Chapter 5
Push-down Automata

by
Dr Zalmiyah Zakaria

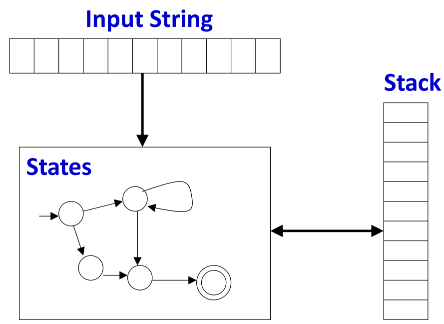
Push-Down Automata

- Regular expressions are *string generators* – *they* tell us how to generate all strings in a language L.
- Finite Automata (DFA, NFA) are *string acceptors* – they tell us if a specific string w is in L.
- CFGs are *string generators*
- Are there string acceptors for Context-Free languages?
- YES! Push-down automata

Push-Down Automata

- DFAs accept regular languages.
- Now, we want to design machines similar to DFAs that will accept context-free languages.
- These machines will need to be more powerful.
- To handle a language like $\{a^n b^n \mid n \geq 0\}$, the machine needs to "remember" the number of a's.
- To do this, we use a *stack*.
- A *push-down automaton* (PDA) is essentially an NFA with a stack.

Push-Down Automaton (PDA)



Definition of a PDA

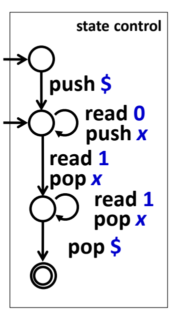
A pushdown automaton (PDA) is a sextuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where:

- Q is a finite set of states;
- Σ is the input alphabet;
- Γ is the stack alphabet
- q_0 in Q is the start state;
- $F \subseteq Q$ is a set of final states;
- δ is the transition function.

  δ: $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow$ subsets of $Q \times (\Gamma \cup \{\varepsilon\})$
- PDA accepts w if we end up in a final state with an empty stack.

Note: Γ - gamma, δ - delta

Building a PDA



$L = \{0^n 1^n : n \geq 1\}$

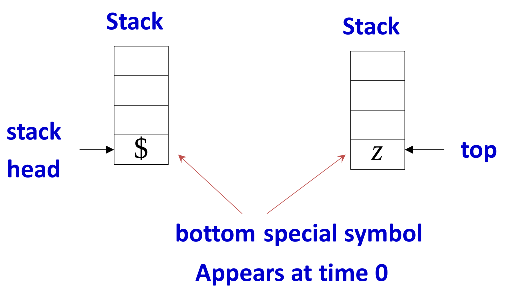
Remember each 0 by pushing x onto the stack

When we see a 1, we pop an x from the stack

We want to accept when we hit the stack bottom

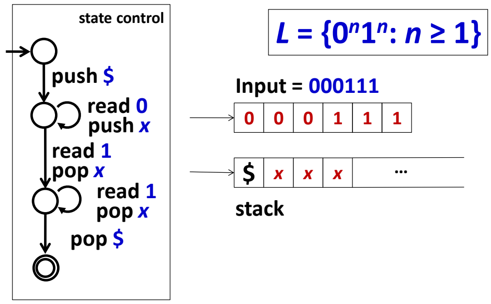
$ = special marker for bottom

Initial Stack Symbol

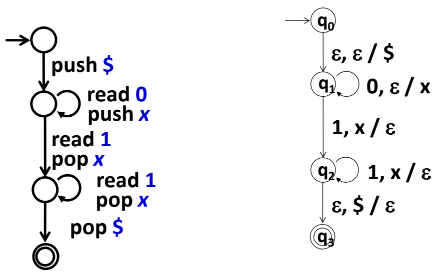


8

A PDA in action

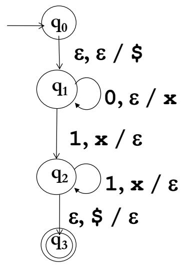


Notation for PDAs



read, pop / push

Example



$\Sigma = \{0, 1\}$
$\Gamma = \{\$, x\}$

$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}$
$\delta(q_1, 0, \varepsilon) = \{(q_1, x)\}$
$\delta(q_1, 1, x) = \{(q_2, \varepsilon)\}$
$\delta(q_2, 1, x) = \{(q_2, \varepsilon)\}$
. . .

