BU CS 332 – Theory of Computation

Lecture 8:

- Pumping lemma for CFLs
- Closure properties for CFLs
- Turing machines

Reading:

Sipser Ch 2.1, 2.3, 3.1, 3.2

Ran Canetti

September 29, 2020

Context-Free Grammar (Formal)

A CFG is a 4-tuple $G = (V, \Sigma, R, S)$

- V is a finite set of variables
- Σ is a finite set of terminal symbols (disjoint from V)
- R is a finite set of production rules of the form $A \to w$, where $A \in V$ and $w \in (V \cup \Sigma)^*$
- $S \in V$ is the start variable

Example: $G = (\{S\}, \Sigma, R, S)$

where

$$\Sigma = \{a, b\}$$
$$R = \{S \to aSb, S \to \varepsilon\}$$

Context-Free Languages

L is a *context-free language* if it is the language of some CFG

Questions about CFLs

1. Which languages are *not* context-free?
2. What are the closure properties of CFLs?
3. How do we recognize whether $w \in L$?

Pumping Lemma for context-free languages

Let L be a context-free language.

Then there exists a "pumping length" p such that

For every $w \in L$ where $|w| \geq p$,

w can be split into five parts $w = uvxyz$ where:

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^ixy^iz \in L$ for all $i \geq 0$

Pumping Lemma: Proof idea

Let L be a context-free language. If $w \in L$ is long enough, then every parse tree for w has a repeated variable.

Pumping Lemma Proof

What does "long enough" mean? (How do we choose the pumping length p?)

- Let G be a CFG for L
- Suppose the right-hand side of every rule in G uses at most b symbols
- Let $p = b^{|V|+1}$

Claim: If $w \in L$ with $|w| \geq p$, then the smallest parse tree for w has height at least $|V| + 1$

Pumping Lemma for context-free languages

Let L be a context-free language.

Then there exists a "pumping length" p such that

For every $w \in L$ where $|w| \geq p$,

w can be split into five parts $w = uvxyz$ where:

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^ixy^iz \in L$ for all $i \geq 0$

Pumping Lemma for context-free languages

Let L be a context-free language.

Then there exists a "pumping length" p such that

For every $w \in L$ where $|w| \geq p$,

w can be split into five parts $w = uvxyz$ where:

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^ixy^iz \in L$ for all $i \geq 0$

Example:

$L = \{w \in \{0,1\}^* | w = w^R\}$

$w = 00000$

Pumping Lemma for context-free languages

Let L be a context-free language.

Then there exists a "pumping length" p such that

For every $w \in L$ where $|w| \geq p$,
w can be split into five parts $w = uvxyz$ where:

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^ixy^iz \in L$ for all $i \geq 0$

Example:
$L = \{w \in \{0,1\}^* | w = w^R\}$
$w = 00$

Pumping Lemma for context-free languages

Let L be a context-free language.

Then there exists a “pumping length” p such that

For every $w \in L$ where $|w| \geq p$,

w can be split into five parts $w = uvxyz$ where:

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^ixy^iz \in L$ for all $i \geq 0$

Example:

$L = \{w \in \{0,1\}^* | w = w^R\}$

$w = 010$

Pumping Lemma as a game

1. YOU pick the language L to be proved non context-free.
2. ADVERSARY picks a possible pumping length p.
3. YOU pick w of length at least p.
4. ADVERSARY divides w into u, v, x, y, z, obeying rules of the Pumping Lemma: $|vy| > 0$ and $|vxy| \leq p$.
5. YOU win by finding $i \geq 0$, for which uv^ixy^iz is not in L.

If *regardless* of how the ADVERSARY plays this game, you can always win, then L is non context-free

Pumping Lemma example

Claim: $L = \{a^n b^n c^n | n \geq 0\}$ is not regular

Proof: Assume L is regular with pumping length p

1. Find $w \in L$ with $|w| \geq p$
2. Show that w cannot be pumped

   If w = $uvxyz$ with $|vy| > 0, |vxy| \leq p$, then…

Context-Free Languages

L is a *context-free language* if it is the language of some CFG

Questions about CFLs

1. Which languages are *not* context-free?
2. What are the closure properties of CFLs?
3. How do we recognize whether $w \in L$?

