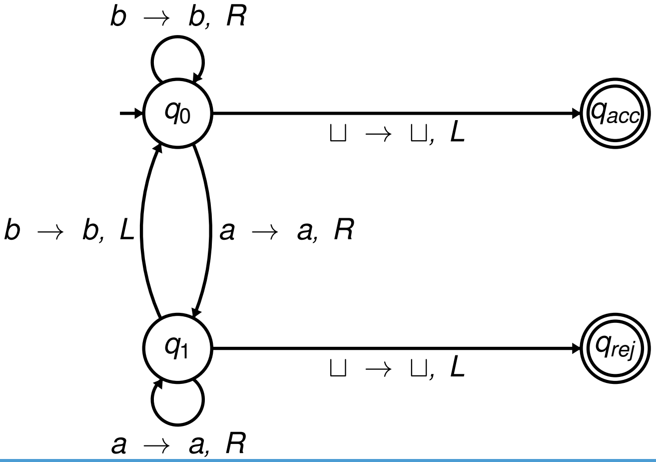


Input/Output Tape

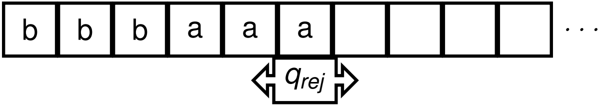


- What happens to the input bbbaaa?

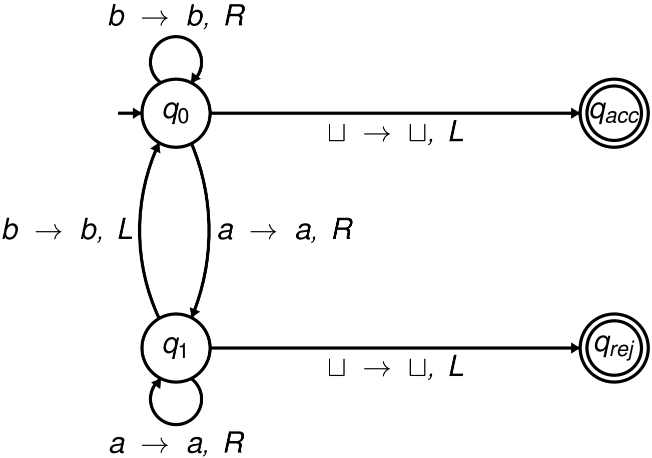




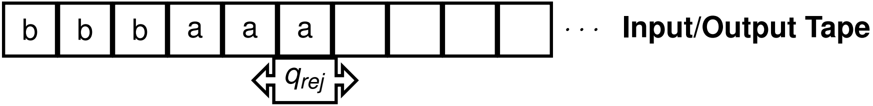
- What happens to the input bbbaaa?

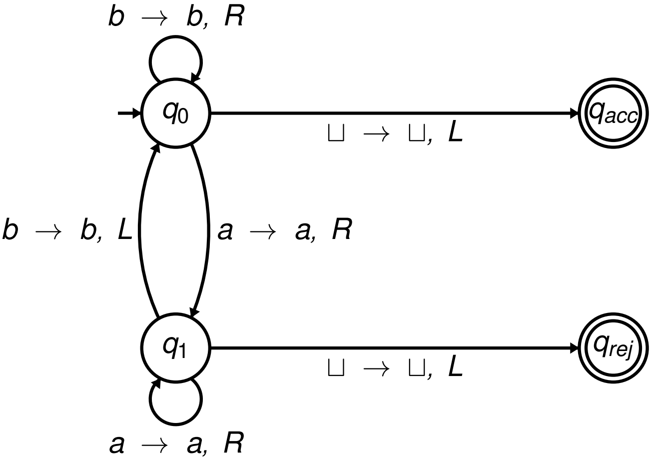


Input/Output Tape

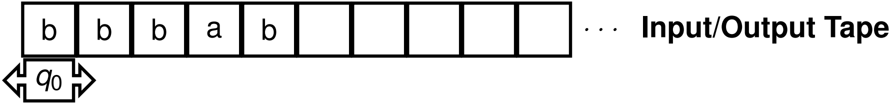


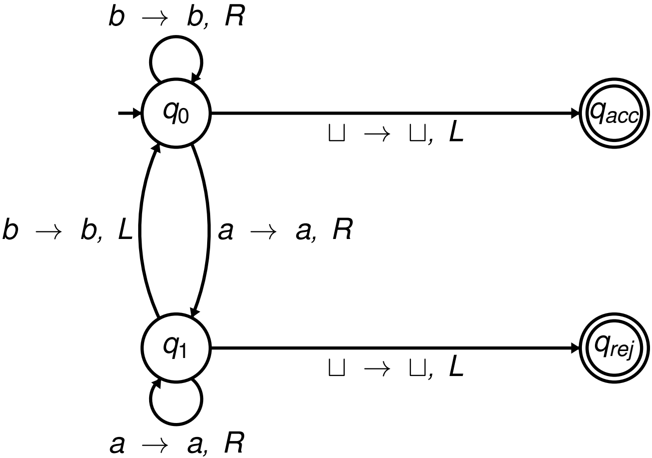
- What happens to the input bbbaaa? We reject it



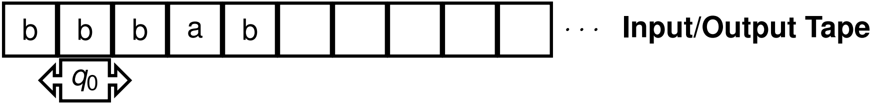


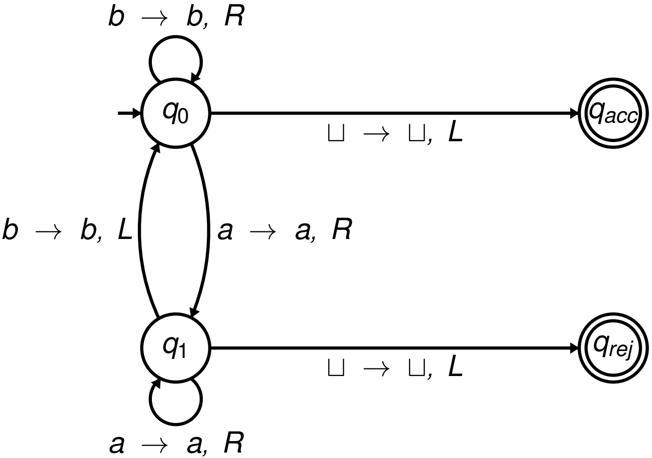
- What happens to the input bbbab?



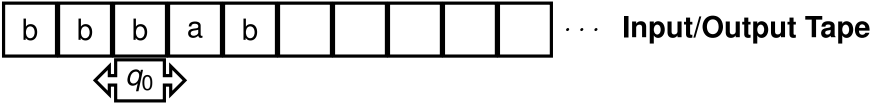


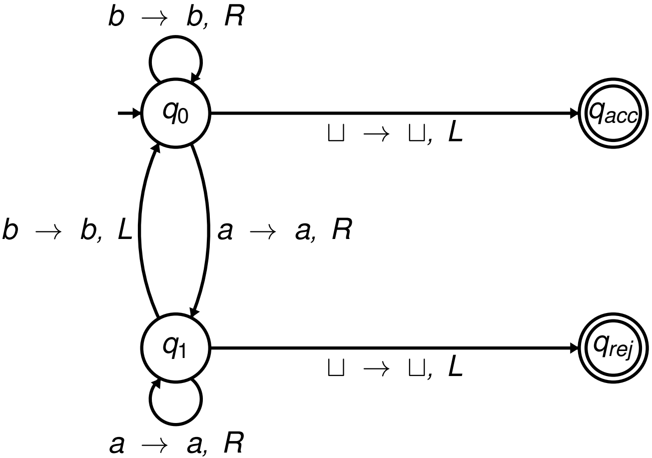
- What happens to the input bbbab?



