Tianjin International Engineering Institute

Formal Languages and Automata

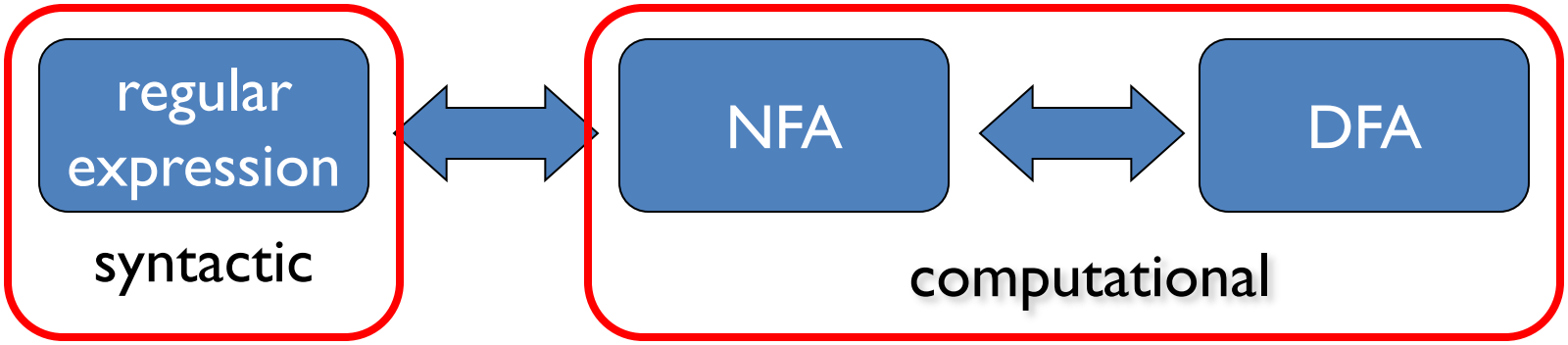
Lesson 11: Pushdown automata

Marc Gaetano

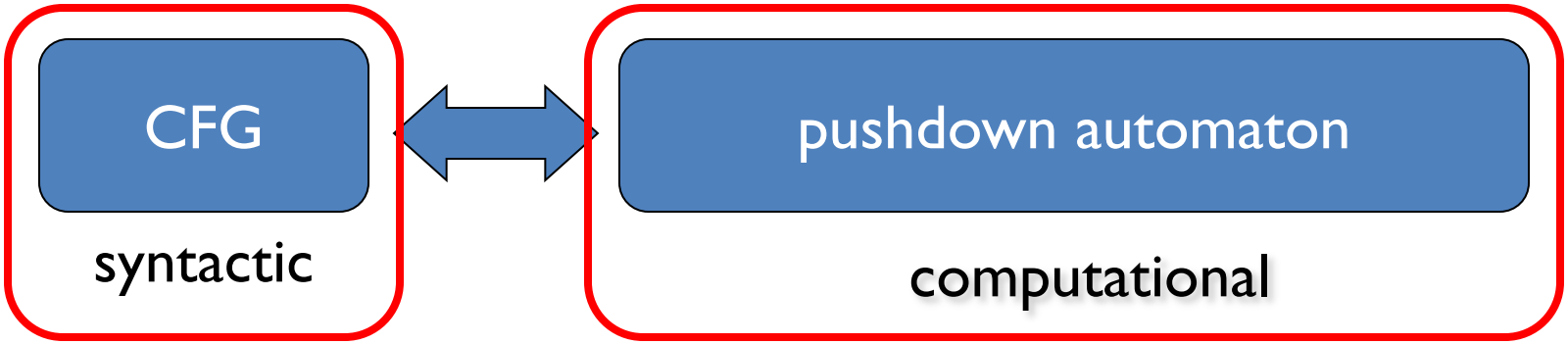
Edition 2018

Motivation

- We had two ways to describe regular languages

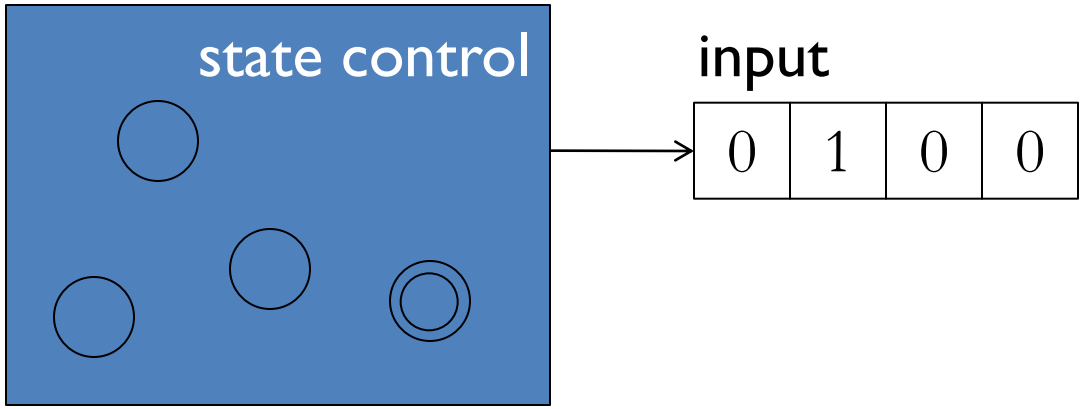


- How about context-free languages?