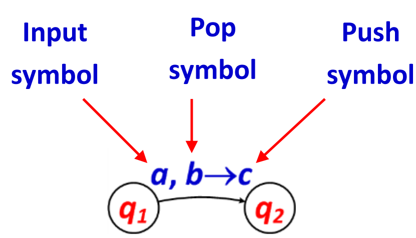
$$\delta: \underbrace{Q}_{\text{state}} \times \underbrace{(\Sigma \cup \{\varepsilon\})}_{\text{input symbol}} \times \underbrace{(\Gamma \cup \{\varepsilon\})}_{\text{pop symbol}} \to \text{subsets of } \underbrace{Q}_{\text{state}} \times \underbrace{(\Gamma \cup \{\varepsilon\})}_{\text{push symbol}}$$

The language of a PDA

- A PDA is **nondeterministic**
  Multiple transitions on same pop/input allowed
- Transitions may but do not have to push or pop

> The **language** of a PDA is the set of all strings in S* that can lead the PDA to an accepting state

The States



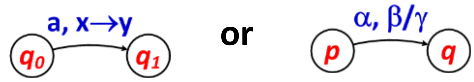
13

Transitions

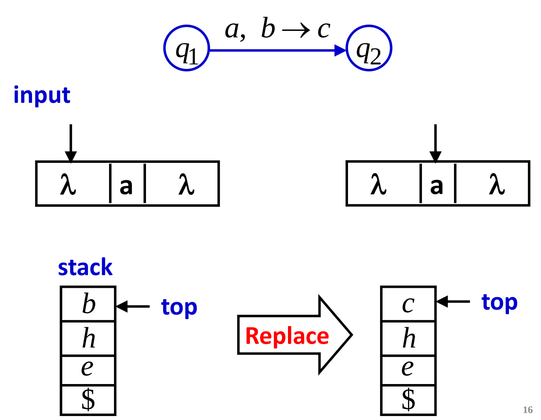
- Let $((p, a, \beta), (q, \gamma)) \in \Delta$ be a transition.
- It means that we
  - Move from state p.
  - Read a from the tape,
  - Pop the string β from the stack,
  - Move to state q,
  - Push string γ onto the stack.
- The first three (p, a, β), are input.
- The last two (q, γ) are output.

