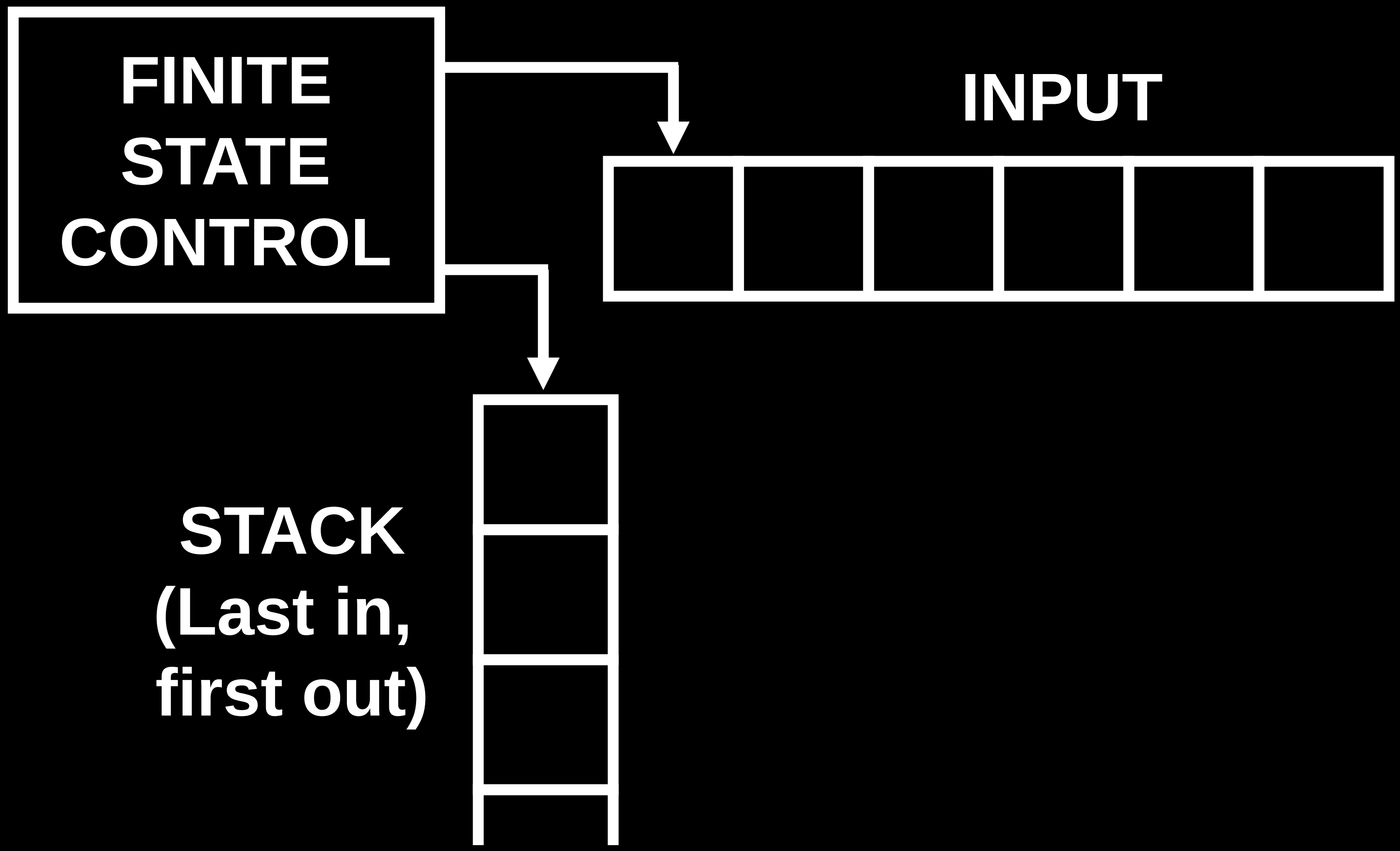
15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