Closure Properties

- The class of CFLs is closed under the regular operations union, concatenation, star

Closure under union

Let A be a CFL generated by CFG G_A and let B be a CFL recognized by CFG G_B

Goal: Construct a CFG G recognizing $A \cup B$

$G_A = (V_A, \Sigma_A, R_A, S_A)$

$G_B = (V_B, \Sigma_B, R_B, S_B)$

Relabel variables so V_A and V_B are disjoint

Construct $G = (V, \Sigma, R, S)$:

$V = V_A \cup V_B \cup \{S\}$, $\Sigma = \Sigma_A \cup \Sigma_B$,

$R =$

Closure under union

Let A be a CFL generated by CFG G_A and let B be a CFL recognized by CFG G_B

Goal: Construct a CFG G recognizing $A \cup B$

$G_A = (V_A, \Sigma_A, R_A, S_A)$

$G_B = (V_B, \Sigma_B, R_B, S_B)$

Relabel variables so V_A and V_B are disjoint

Construct $G = (V, \Sigma, R, S)$:

$$V = V_A \cup V_B \cup \{S\}, \qquad \Sigma = \Sigma_A \cup \Sigma_B,$$
$$R = R_{\mathrm{A}} \cup R_B \cup \{S \to S_A | S_B\}$$

Closure under concatenation

Let A be a CFL generated by CFG G_A and let B be a CFL recognized by CFG G_B

Goal: Construct a CFG G recognizing $A \cup B$

$G_A = (V_A, \Sigma_A, R_A, S_A)$
$G_B = (V_B, \Sigma_B, R_B, S_B)$
Relabel variables so V_A and V_B are disjoint

Construct $G = (V, \Sigma, R, S)$:

$V = V_A \cup V_B \cup \{S\}$, $\Sigma = \Sigma_A \cup \Sigma_B$,
$R =$

Closure under concatenation

Let A be a CFL generated by CFG G_A and let B be a CFL recognized by CFG G_B

Goal: Construct a CFG G recognizing $A \cup B$

$G_A = (V_A, \Sigma_A, R_A, S_A)$

$G_B = (V_B, \Sigma_B, R_B, S_B)$

Relabel variables so V_A and V_B are disjoint

Construct $G = (V, \Sigma, R, S)$:

$$V = V_A \cup V_B \cup \{S\}, \qquad \Sigma = \Sigma_A \cup \Sigma_B,$$
$$R = R_\mathrm{A} \cup R_B \cup \{S \to S_A S_B\}$$

Closure under star

Let A be a CFL generated by CFG G_A and let B be a CFL recognized by CFG G_B

Goal: Construct a CFG G recognizing $A \cup B$

$G_A = (V_A, \Sigma_A, R_A, S_A)$
$G_B = (V_B, \Sigma_B, R_B, S_B)$
Relabel variables so V_A and V_B are disjoint

Construct $G = (V, \Sigma, R, S)$:

$V = V_A \cup \{S\}, \qquad \Sigma = \Sigma_A,$
$R =$

Closure under star

Let A be a CFL generated by CFG G_A and let B be a CFL recognized by CFG G_B

Goal: Construct a CFG G recognizing $A \cup B$

$G_A = (V_A, \Sigma_A, R_A, S_A)$

$G_B = (V_B, \Sigma_B, R_B, S_B)$

Relabel variables so V_A and V_B are disjoint

Construct $G = (V, \Sigma, R, S)$:

$$V = V_A \cup \{S\}, \qquad \Sigma = \Sigma_A,$$

$$R = R_{\mathrm{A}} \cup R_B \cup \{S \to \epsilon | S\ S_A\}$$

Closure Properties

- The class of CFLs is closed under the regular operations union, concatenation, star

- Are CFLs closed under complement?

