Intro to Theory of Computation

CS
332

LECTURE 8

Last time:

- Pumping lemma for CFLs
- Proving that a language is not CF

Today:

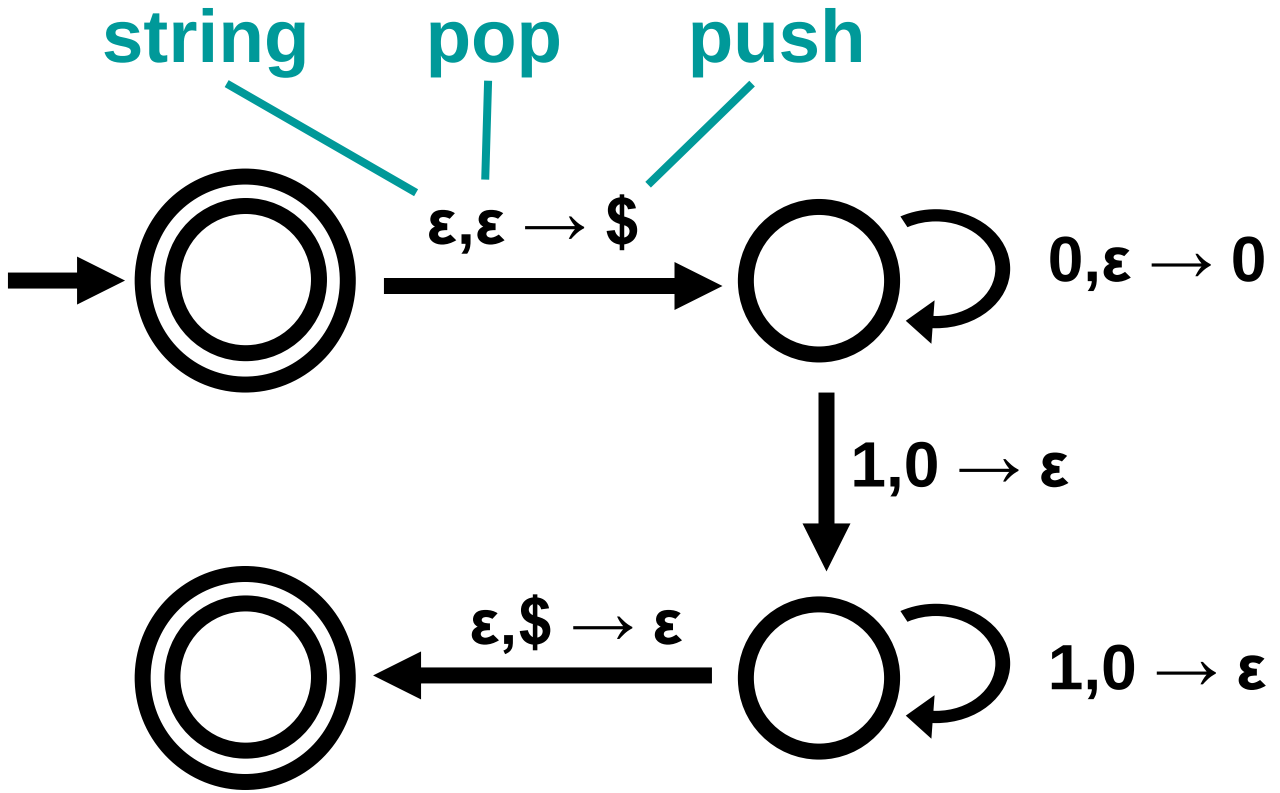
- Equivalence of CFGs and PDAs
- Turing Machines

Sofya Raskhodnikova

Read on your own

1. Ambiguous CFGs.
2. Chomsky normal form for CFGs
3. (Skipping Chapter 2.4 in Sipser).

PDAs: reminder

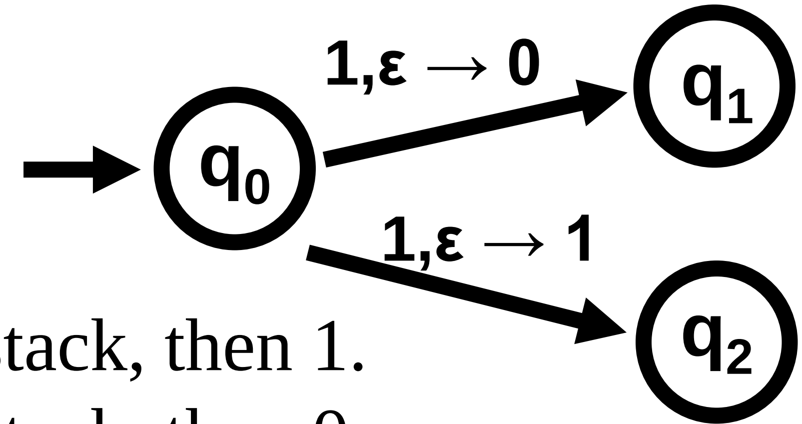


The language of P is the set of strings it accepts.
PDAs are nondeterministic.

CS 332

Exercise

When PDA takes a nondeterministic step, what happens to the stack?



A. First 0 is pushed onto the stack, then 1.
B. First 1 is pushed onto the stack, then 0.
C. Now PDA has access to two stacks and it pushes 0 onto one and 1 onto the other.
D. Two different computational branches are available to the PDA: it pushes exactly one symbol (0 or 1) onto the stack on each branch.
E. None of the above

Equivalence of CFGs & PDAs

A language is generated by a CFG

$\Leftrightarrow$

It is recognized by a PDA

Converting a CFG to a PDA

Suppose L is generated by a CFG G = (V, Σ, R, S).

Construct a PDA P = (Q, Σ, Γ, δ, q, F)
that recognizes L.

Idea: P will guess steps of a derivation of its input w and use its stack to derive it.

Algorithmic description of PDA

(1) Place the marker symbol \$ and the start variable on the stack.

(2) Repeat forever:

(a) If a variable A is on top of the stack, and $(A \to s) \in R$, pop A and push string s on the stack in reverse order.

Choose the rule from R nondeterministically.

(b) If a terminal is on top of the stack, pop it and match it with input.

(3) On $(\varepsilon, \$)$, accept.

CS 332

Designing states of PDA

(q_{start}) Push S$ and go to q_{loop}

(q_{loop}) Repeat the following steps forever:

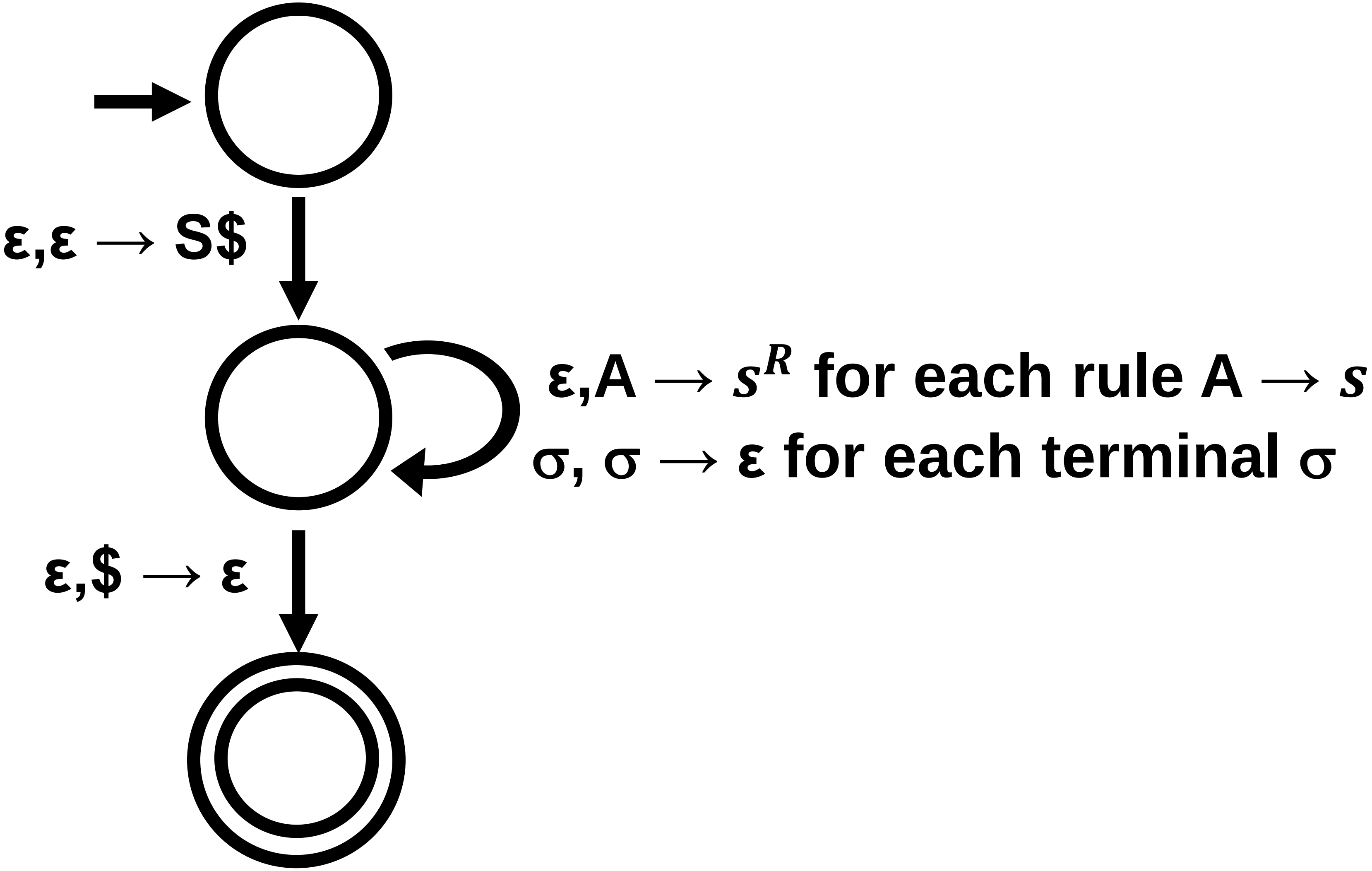
(a) On (ε,A) where ($A \to s$) $\in$ R, push s^R and go to q_{loop}

(b) On (σ,σ), pop σ and go to q_{loop}

(c) On (ε,$) go to q_{accept}

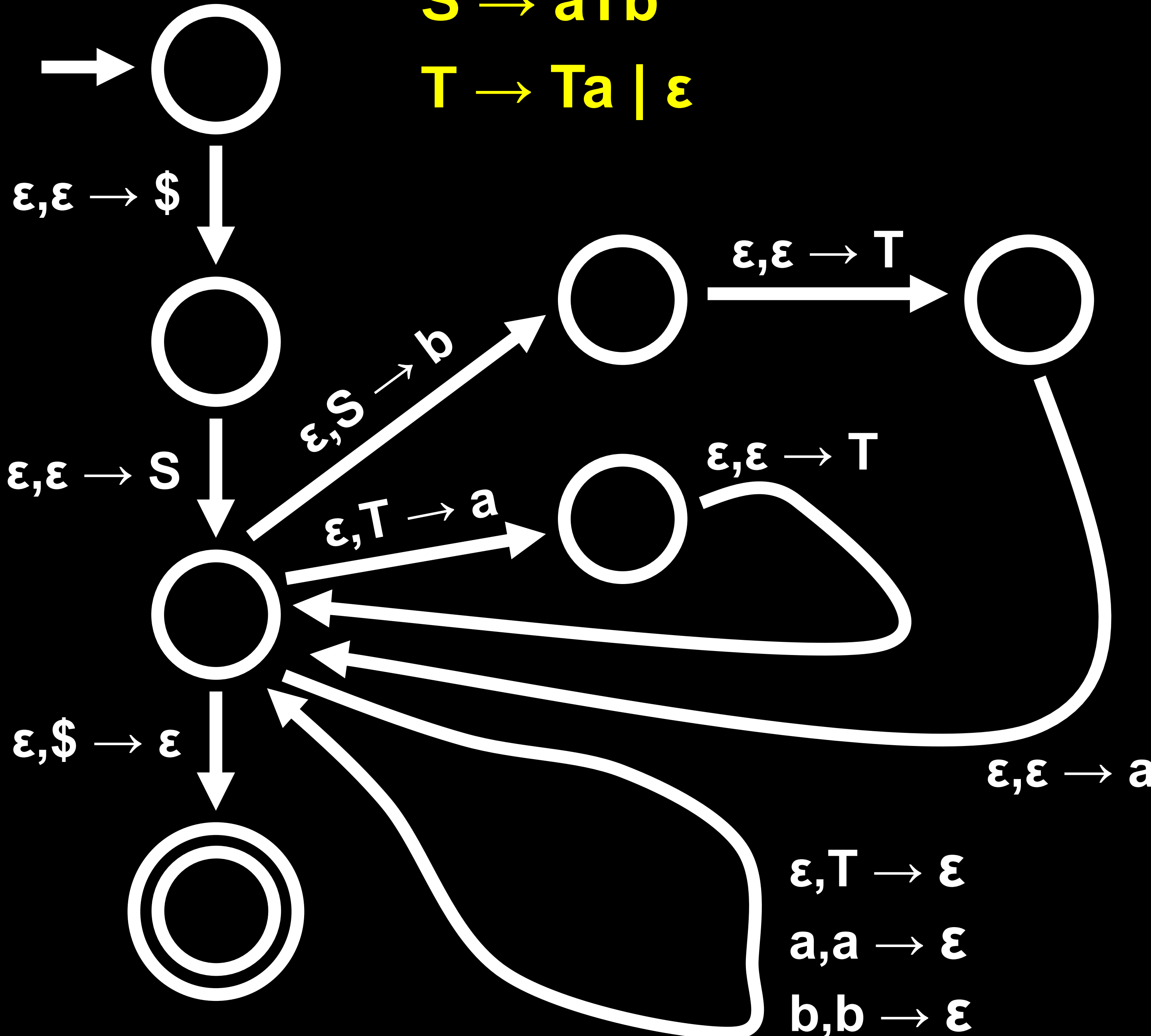
Otherwise, the PDA will get stuck!

