Programming Language Principles

Programming Language Theory

Topics

- What is a Computer?
- **Turing Machine**
- How to implement a PL?
    - Compiler & Interpreter

Turing Machine

- Turing machine consists of a control unit and an infinite tape.
- **Tape**: an infinite tape, each **cell** contains one symbol.
- **Read-write Head**: read the current symbol, or write a symbol to the current cell.
- **Control unit**: a unit defines internal states and transition function.

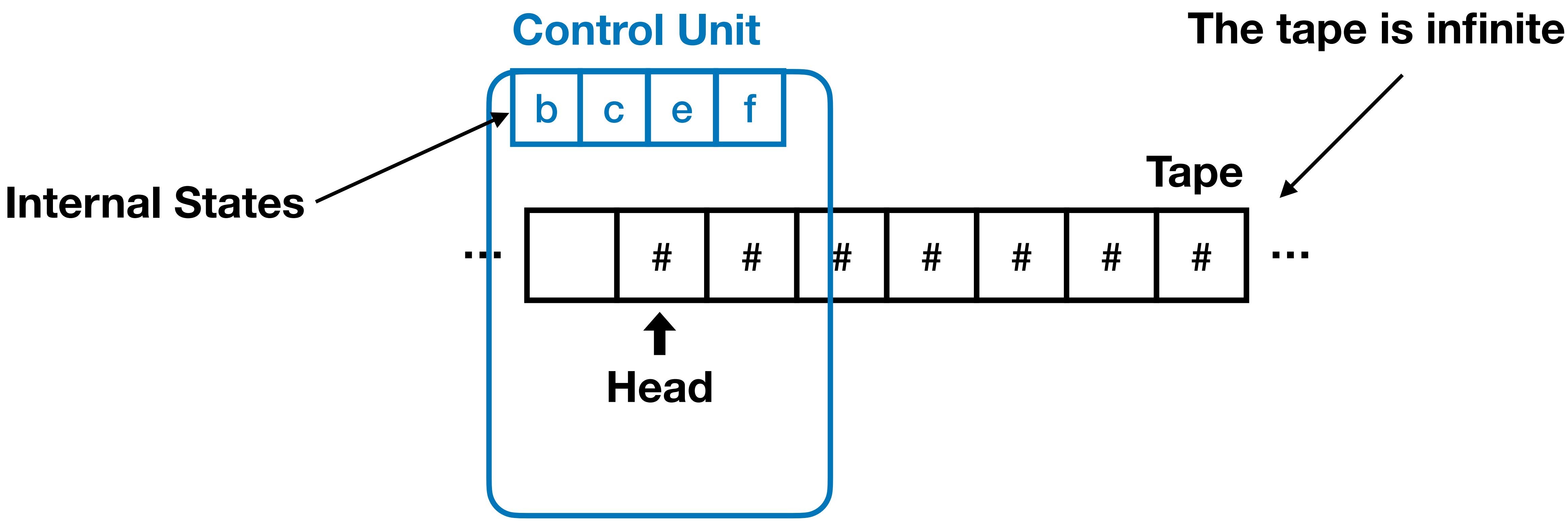
How does it work?

- Based on the current state and a symbol read by head,
  - Move Head: **L**eft, **R**ight, **S**tay.
  - Print to Tape: **P**X → X is a symbol, e.g.) P0, Pa, Pb
  - Decided the next state.
- All these are defined in transition function $\boldsymbol{\delta}$, which should be stored in control unit.
- The tape works as an input.

An Example

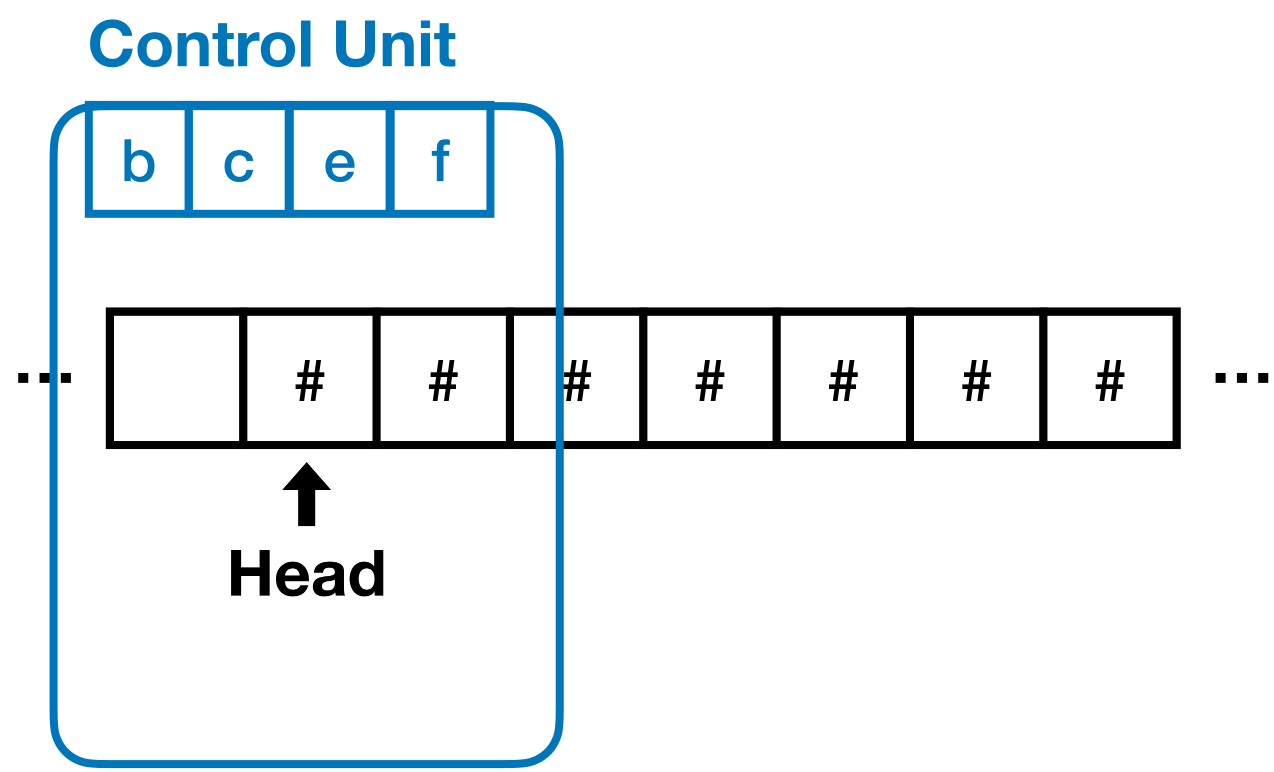
- A Turing machine computes a sequence 01010101…
- ‘#’ indicates a blank.
- Let’s consider that the input is a blank tape - ########
- We will use **S**tay operation - this was not in the original example of Turing’s paper.

0101010101...



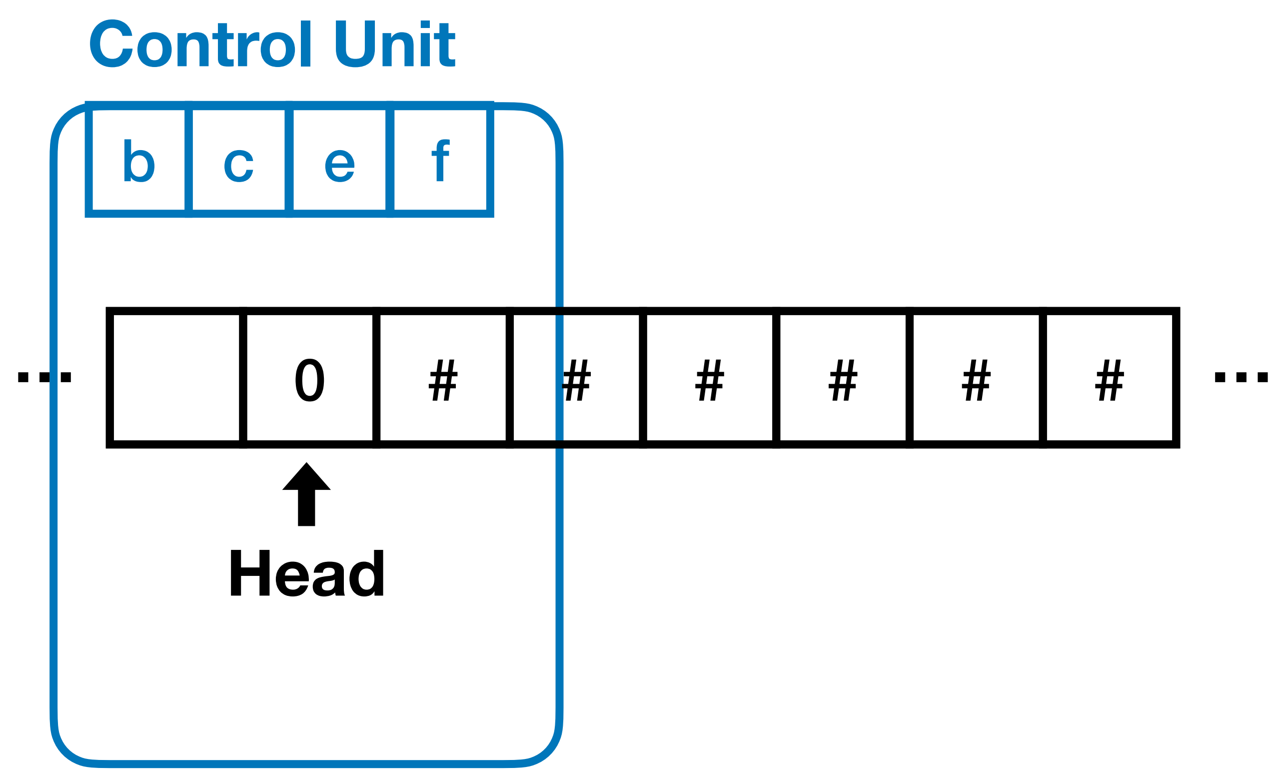
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

0...