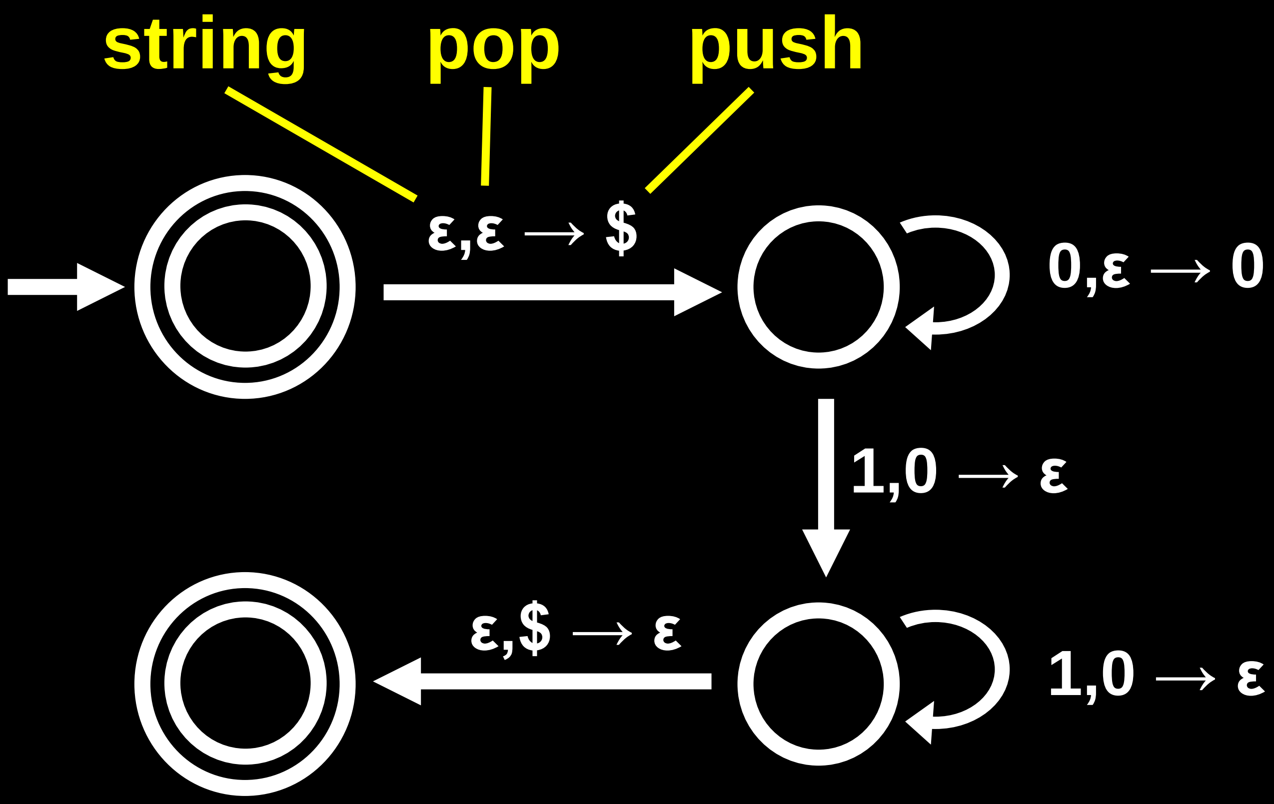
PDAs ARE EQUIVALENT TO CFGs

THURSDAY Jan 30

PUSHDOWN AUTOMATA (PDA)



PUSHDOWN AUTOMATA (PDA)



CONTEXT-FREE GRAMMARS

Production rules

variable → A

terminals → 0, 1

$A \rightarrow 0A1$

$A \rightarrow \varepsilon$

$\in (V \cup \Sigma)^*$

$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000111$

(yields)

$A \Rightarrow^* 000111$

(derives)