Designing PDA



$S \rightarrow aTb$

$T \rightarrow Ta \mid \varepsilon$



A language is generated by a CFG

$\Rightarrow$

It is recognized by a PDA

Converting a PDA to a CFG

Given PDA $P = (Q, \Sigma, \Gamma, \delta, q, F)$

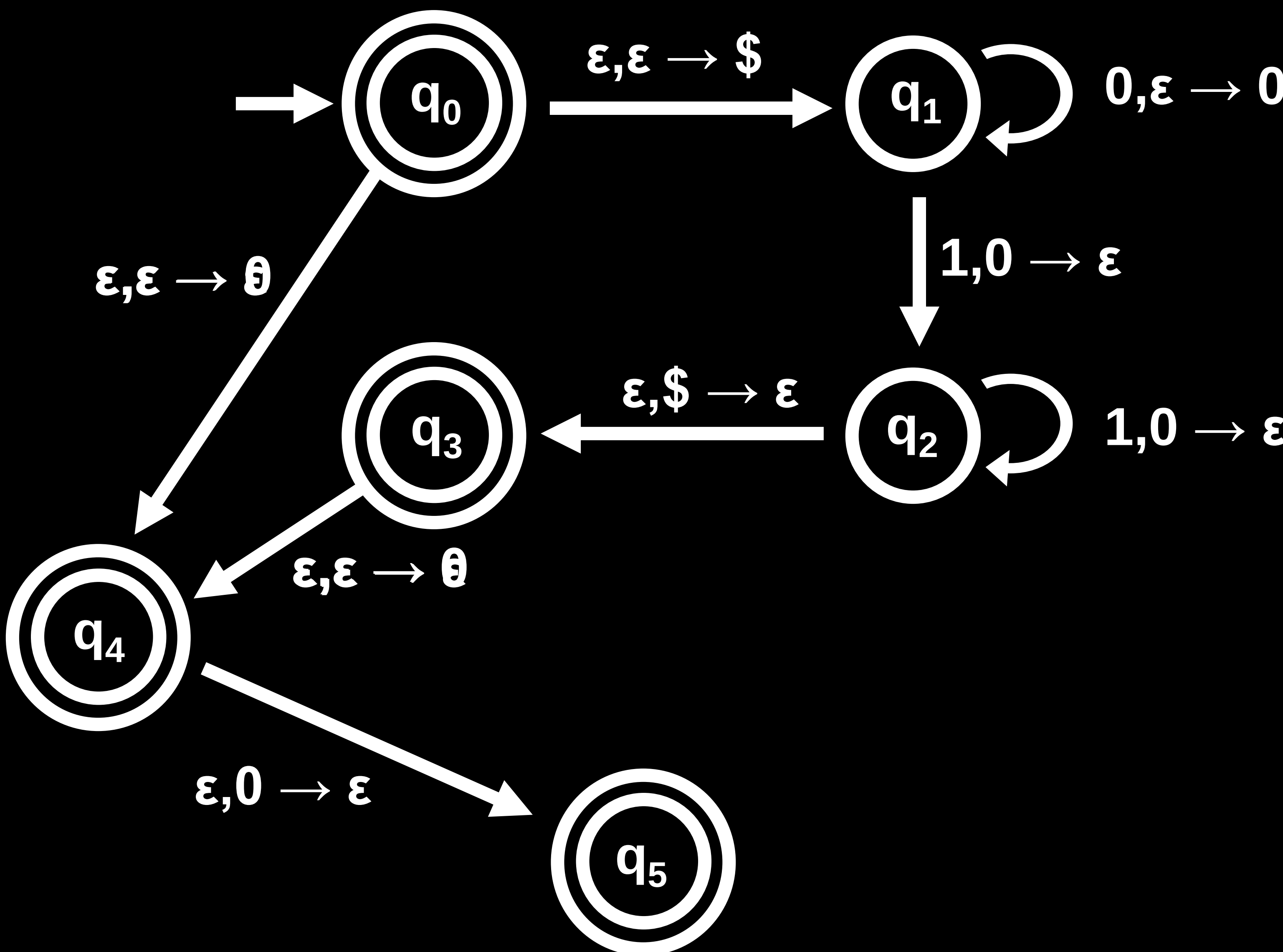
Construct a CFG $G = (V, \Sigma, R, S)$ with $L(G)=L(P)$

First, simplify P so that:

1. **It has a single accept state, q_{accept}**
2. **It empties the stack before accepting**
3. **Each transition does exactly one of:**

- **pushes a symbol;**
- **pops a symbol.**

SIMPLIFY



From PDA to CFG: main idea

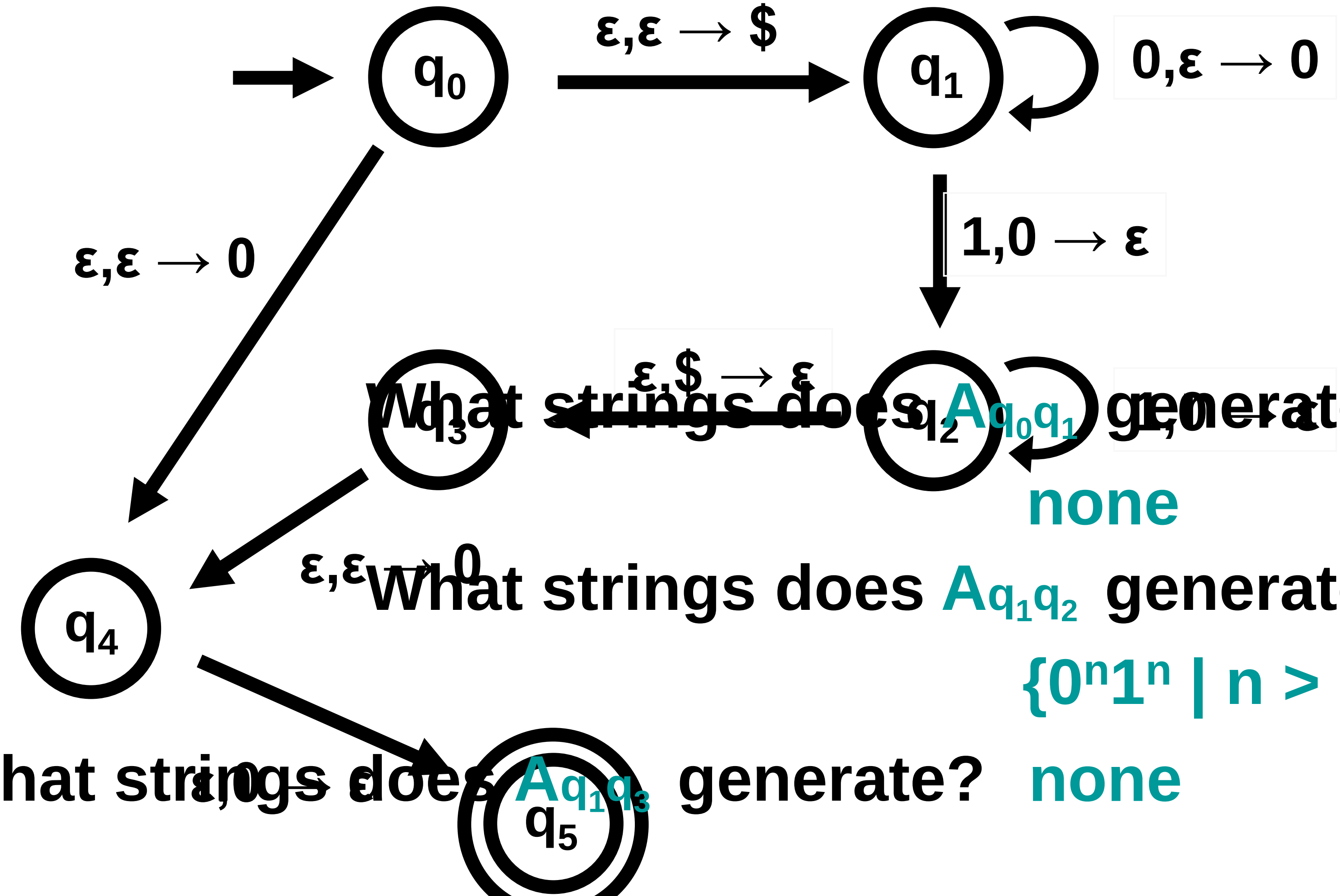
For each pair of states p and q in P, add a variable A_{pq} to CFG that generates all strings that that can take P from p to q without changing the stack*

$$V = \{A_{pq} \mid p,q \in Q\}$$

$$S = A_{q_0 q_{accept}}$$

*Starting from any stack S in p, including empty stack, P has stack S at q.