Closure Properties

- The class of CFLs is closed under the regular operations union, concatenation, star

- Are CFLs closed under complement?
- What about intersection?

Context-Free Languages

L is a *context-free language* if it is the language of some CFG

Questions about CFLs

1. Which languages are *not* context-free?
2. What are the closure properties of CFLs?
3. How do we recognize whether $w \in L$?

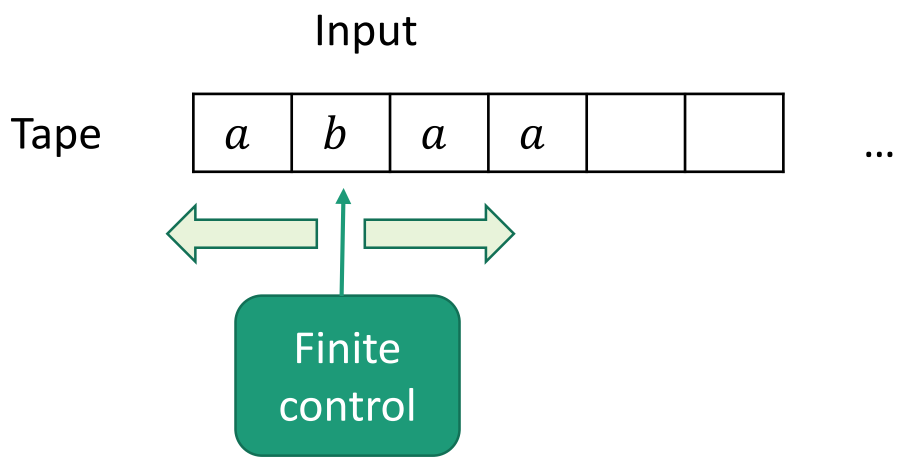
Recognizing CFLs

- Need to somehow extend NDAs… (Need memory!)
- Standard extension: “Pushdown automata (PDAs)”
  - NDA’s with limited memory (arranged as a stack)
  - Can:
    - Given any CFG G, construct a PDA P s.t. L(G)=L(P)
    - Given any PDA P, construct a CFG G s.t. L(G)=L(P)
  - Still, a bit unsatisfying since PDAs are non-deterministic…
- Non-determinism seems “inherent”: There exist “ambiguous CFGs” where some words have several parse-trees
- Can overcome by transforming a CFG to an equivalent one that is unambiguous.