A Language L is generated by a CFG

$\Leftrightarrow$

L is recognized by a PDA

A Language L is generated by a CFG

$\Rightarrow$

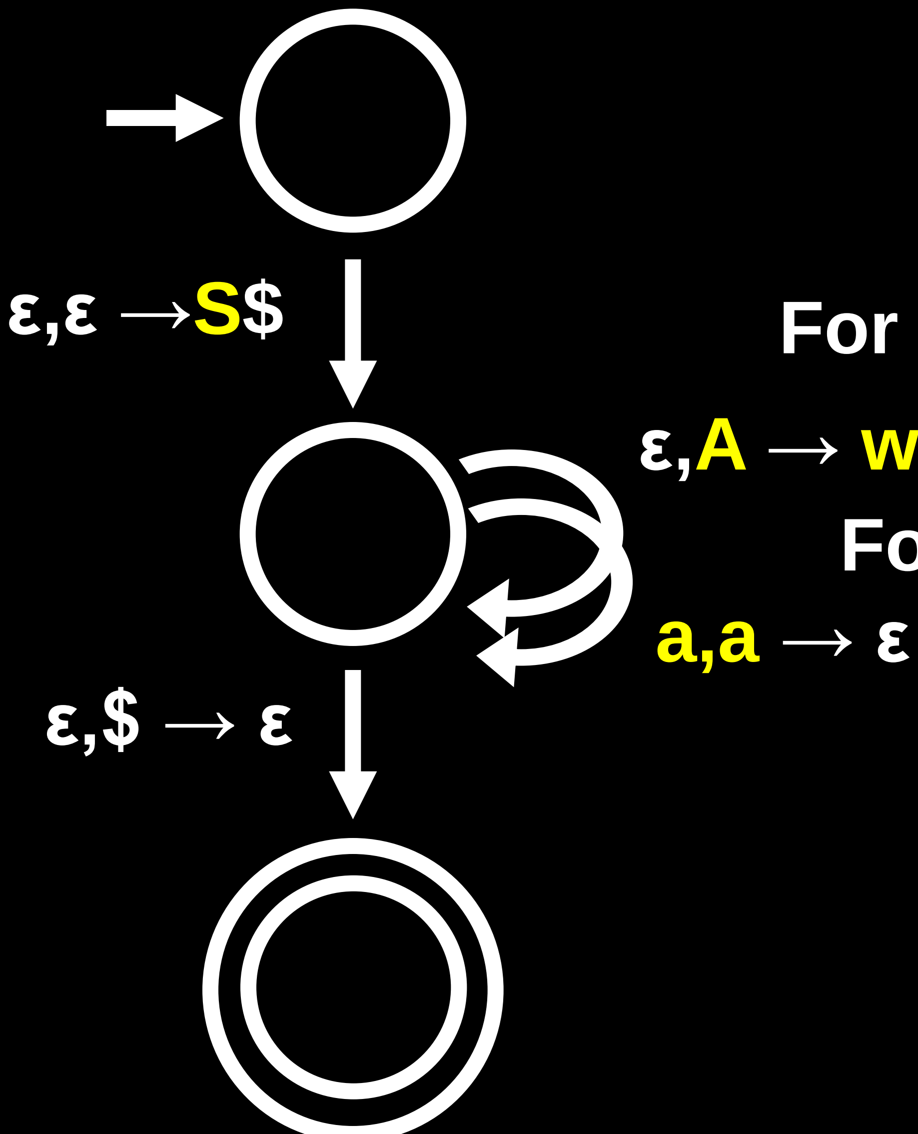
L is recognized by a PDA

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

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

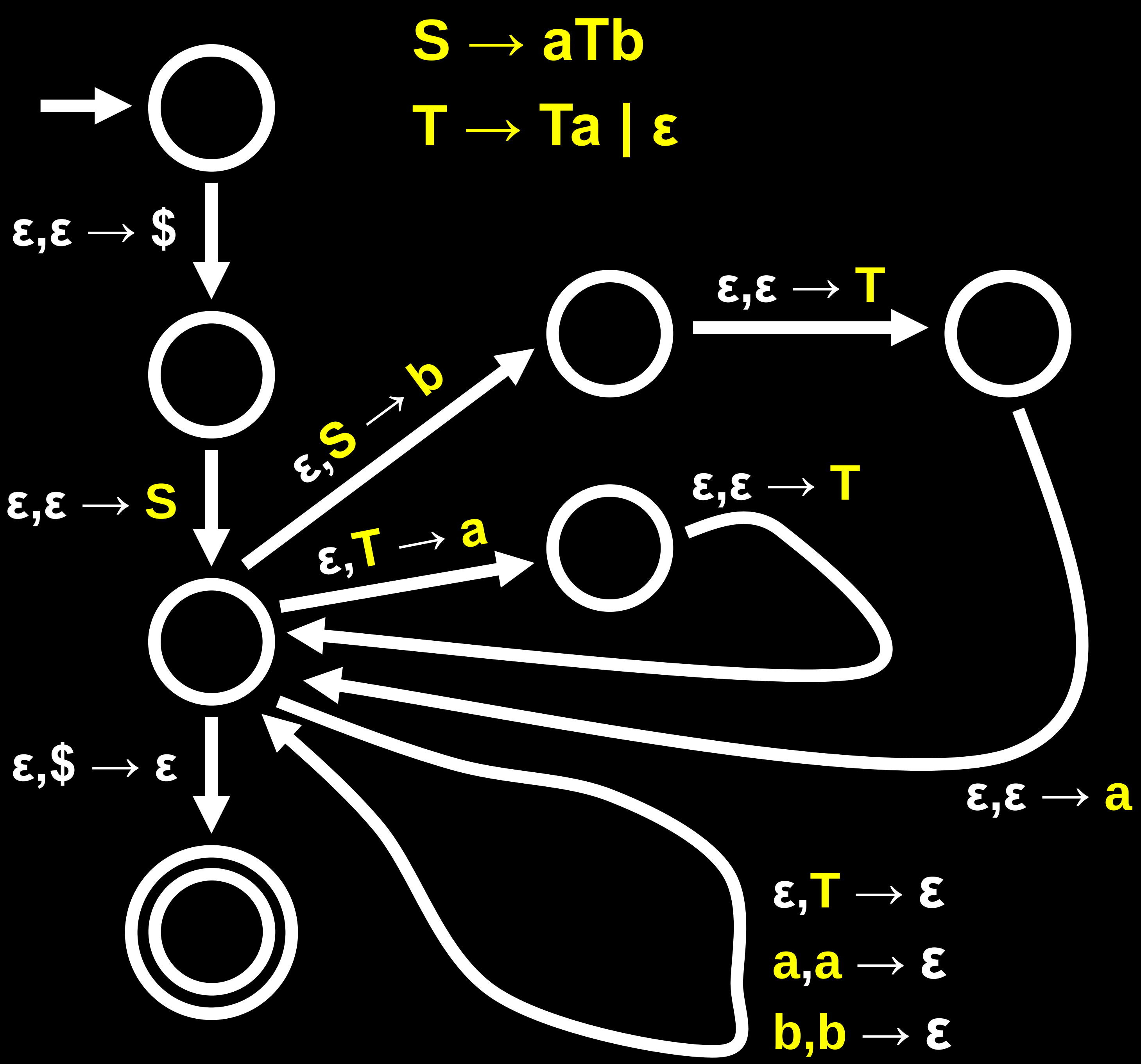
Suppose L is generated by a CFG G = (V, Σ, R, S)

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



For each rule 'A → w' ∈ R:

For each terminal a ∈ Σ:



S → aTb
T → Ta | ε
ε,ε → $
ε,ε → S
ε,S → b
ε,ε → T
ε,T → a
ε,ε → T
ε,ε → a
ε,$ → ε
ε,T → ε
a,a → ε
b,b → ε

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

Describe P = (Q, Σ, Γ, δ, q, F) that recognizes L (via pseudocode):

(1) Push $ and then S on the stack

(2) Repeat the following steps forever:

(a) Suppose x is now on top of stack

(b) If x is a variable A, guess a rule for A and push yield (in reverse) into the stack and Go to (a).