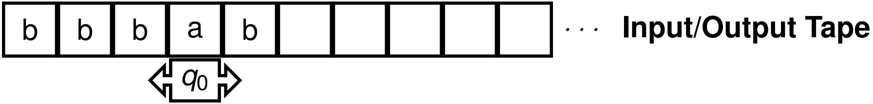


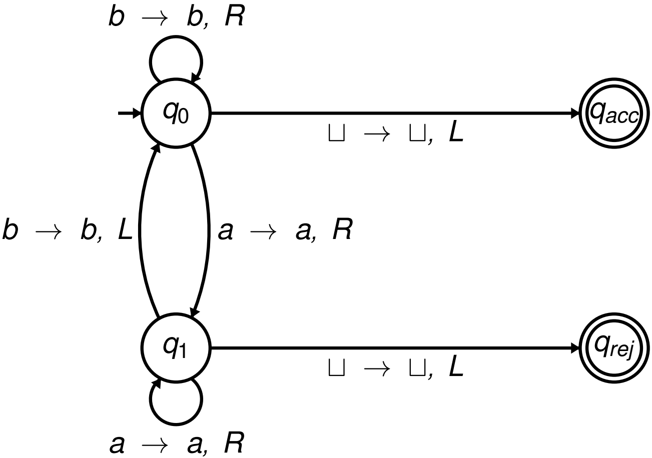
- What happens to the input bbbab?



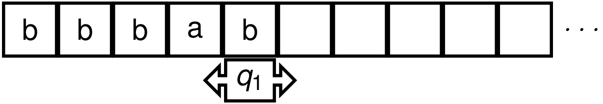


- What happens to the input bbbab?

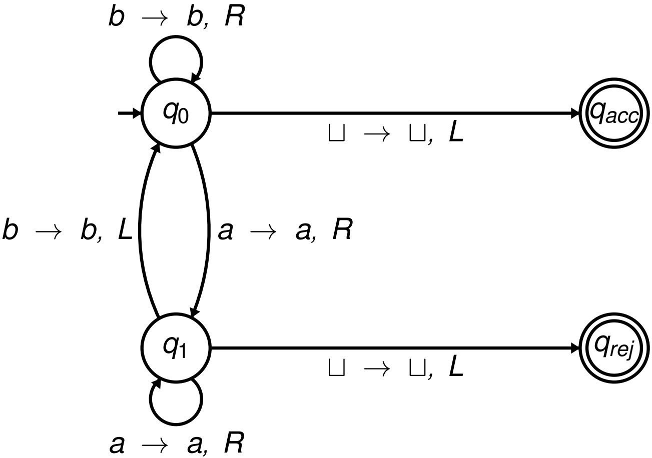




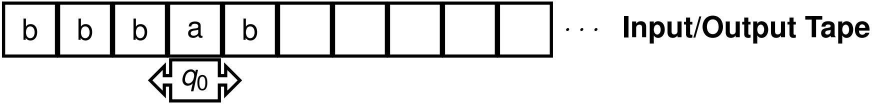
- What happens to the input bbbab?

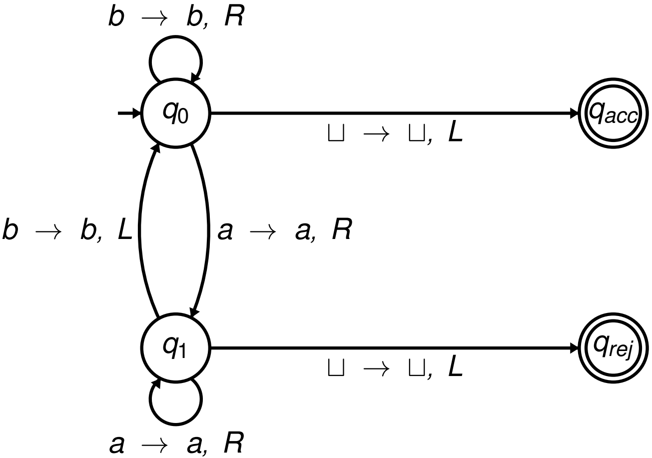


Input/Output Tape

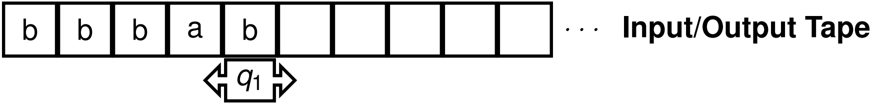


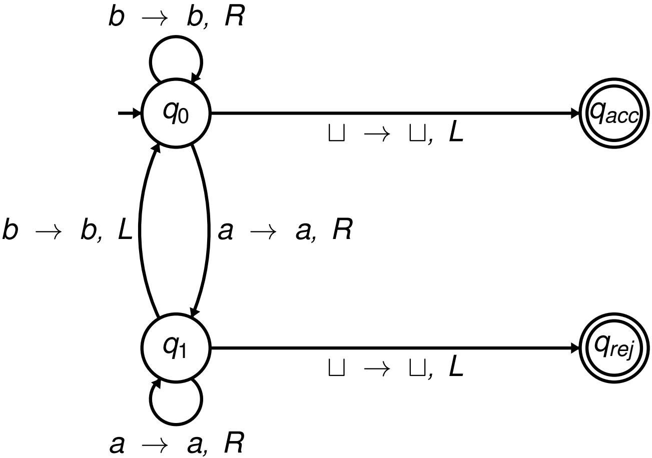
- What happens to the input bbbab?



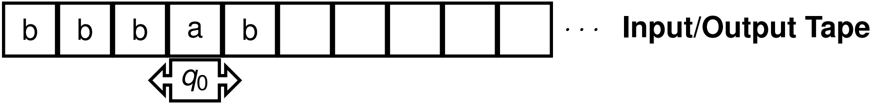


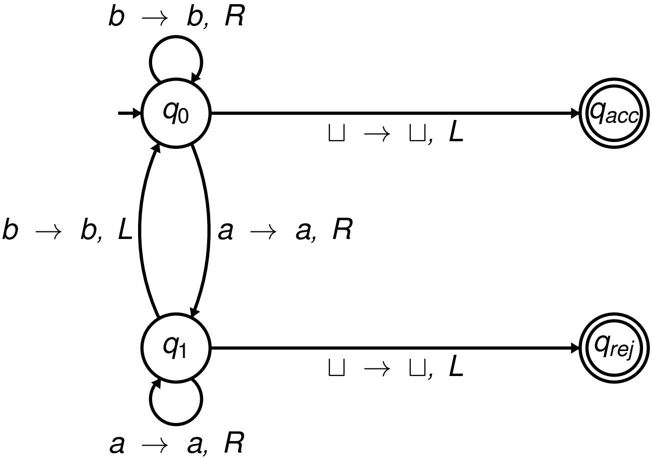
- What happens to the input bbbab?



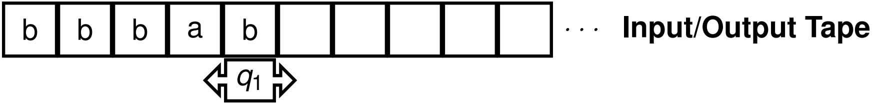


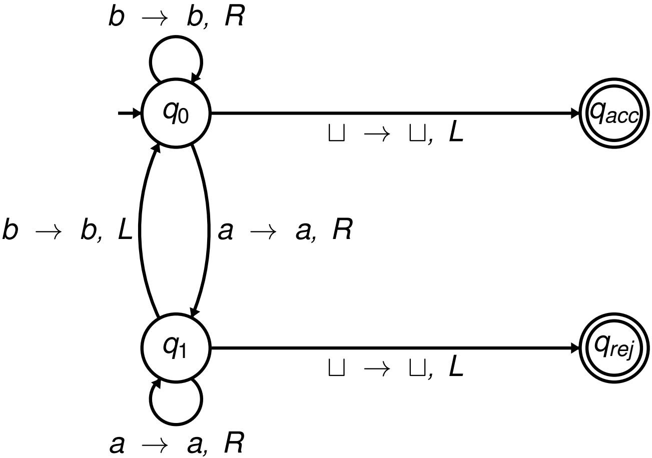
- What happens to the input bbbab?