We will skip this part, and answer the recognizability question more generally…

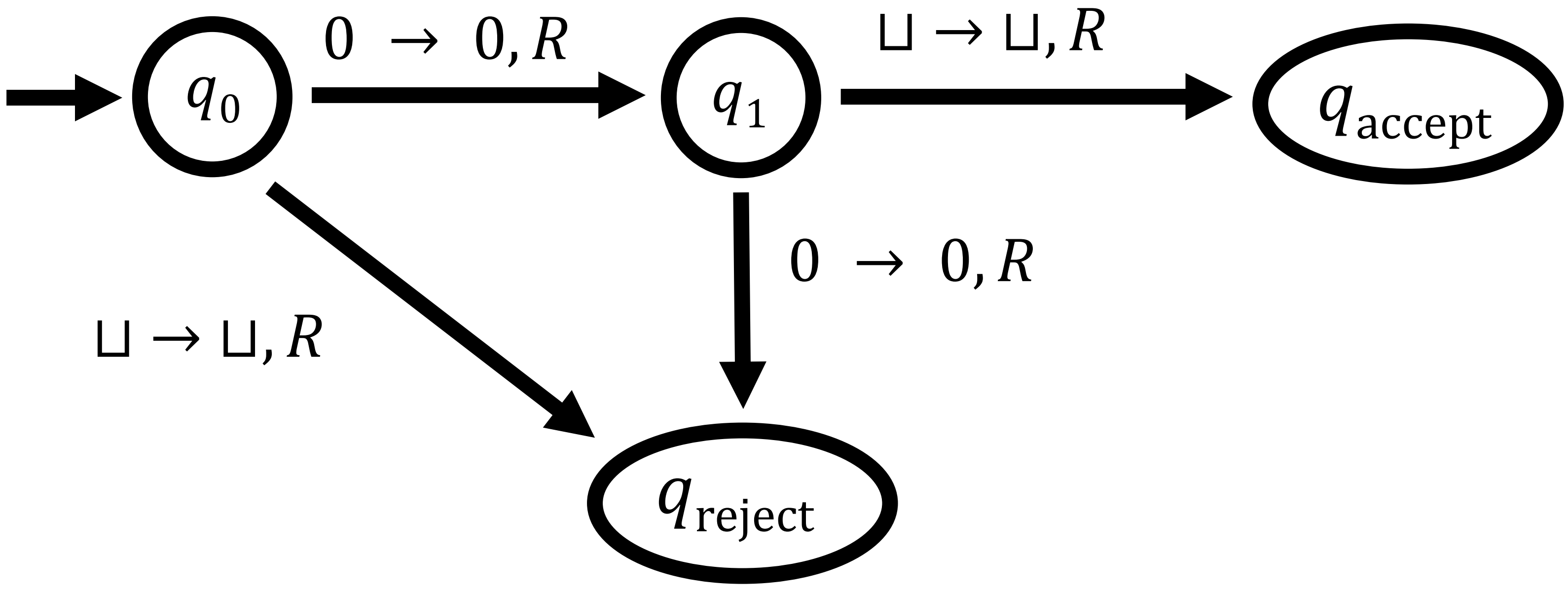
Turing Machines

The Basic Turing Machine (TM)