Example



What strings does $A_{q_0q_1}$ generate?

none

What strings does $A_{q_1q_2}$ generate?

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

What strings does $A_{q_1q_3}$ generate? none

From PDA to CFG: main idea

A_{pq} generates all strings that take P from p to q without changing the stack

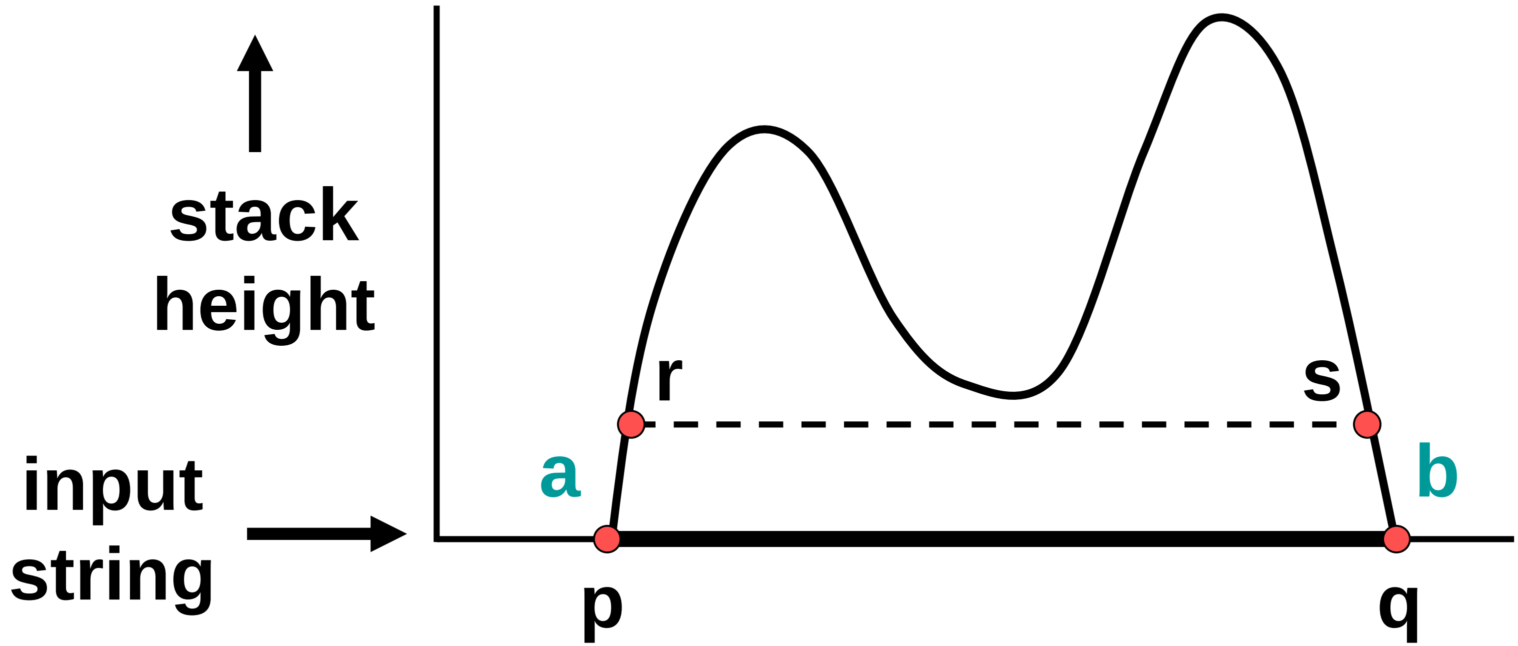
Let x be such a string

- P's **first** move on x must be a **push**
- P's **last** move on x must be a **pop**

Consider the stack while reading x. Either:

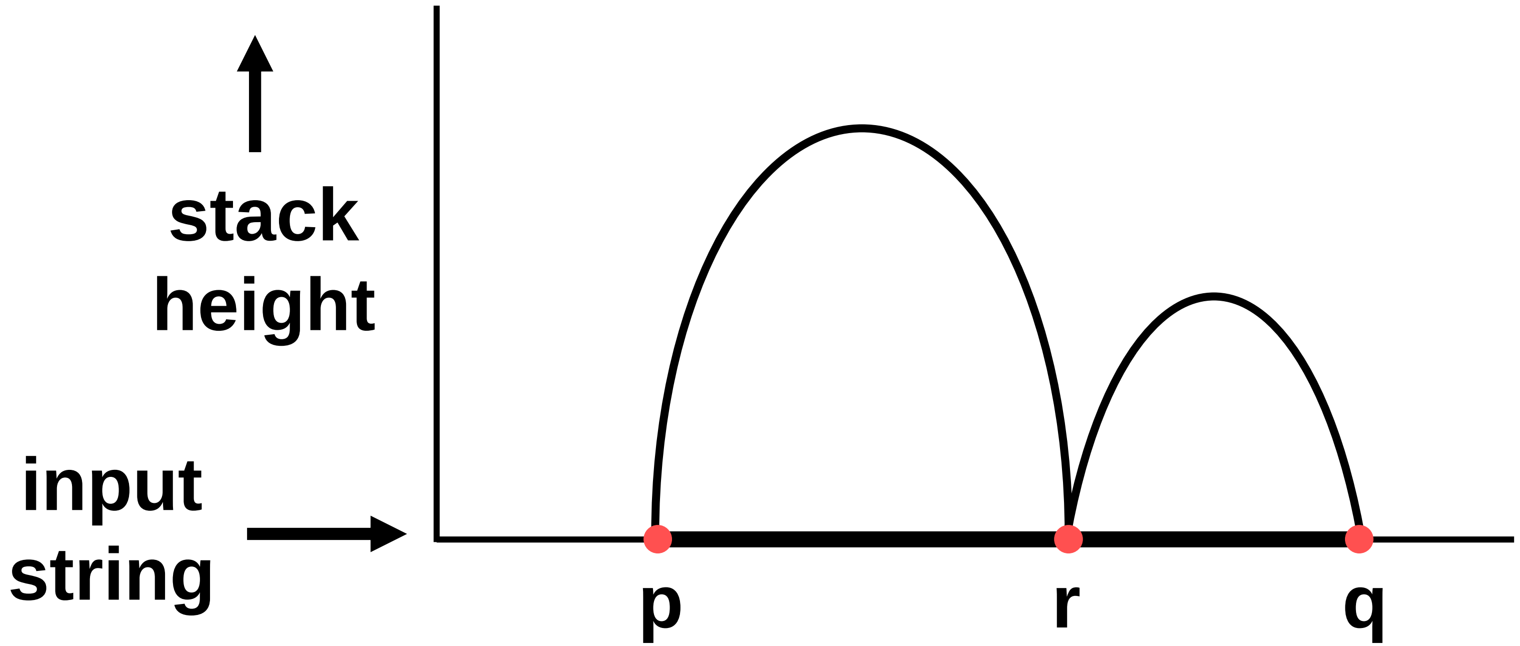
1. New portion of the stack first empties **only at the end of x**
2. New portion empties **before the end of x**