Transitions

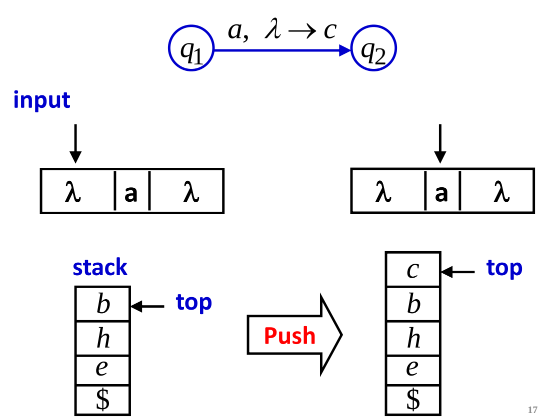
- We will draw it as



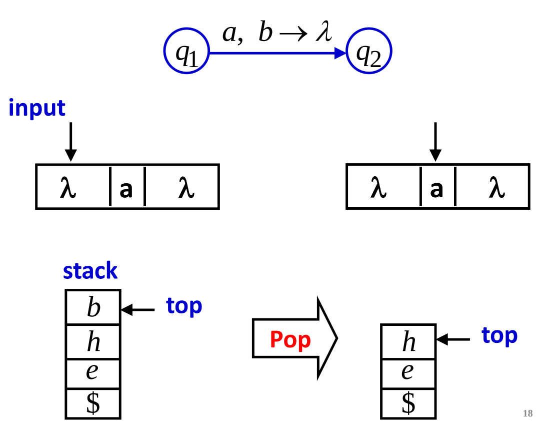
If at q_0 (p), with next input symbol a (α) and top of stack x (β), then can consume a (α), pop x (β), push y (γ) onto stack and move to q_1 (q)



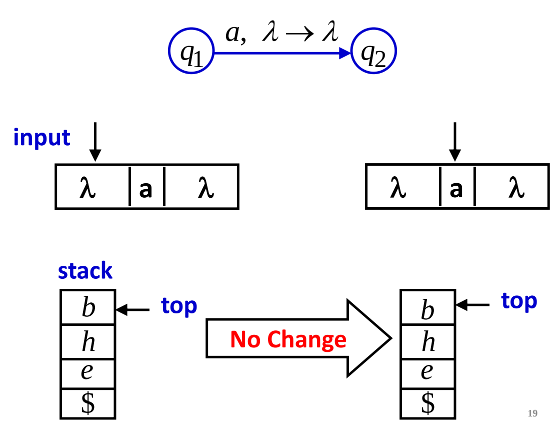
16



17



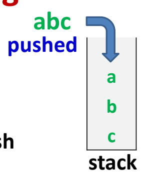
18



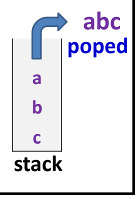
19

Pushing and Popping

- When we push β, we push the symbols of β as we read them *right to left*.
  - When we push the string ***abc***, we push ***c***, then push ***b***, then push ***a***.

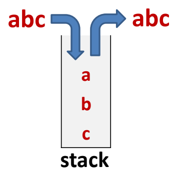


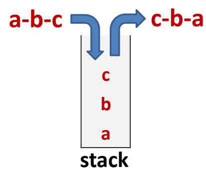
- When we pop γ, we pop the symbols of γ as we read them from *left to right* (reverse order).
  - When we pop the string ***abc***, we pop ***a***, then pop ***b***, then pop ***c***.



Pushing and Popping

- Thus, if we push the string ***abc*** and then pop it, we will get back ***abc***, not ***cba***.
- If we wanted to reverse the order, we would use three separate transitions:
  - Push ***a***
  - Push ***b***
  - Push ***c***





Configurations

- A *configuration* fully describes the current (*state*, *string*, *stack*) of the PDA.
  - The current state p.
  - The remaining symbols left in the input string w.
  - The current contents of the stack α.
- Thus, a configuration is a triple

$$(p, w, \alpha) \in (K, \Sigma^*, \Gamma^*).$$

Computations

- A configuration (p, w, α) *yields* a configuration (p', w', α') *in one step*, denoted
$$(p, w, \alpha) \vdash (p', w', \alpha'),$$
if there is a transition $((p, a, \beta), (p', \gamma)) \in \Delta$ such that $w = aw'$, $\alpha = \beta\eta$, and $\alpha' = \gamma\eta$ for some $\eta \in \Gamma^*$.
- The reflexive, transitive closure of $\vdash$ is denoted $\vdash^*$.
- A computation of a PDA is a sequence of transitions beginning with *start state*.

Accepting Strings

- After processing the string on the tape,
  - The PDA is in either a *final* or a *nonfinal* state, and
  - The stack is either *empty* or *not empty*.
- The input string is *accepted* if
  - The ending state is a final state, *and*
  - The stack is empty.
- That is, the string $w \in \Sigma^*$ is *accepted* if
$$(s, w, \varepsilon) \vdash^* (f, \varepsilon, \varepsilon)$$
for some $f \in F$.