(c) If x is a terminal, read next symbol from input and compare it to x. If they're different, *reject*. If same, pop x and Go to (a).

(d) If x is $: then *accept* iff no more input

A Language L is generated by a CFG <= L is recognized by a PDA

Given PDA P = (Q, Σ, Γ, δ, q, F)

Construct a CFG G = (V, Σ, R, S) such that L(G) = L(P)

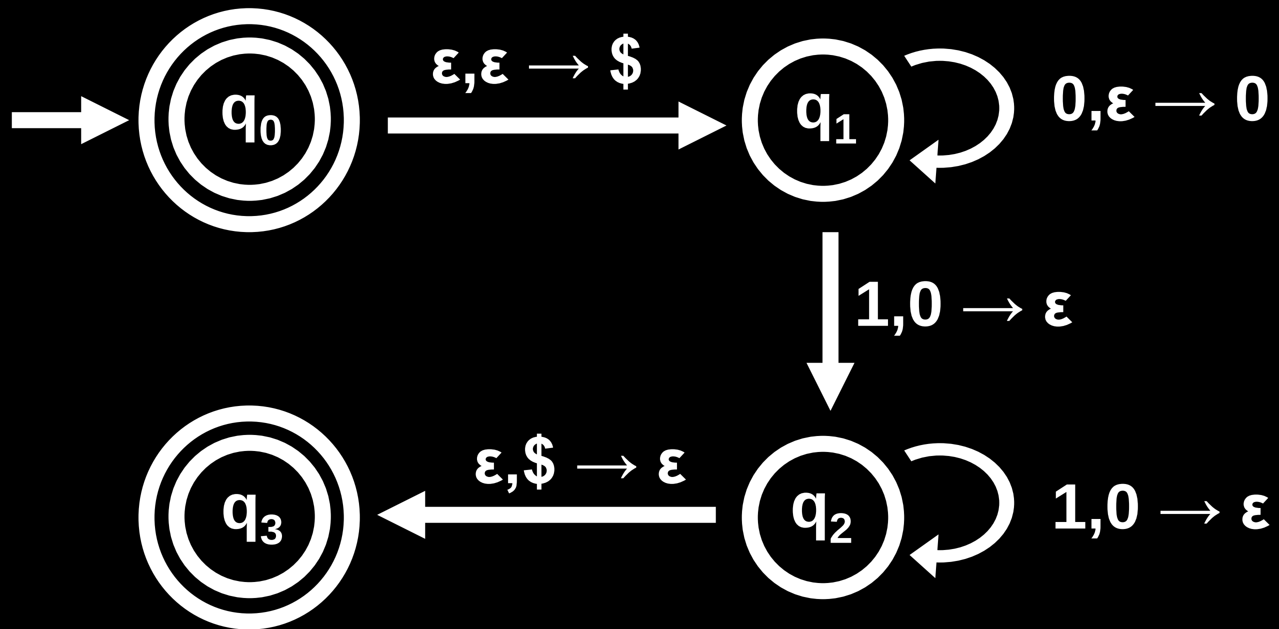
First, simplify P to have the following form:

(1) It has a unique accept state, q_{acc}

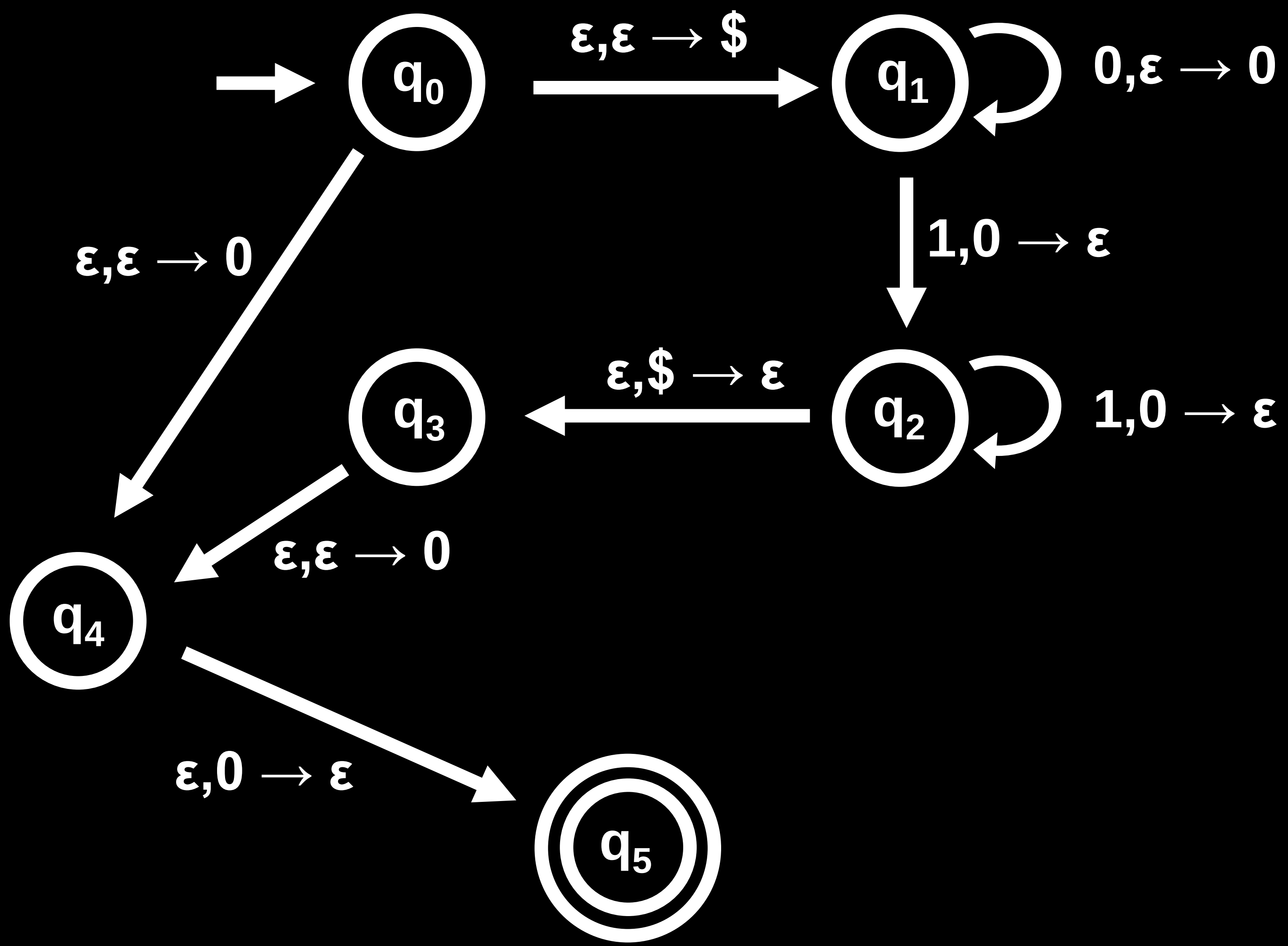
(2) It empties the stack before accepting

(3) Each transition either pushes a symbol or pops a symbol, but not both at the same time

SIMPLIFY



SIMPLIFY



Our task is to construct Grammar G to generate exactly the words that PDA P accepts.

Idea For Our Grammar G:

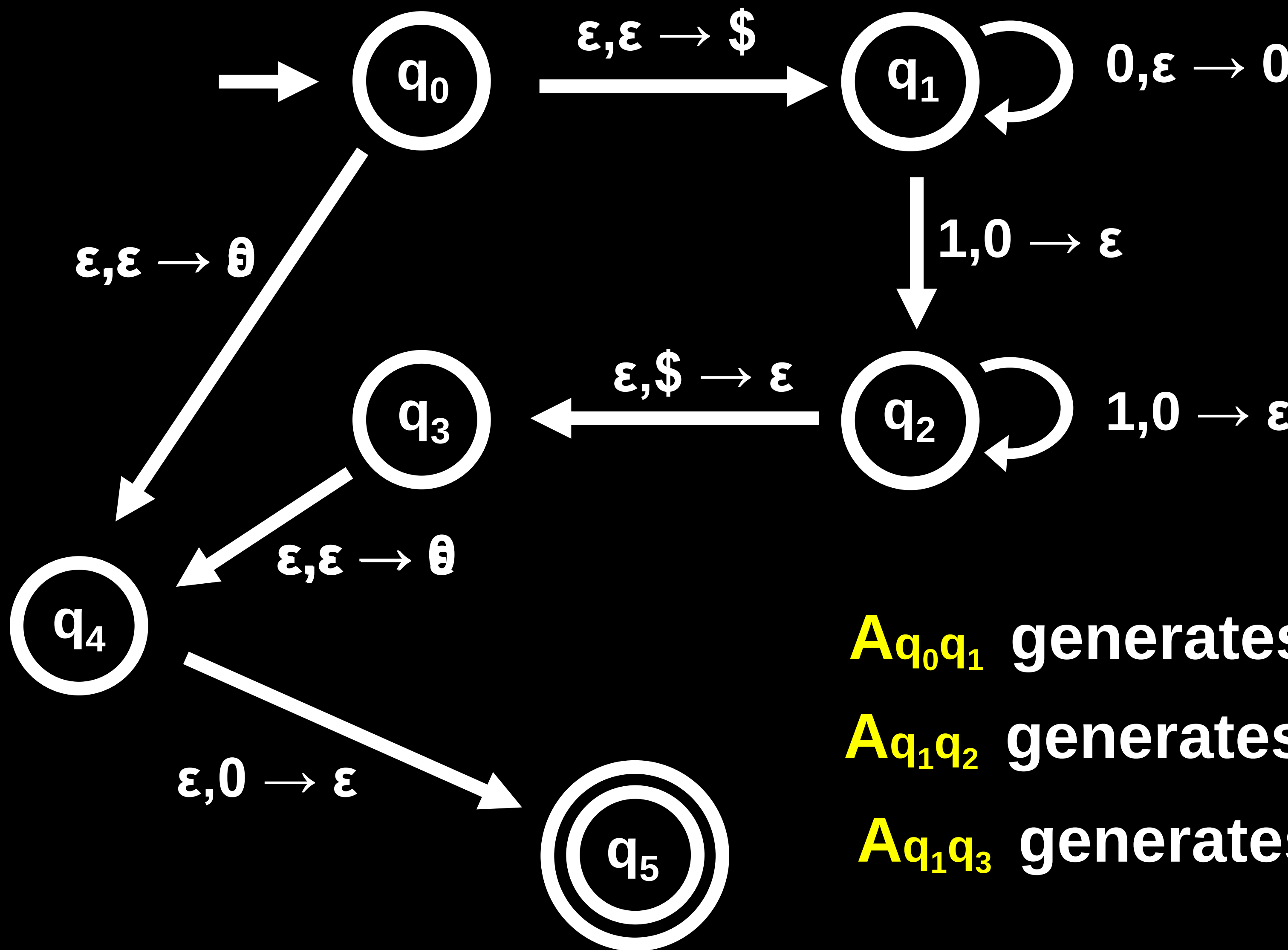
For every pair of states p and q in PDA P,