1. New portion of the stack first empties only at the end of x



$$A_{pq} \rightarrow aA_{rs}b$$

2. New portion empties before the end of x



$$A_{pq} \rightarrow A_{pr}A_{rq}$$

CFG construction

$V = \{A_{pq} \mid p,q \in Q\}$

$S = A_{q_0 q_{accept}}$

For each $p,q,r,s \in Q$, $t \in \Gamma$ and $a,b \in \Sigma_\varepsilon$

If $(r,t) \in \delta(p,a,\varepsilon)$ and $(q,\varepsilon) \in \delta(s,b,t)$

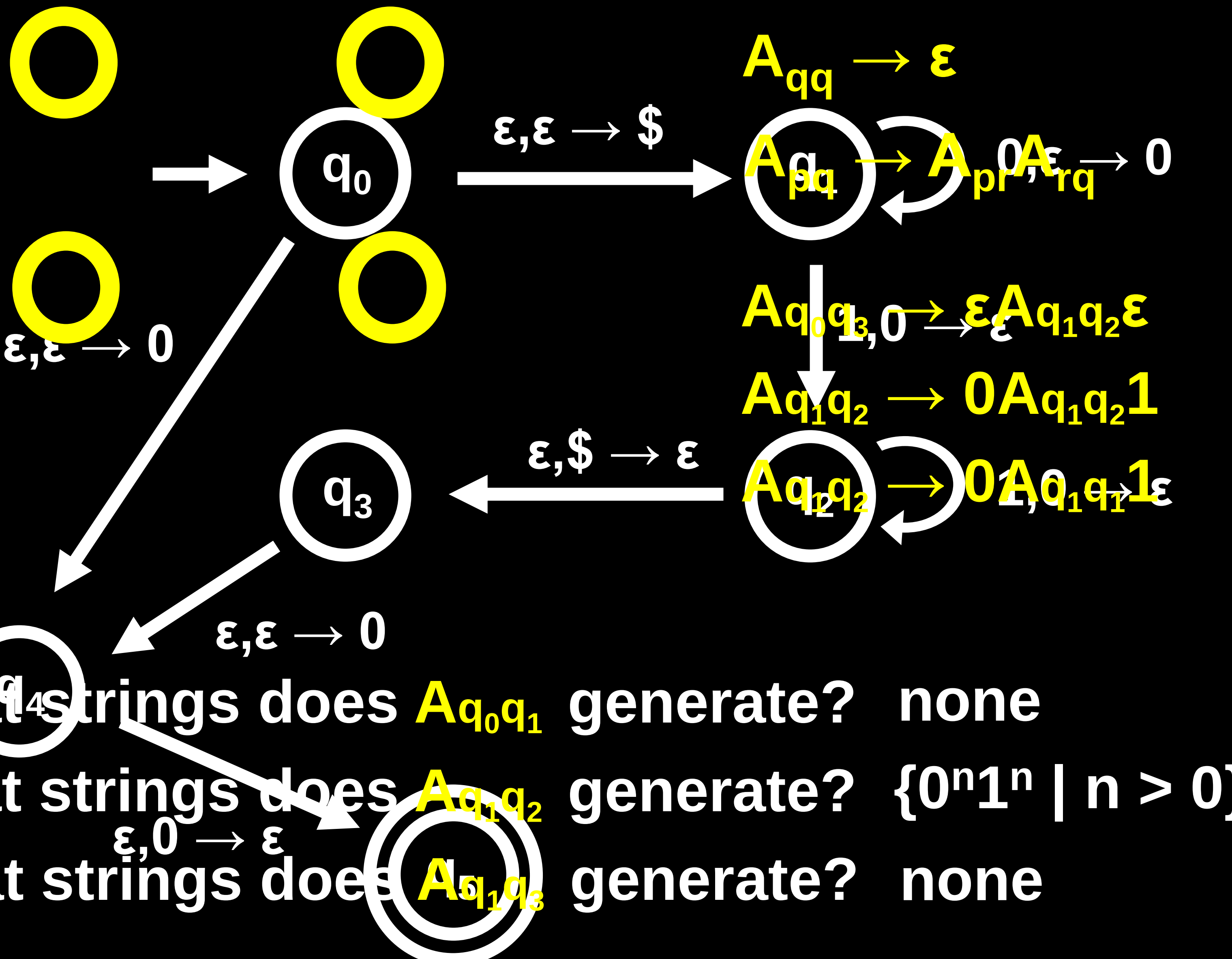
Then add the rule $A_{pq} \rightarrow aA_{rs}b$