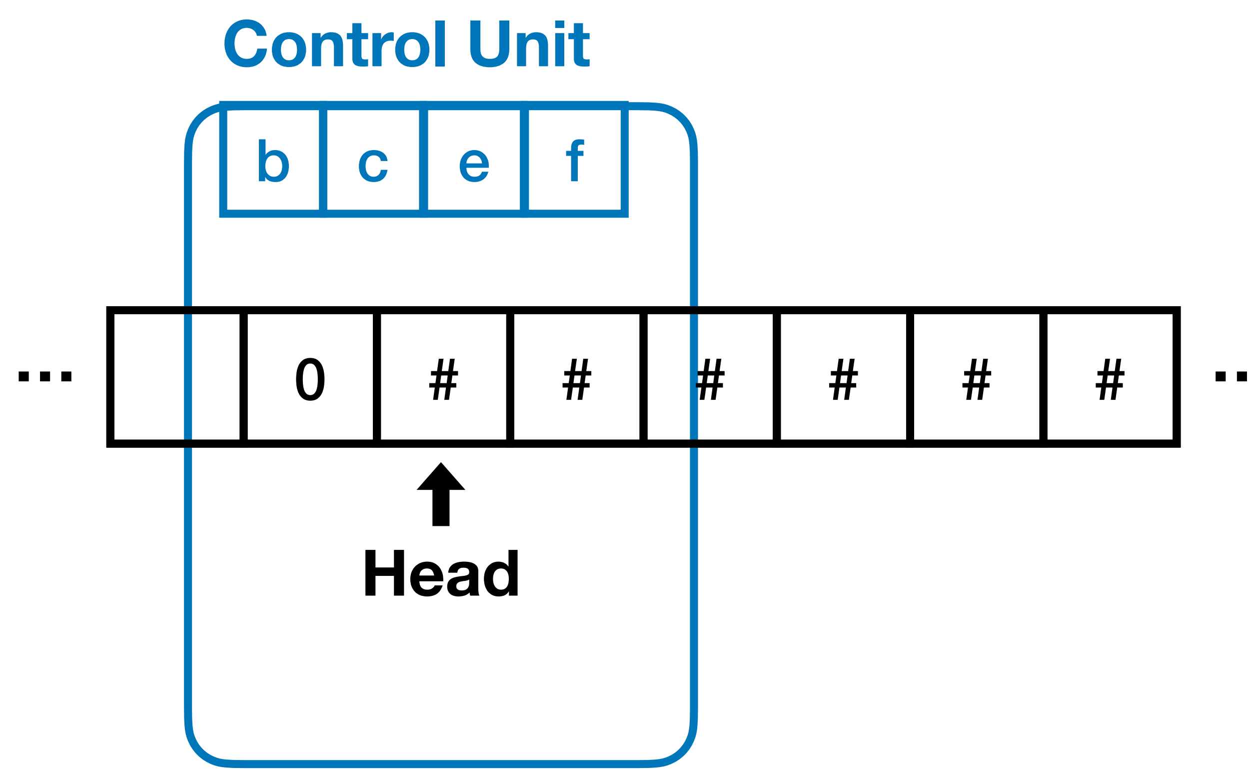
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

0...



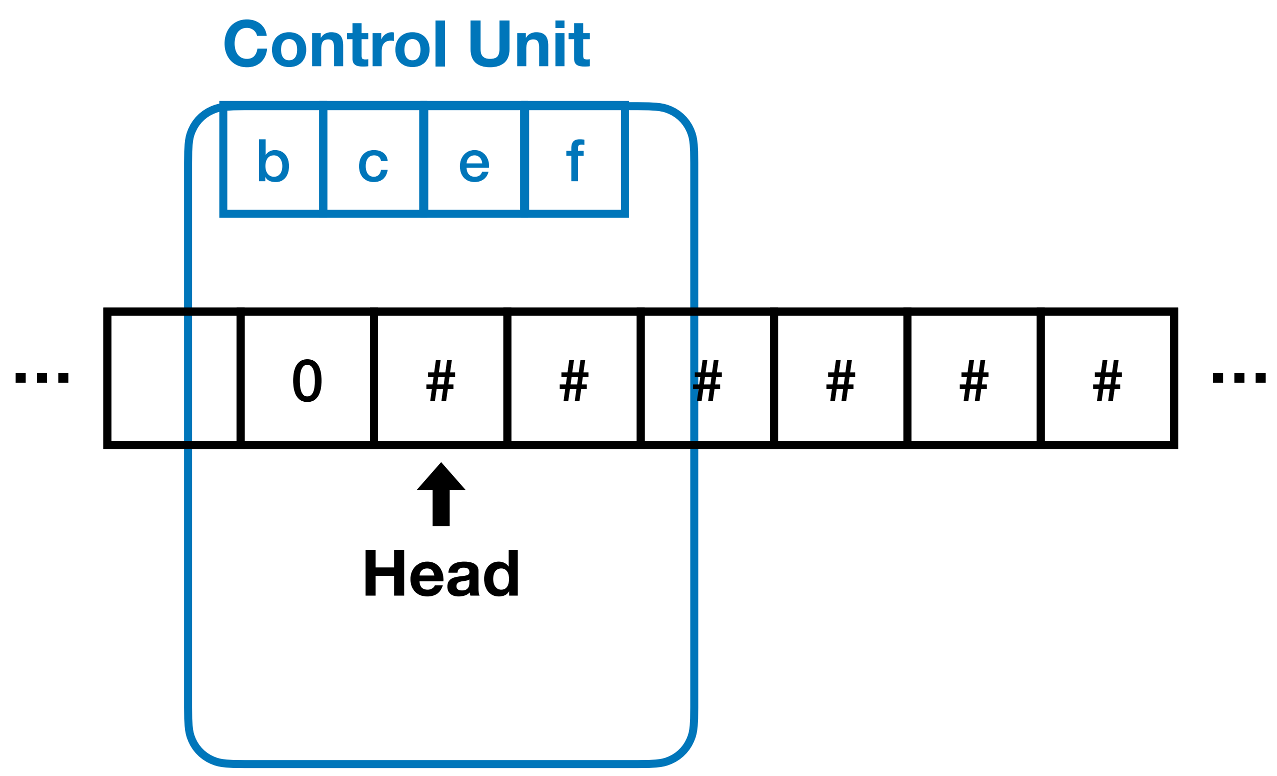
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

0...



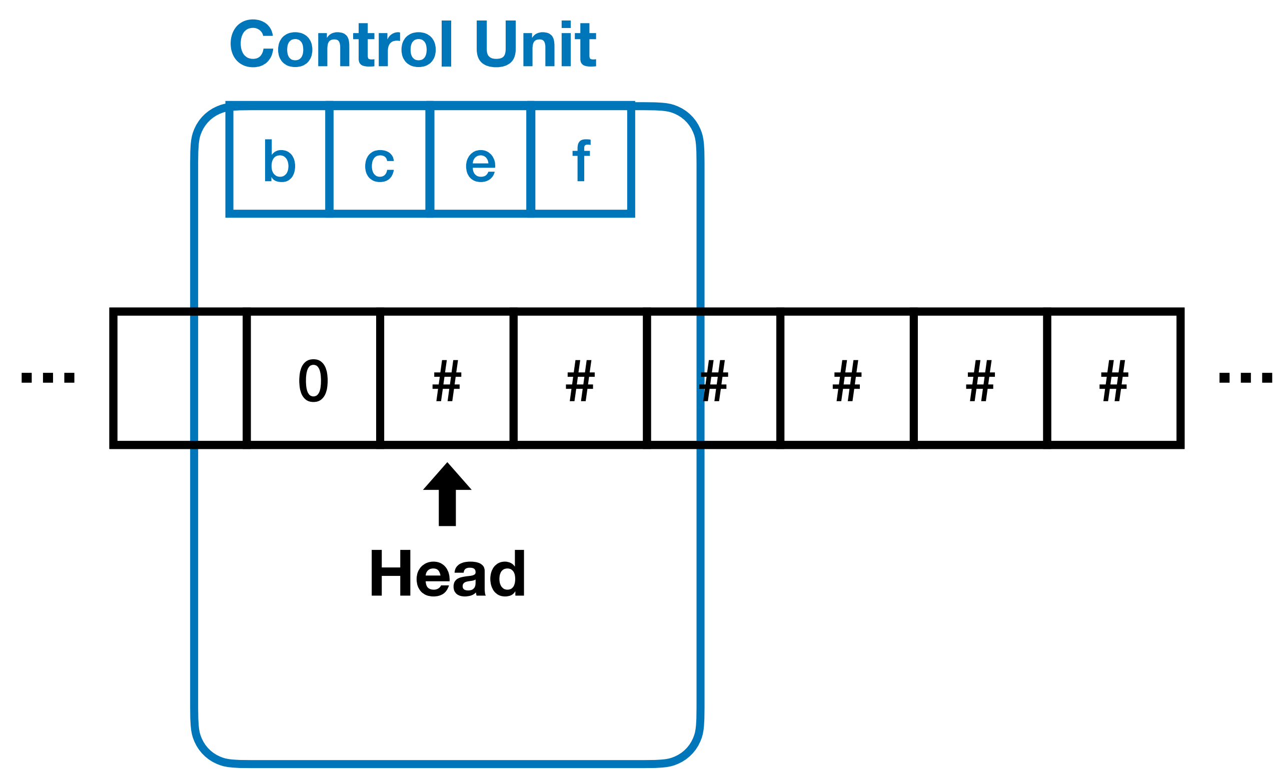
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