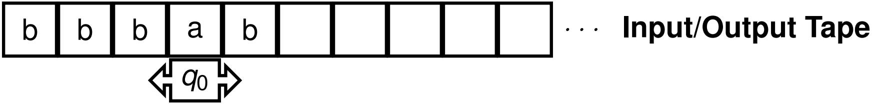


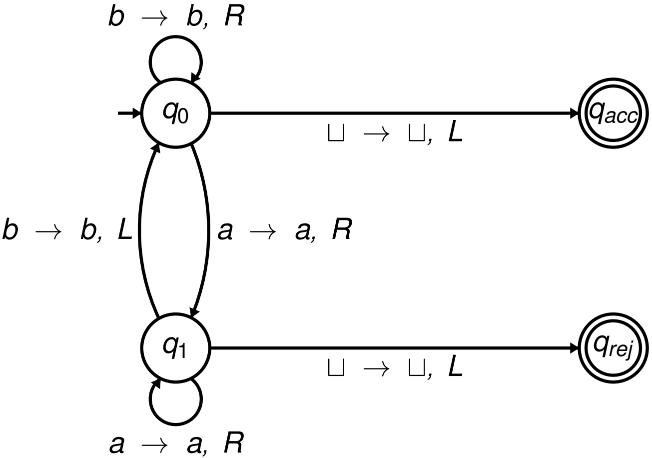
- What happens to the input bbbab?



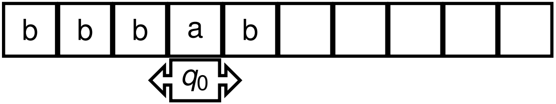


- What happens to the input bbbab?

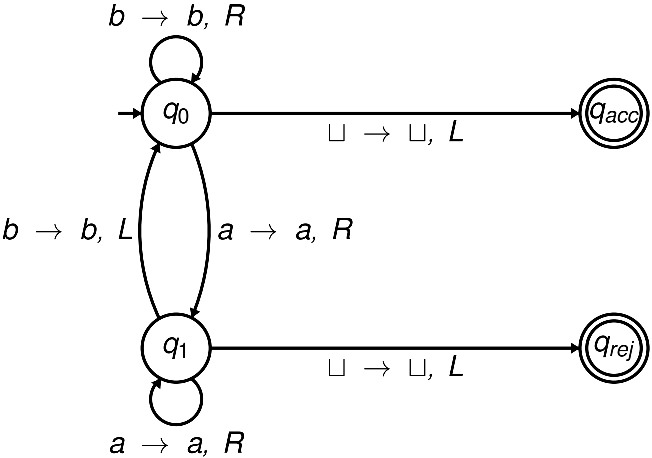




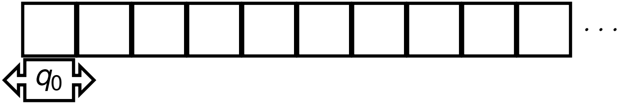
- What happens to the input bbbab? We get stuck in an infinite loop



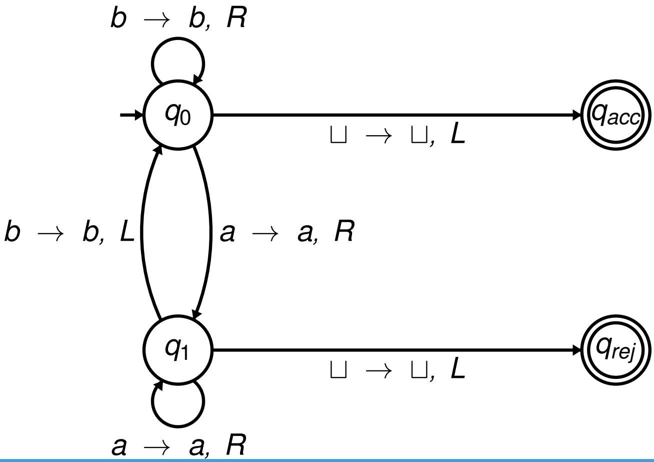
. . . **Input/Output Tape**



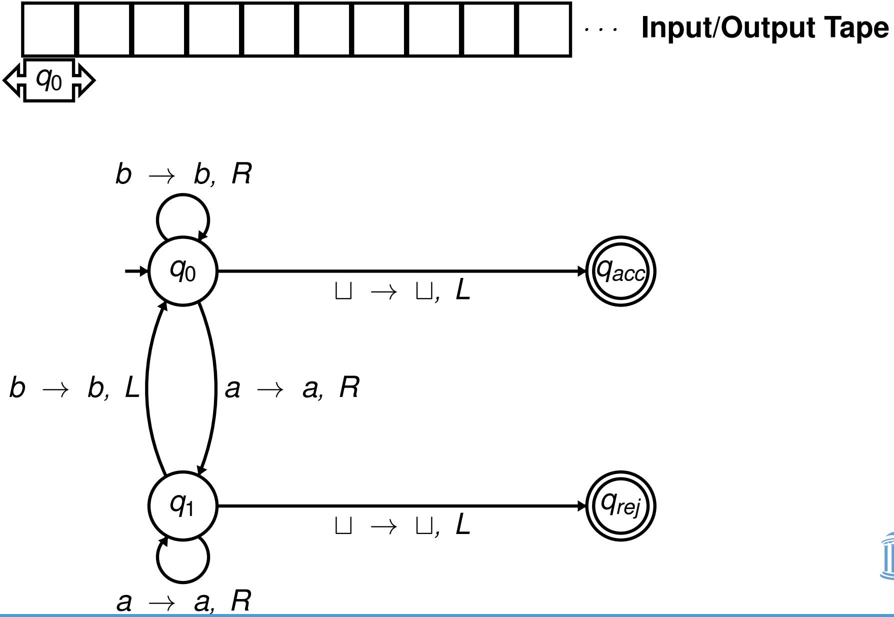
- What language does this machine recognize?



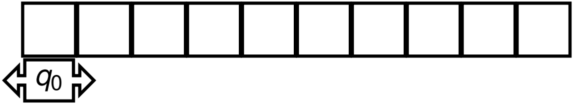
Input/Output Tape



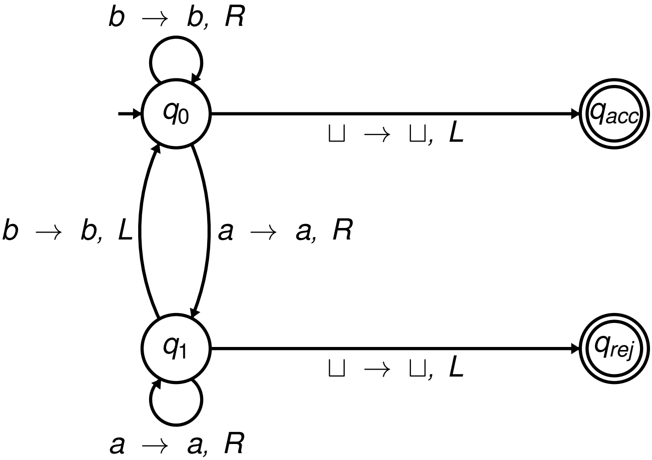
- What language does this machine recognize? $\{b^n \mid n \geq 0\}$



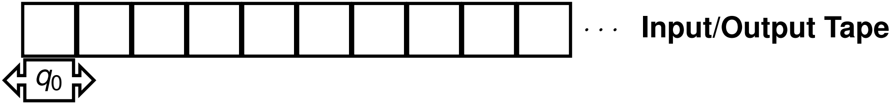
- Is this machine a decider?