For each $p,q,r \in Q$,

add the rule $A_{pq} \rightarrow A_{pr}A_{rq}$

For each $p \in Q$,

add the rule $A_{pp} \rightarrow \varepsilon$



$A_{qq} \to \varepsilon$

$A_{pq} \to A_{pr}A_{rq}$

$A_{q_0q_3} \to \varepsilon A_{q_1q_2}\varepsilon$

$A_{q_1q_2} \to 0A_{q_1q_2}1$

$A_{q_1q_2} \to 0A_{q_1q_1}1$

What strings does $A_{q_0q_1}$ generate? none

What strings does $A_{q_1q_2}$ generate? $\{0^n1^n \mid n > 0\}$

What strings does $A_{q_1q_3}$ generate? none

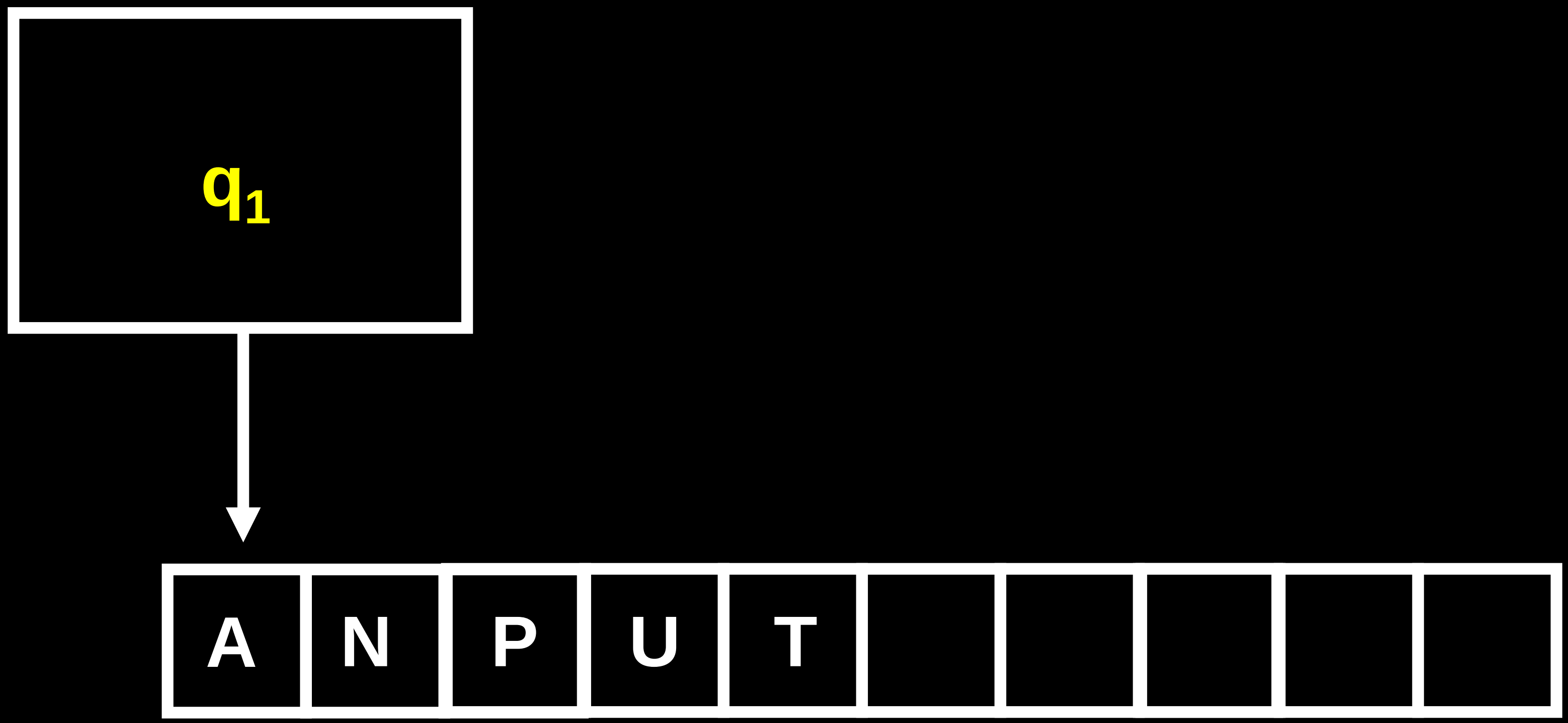
Equivalence of CFGs & PDAs

A language is generated by a CFG

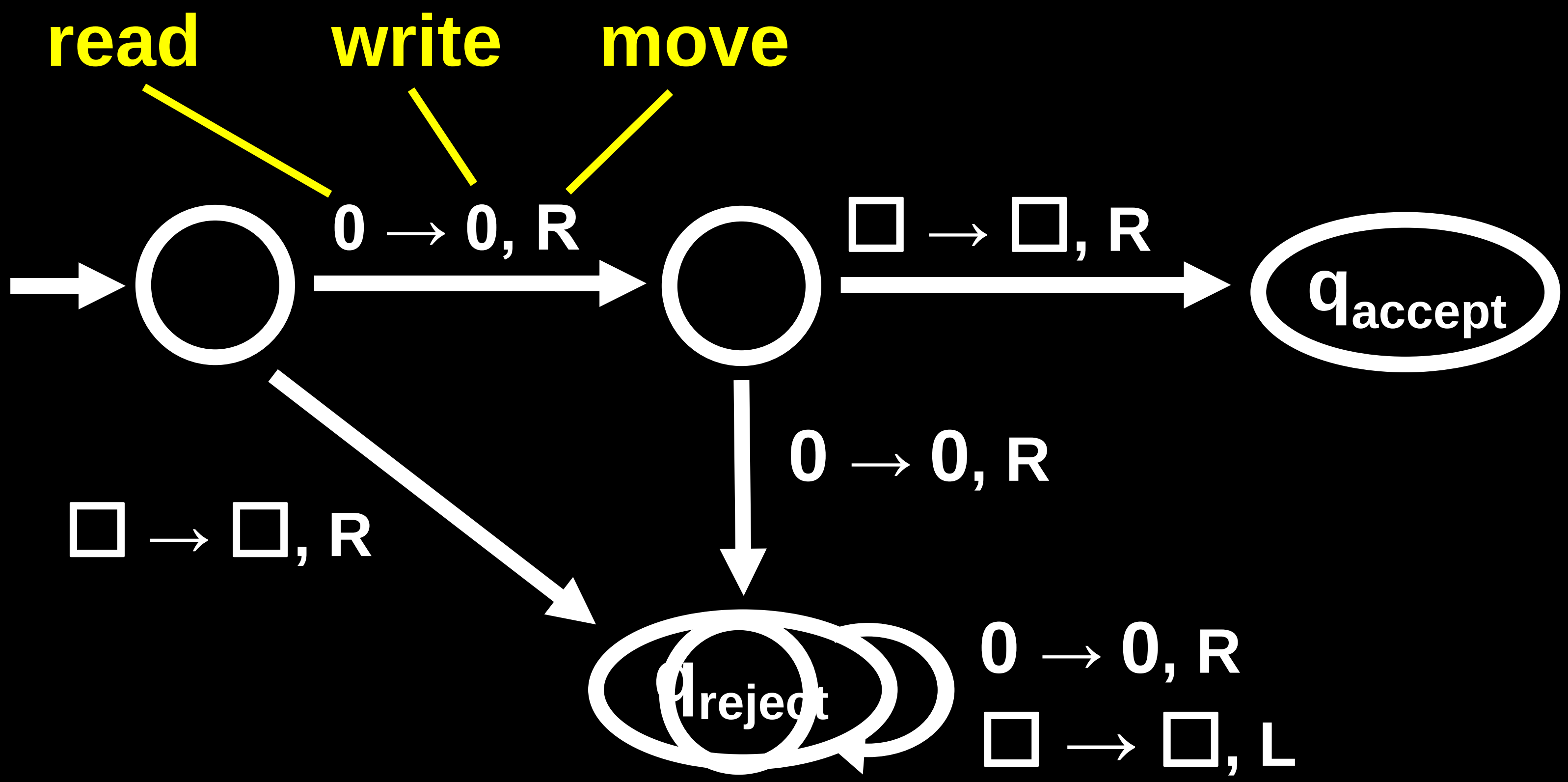
$\Leftrightarrow$

It is recognized by a PDA

TURING MACHINE (TM)