0…



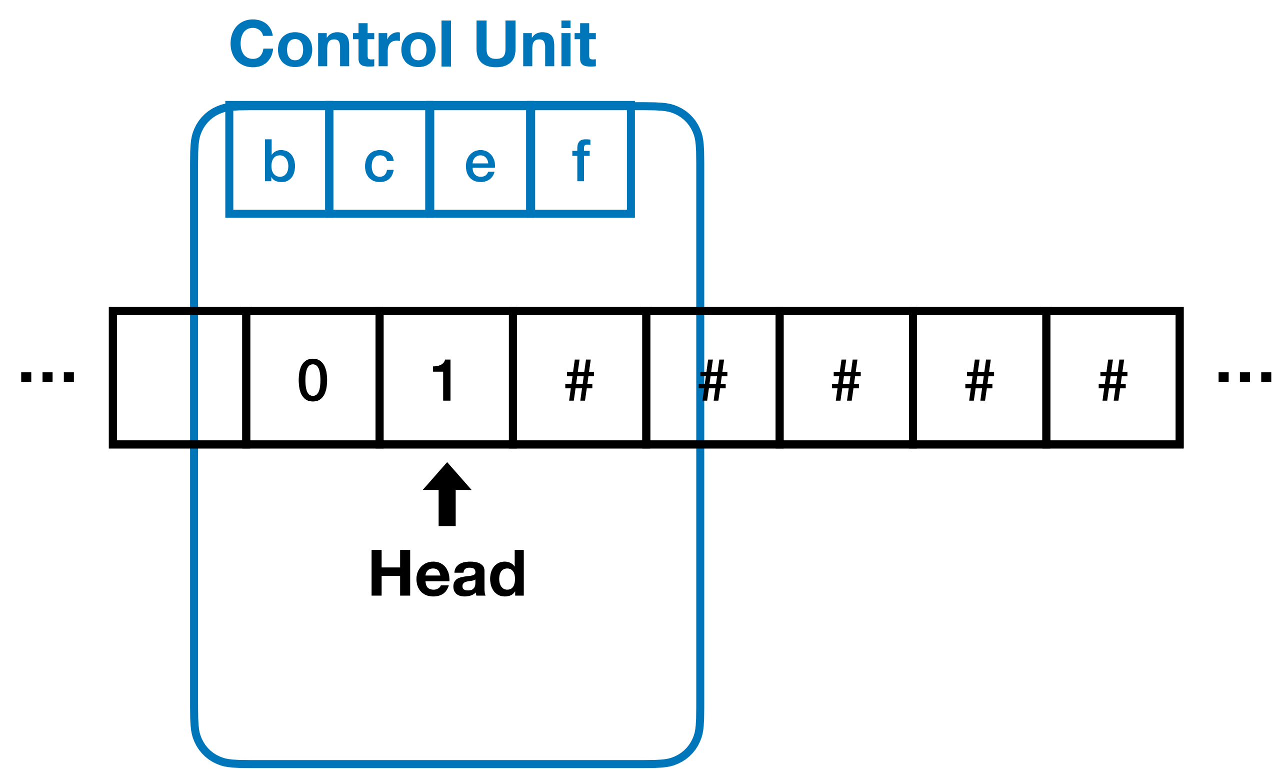
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

01...



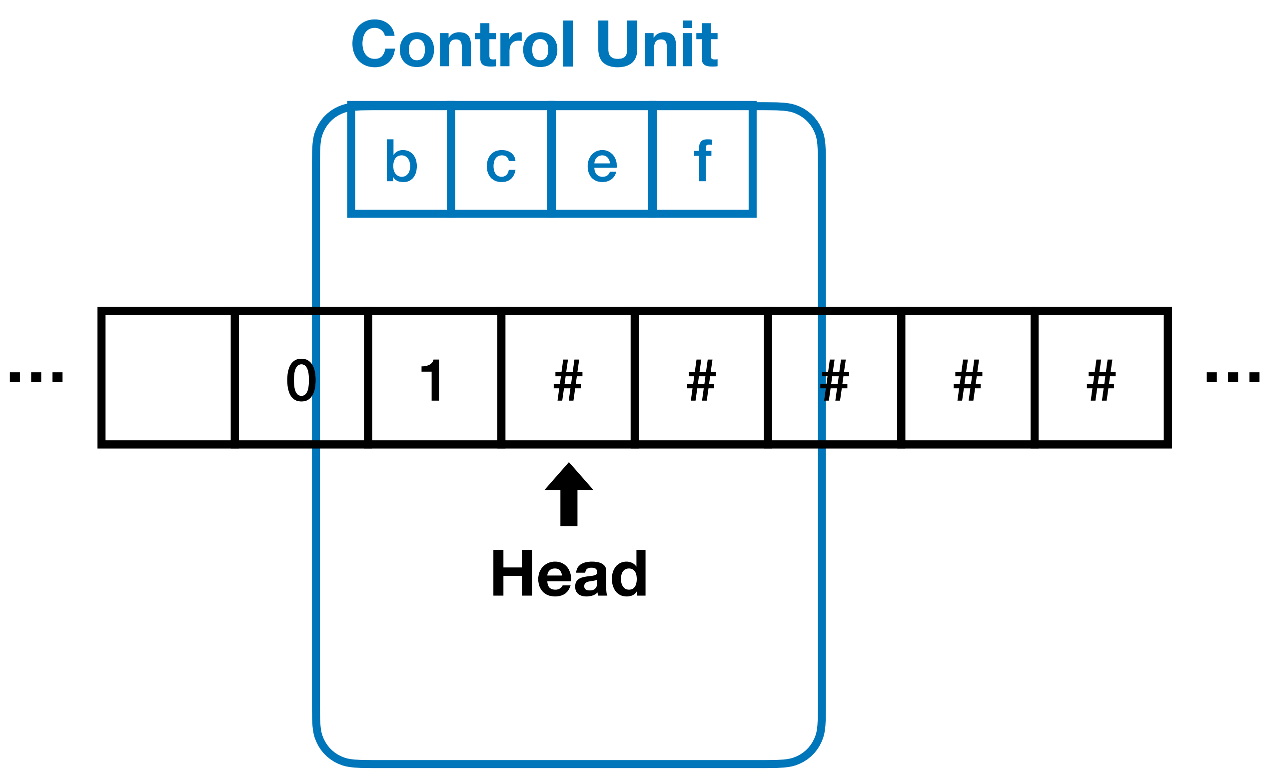
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

01...



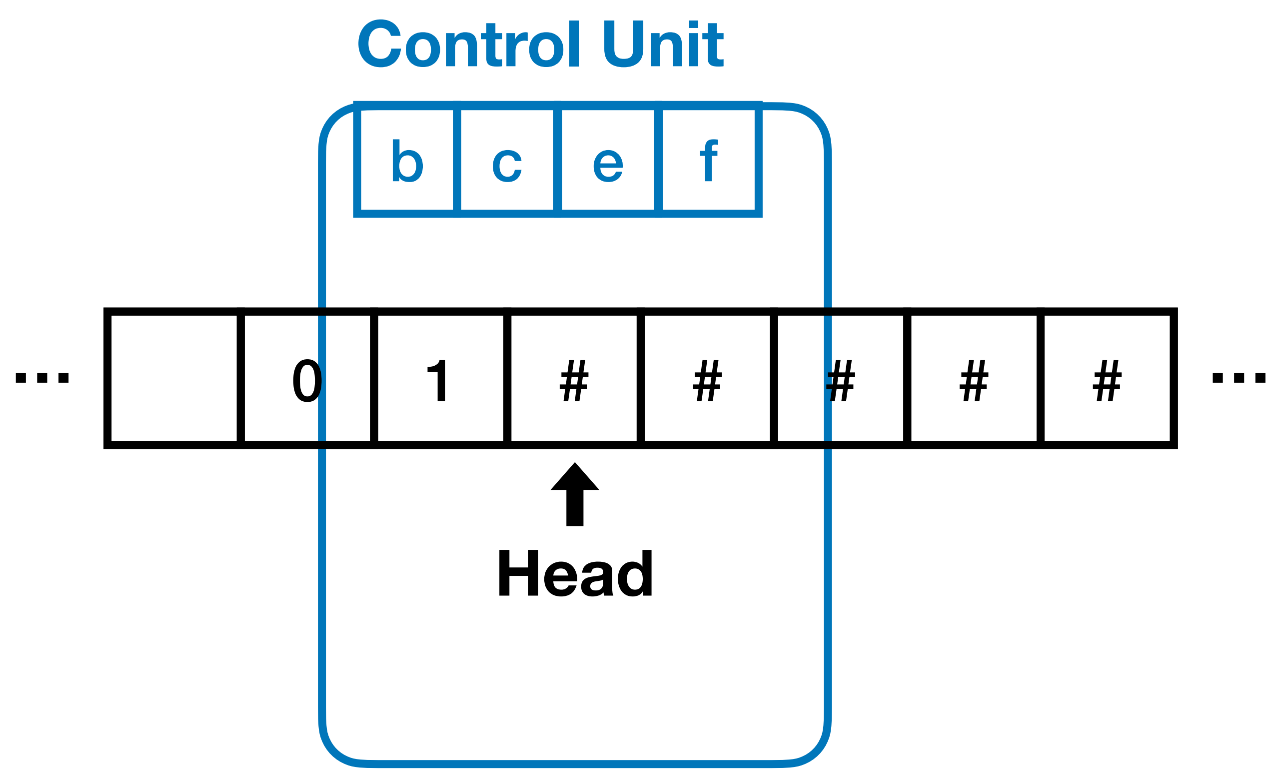
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

01...