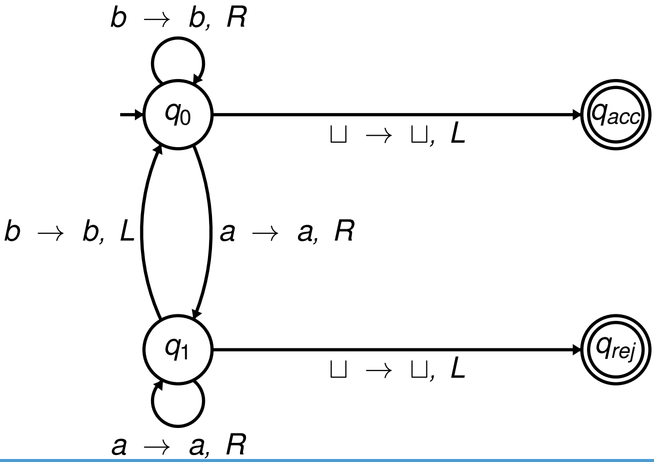


Input/Output Tape

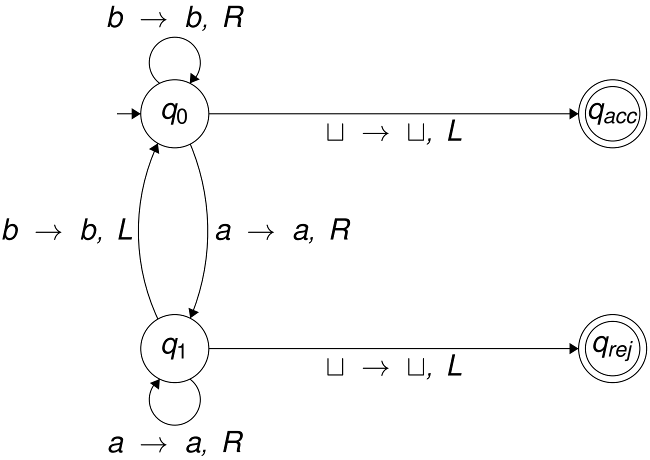


- Is this machine a decider? No, since it loops infinitely on bbbab

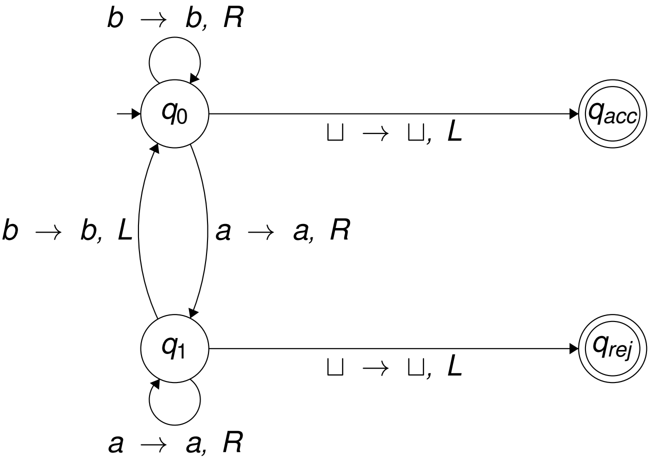




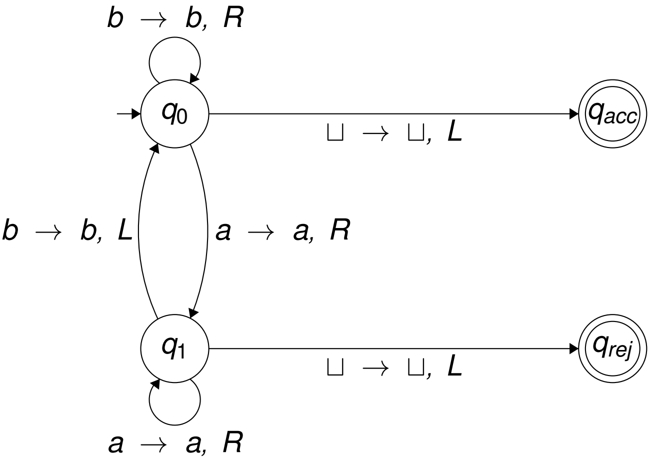
- This machine does ***not decide*** $\{b^n \mid n \geq 0\}$



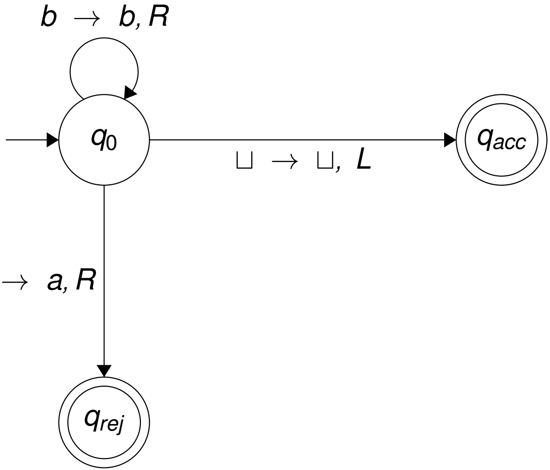
- This machine does ***not decide*** $\{b^n \mid n \geq 0\}$
- This machine does ***recognize*** $\{b^n \mid n \geq 0\}$



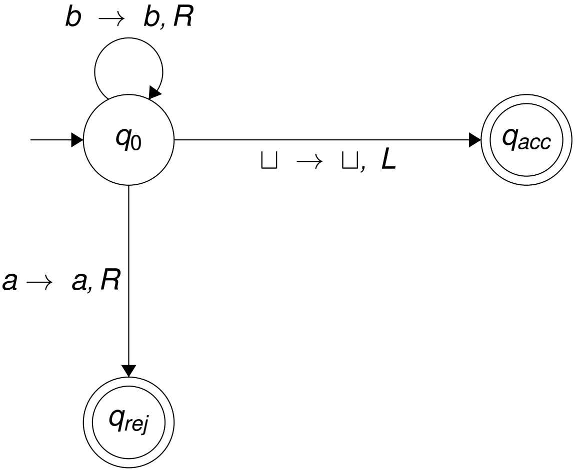
- This machine does ***not decide*** $\{b^n \mid n \geq 0\}$
- This machine does ***recognize*** $\{b^n \mid n \geq 0\}$
- IMPORTANT: Even though this particular machine does not decide $\{b^n \mid n \geq 0\}$, some other machine might decide it



- The following machine ***decides*** $\{b^n \mid n \geq 0\}$



- The following machine ***decides*** $\{b^n \mid n \geq 0\}$
- So, $\{b^n \mid n \geq 0\}$ is a ***decidable language***



Decidable vs Recognizable (Languages)

Decidable vs Recognizable (Languages)

Definition (Decidable)

A language L is ***decidable*** if and only if <u>there exists</u> a Turing machine M that ***decides*** L.

Decidable vs Recognizable (Languages)

Definition (Decidable)

A language L is ***decidable*** if and only if there exists a Turing machine M that ***decides*** L.

Definition (Recognizable)

A language L is ***recognizable*** if and only if there exists a Turing machine M that ***recognizes*** L.

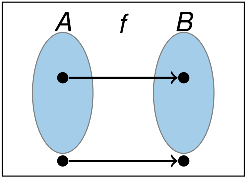
Mapping Reduction

Mapping Reduction

What $A \leq_m B$ means conceptually:

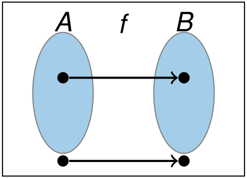
Mapping Reduction

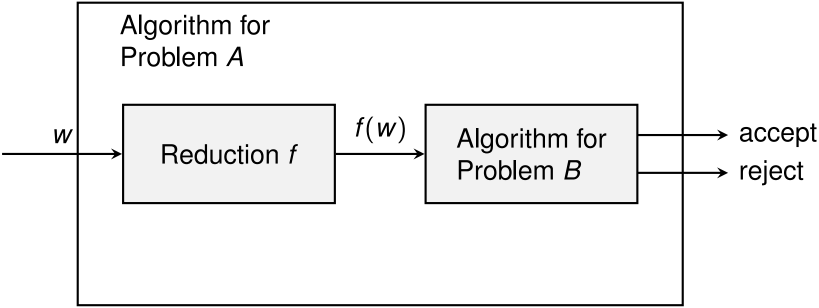
What $A \leq_m B$ means conceptually:



Mapping Reduction

What $A \leq_m B$ means conceptually:

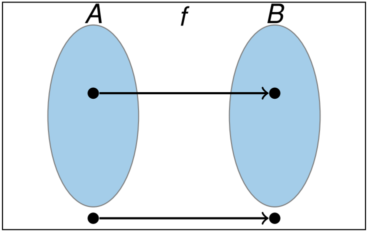




Mapping Reduction

Definition

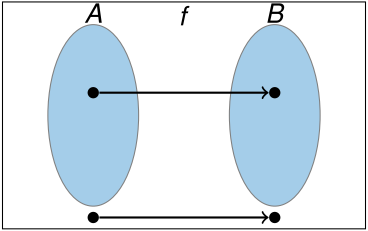
A function $f : \Sigma^* \to \Sigma^*$ is ***computable*** if there exists a Turing machine M that on every input w, halts with $f(w)$ on the tape.