Pushdown automata versus NFA

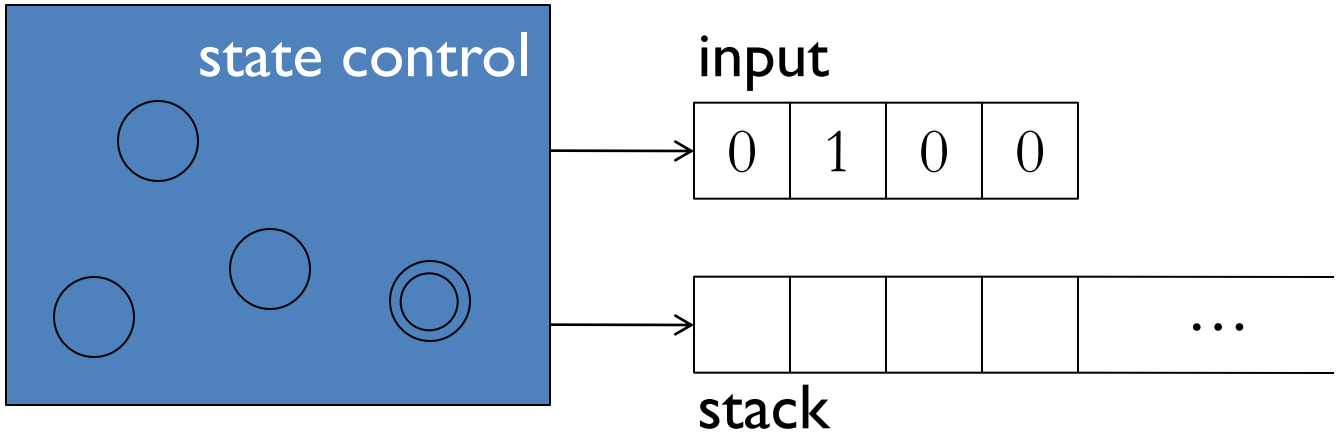
- Since context-free is more powerful than regular, pushdown automata must generalize NFAs



NFA

Pushdown automata

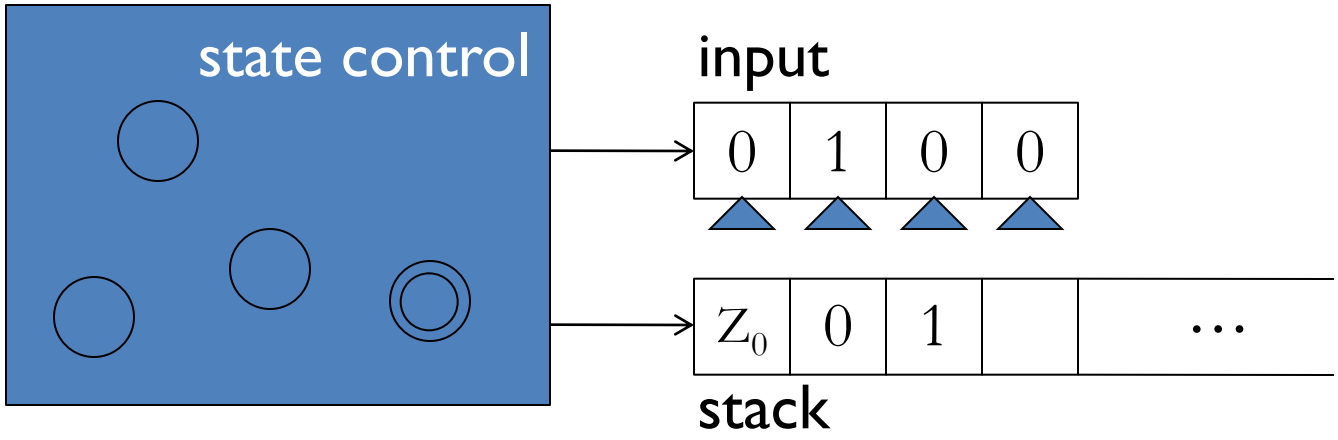
- A pushdown automaton has access to a stack, which is a potentially infinite supply of memory



pushdown automaton (PDA)

Pushdown automata

- As the PDA is reading the input, it can push / pop symbols in / out of the stack