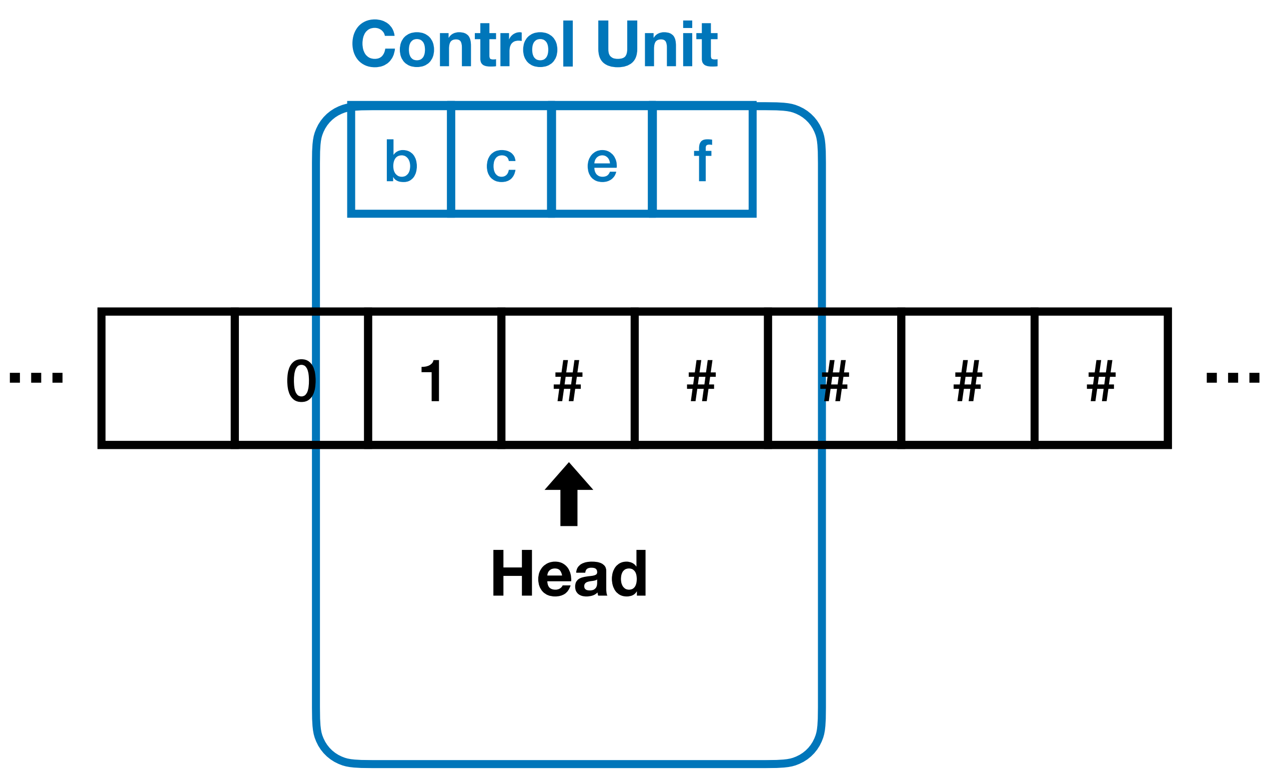
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

01...



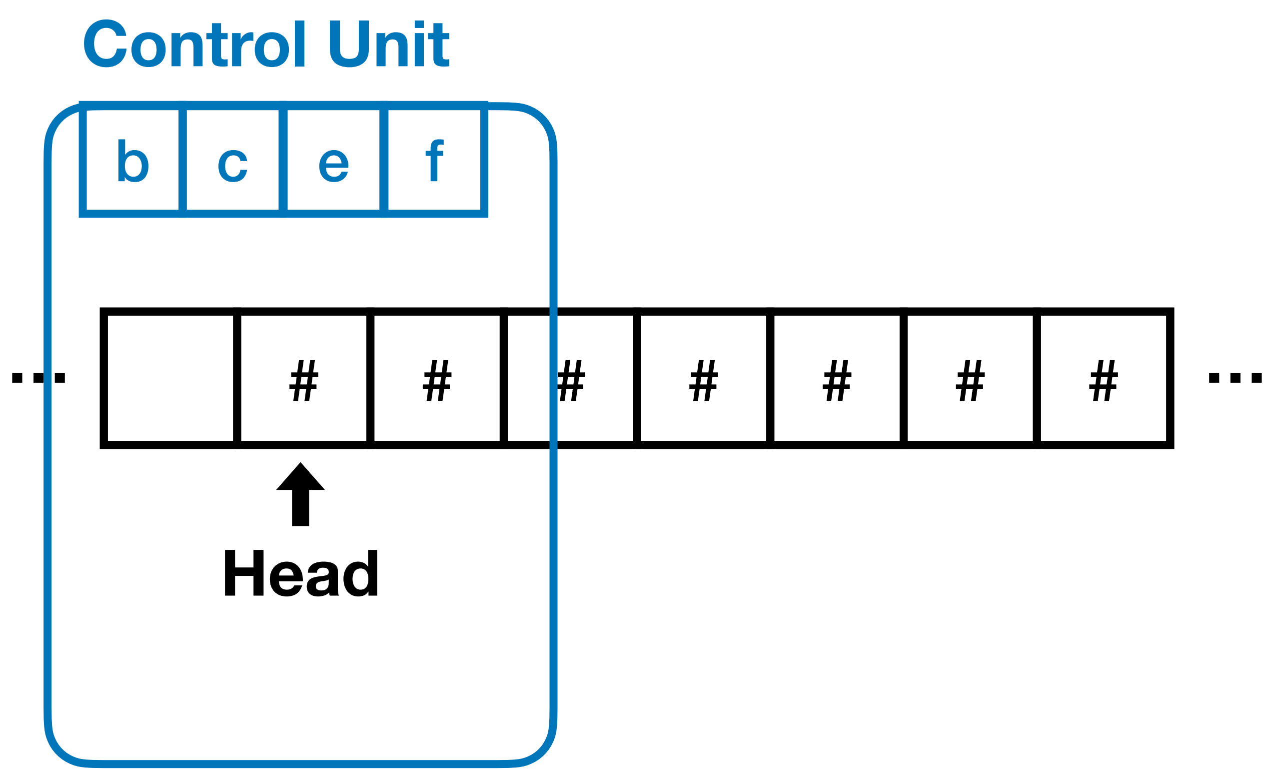
current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

01...



current state	symbol	operations	final state
b	#	P0, R	c
c	#	S	e
e	#	P1, R	f
f	#	S	b

More Compact Transition



current state	symbol	operations	final state
b	#	P0, R	e
e	#	P1, R	b

Formal Definition

- A Turing machine M is defined by
  - $M = (Q, \Sigma, \Gamma, \delta, q_0, \#, H)$

- Q: the set of internal states
- Σ: the input alphabet
- Γ: the tape alphabet
- δ: the transition function
- $q_0 \in Q$: the initial state
- $\#$: blank symbol
- $H \subseteq Q$: a set of final states

Formal Definition

- Transition Function $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{ L, R, S \}$
- $Q \times \Gamma$: current state & symbol
- $Q \times \Gamma \times \{ L, R, S \}$: next state & new symbol + head movement
  - ***L**eft, **R**ight, **S**tay*
- Machine halts (or at halting state) if there is no available transition.

Formal Definition

- $\delta\,(q_0, \text{a}) = (q_1, \text{b}, R)$
    - current state: q0, head: ‘a’
    - change state to $\boldsymbol{q_1}$, replace ‘a’ with ‘**b**’ and move head to ***Right***.
- $\delta\,(q_1, \text{b}) = (q_0, \text{a}, R)$

current state	symbol	operations	final state
q_0	a	Pb, R	q_1
q_1	b	Pa, R	q_0

Accepting Languages

- We can design a Turing machine which accepts a specific language.
- A string is given as an input (appeared in the tape).
- Turing machine examines the input string, and verifies whether the string follows a specific rule or not.

Language *L*

- L = { $a^n b^n$: n >= 1}
    - A string belongs to language *L* starts with ‘a’.
    - ‘a’ repeats n times.
    - Then ‘b’ follows ‘a’s, repeat exactly n times too.
- e.g.)
    - aabb, ab, aaabbb in *L*.
    - abb, #, aaabb, ba not in *L*.

How to Design *M*?

- A Turing machine *M* accepting $L = \{ a^n b^n : n >= 1\}$.
- $Q = \{ q_0, q_1, q_2, q_3, q_4 \}$
- $H = \{ q_4 \}$
- $\Sigma = \{ a, b \}$
- $\Gamma = \{ a, b, x, y, \# \}$

Basic Idea

1. Replace leftmost **a** with an **x**.
2. Move head to right until the first **b**, replace it with **y**.
3. Go back to left until **x** is found.
4. Repeat 1~3 until no more **a**, **b**.

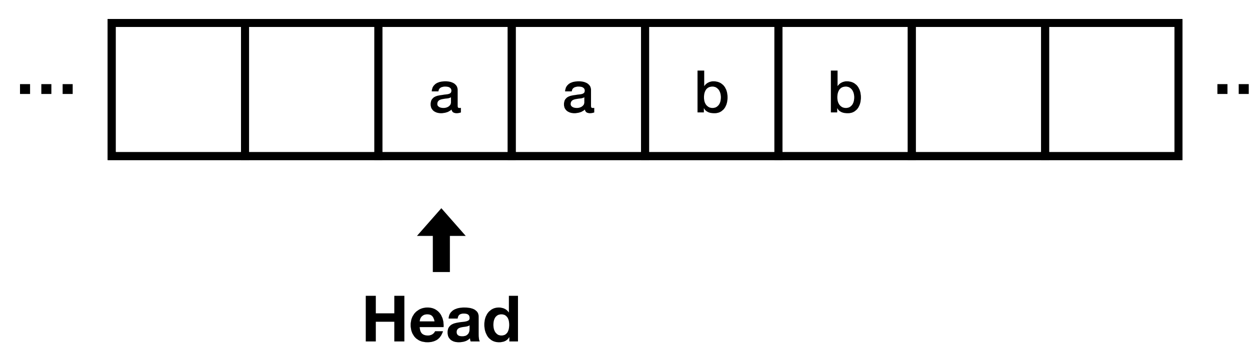
Transition Function δ

- δ $(q_0, a) = (q_1, x, R)$
- δ $(q_1, a) = (q_1, a, R)$
- δ $(q_1, y) = (q_1, y, R)$
- δ $(q_0, b) = (q_2, y, L)$

Example

- δ **(q_0, a) = (q_1, x, R)**
- δ (q_1, a) = (q_1, a, R)
- δ (q_1, y) = (q_1, y, R)
- δ (q_1, b) = (q_2, y, L)

Finding leftmost 'a'