Mapping Reduction

Definition

A ***mapping reduction*** from a language A to a language B is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \iff f(w) \in B$. We say that A is **reducible to** B, and we denote it by $A \leq_m B$.

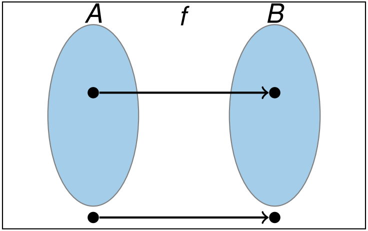


Mapping Reduction

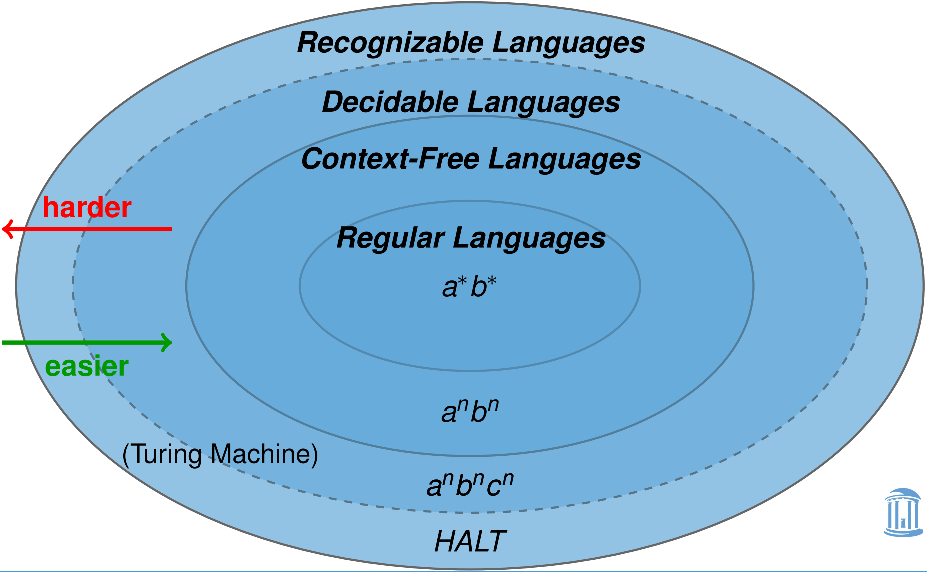
Definition

A ***mapping reduction*** from a language A to a language B is a computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in A \iff f(w) \in B$. We say that A is **reducible to** B, and we denote it by $A \leq_m B$.

- Informally, $A \leq_m B$ means “Problem A is no harder than B," and all decidable problems are equally difficult under mapping reductions (e.g., regular vs. context-free languages).



Mapping Reductions



Suppose we have $X \leq_m Y$

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is ___

Options:

A: Decidable

B: Undecidable

C: Recognizable

D: Unrecognizable

W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**

Options:

A: Decidable

B: Undecidable

C: Recognizable

D: Unrecognizable

W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is ___

Options:

- A: Decidable
- B: Undecidable
- C: Recognizable
- D: Unrecognizable
- W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**

Options:

A: Decidable

B: Undecidable

C: Recognizable

D: Unrecognizable

W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is ___

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is ___

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**

Options:

A: Decidable

B: Undecidable

C: Recognizable

D: Unrecognizable

W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is ___

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is **(W)**

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is **(W)**
6. Y is recognizable. X is ___

Options:

- A: Decidable
- B: Undecidable
- C: Recognizable
- D: Unrecognizable
- W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is **(W)**
6. Y is recognizable. X is **(C)**

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is **(W)**
6. Y is recognizable. X is **(C)**
7. X is unrecognizable. Y is ___

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is **(W)**
6. Y is recognizable. X is **(C)**
7. X is unrecognizable. Y is **(D)**

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is **(W)**
6. Y is recognizable. X is **(C)**
7. X is unrecognizable. Y is **(D)**
8. Y is unrecognizable. X is ___

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Suppose we have $X \leq_m Y$

Conclusion (choose the smallest one that must be true):

1. X is decidable. Y is **(W)**
2. Y is decidable. X is **(A)**
3. X is undecidable. Y is **(B)**
4. Y is undecidable. X is **(W)**
5. X is recognizable. Y is **(W)**
6. Y is recognizable. X is **(C)**
7. X is unrecognizable. Y is **(D)**
8. Y is unrecognizable. X is **(W)**

Options:

A: Decidable
B: Undecidable
C: Recognizable
D: Unrecognizable
W: Cannot tell

Decidable Languages: Mapping Reduction Example

Theorem

Let

$$X = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is even}\}$$

and

$$Y = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is divisible by 6}\}.$$

Then $X \leq_m Y$.

Decidable Languages: Mapping Reduction Example

Theorem

Let

$$X = \{w \in \{0,1\}^* \mid \text{\#0s in } w \text{ is even}\}$$

and

$$Y = \{w \in \{0,1\}^* \mid \text{\#0s in } w \text{ is divisible by 6}\}.$$

Then $X \leq_m Y$.

Proof Idea.

Decidable Languages: Mapping Reduction Example

Theorem

Let

$$X = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is even}\}$$

and

$$Y = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is divisible by 6}\}.$$

Then $X \leq_m Y$.

Proof Idea.

- We will define a mapping $f : \{0,1\}^* \to \{0,1\}^*$ such that $w \in X \iff f(w) \in Y$

Decidable Languages: Mapping Reduction Example

Theorem

Let

$$X = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is even}\}$$

and

$$Y = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is divisible by 6}\}.$$

Then $X \leq_m Y$.

Proof Idea.

- We will define a mapping $f : \{0,1\}^* \to \{0,1\}^*$ such that $w \in X \iff f(w) \in Y$
- Idea: for each 0 in w, remove it from its position and append 3 zeros at the end; leave 1's unchanged

Decidable Languages: Mapping Reduction Example

Theorem

Let

$$X = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is even}\}$$

and

$$Y = \{w \in \{0,1\}^* \mid \textit{\#0s in } w \textit{ is divisible by 6}\}.$$

Then $X \leq_m Y$.

Proof Idea.

- We will define a mapping $f : \{0,1\}^* \to \{0,1\}^*$ such that $w \in X \iff f(w) \in Y$
- Idea: for each 0 in w, remove it from its position and append 3 zeros at the end; leave 1's unchanged (e.g., 0101 $\to$ 11000000)

Mapping Reduction: Even 0s $\to$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

Mapping Reduction: Even 0s $\to$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

Mapping Reduction: Even 0s $\rightarrow$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$

Mapping Reduction: Even 0s $\to$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).

Mapping Reduction: Even 0s $\rightarrow$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot 2k = 6k$, which is divisible by 6.

Mapping Reduction: Even 0s $\rightarrow$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot 2k = 6k$, which is divisible by 6.
  - So, $f(w) \in Y$.

Mapping Reduction: Even 0s $\rightarrow$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot 2k = 6k$, which is divisible by 6.
  - So, $f(w) \in Y$.
- If $w \notin X$

Mapping Reduction: Even 0s $\to$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot 2k = 6k$, which is divisible by 6.
  - So, $f(w) \in Y$.
- If $w \notin X$
  - w has an odd number of 0's ($2k + 1$ for some $k \in \mathbb{Z}$).