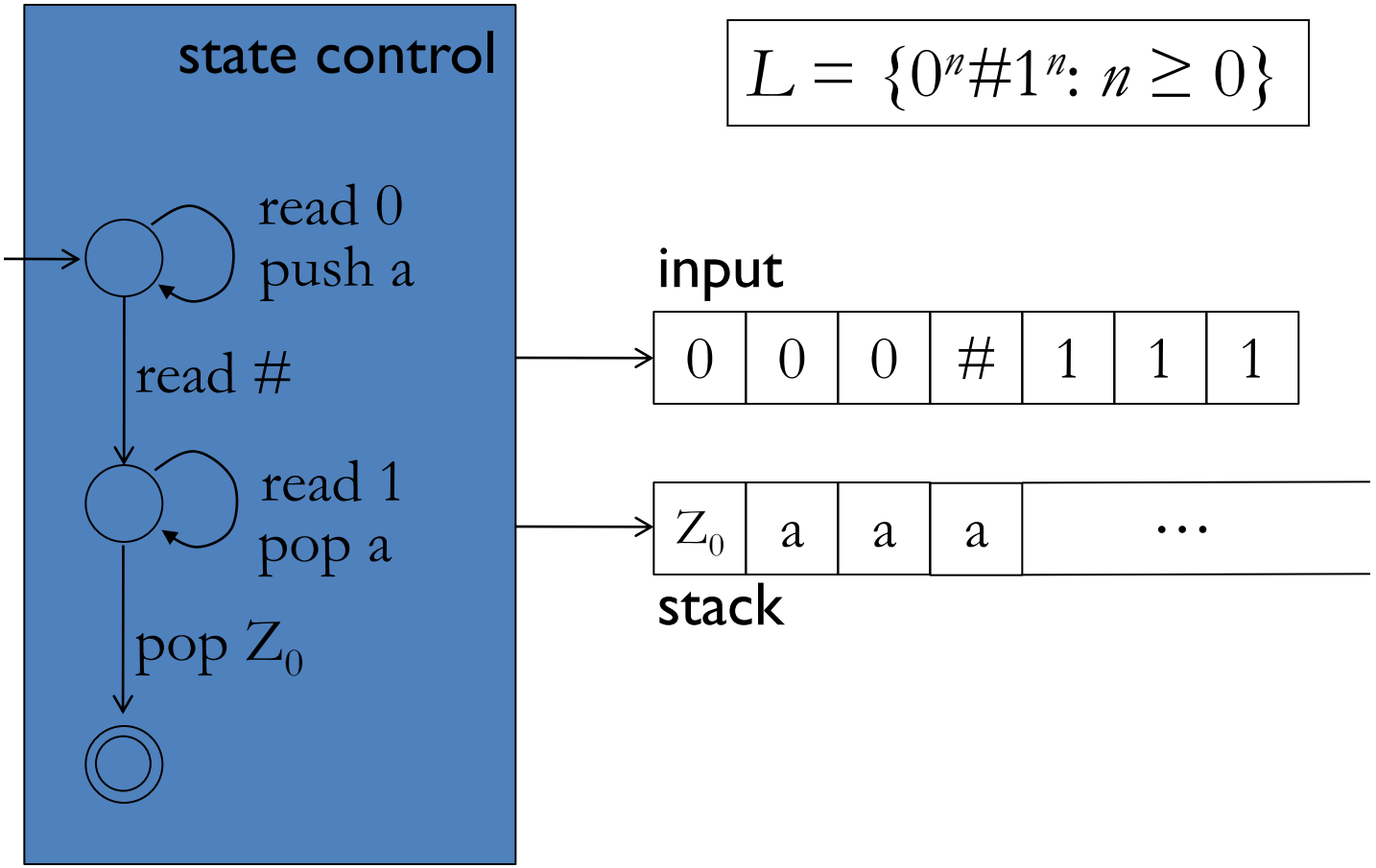
pushdown automaton (PDA)

Rules for pushdown automata

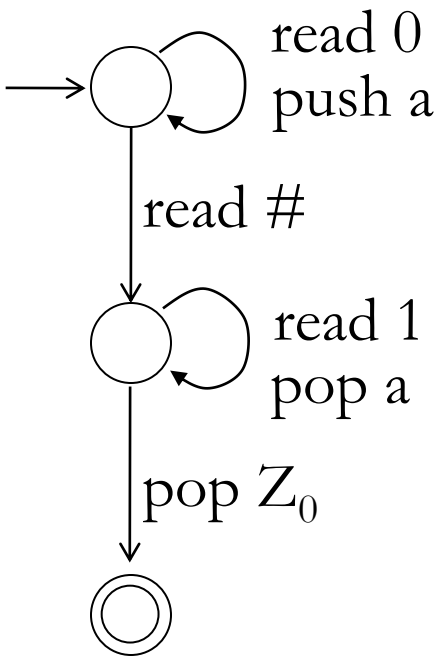
- The transitions are nondeterministic
- Stack is always accessed from the top
- Each transition can pop a symbol from the stack and / or push another symbol onto the stack
- Transitions depend on input symbol and on last symbol popped from stack

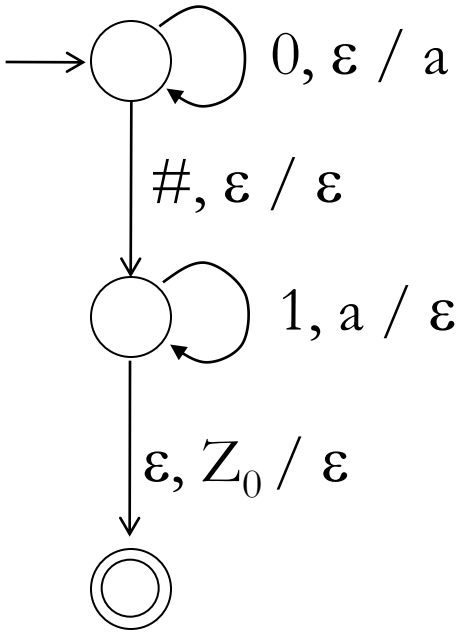
- Automaton accepts

Example


state control
read 0
push a
read #
read 1
pop a
pop Z_0
$L = \{0^n\#1^n : n \geq 0\}$
input
0 0 0 # 1 1 1
Z_0 a a a …
stack