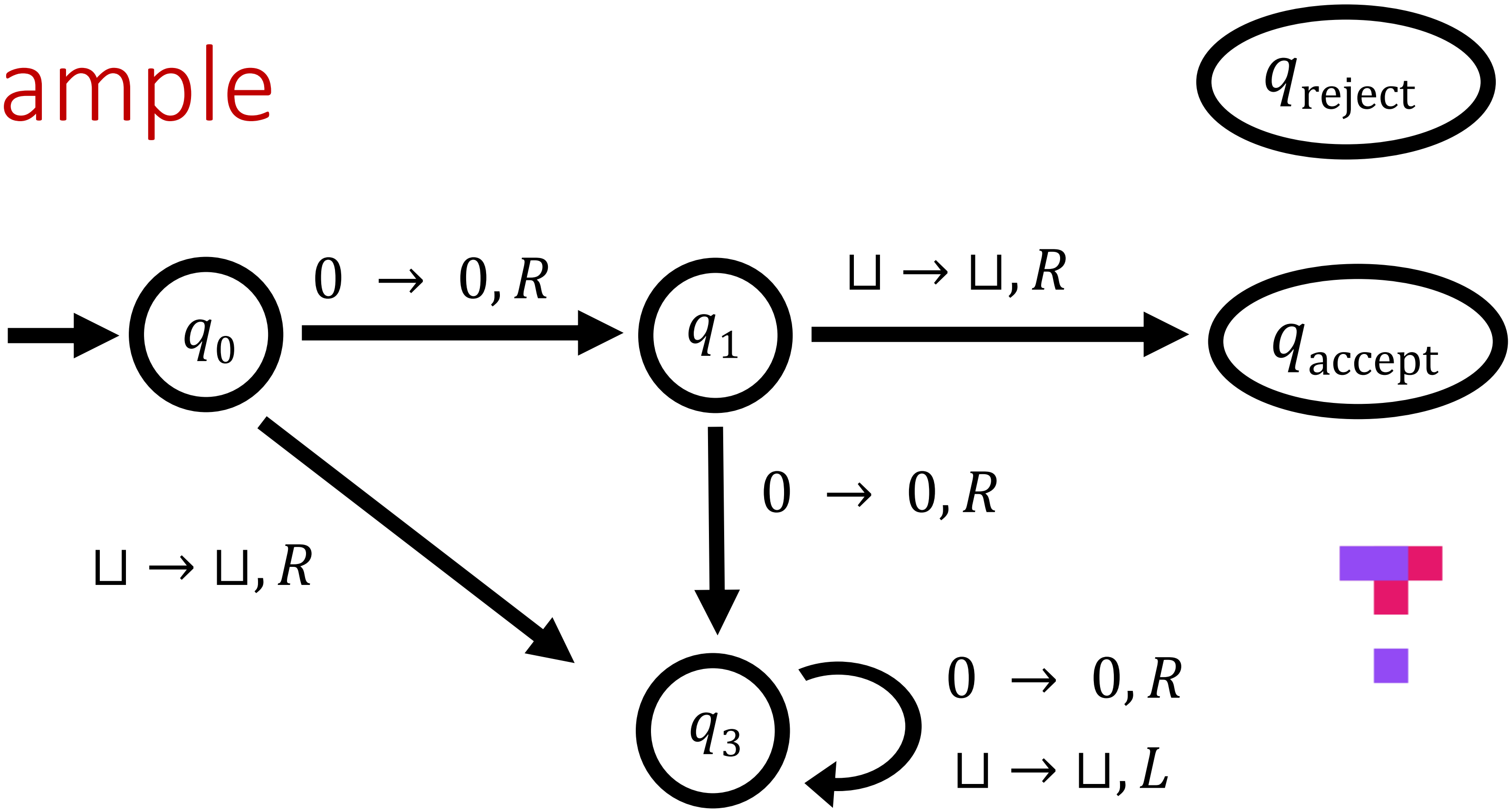


- Input is written on an infinitely long tape
- Head can both read and write, and move in both directions
- Computation halts when control reaches "accept" or "reject" state

Example


q_0
$0 \rightarrow 0, R$
q_1
$\sqcup \rightarrow \sqcup, R$
q_{accept}
$0 \rightarrow 0, R$
$\sqcup \rightarrow \sqcup, R$
q_{reject}

Example



q_{reject}
q_0
$0 \rightarrow 0, R$
q_1
$\sqcup \rightarrow \sqcup, R$
q_{accept}
$0 \rightarrow 0, R$
$\sqcup \rightarrow \sqcup, R$
q_3
$0 \rightarrow 0, R$
$\sqcup \rightarrow \sqcup, L$

TMs vs. Finite / Pushdown Automata

Three Levels of Abstraction

High-Level Description

An algorithm (like CS 330)

Implementation-Level Description

Describe (in English) the instructions for a TM

- How to move the head
- What to write on the tape

Low-Level Description

State diagram or formal specification

Example

Decide if $w \in A = \{0^{2^n} \mid n \geq 0\}$

High-Level Description

Repeat the following:

- If there is exactly one 0 in w, accept
- If there is an odd number of 0s in w $(> \ 1)$, reject
- Delete half of the 0s in w