G will have a variable A_{pq} whose production rules will generate all strings x that can take:

P from p with an empty stack
to q with an empty stack

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

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



$A_{q_0q_1}$ generates?

$A_{q_1q_2}$ generates?

$A_{q_1q_3}$ generates?

WANT: A_{pq} generates all strings that take p with an empty stack to q with empty stack

Let x be such a string

- P's first move on x must be a push (why?)
- P's last move on x must be a pop

Two possibilities:

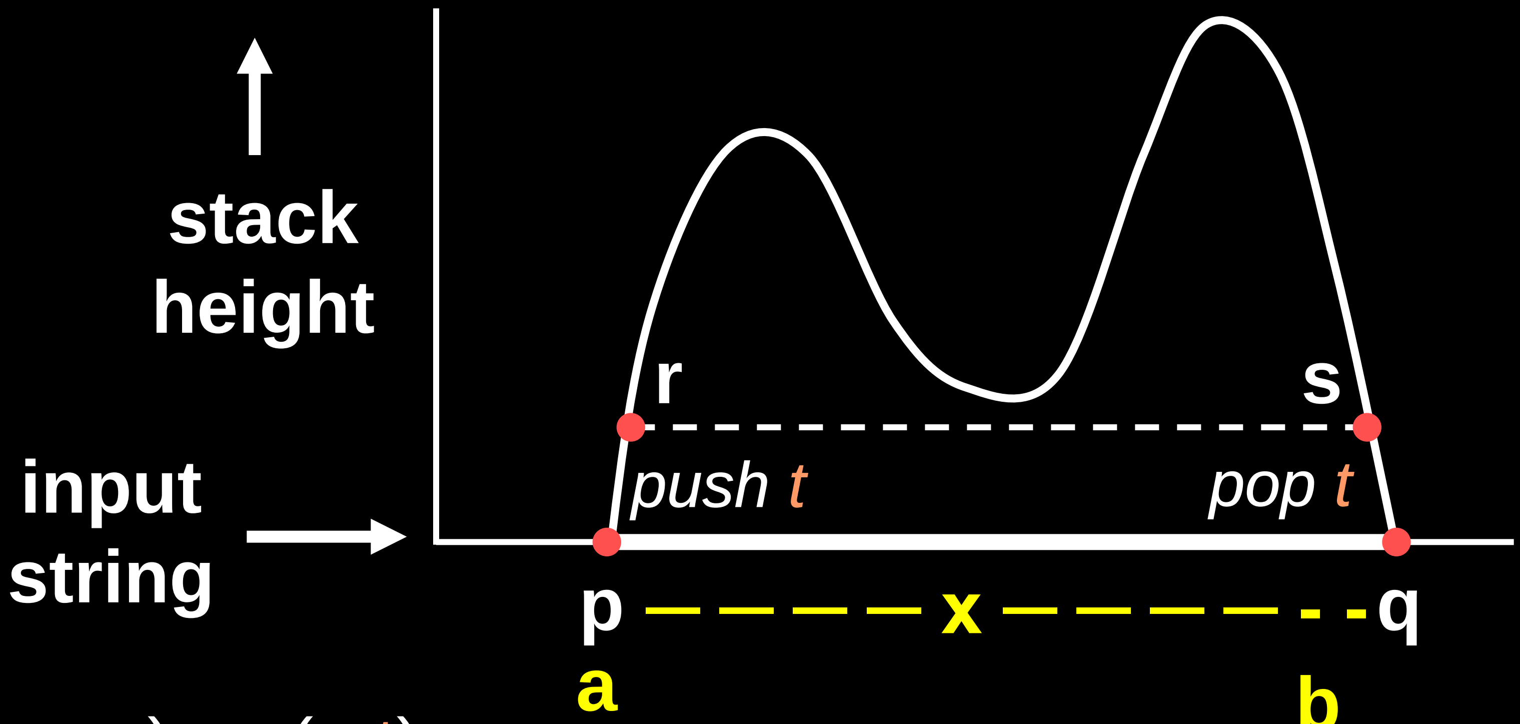
1. The symbol popped at the end is the one pushed at the beginning