UNBOUNDED (on the right) TAPE



A TM can loop forever

TM versus PDA

TM can both write to and read from the tape

The head can move left and right

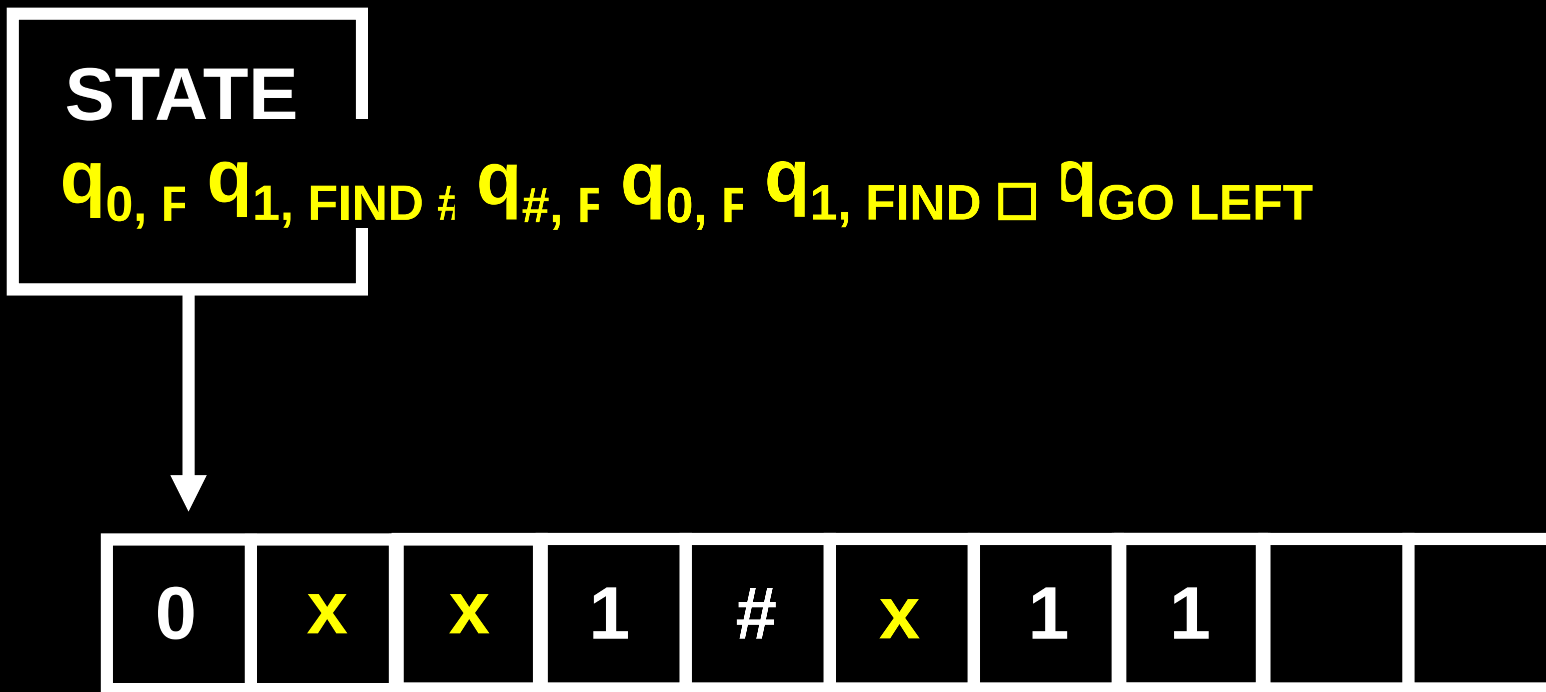
The input does not have to be read entirely

Accept and Reject take immediate effect

Infinite tape on the right, stick on the left

TM is deterministic (will consider NTMs later)