Example

Decide if $w \in A = \{0^{2^n} \mid n \geq 0\}$

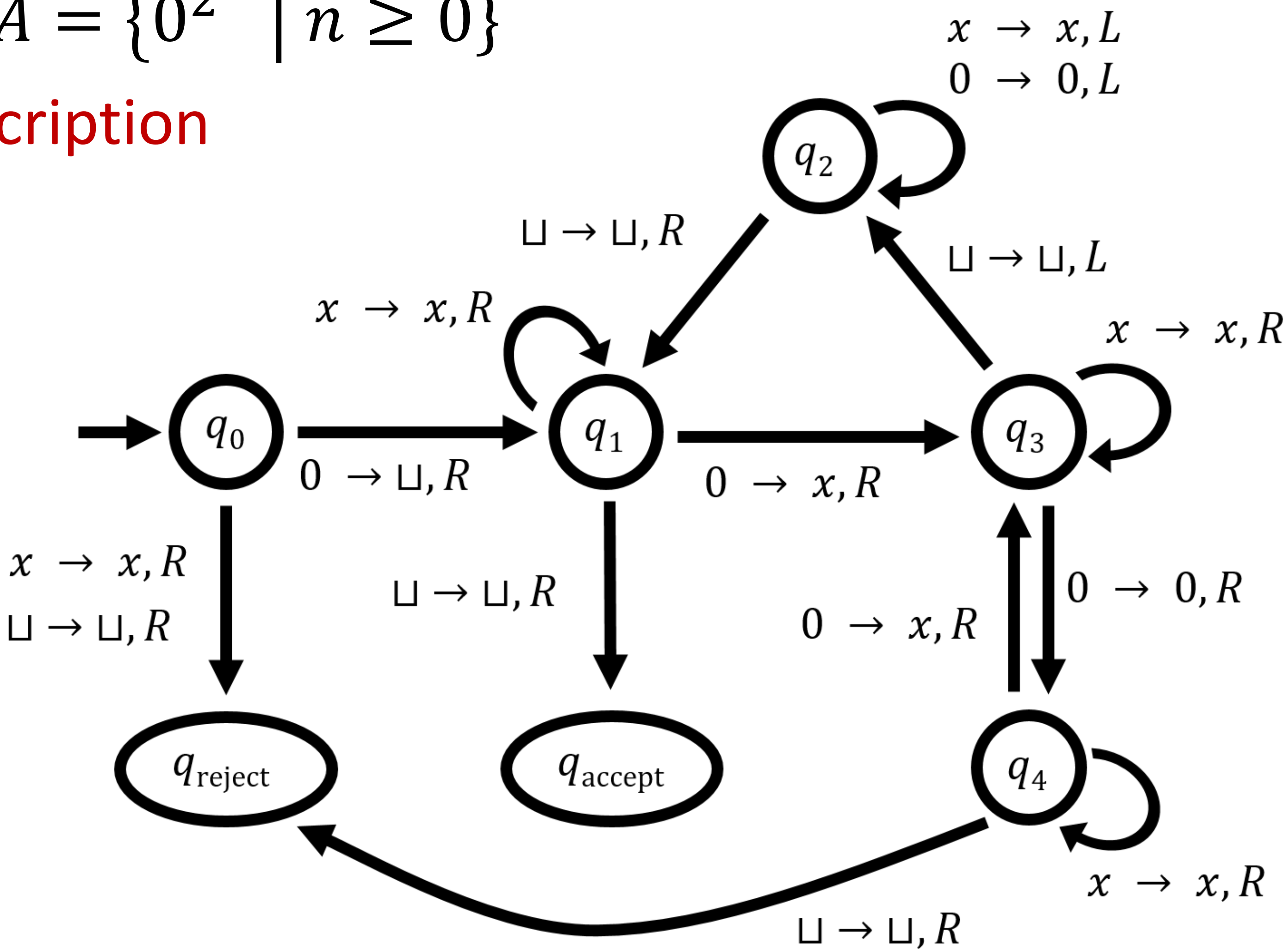
Implementation-Level Description

1. While moving the tape head left-to-right:
   a) Cross off every other 0
   b) If there is exactly one 0 when we reach the right end of the tape, accept
   c) If there is an odd number of 0s when we reach the right end of the tape, reject
2. Return the head to the left end of the tape
3. Go back to step 1

Example

Decide if $w \in A = \{0^{2^n} \mid n \geq 0\}$

Low-Level Description



Formal Definition of a TM

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

- Q is a finite set of states
- Σ is the input alphabet (does **not** include ⊔)
- Γ is the tape alphabet (contains ⊔ and Σ)
- δ is the transition function

…more on this later

- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state ($q_{\text{reject}} \neq q_{\text{accept}}$)

TM Transition Function

$$\delta : Q \times \Gamma \ \rightarrow Q \times \Gamma \ \times \{L, R\}$$

L means "move left" and R means "move right"

$\delta(p, a) = (q, b, R)$ means:

- Replace a with b in current cell
- Transition from state p to state q
- Move tape head right

$\delta(p, a) = (q, b, L)$ means:

- Replace a with b in current cell
- Transition from state p to state q
- Move tape head left UNLESS we are at left end of tape, in which case don't move