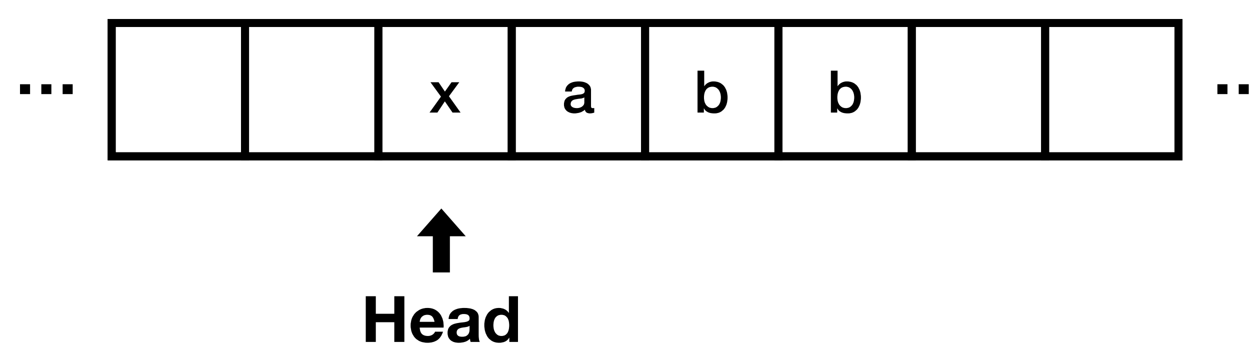


Internal State: q_0

Example

- **δ (q_0, a) = (q_1, x, R)**
- δ (q_1, a) = (q_1, a, R)
- δ (q_1, y) = (q_1, y, R)
- δ (q_1, b) = (q_2, y, L)

Replace it with ‘x’

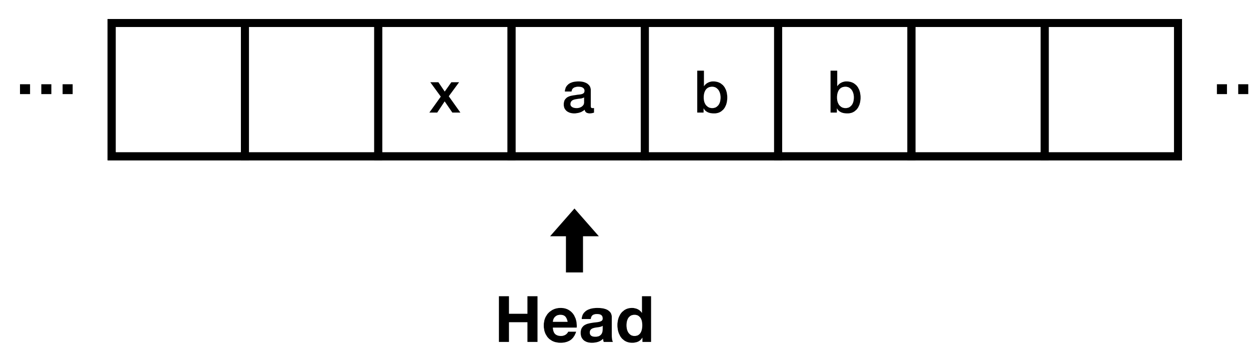


Internal State: q_0

Example

Replace it with ‘x’

- **δ (q_0, a) = (q_1, x, R)**
- δ (q_1, a) = (q_1, a, R)
- δ (q_1, y) = (q_1, y, R)
- δ (q_1, b) = (q_2, y, L)

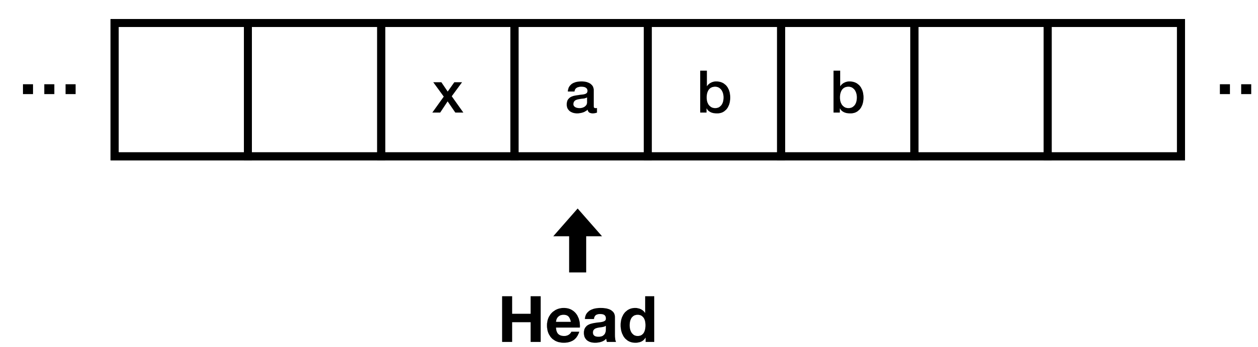


Internal State: q_1

Example

- δ (q_0, a) = (q_1, x, R)
- **δ (q_1, a) = (q_1, a, R)**
- δ (q_1, y) = (q_1, y, R)
- **δ (q_1, b) = (q_2, y, L)**

Finding first ‘b’ on the right

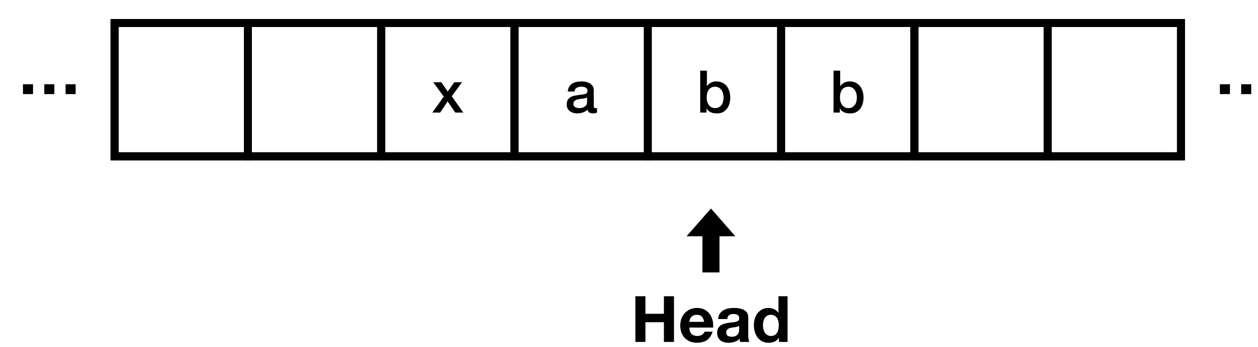


Internal State: q_1

Example

- δ (q_0, a) = (q_1, x, R)
- **δ (q_1, a) = (q_1, a, R)**
- δ (q_1, y) = (q_1, y, R)
- **δ (q_1, b) = (q_2, y, L)**

Finding first 'b' on the right

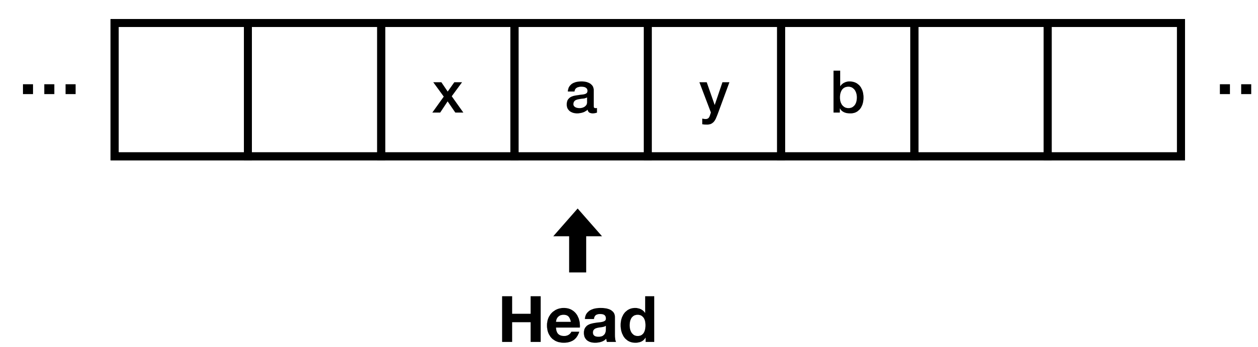


Internal State: q_1

Example

- δ (q_0, a) = (q_1, x, R)
- **δ (q_1, a) = (q_1, a, R)**
- δ (q_1, y) = (q_1, y, R)
- **δ (q_1, b) = (q_2, y, L)**

Replace it with ‘y’

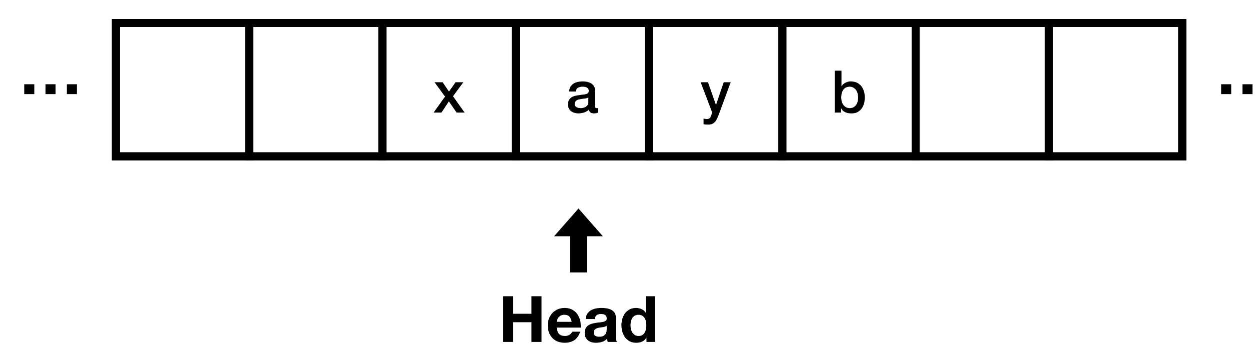


Internal State: q_2

Example

Go back to ‘x’

- δ (q_2, y) = (q_2, y, L)
- **δ (q_2, a) = (q_2, a, L)**
- **δ (q_2, x) = (q_0, x, R)**