Mapping Reduction: Even 0s $\rightarrow$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot 2k = 6k$, which is divisible by 6.
  - So, $f(w) \in Y$.
- If $w \notin X$
  - w has an odd number of 0's ($2k + 1$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot (2k + 1) = 6k + 3$, which is not divisible by 6.

Mapping Reduction: Even 0s $\rightarrow$ #0s divisible by 6

We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot 2k = 6k$, which is divisible by 6.
  - So, $f(w) \in Y$.
- If $w \notin X$
  - w has an odd number of 0's ($2k + 1$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot (2k + 1) = 6k + 3$, which is not divisible by 6.
  - So, $f(w) \notin Y$.

Mapping Reduction: Even 0s $\rightarrow$ #0s divisible by 6

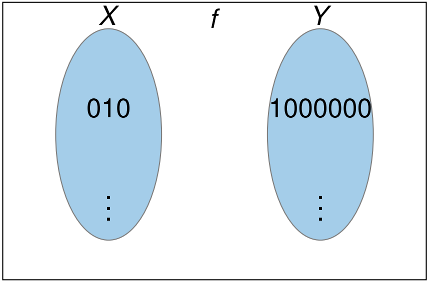
We will map the input w to the string obtained by replacing each 0 in w with three 0s at the end and leaving 1's unchanged.

- If $w \in X$
  - w has an even number of 0's ($2k$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot 2k = 6k$, which is divisible by 6.
  - So, $f(w) \in Y$.
- If $w \notin X$
  - w has an odd number of 0's ($2k + 1$ for some $k \in \mathbb{Z}$).
  - The #0s in $f(w)$ is $3 \cdot (2k + 1) = 6k + 3$, which is not divisible by 6.
  - So, $f(w) \notin Y$.

Hence, $w \in X \iff f(w) \in Y$.

Some Properties of the Reduction f

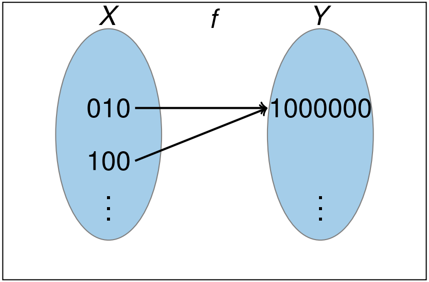
The previous proof is complete, but I just want to emphasize some things about f:



Some Properties of the Reduction f

The previous proof is complete, but I just want to emphasize some things about f:

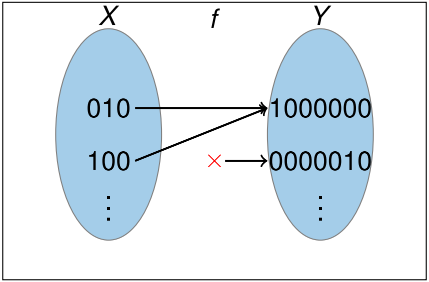
1. f does not need to be one-to-one (e.g., 010 and 100 both map to 1000000)



Some Properties of the Reduction f

The previous proof is complete, but I just want to emphasize some things about f:

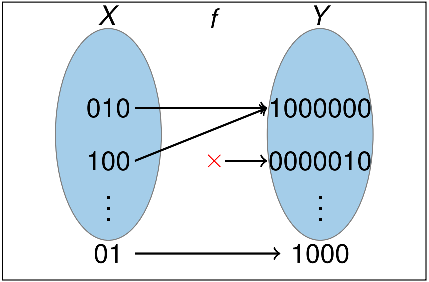
1. f does not need to be one-to-one (e.g., 010 and 100 both map to 1000000)
2. f does not need to be onto (e.g., nothing maps to 0000010 even though it has 6 zeros)



Some Properties of the Reduction f

The previous proof is complete, but I just want to emphasize some things about f:

1. f does not need to be one-to-one (e.g., 010 and 100 both map to 1000000)
2. f does not need to be onto (e.g., nothing maps to 0000010 even though it has 6 zeros)
3. We need: All elements in X map to elements in Y, and all elements not in X map outside Y



Undecidability of `REGULAR`

Theorem

The language

$$REGULAR = \{\langle M \rangle \mid L(M) \textit{ is regular}\}$$

is undecidable.

Undecidability of REGULAR

Theorem

The language

$$REGULAR = \{\langle M \rangle \mid L(M) \textit{ is regular}\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ to $REGULAR$.

Undecidability of REGULAR

Theorem

The language

$$REGULAR = \{\langle M \rangle \mid L(M) \text{ is regular}\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to $REGULAR$.
- We want the inputs of f to be of the form:

Undecidability of REGULAR

Theorem

The language

$$REGULAR = \{\langle M \rangle \mid L(M) \text{ is regular}\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ to *REGULAR*.
- We want the inputs of f to be of the form:
  - $\langle M, w\rangle$ where M is a Turing machine and w is a string

Undecidability of REGULAR

Theorem

The language

$$REGULAR = \{\langle M \rangle \mid L(M) \text{ is regular}\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to *REGULAR*.
- We want the inputs of f to be of the form:
  - $\langle M, w \rangle$ where M is a Turing machine and w is a string (If the input was not of the right form, we could just trivially map it to something not in *REGULAR*)

Undecidability of REGULAR

Theorem

The language

$$REGULAR = \{\langle M \rangle \mid L(M) \text{ is regular}\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to *REGULAR*.
- We want the inputs of f to be of the form:
  - $\langle M, w \rangle$ where M is a Turing machine and w is a string (If the input was not of the right form, we could just trivially map it to something not in *REGULAR*)
- We want the outputs of f to be of the form:

Undecidability of REGULAR

Theorem

The language

$$REGULAR = \{\langle M \rangle \mid L(M) \text{ is regular}\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to *REGULAR*.
- We want the inputs of f to be of the form:
  - $\langle M, w \rangle$ where M is a Turing machine and w is a string (If the input was not of the right form, we could just trivially map it to something not in *REGULAR*)
- We want the outputs of f to be of the form:
  - $\langle N \rangle$ where N is a Turing machine

Undecidability of REGULAR

Theorem

The language

$$REGULAR = \{\langle M\rangle \mid L(M) \text{ is regular}\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ to *REGULAR*.
- We want the inputs of f to be of the form:
  - $\langle M, w\rangle$ where M is a Turing machine and w is a string (If the input was not of the right form, we could just trivially map it to something not in *REGULAR*)
- We want the outputs of f to be of the form:
  - $\langle N\rangle$ where N is a Turing machine
- Again, we want $\langle M, w\rangle \in A_{TM} \iff f(\langle M, w\rangle) \in REGULAR$

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

Construction of N for $f(\langle M, w\rangle)$

We will map the input $\langle M, w\rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

Construction of N for $f(\langle M, w\rangle)$

We will map the input $\langle M, w\rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w\rangle \in A_{TM}$

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.
  - So, $\langle N \rangle \in REGULAR$.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.
  - So, $\langle N \rangle \in REGULAR$.
- If $\langle M, w \rangle \notin A_{TM}$

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.
  - So, $\langle N \rangle \in REGULAR$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.
  - So, $\langle N \rangle \in REGULAR$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N accepts strings if and only if they are of the form $0^n 1^n$.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.
  - So, $\langle N \rangle \in REGULAR$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N accepts strings if and only if they are of the form $0^n 1^n$. So, $L(N) = \{0^n 1^n \mid n \geq 0\}$, which is not regular.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.
  - So, $\langle N \rangle \in REGULAR$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N accepts strings if and only if they are of the form $0^n 1^n$. So, $L(N) = \{0^n 1^n \mid n \geq 0\}$, which is not regular.
  - So, $\langle N \rangle \notin REGULAR$.