2. The symbol popped at the end is not the one pushed at the beginning

(so P must empty stack somewhere in the middle, and then start pushing symbols on it again)

x = ayb takes **p** with empty stack to **q** with empty stack

1. The symbol t popped at the end is *exactly* the one pushed at the beginning

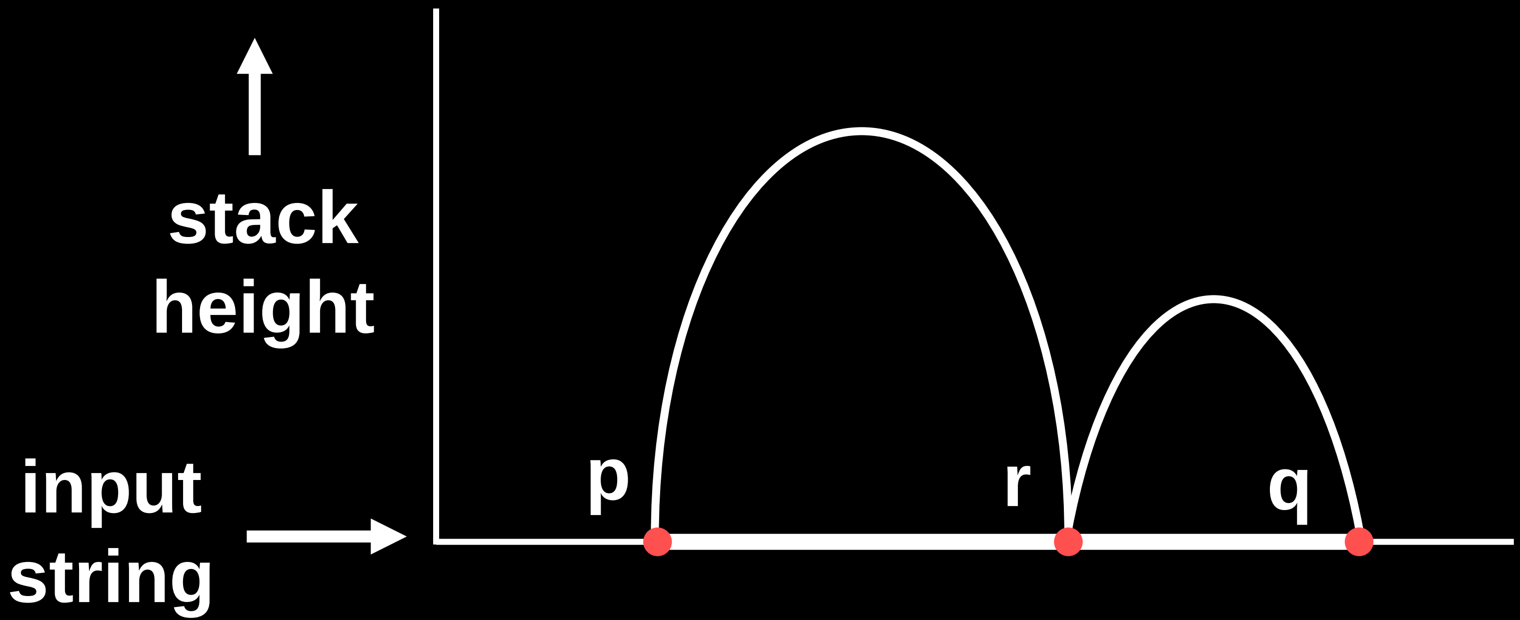


$\delta(p, a, \varepsilon) \rightarrow (r, t)$

$\delta(s, b, t) \rightarrow (q, \varepsilon)$

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

2. The symbol popped at the end is **not** the one pushed at the beginning



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

Formally:

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

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

For every $p, q, r, s \in Q$, $t \in \Gamma$ 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 every $p, q, r \in Q$,

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

For every $p \in Q$,

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

Show, for all x,

A_{pq} generates x

$\Leftrightarrow$

x can bring P from p with an empty stack to q with an empty stack

Show, for all x,

A_{pq} generates x

$\Rightarrow$

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the derivation of x from A_{pq}):

Base Case: The derivation has 1 step: $A_{pp} \Rightarrow^* \varepsilon$

Show, for all x,

A_{pq} **generates x**

$\Rightarrow$

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the derivation of x from A_{pq}):

Inductive Step:

Assume true for derivations of length ≤ k and prove true for derivations of length k+1:

$$A_{pq} \Rightarrow^* x \quad \text{in} \quad k+1 \text{ steps}$$

First step in derivation: $A_{pq} \rightarrow A_{pr}A_{rq}$ **or** $A_{pq} \rightarrow aA_{rs}b$

Show, for all x, A_{pq} generates x

$$\Rightarrow$$

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the derivation of x from A_{pq}):

Inductive Step:

Assume true for derivations of length ≤ k and prove true for derivations of length k+1:

$$A_{pq} \Rightarrow^* x \quad \text{in} \quad k+1 \text{ steps}$$

First step in derivation: $A_{pq} \rightarrow A_{pr}A_{rq}$

Then, x = yz with $A_{pr} \Rightarrow^* y$, $A_{rq} \Rightarrow^* z$

By IH, y can take p with empty stack to r with empty stack; similarly for z from r to q. So, …

Show, for all x,

A_{pq} generates x
$\Rightarrow$
x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the derivation of x from A_{pq}):

Inductive Step:

Assume true for derivations of length ≤ k and prove true for derivations of length k+1:

$A_{pq} \Rightarrow^* x$ in k+1 steps

First step in derivation: or $A_{pq} \rightarrow aA_{rs}b$

Then $x = ayb$ with $A_{rs} \Rightarrow^* y$.