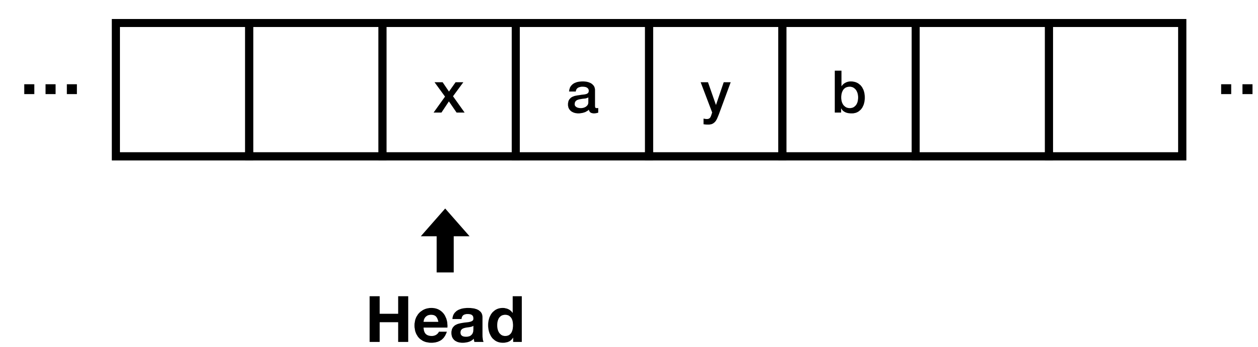


Internal State: q_2

Example

- δ (q_2, y) = (q_2, y, L)
- δ **(q_2, a) = (q_2, a, L)**
- δ **(q_2, x) = (q_0, x, R)**

Go back to ‘x’

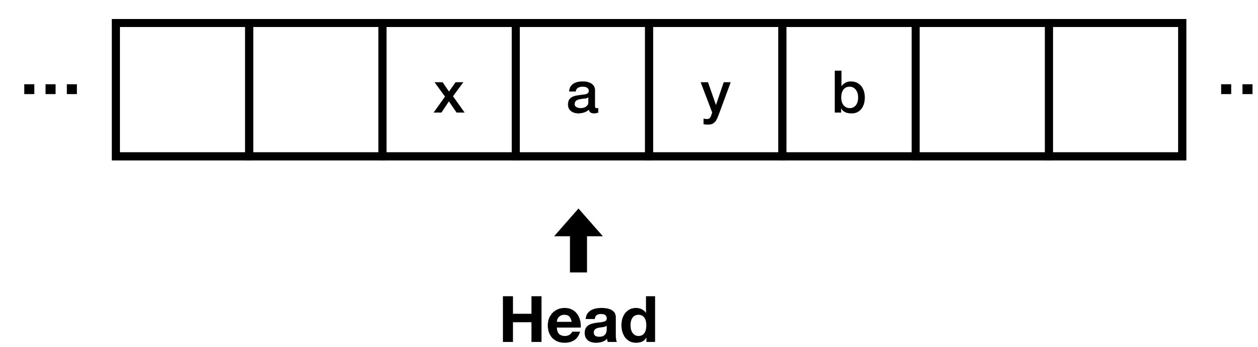


Internal State: q_2

Example

- δ (q_2, y) = (q_2, y, L)
- **δ (q_2, a) = (q_2, a, L)**
- **δ (q_2, x) = (q_0, x, R)**

Go back to ‘x’

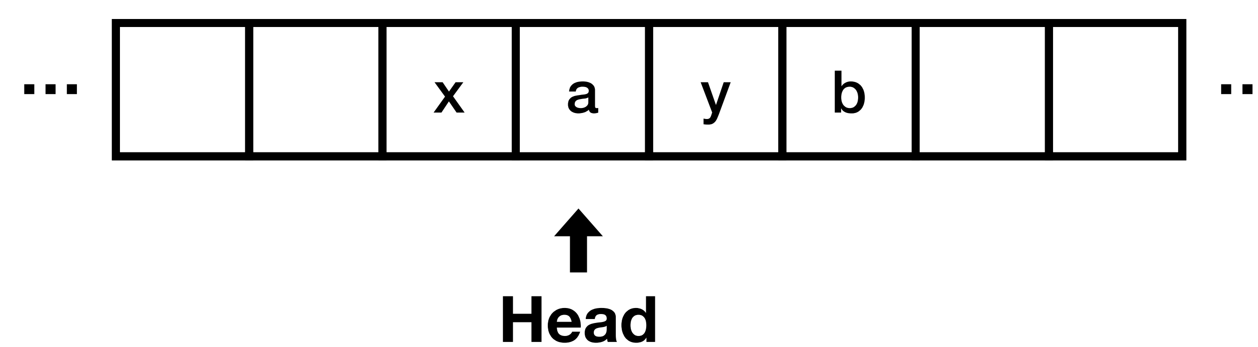


Internal State: q_2

Example

- δ (q_2, y) = (q_2, y, L)
- δ **(q_2, a) = (q_2, a, L)**
- δ **(q_2, x) = (q_0, x, R)**

Now it's back to the initial state

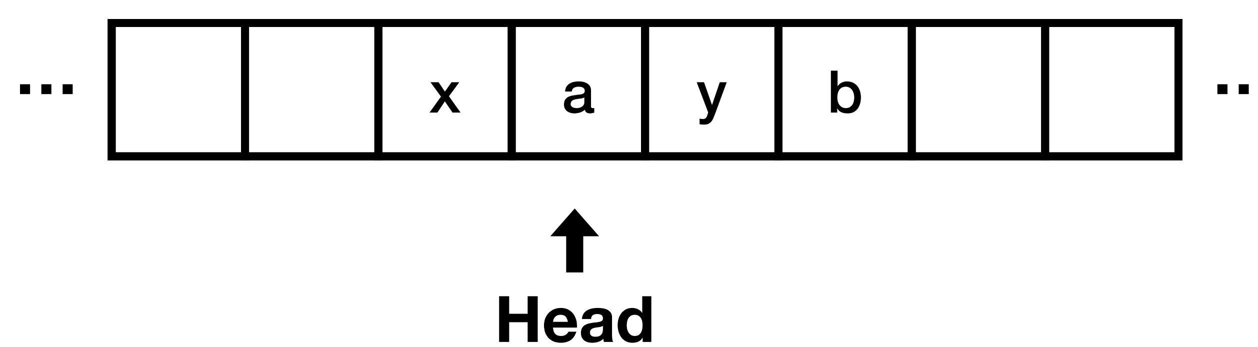


Internal State: q_0

Example

- δ **(q_0, a) = (q_1, x, R)**
- δ (q_1, a) = (q_1, a, R)
- δ (q_1, y) = (q_1, y, R)
- δ (q_1, b) = (q_2, y, L)

Finding leftmost 'a'

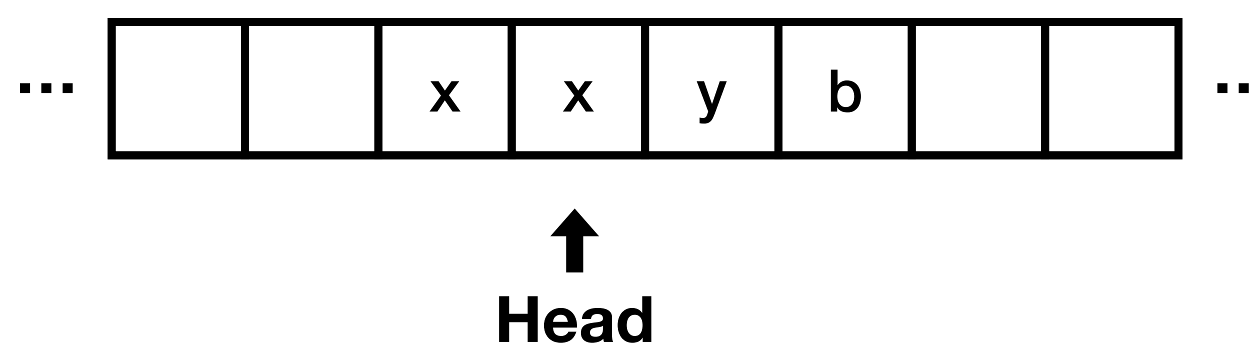


Internal State: q_0

Example

- δ **(q_0, a) = (q_1, x, R)**
- δ (q_1, a) = (q_1, a, R)
- δ (q_1, y) = (q_1, y, R)
- δ (q_1, b) = (q_2, y, L)

Replace it with ‘x’

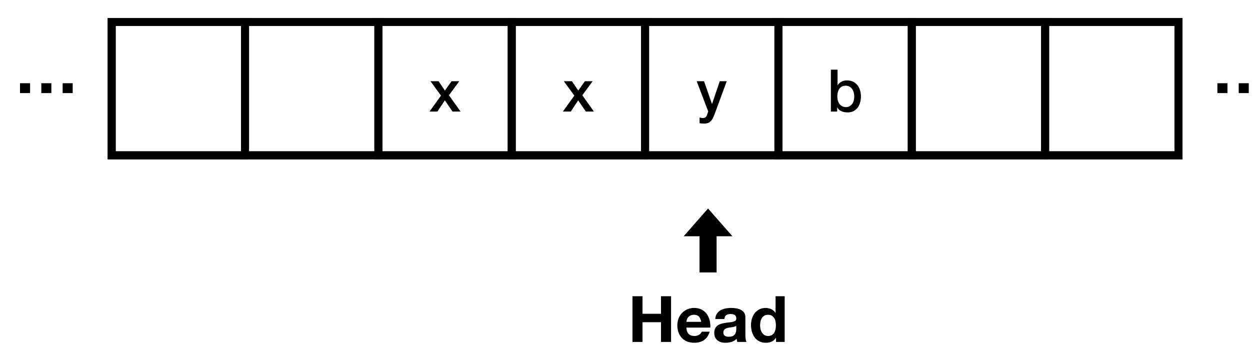


Internal State: q_0

Example

- δ **(q_0, a) = (q_1, x, R)**
- δ (q_1, a) = (q_1, a, R)
- δ (q_1, y) = (q_1, y, R)
- δ (q_1, b) = (q_2, y, L)

Replace it with ‘x’