Construction of N for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to the following Turing machine N:

```
On input x:
    If x is of the form 0^n 1^n then
        accept x
    else
        run M on w
        accept x if and only if M accepts w
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is regular.
  - So, $\langle N \rangle \in REGULAR$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N accepts strings if and only if they are of the form $0^n 1^n$. So, $L(N) = \{0^n 1^n \mid n \geq 0\}$, which is not regular.
  - So, $\langle N \rangle \notin REGULAR$.
- Hence, $\langle M, w \rangle \in A_{TM} \iff \langle N \rangle \in REGULAR$

Undecidability of *VST*

Theorem

The language

$$VST = \{\langle M, w, q\rangle \mid \textit{Turing Machine } M \textit{ visits state } q \textit{ on input } w\}$$

is undecidable.

Undecidability of *VST*

Theorem

The language

$$VST = \{\langle M, w, q\rangle \mid \textit{Turing Machine } M \textit{ visits state } q \textit{ on input } w\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from HALT $= \{\langle M, w\rangle \mid M$ halts on $w\}$ to VST (i.e., $HALT \leq_m VST$)

Undecidability of *VST*

Theorem

The language

$$VST = \{\langle M, w, q \rangle \mid \text{Turing Machine } M \text{ visits state } q \text{ on input } w\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from HALT $= \{\langle M, w \rangle \mid M \text{ halts on } w\}$ to VST (i.e., $HALT \leq_m VST$)
- We want the inputs of f to be of the form:

Undecidability of *VST*

Theorem

The language

$$VST = \{\langle M, w, q\rangle \mid \textit{Turing Machine } M \textit{ visits state } q \textit{ on input } w\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from HALT $= \{\langle M, w\rangle \mid M$ halts on $w\}$ to VST (i.e., $HALT \leq_m VST$)
- We want the inputs of f to be of the form:
  - $\langle M, w\rangle$ where M is a Turing machine and w is a string

Undecidability of VST

Theorem

The language

$$VST = \{\langle M, w, q\rangle \mid \text{Turing Machine } M \text{ visits state } q \text{ on input } w\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from HALT $= \{\langle M, w\rangle \mid M$ halts on $w\}$ to VST (i.e., $HALT \leq_m VST$)
- We want the inputs of f to be of the form:
  - $\langle M, w\rangle$ where M is a Turing machine and w is a string
- We want the outputs of f to be of the form

Undecidability of *VST*

Theorem

The language

$$VST = \{\langle M, w, q\rangle \mid \textit{Turing Machine } M \textit{ visits state } q \textit{ on input } w\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from HALT $= \{\langle M, w\rangle \mid M$ halts on $w\}$ to VST (i.e., $HALT \leq_m VST$)
- We want the inputs of f to be of the form:
  - $\langle M, w\rangle$ where M is a Turing machine and w is a string
- We want the outputs of f to be of the form
  - $\langle N, s, q\rangle$ where N is a Turing machine, s is a string, and q is a state.

Undecidability of *VST*

Theorem

The language

$$VST = \{\langle M, w, q\rangle \mid \textit{Turing Machine } M \textit{ visits state } q \textit{ on input } w\}$$

is undecidable.

Proof Idea.

- We will provide a reduction f from HALT $= \{\langle M, w\rangle \mid M \text{ halts on } w\}$ to VST (i.e., $HALT \leq_m VST$)
- We want the inputs of f to be of the form:
  - $\langle M, w\rangle$ where M is a Turing machine and w is a string
- We want the outputs of f to be of the form
  - $\langle N, s, q\rangle$ where N is a Turing machine, s is a string, and q is a state.
- Again, we want $\langle M, w\rangle \in HALT \iff f(\langle M, w\rangle) \in VST$

Construction of $\langle N, s, q \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N, s, q \rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

Construction of $\langle N, s, q \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N, s, q \rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

Construction of $\langle N, s, q \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N, s, q \rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

- If $\langle M, w \rangle \in \mathrm{HALT}$

Construction of $\langle N, s, q\rangle$ for $f(\langle M, w\rangle)$

We will map the input $\langle M, w\rangle$ to $\langle N, s, q\rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

- If $\langle M, w\rangle \in$ HALT
  - Then, by construction, N visits the state q on input w.

Construction of $\langle N, s, q \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N, s, q \rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

- If $\langle M, w \rangle \in$ HALT
  - Then, by construction, N visits the state q on input w.
  - So, $\langle N, s, q \rangle \in VST$.

Construction of $\langle N, s, q\rangle$ for $f(\langle M, w\rangle)$

We will map the input $\langle M, w\rangle$ to $\langle N, s, q\rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

- If $\langle M, w\rangle \in$ HALT
  - Then, by construction, N visits the state q on input w.
  - So, $\langle N, s, q\rangle \in VST$.
- If $\langle M, w\rangle \notin HALT$

Construction of $\langle N, s, q \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N, s, q \rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

- If $\langle M, w \rangle \in$ HALT
  - Then, by construction, N visits the state q on input w.
  - So, $\langle N, s, q \rangle \in VST$.
- If $\langle M, w \rangle \notin HALT$
  - Then N simulates M on w forever and never reaches q.

Construction of $\langle N, s, q \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N, s, q \rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

- If $\langle M, w \rangle \in \text{HALT}$
  - Then, by construction, N visits the state q on input w.
  - So, $\langle N, s, q \rangle \in VST$.
- If $\langle M, w \rangle \notin HALT$
  - Then N simulates M on w forever and never reaches q.
  - So, $\langle N, s, q \rangle \notin VST$

Construction of $\langle N, s, q \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N, s, q \rangle$ where $s = w$, q is a state not in M, and N is the following Turing machine:

```
On input x
    run M on x
    If M halts and accepts or halts and rejects x,
        go to the newly created state q
```

- If $\langle M, w \rangle \in$ HALT
  - Then, by construction, N visits the state q on input w.
  - So, $\langle N, s, q \rangle \in VST$.
- If $\langle M, w \rangle \notin HALT$
  - Then N simulates M on w forever and never reaches q.
  - So, $\langle N, s, q \rangle \notin VST$
- Hence, $\langle M, w \rangle \in HALT \iff \langle N, s, q \rangle \in VST$

Undecidability of NE_{TM}

Theorem

The language $NE_{TM} = \{\langle M \rangle \mid \boldsymbol{L}(M) \neq \emptyset\}$ *is undecidable.*

Undecidability of NE_{TM}

Theorem

The language $NE_{TM} = \{\langle M \rangle \mid \mathbf{L}(M) \neq \emptyset\}$ is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to NE_{TM} (i.e., $A_{TM} \leq_m NE_{TM}$)

Undecidability of NE_{TM}

Theorem

The language $NE_{TM} = \{\langle M \rangle \mid \boldsymbol{L}(M) \neq \emptyset\}$ *is undecidable.*

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to NE_{TM} (i.e., $A_{TM} \leq_m NE_{TM}$)
- We want the inputs of f to be of the form:

∎

Undecidability of NE_{TM}

Theorem

The language $NE_{TM} = \{\langle M \rangle \mid \boldsymbol{L}(M) \neq \emptyset\}$ is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to NE_{TM} (i.e., $A_{TM} \leq_m NE_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M, w \rangle$ where M is a Turing machine and w is a string

Undecidability of NE_{TM}

Theorem

The language $NE_{TM} = \{\langle M \rangle \mid \boldsymbol{L}(M) \neq \emptyset\}$ is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to NE_{TM} (i.e., $A_{TM} \leq_m NE_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M, w \rangle$ where M is a Turing machine and w is a string
- We want the outputs of f to be of the form

Undecidability of NE_{TM}

Theorem

The language $NE_{TM} = \{\langle M \rangle \mid \mathbf{L}(M) \neq \emptyset\}$ is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$ to NE_{TM} (i.e., $A_{TM} \leq_m NE_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M, w \rangle$ where M is a Turing machine and w is a string
- We want the outputs of f to be of the form
  - $\langle N \rangle$ where N is a Turing machine

■

Undecidability of NE_{TM}

Theorem

The language $NE_{TM} = \{\langle M\rangle \mid \mathbf{L}(M) \neq \emptyset\}$ is undecidable.

Proof Idea.

- We will provide a reduction f from $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ to NE_{TM} (i.e., $A_{TM} \leq_m NE_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M, w\rangle$ where M is a Turing machine and w is a string
- We want the outputs of f to be of the form
  - $\langle N\rangle$ where N is a Turing machine
- Again, we want $\langle M, w\rangle \in A_{TM} \iff f(\langle M, w\rangle) \in NE_{TM}$

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.
  - So, $\langle N \rangle \in NE_{TM}$.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.
  - So, $\langle N \rangle \in NE_{TM}$.
- If $\langle M, w \rangle \notin A_{TM}$

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.
  - So, $\langle N \rangle \in NE_{TM}$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.
  - So, $\langle N \rangle \in NE_{TM}$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N does not accept anything.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.
  - So, $\langle N \rangle \in NE_{TM}$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N does not accept anything. So, $L(N) = \emptyset$.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.
  - So, $\langle N \rangle \in NE_{TM}$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \notin NE_{TM}$.

Construction of $\langle N \rangle$ for $f(\langle M, w \rangle)$

We will map the input $\langle M, w \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on w
    If M accepts w,
        accept x
    else
        reject x
```