Shorthand notation





read, pop / push

Formal definition

A pushdown automaton is $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$:

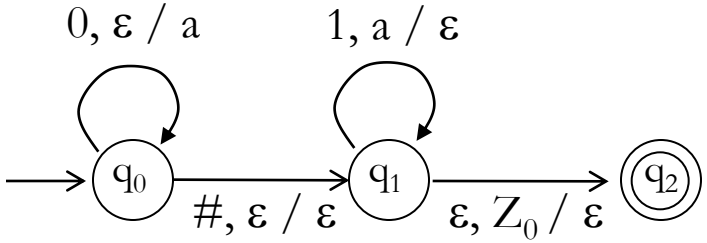
- Q is a finite set of states;
- Σ is the input alphabet;
- Γ is the stack alphabet, including a special symbol Z_0;
- q_0 in Q is the initial state;
- Z_0 in Γ is the start symbol;
- $F \subseteq Q$ is a set of final states;
- δ is the transition function

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \text{subsets of } Q \times (\Gamma \cup \{\varepsilon\})$$

state input symbol pop symbol state push symbol

Notes on definition

- We use slightly different definition than textbook
- Example

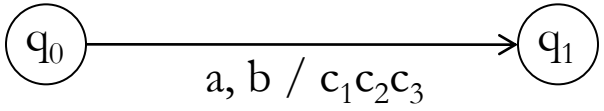


$$\delta(q_0, 0, \varepsilon) = \{(q_0, a)\}$$
$$\delta(q_0, 1, \varepsilon) = \emptyset$$
$$\delta(q_0, \#, \varepsilon) = \{(q_1, \varepsilon)\}$$
$$\delta(q_0, 0, \alpha) = \emptyset$$
$$\ldots$$

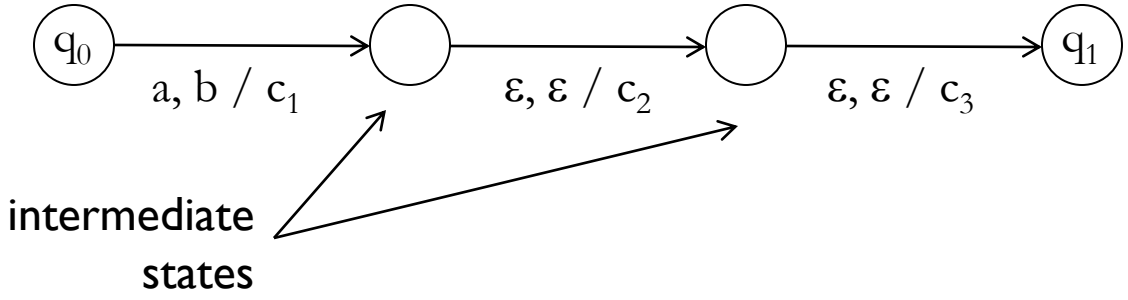
$$\delta\text{: } Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \text{subsets of } Q \times (\Gamma \cup \{\varepsilon\})$$

A convention

- Sometimes we denote "transitions" by:



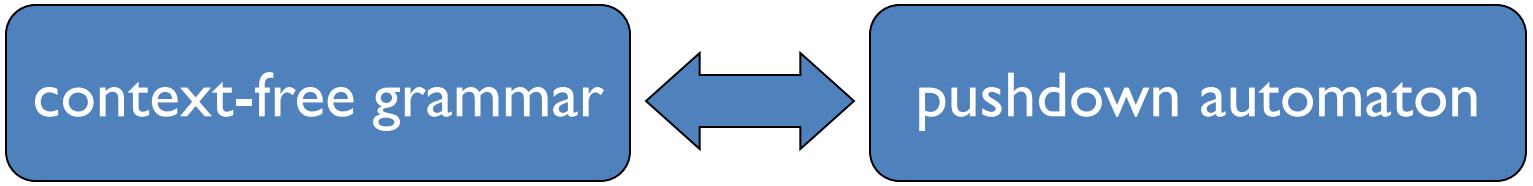
- This will mean:



  - Intuitively, pop b, then push c_1, c_2, and c_3

Main theorem

A language L is context-free if and only if it is accepted by some pushdown automaton.