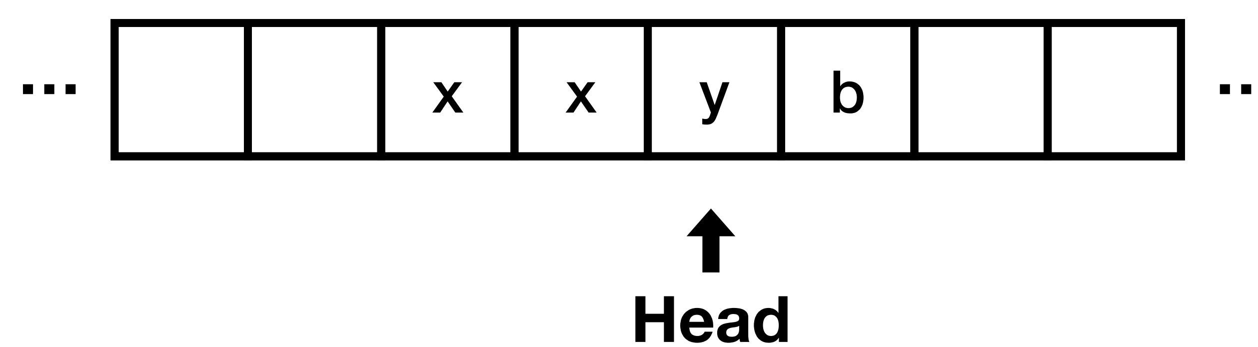


Internal State: q_1

Example

- δ (q_0, a) = (q_1, x, R)
- δ (q_1, a) = (q_1, a, R)
- **δ (q_1, y) = (q_1, y, R)**
- **δ (q_1, b) = (q_2, y, L)**

Finding first ‘b’ on the right

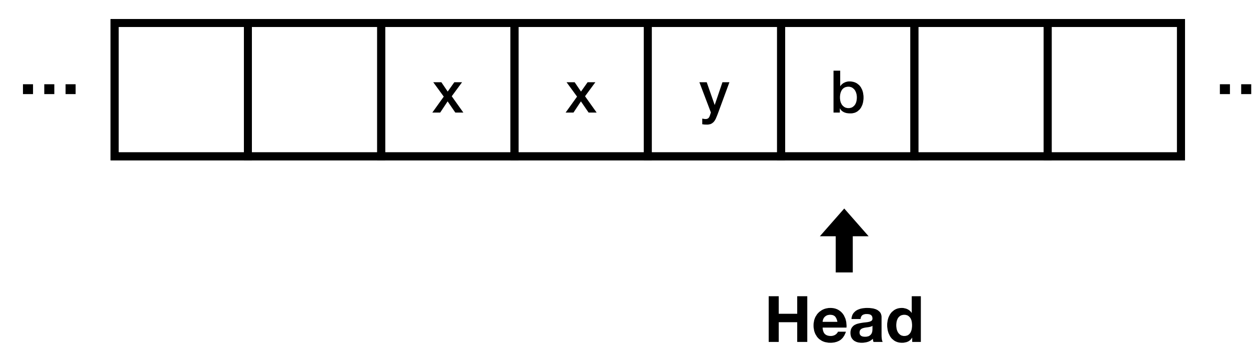


Internal State: q_1

Example

- δ (q_0, a) = (q_1, x, R)
- δ (q_1, a) = (q_1, a, R)
- **δ (q_1, y) = (q_1, y, R)**
- **δ (q_1, b) = (q_2, y, L)**

Finding first ‘b’ on the right

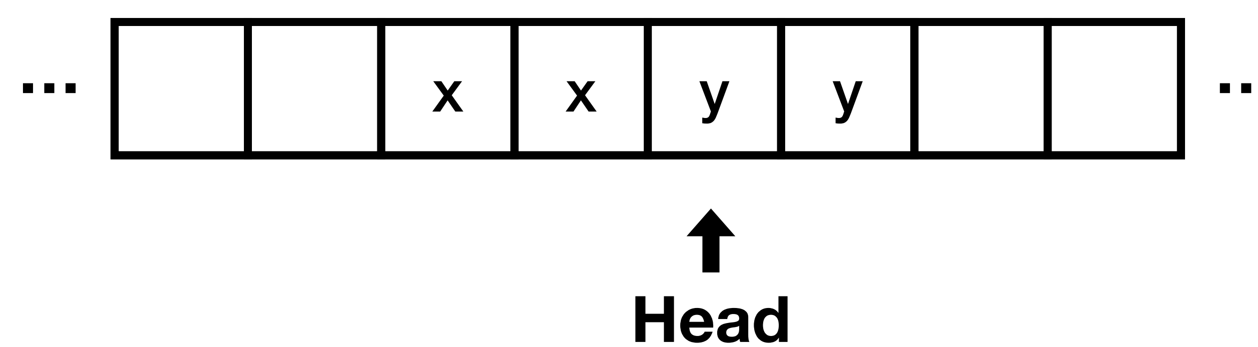


Internal State: q_1

Example

- δ (q_0, a) = (q_1, x, R)
- δ (q_1, a) = (q_1, a, R)
- **δ (q_1, y) = (q_1, y, R)**
- **δ (q_1, b) = (q_2, y, L)**

Replace it with ‘y’

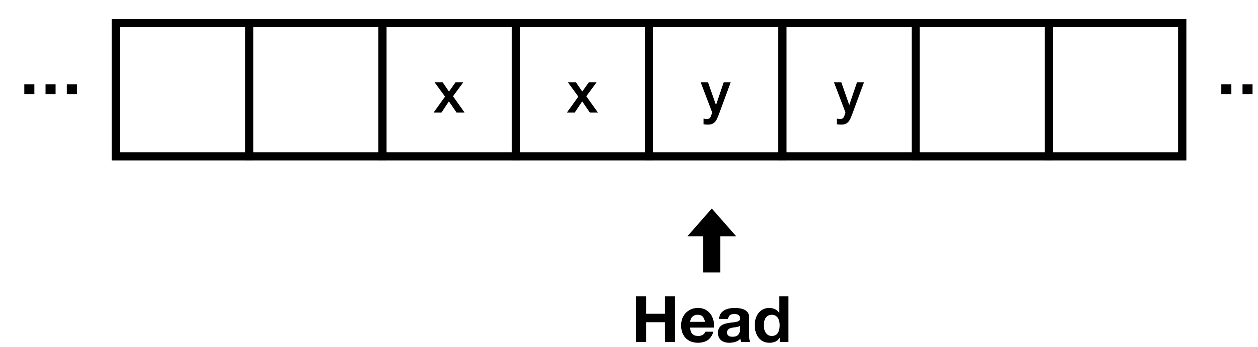


Internal State: q_2

Example

- **δ (q_2, y) = (q_2, y, L)**
- δ (q_2, a) = (q_2, a, L)
- **δ (q_2, x) = (q_0, x, R)**

Go back to 'x'

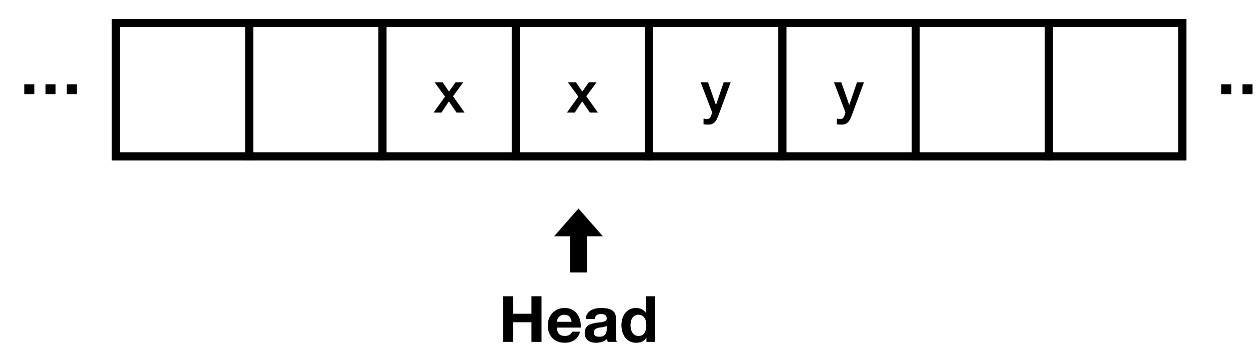


Internal State: q_2

Example

Go back to ‘x’

- **δ (q_2, y) = (q_2, y, L)**
- δ (q_2, a) = (q_2, a, L)
- **δ (q_2, x) = (q_0, x, R)**

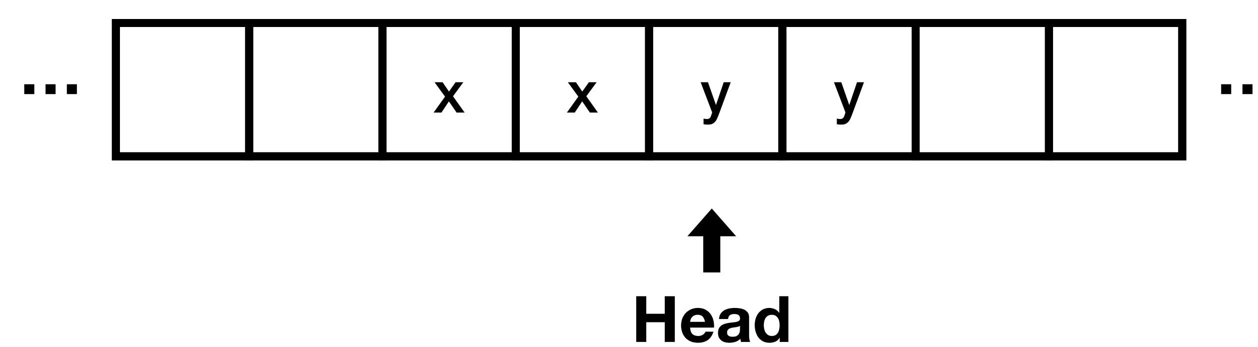


Internal State: q_2

Example

- δ **(q_2, y) = (q_2, y, L)**
- δ (q_2, a) = (q_2, a, L)
- δ **(q_2, x) = (q_0, x, R)**

Go back to ‘x’