By IH, y can take r with empty stack to s with empty stack

Show, for all x, A_{pq} generates x

$\Rightarrow$

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the derivation of x from A_{pq}):

Inductive Step:

Assume true for derivations of length ≤ k and prove true for derivations of length k+1:

$$A_{pq} \Rightarrow^* x \quad \text{in} \quad k+1 \text{ steps}$$

First step in derivation: or $A_{pq} \rightarrow aA_{rs}b$

By def of rules of G, $(r,t) \in \delta(p,a,\varepsilon)$ and $(q, \varepsilon) \in \delta(s,b,t)$

state push state alphabet pop

Show, for all x,

A_{pq} generates x

$\Rightarrow$

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the derivation of x from A_{pq}):

Inductive Step:

Assume true for derivations of length ≤ k and prove true for derivations of length k+1:

$A_{pq} \Rightarrow^* x$ in k+1 steps

First step in derivation: **or $A_{pq} \rightarrow aA_{rs}b$**

So if P starts in p then after reading a, it can go to r and push t.
By IH, y can bring P from r to s, with t at the top of the stack.
Then from s reading b, it can pop t and end in state q.

Show, for all x,

A_{pq} generates x

⇐

x can bring P from p with an empty stack to q with an empty stack

Proof (by induction on the number of steps in the computation of P from p to q with empty stacks on input x):

Base Case: The computation has 0 steps

So it starts and ends in the same state.
The only string that can do that in 0 steps is ε.

Since $A_{pp} \to \varepsilon$ is a rule of G, $A_{pp} \Rightarrow^* \varepsilon$

Inductive Step:

Assume true for computations of length ≤ k, we'll prove true for computations of length k+1

Suppose that P has a computation where x brings p to q with empty stacks in k+1 steps

Two cases: (idea!)

1. The stack is empty only at the beginning and the end of this computation

2. The stack is empty somewhere in the middle of the computation

Inductive Step:

Assume true for computations of length ≤ k, we'll prove true for computations of length k+1

Suppose that P has a computation where x brings p to q with empty stacks in k+1 steps

Two cases: (idea!)

1. The stack is empty only at the beginning and the end of this computation

To Show: Can write x as ayb where $A_{rs} \Rightarrow^* y$ and $A_{pq} \rightarrow aA_{rs}b$ is a rule in G. So $A_{pq} \Rightarrow^* x$

2. The stack is empty somewhere in the middle of the computation

Inductive Step:

Assume true for computations of length ≤ k, we'll prove true for computations of length k+1

Suppose that P has a computation where x brings p to q with empty stacks in k+1 steps

Two cases: (idea!)

1. The stack is empty only at the beginning and the end of this computation

To Show: Can write x as ayb where $A_{rs} \Rightarrow^* y$ and $A_{pq} \rightarrow aA_{rs}b$ is a rule in G. So $A_{pq} \Rightarrow^* x$

2. The stack is empty somewhere in the middle of the computation

To Show: Can write x as yz where $A_{pr} \Rightarrow^* y$, $A_{rq} \Rightarrow^* z$ and $A_{pq} \rightarrow A_{pr}A_{rq}$ is a rule in G. So $A_{pq} \Rightarrow^* x$

Inductive Step:

1. The stack is empty *only* at the beginning and the end of this computation

To Show: Can write x as ayb where $A_{rs} \Rightarrow^* y$ and $A_{pq} \rightarrow aA_{rs}b$ is a rule in G. So $A_{pq} \Rightarrow^* x$

The symbol t pushed at the beginning must be the same symbol popped at the end. **why?)**

Let a be input symbol read at beginning, b read at end.

- So $x = ayb$, for some y.

Let r be the state after the first step, let s be the state before the last step.

- y can bring P from r with an empty stack to s with an empty stack. (why?) So by IH, $A_{rs} \Rightarrow^* y$.
- Also, $A_{pq} \rightarrow aA_{rs}b$ **must be a rule in G. (why?)**

Inductive Step:

2. The stack is empty somewhere in the middle of the computation

To Show: Can write x as yz where $A_{pr} \Rightarrow^* y$, $A_{rq} \Rightarrow^* z$ and $A_{pq} \rightarrow A_{pr}A_{rq}$ is a rule in G. So $A_{pq} \Rightarrow^* x$

Let r be a state in which the stack becomes empty in the middle.

Let y be the input read to that point, z be input read after. So, x = yz where |y|, |z| > 0.

By IH, both $A_{pr} \Rightarrow^* y$, $A_{rq} \Rightarrow^* z$

By construction of G, $A_{pq} \rightarrow A_{pr}A_{rq}$ **is a rule in G**

A Language L is generated by a CFG

$\Leftrightarrow$

L is recognized by a PDA

Corollary: Every regular language is context-free

WWW.FLAC.WS

Read Chapters 2 and 3 of the book for next time