From CFGs to PDAs

- Idea: Use PDA to simulate (rightmost) derivations

CFG:

$$\begin{aligned} A &\to 0A1 \\ A &\to B \\ B &\to \# \end{aligned}$$

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$$

PDA control:	stack:	input:
write start variable	Z_0A	$00\#11$
replace production in reverse	Z_01A0	$00\#11$
pop terminals and match	Z_01A	$0\#11$
replace production in reverse	Z_011A0	$0\#11$
pop terminals and match	Z_011A	$\#11$
replace production in reverse	Z_011B	$\#11$

From CFGs to PDAs

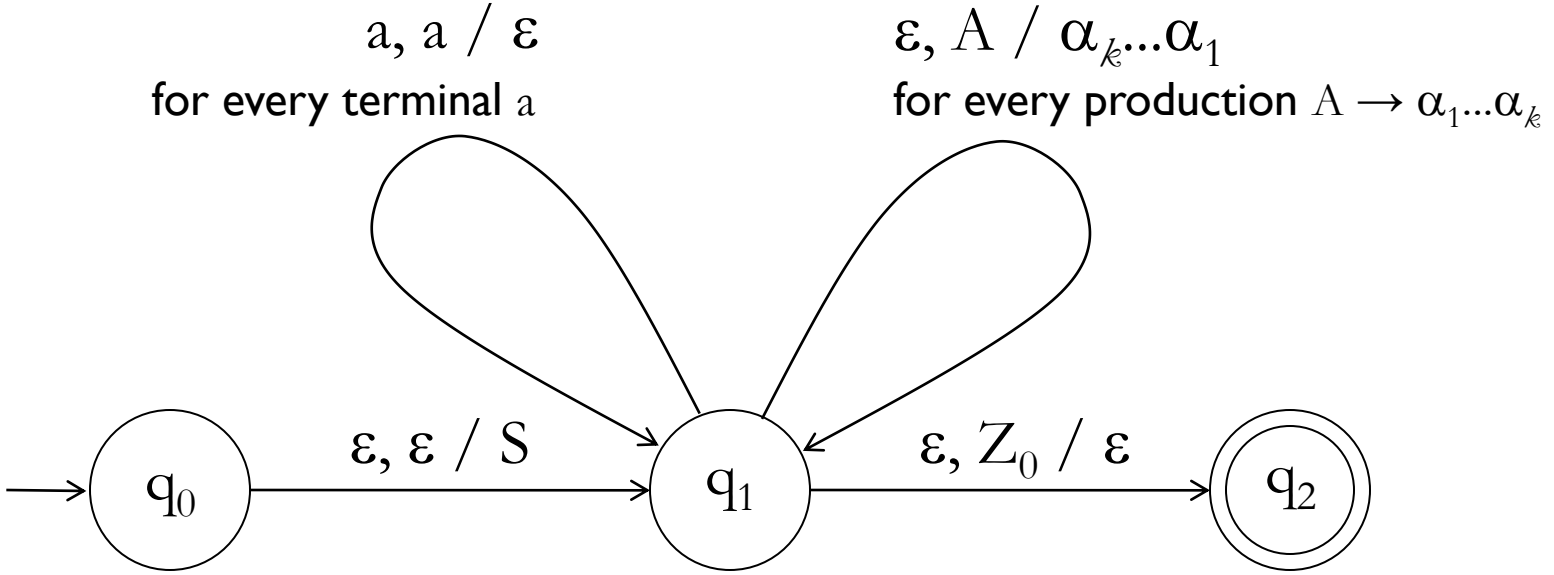
- If, after reading whole input, PDA ends up with an empty stack, derivation must be valid

- Conversely, if there is no valid derivation, PDA will get stuck somewhere
  - Either unable to match next input symbol,
  - Or match whole input but stack non empty

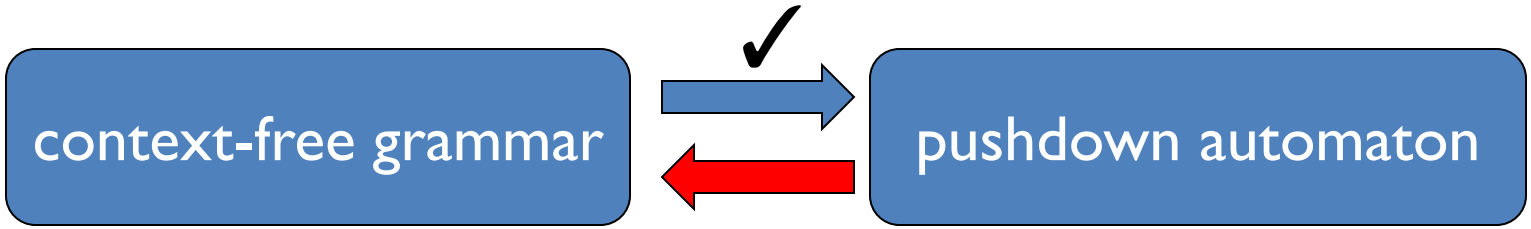
Description of PDA for CFGs

- Repeat the following steps:
  - If the top of the stack is a variable A:
    Choose a rule $\mathrm{A} \rightarrow \alpha$ and substitute A with α
  - If the top of the stack is a terminal a:
    Read next input symbol and compare to a
    If they don't match, reject (die)
  - If top of stack is Z_0, go to accept state