Testing membership in B = { w#w | w $\in$ {0,1}* }



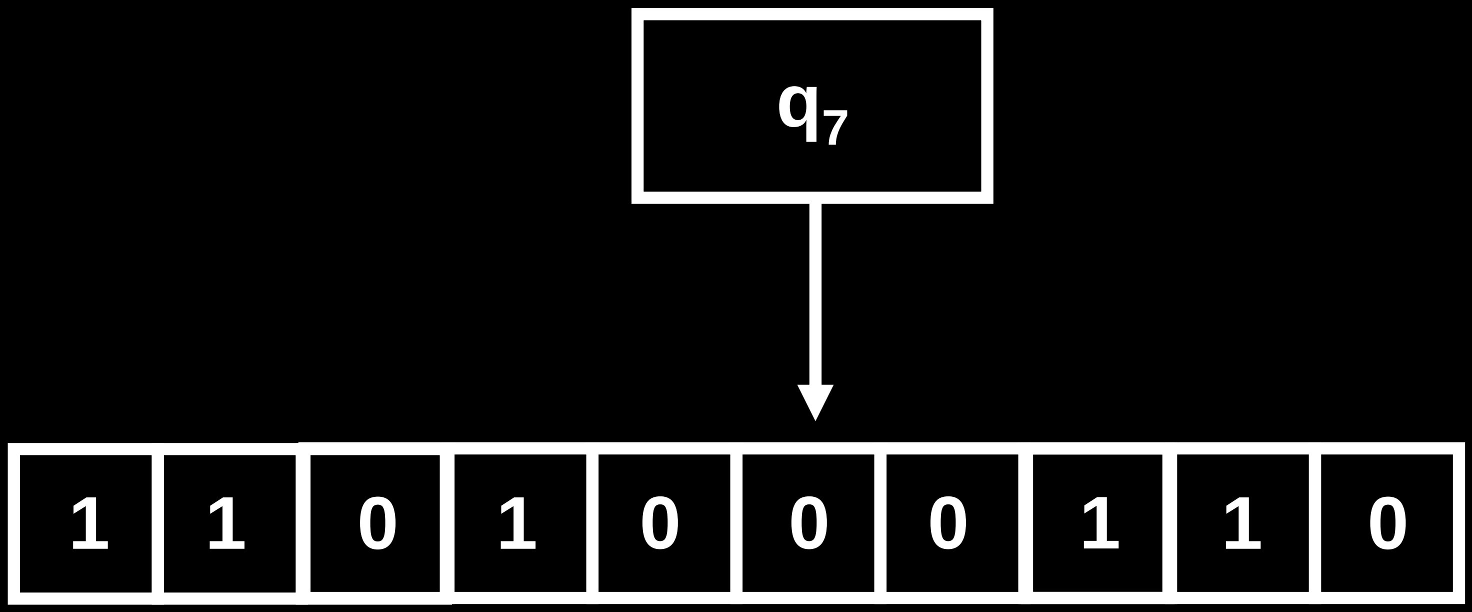
Definition of a TM

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

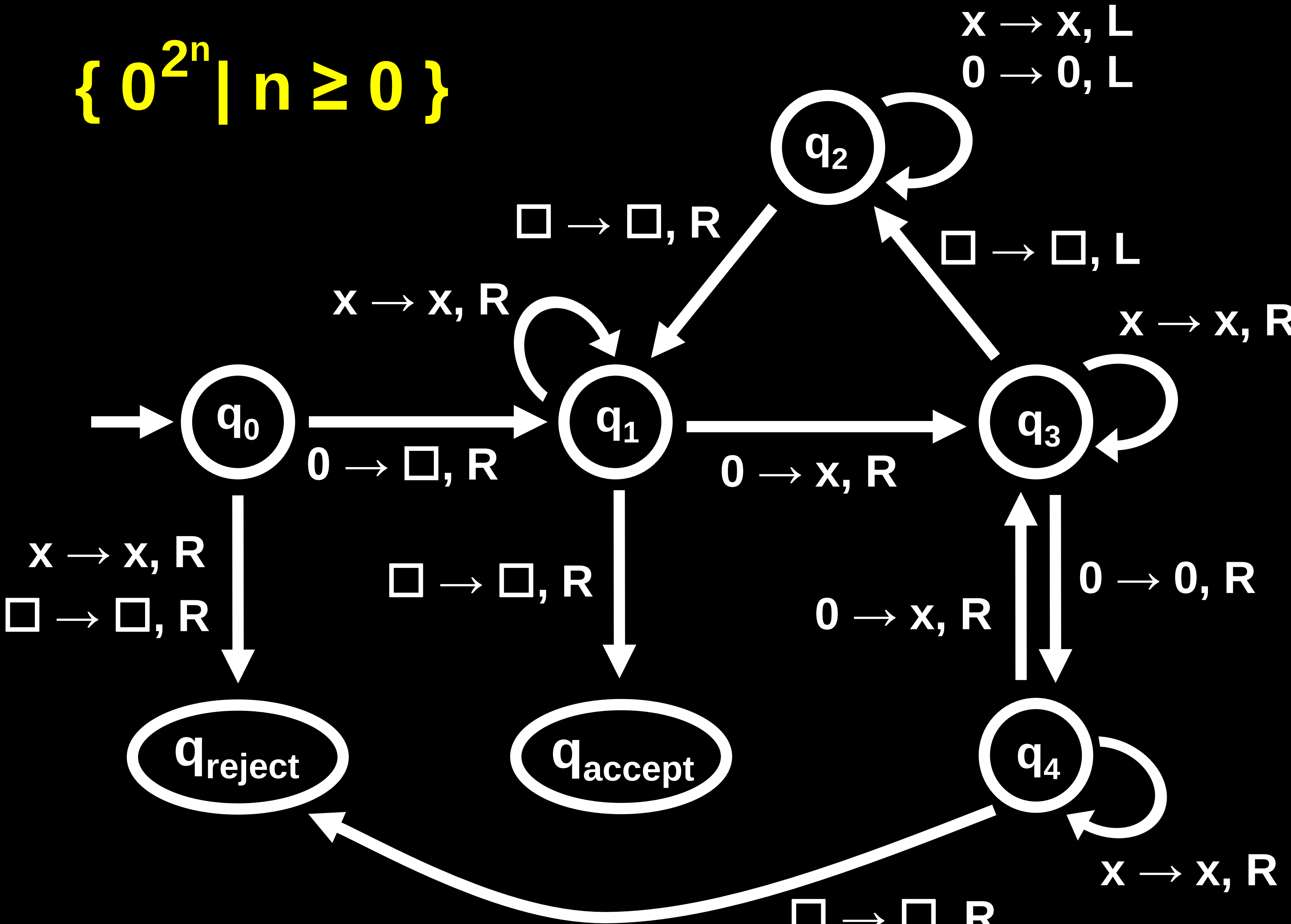
- Q is a finite set of states
- Σ is the input alphabet, where $\square \notin \Sigma$
- Γ is the tape alphabet, where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$
- $q_0 \in Q$ is the start state
- $q_{accept} \in Q$ is the accept state
- $q_{reject} \in Q$ is the reject state, and $q_{reject} \neq q_{accept}$

CONFIGURATIONS

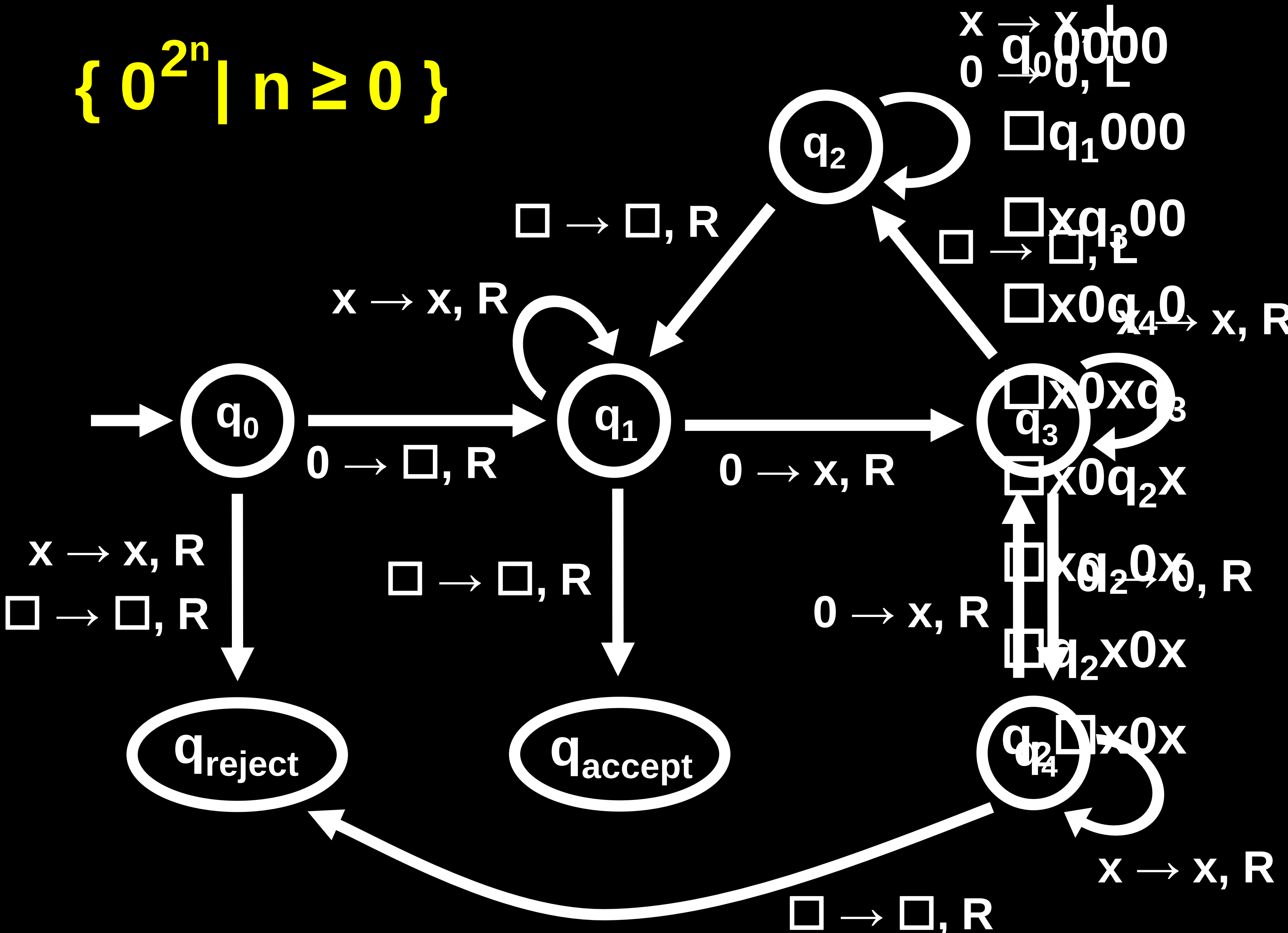
$$11010q_7 00110$$



$\{ 0^{2^n} \mid n \geq 0 \}$



$\{ 0^{2^n} \mid n \geq 0 \}$



CS 332

Accepting and rejecting

A TM on input sting w may

- **either halt (enter q_{accept} or q_{reject})**
- **or never halt (loop)**

A TM is a decider if it halts on every input.

Language of a TM

A TM **recognizes** a language L if it accepts all strings in L and no other strings.

- A language is called **recognizable** (or enumerable) if some TM recognizes it.

A TM **decides** a language L if it accepts all strings in L and rejects all strings not in L.

- A language is called **decidable** (or recursive) if some TM decides it.