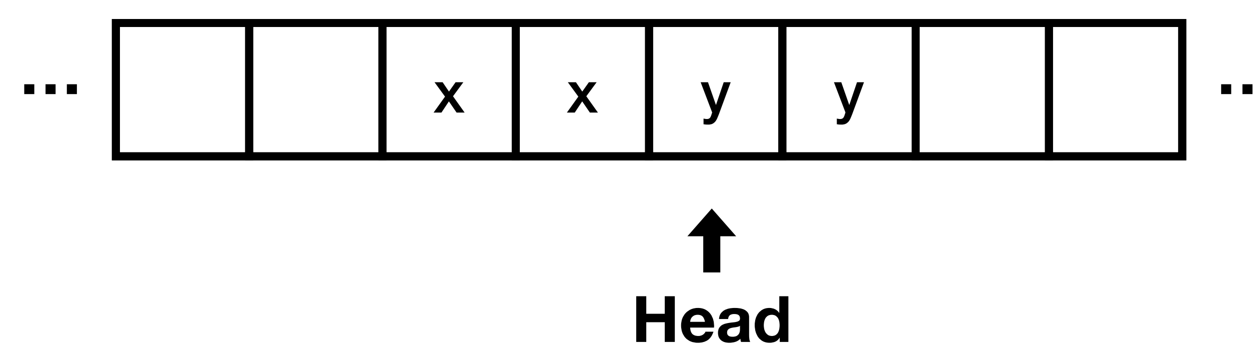


Internal State: q_2

Example

- **δ (q_2, y) = (q_2, y, L)**
- δ (q_2, a) = (q_2, a, L)
- **δ (q_2, x) = (q_0, x, R)**

Now it's back to the initial state

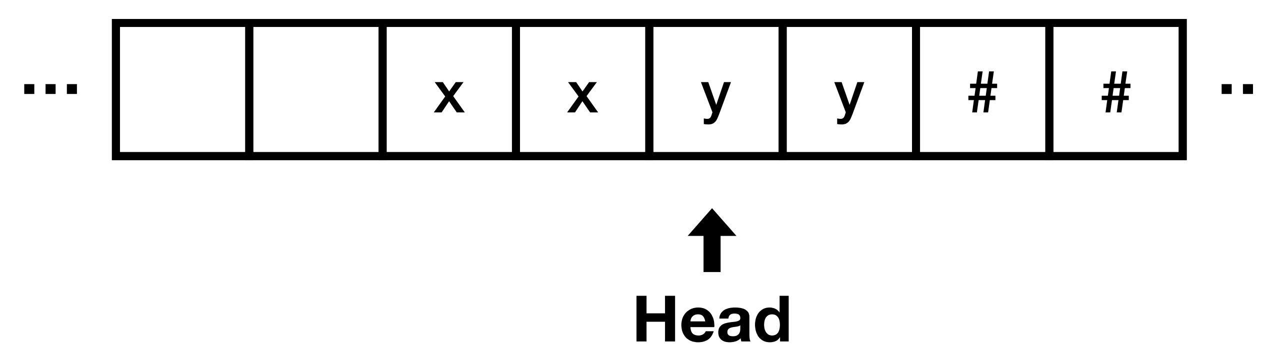


Internal State: q_0

Example

No more b → Finish

- δ **(q_0, y) = (q_3, y, R)**
- δ **(q_3, y) = (q_3, y, R)**
- δ **(q_3, #) = (q_4, #, R)**

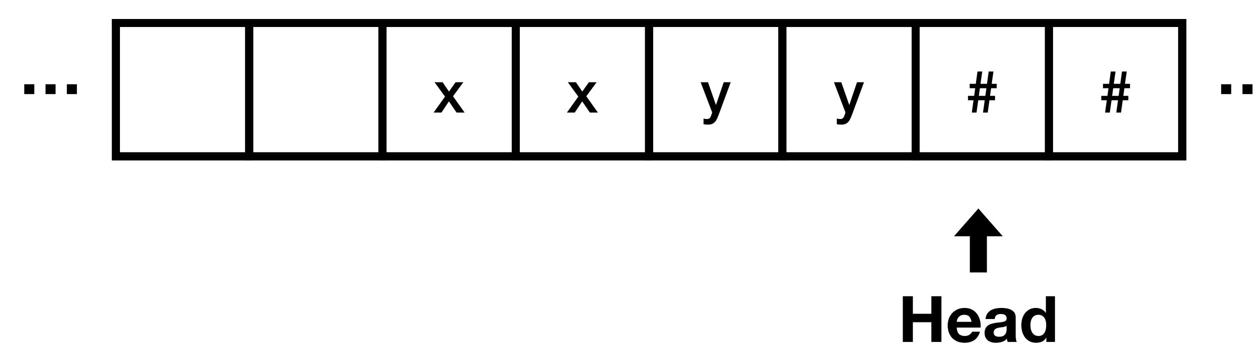


Internal State: q_0

Example

- δ **(q_0, y) = (q_3, y, R)**
- δ **(q_3, y) = (q_3, y, R)**
- δ **(q_3, #) = (q_4, #, R)**

No more b → Finish

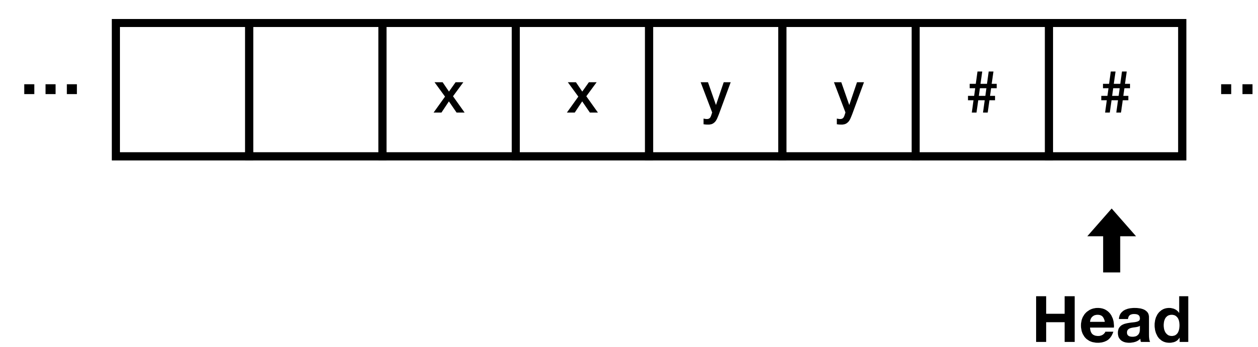


Internal State: q_3

Example

- δ **(q_0, y) = (q_3, y, R)**
- δ **(q_3, y) = (q_3, y, R)**
- δ **(q_3, #) = (q_4, #, R)**

No more b → Finish

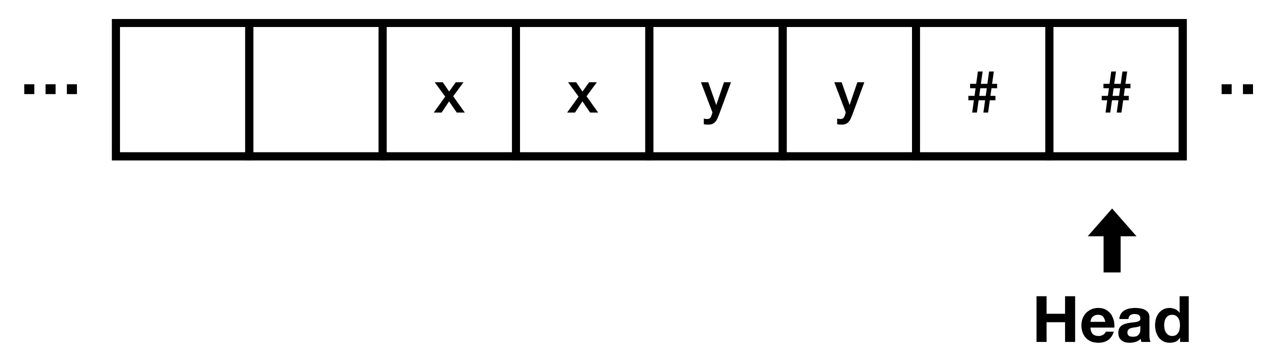


Internal State: q_4

Example

No more b → Finish

- δ **(q_0, y) = (q_3, y, R)**
- δ **(q_3, y) = (q_3, y, R)**
- δ **(q_3, #) = (q_4, #, R)**



q_4 is a halting state!

Internal State: q_4

Table Notation

State	Input Symbol a	b	x	y	#
q_0	(q_1, x, R)			(q_3, y, R)	
q_1	(q_1, a, R)	(q_2, y, L)		(q_1, y, R)	
q_2	(q_2, a, L)		(q_0, x, R)	(q_2, y, L)	
q_3				(q_3, y, R)	$(q_4, \#, R)$
q_4					

No transition for q_4 since it is a final state.

- δ **$(q_0, a) = (q_1, x, R)$**
- $\delta\ (q_1, a) = (q_1, a, R)$
- $\delta\ (q_1, y) = (q_1, y, R)$
- $\delta\ (q_1, b) = (q_2, y, L)$
- $\delta\ (q_2, y) = (q_2, y, L)$
- $\delta\ (q_2, a) = (q_2, a, L)$
- $\delta\ (q_2, x) = (q_0, x, R)$
- $\delta\ (q_0, y) = (q_3, y, R)$
- $\delta\ (q_3, y) = (q_3, y, R)$
- $\delta\ (q_3, \#) = (q_4, \#, R)$

$\vdash$ Notation

- We can represent transition with ‘$\vdash$’, indicating a move from one configuration to another configuration.
- δ (q0, a) = (q1, x, R), head at the first a of “aabb”
  - $q_0aabb \vdash xq_1abb$
- Put state name in front of a symbol which the head is pointing.
- We can also combine many transitions into one.
  - $q_0aabb \vdash^* xxyy\#q_4\#$

Universal Turing Machine

- What if we can provide a Turing machine ***M*** as an input to another Turing machine ***U***?
- Then ***U*** can compute the same as ***M***.
- We can think that ***U*** is a computer, ***M*** is a program.
- This is the idea of a ***stored-program computer***.

Summary

- Turing Machine
- Accepting Languages
- Universal Turing Machine