Description of PDA for CFGs



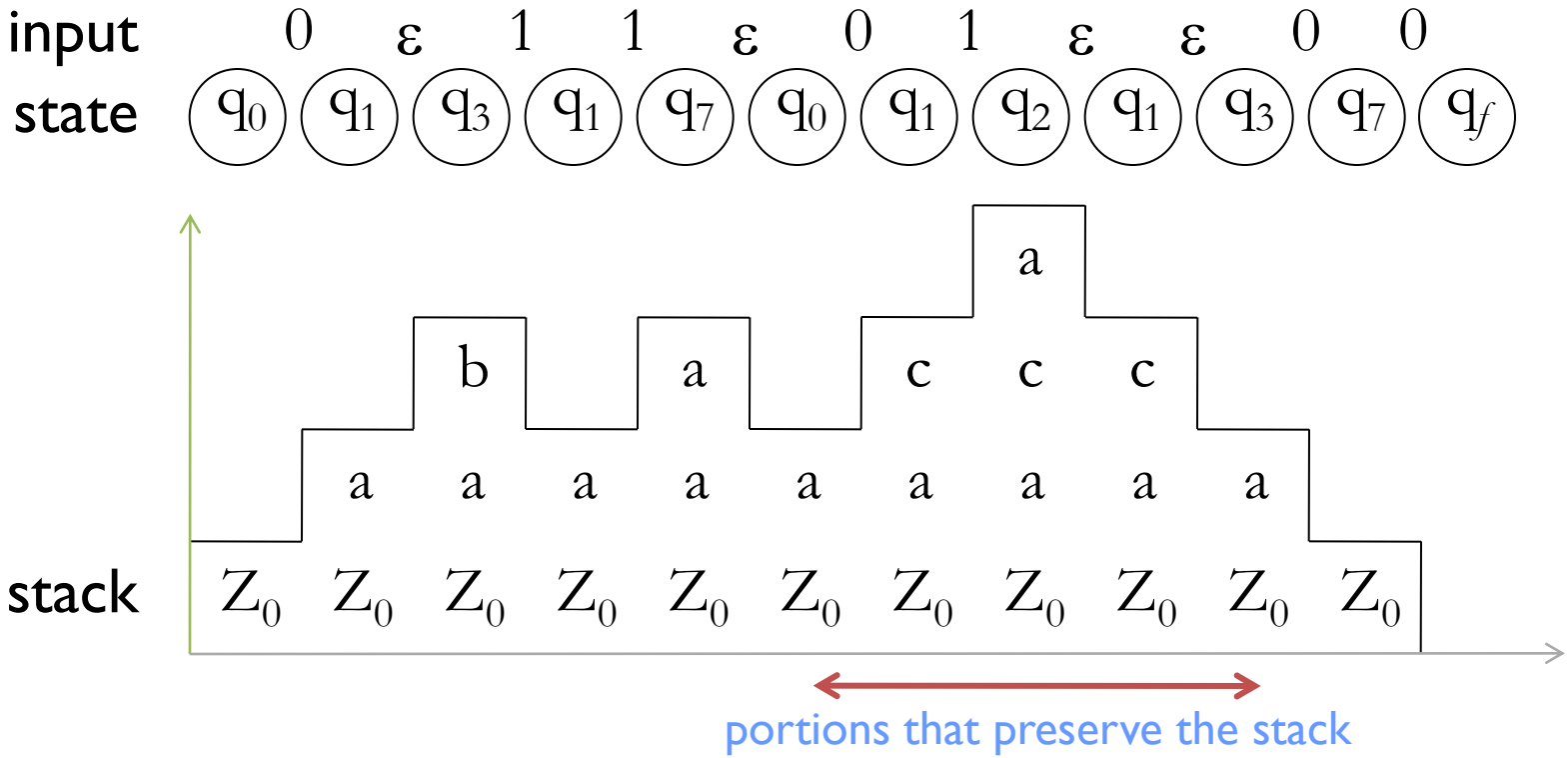
From PDAs to CFGs



- First, we simplify the PDA:
  - It has a single accept state q_f
  - Z_0 is always popped exactly before accepting
  - Each transition is either a push, or a pop, but not both