- If $\langle M, w \rangle \in A_{TM}$
  - Then M accepts w.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is nonempty.
  - So, $\langle N \rangle \in NE_{TM}$.
- If $\langle M, w \rangle \notin A_{TM}$
  - Then M does not accept w.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \notin NE_{TM}$.
- Hence, $\langle M, w \rangle \in A_{TM} \iff \langle N \rangle \in NE_{TM}$

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ *is not recognizable.*

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ *is not recognizable.*

- Suppose for contradiction that some machine D recognizes L_d

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ *is not recognizable.*

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ is not recognizable.

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.
  - Since D recognizes L_d, D shouldn't accept encodings of machines that accept their own encoding.

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ *is not recognizable.*

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.
  - Since D recognizes L_d, D shouldn't accept encodings of machines that accept their own encoding.
  - So D doesn't accept $\langle D \rangle$.

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ *is not recognizable.*

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.
  - Since D recognizes L_d, D shouldn't accept encodings of machines that accept their own encoding.
  - So D doesn't accept $\langle D \rangle$.
  - This is a contradiction.

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ *is not recognizable.*

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.
  - Since D recognizes L_d, D shouldn't accept encodings of machines that accept their own encoding.
  - So D doesn't accept $\langle D \rangle$.
  - This is a contradiction.
- If D does not accept $\langle D \rangle$, then $\langle D \rangle \in L_d$.

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ is not recognizable.

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.
  - Since D recognizes L_d, D shouldn't accept encodings of machines that accept their own encoding.
  - So D doesn't accept $\langle D \rangle$.
  - This is a contradiction.
- If D does not accept $\langle D \rangle$, then $\langle D \rangle \in L_d$.
  - Since D recognizes L_d and $\langle D \rangle \in L_d$, D accepts $\langle D \rangle$.

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ is not recognizable.

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.
  - Since D recognizes L_d, D shouldn't accept encodings of machines that accept their own encoding.
  - So D doesn't accept $\langle D \rangle$.
  - This is a contradiction.
- If D does not accept $\langle D \rangle$, then $\langle D \rangle \in L_d$.
  - Since D recognizes L_d and $\langle D \rangle \in L_d$, D accepts $\langle D \rangle$.
  - This is a contradiction.

(Temporary Detour from Reductions)

Theorem

The language $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ is not recognizable.

- Suppose for contradiction that some machine D recognizes L_d
- If D accepts $\langle D \rangle$, then $\langle D \rangle \notin L_d$.
    - Since D recognizes L_d, D shouldn't accept encodings of machines that accept their own encoding.
    - So D doesn't accept $\langle D \rangle$.
    - This is a contradiction.
- If D does not accept $\langle D \rangle$, then $\langle D \rangle \in L_d$.
    - Since D recognizes L_d and $\langle D \rangle \in L_d$, D accepts $\langle D \rangle$.
    - This is a contradiction.
- So, no machine D can recognize L_d.

E_{TM} is not recognizable

Theorem

The language $E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$ *is not recognizable.*

E_{TM} is not recognizable

Theorem

The language $E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$ is not recognizable.

Proof Idea.

- We will provide a reduction f from $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ to E_{TM} (i.e., $L_d \leq_m E_{TM}$)

E_{TM} is not recognizable

Theorem

The language $E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$ *is not recognizable.*

Proof Idea.

- We will provide a reduction f from $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ to E_{TM} (i.e., $L_d \leq_m E_{TM}$)
- We want the inputs of f to be of the form:

E_{TM} is not recognizable

Theorem

The language $E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$ is not recognizable.

Proof Idea.

- We will provide a reduction f from $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ to E_{TM} (i.e., $L_d \leq_m E_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M \rangle$ where M is a Turing machine

E_{TM} is not recognizable

Theorem

The language $E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$ is not recognizable.

Proof Idea.

- We will provide a reduction f from $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ to E_{TM} (i.e., $L_d \leq_m E_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M \rangle$ where M is a Turing machine
- We want the outputs of f to be of the form

E_{TM} is not recognizable

Theorem

The language $E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$ is not recognizable.

Proof Idea.

- We will provide a reduction f from $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ to E_{TM} (i.e., $L_d \leq_m E_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M \rangle$ where M is a Turing machine
- We want the outputs of f to be of the form
  - $\langle N \rangle$ where N is a Turing machine

E_{TM} is not recognizable

Theorem

The language $E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$ is not recognizable.

Proof Idea.

- We will provide a reduction f from $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ to E_{TM} (i.e., $L_d \leq_m E_{TM}$)
- We want the inputs of f to be of the form:
  - $\langle M \rangle$ where M is a Turing machine
- We want the outputs of f to be of the form
  - $\langle N \rangle$ where N is a Turing machine
- Again, we want $\langle M \rangle \in L_d \iff f(\langle M \rangle) \in E_{TM}$

∎

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

On input x
    run M on $\langle M \rangle$
    accept if and only if M accepts $\langle M \rangle$

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

On input x
    run M on $\langle M \rangle$
    accept if and only if M accepts $\langle M \rangle$

- If $\langle M \rangle \in L_d$

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset$.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \in E_{TM}$.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset.$
  - So, $\langle N \rangle \in E_{TM}$.
- If $\langle M \rangle \notin L_d$

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \in E_{TM}$.
- If $\langle M \rangle \notin L_d$
  - Then M accepts $\langle M \rangle$.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

> On input x
>
> run M on $\langle M \rangle$
>
> accept if and only if M accepts $\langle M \rangle$

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \in E_{TM}$.
- If $\langle M \rangle \notin L_d$
  - Then M accepts $\langle M \rangle$.
  - Then N accepts everything.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \in E_{TM}$.
- If $\langle M \rangle \notin L_d$
  - Then M accepts $\langle M \rangle$.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is not empty.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \in E_{TM}$.
- If $\langle M \rangle \notin L_d$
  - Then M accepts $\langle M \rangle$.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is not empty.
  - So, $\langle N \rangle \notin E_{TM}$.

Construction of $\langle N \rangle$ for $f(\langle M \rangle)$

We will map the input $\langle M \rangle$ to $\langle N \rangle$ where N is the following Turing machine:

```
On input x
    run M on ⟨M⟩
    accept if and only if M accepts ⟨M⟩
```

- If $\langle M \rangle \in L_d$
  - Then M does not accept $\langle M \rangle$.
  - Then N does not accept anything. So, $L(N) = \emptyset$.
  - So, $\langle N \rangle \in E_{TM}$.
- If $\langle M \rangle \notin L_d$
  - Then M accepts $\langle M \rangle$.
  - Then N accepts everything. So, $L(N) = \Sigma^*$, which is not empty.
  - So, $\langle N \rangle \notin E_{TM}$.
- Hence, $\langle M \rangle \in L_d \iff \langle N \rangle \in E_{TM}$