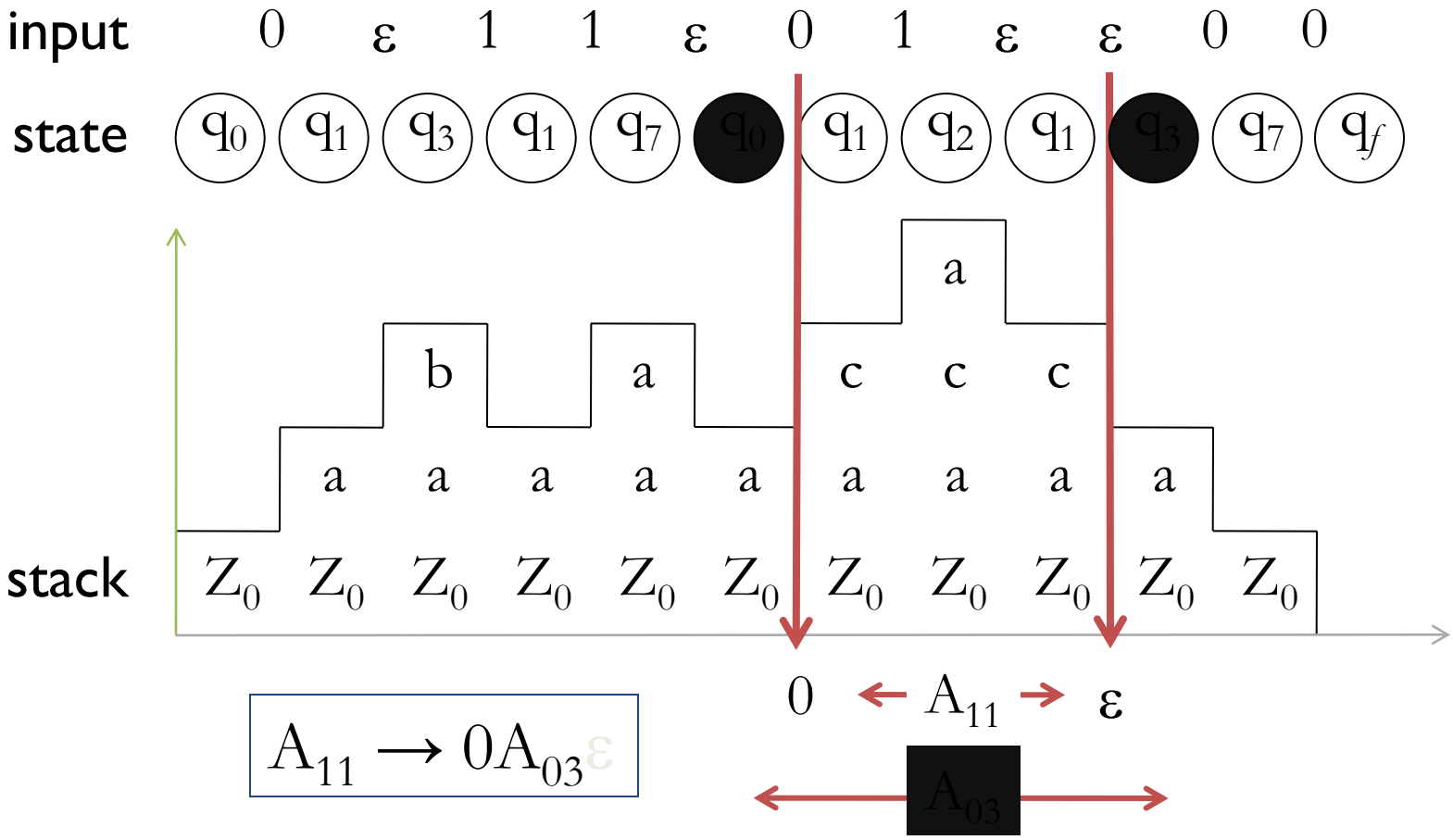
From PDAs to CFGs

- We look at the stack in an accepting computation:

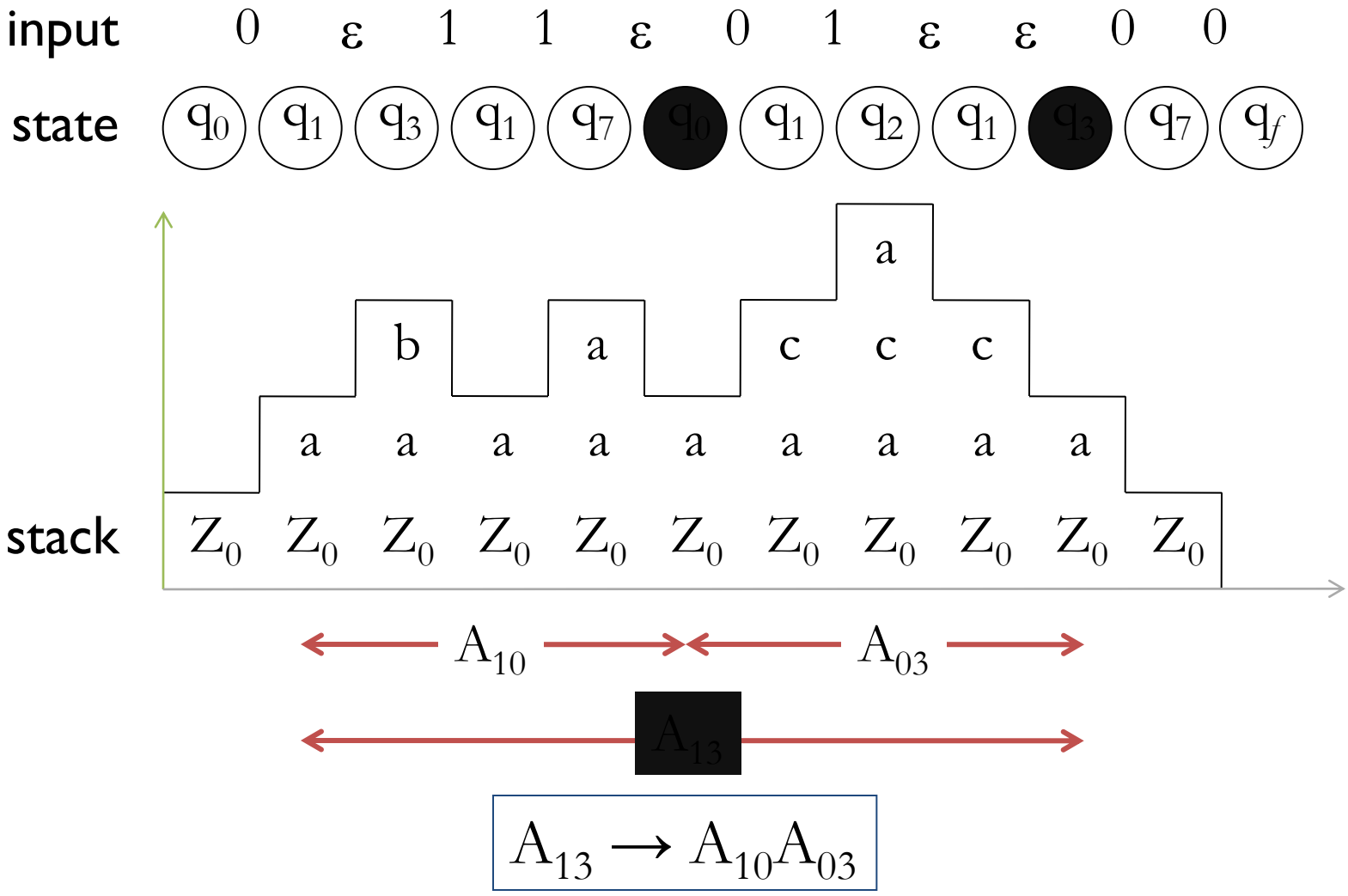


$A_{03} = \{x: x \text{ leads from } q_0 \text{ to } q_3 \text{ and preserves stack}\}$

From PDAs to CFGs



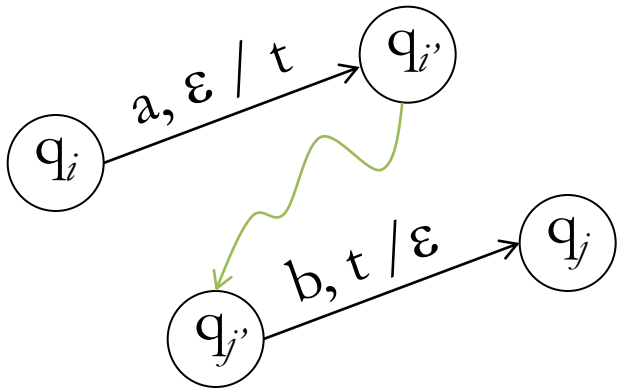
From PDAs to CFGs



From PDAs to CFGs

variables: A_{ij}

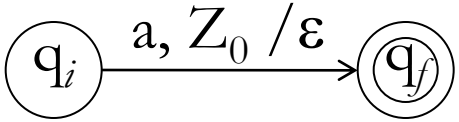
start variable: A_{0f}



$A_{ij} \rightarrow aA_{i'j'}b$

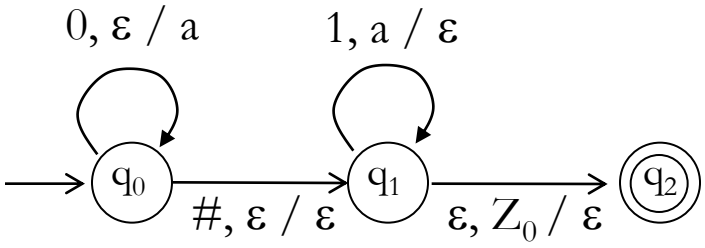
$A_{ik} \rightarrow A_{ij}A_{jk}$

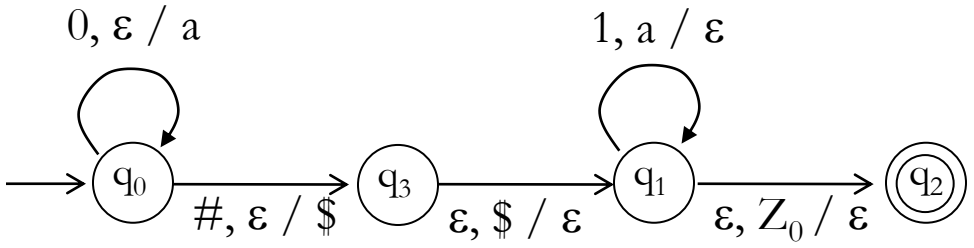
$A_{ii} \rightarrow \varepsilon$



$A_{0f} \rightarrow A_{0i}a$

Example





start variable: A_{02}

productions:

$A_{00} \rightarrow A_{00}A_{00}$
$A_{00} \rightarrow A_{01}A_{10}$
$A_{00} \rightarrow A_{03}A_{30}$
$A_{01} \rightarrow A_{01}A_{11}$
$A_{01} \rightarrow A_{02}A_{21}$
...

$A_{00} \rightarrow \varepsilon$
$A_{11} \rightarrow \varepsilon$
$A_{22} \rightarrow \varepsilon$
$A_{33} \rightarrow \varepsilon$

$A_{01} \rightarrow 0A_{01}1$
$A_{01} \rightarrow \#A_{33}$

$A_{02} \rightarrow A_{01}$