Accepting Strings

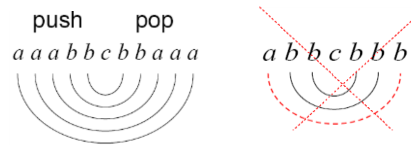
- One may define acceptance by final state only.
  - The input is accepted if and only if the last state is a final state, regardless of whether the stack is empty.
- One may define acceptance by empty stack only.
  - The input is accepted if and only if the stack is empty once the input is processed, regardless of which state the PDA is in.

Example of a PDA

- The language $L = \{wcw^R : w \in \{a, b\}^*\}$ is CFL but not RL
  - The grammar for L is $S \rightarrow aSb \mid ab$
  - We can not have a DFA for L
  - Problem is memory, DFA's cannot remember left hand substring
- What kind of memory do we need to be able to recognize strings in L? Answer: a stack

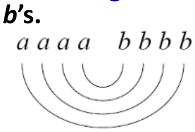
Example of a PDA

- $u = aaabbcbbaaa \in L$
  - We push the first part of the string onto the stack
  - after the c is encountered
  - start popping characters off of the stack and matching them with each character.
  - if everything matches, this string $\in L$



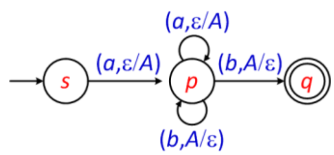
Example of a PDA

- We can also use a stack for counting out equal numbers of a's and b's.
- Example:
  - $L = \{a^n b^n : n \geq 0\}$
  - $w = aaaabbbb \in L$
  - Push the a's onto the stack, then pop an a off and match it with each b.
  - If we finish processing the string successfully (and there are no more a's on our stack), then the string belongs to L.



Example of a PDA

- Run the following PDA on the input string *aaabbb*.



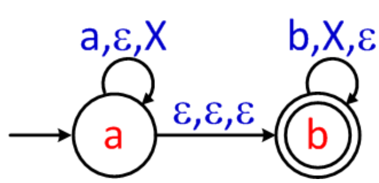
{w | $w \in \{a,b\}^*$ and w has equal number of a's and b's}

Example of a PDA

- The steps in the processing are
  - $(s, aaabbb, \varepsilon) \vdash (p, aabbb, A)$
    $\vdash (p, abbb, AA)$
    $\vdash (p, bbb, AAA)$
    $\vdash (p, bb, AA)$
    $\vdash (p, b, A)$
    $\vdash (q, \varepsilon, \varepsilon)$.

Example of a PDA

- What is the language of the following PDA?



$\{a^n b^n \mid n > 0\}$.

PUSHDOWN AUTOMATA (PDA) Example 1

- L(M) = {w ∈ {a,b}* : w consists of **equal number of a's and b's.**
- M = (Q, Σ, Γ, δ, s, f) where M = ({s, q, f}, {a, b}, {A, B, C}, δ, {s}, {f}) and …

PUSHDOWN AUTOMATA (PDA) Example 1

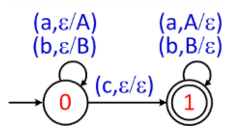
- δ is defined as follows:
  1. δ(s, λ, λ) = {[q, C]}
  2. δ(q, a, C) = {[q, AC]}
  3. δ(q, a, A) = {[q, AA]}
  4. δ(q, a, B) = {[q, λ]}
  5. δ(q, b, C) = {[q, BC]}
  6. δ(q, b, B) = {[q, BB]}
  7. δ(q, b, A) = {[q, λ]}
  8. δ(q, λ, C) = {[f, λ]}
- Test the string: ***abbbabaa***

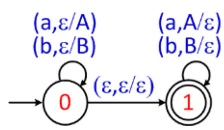
Examples of PDAs

- Let $\Sigma = \{a, b\}$.
- Design a PDA that accepts the language $\{wcw^R \mid w \in \Sigma^*\}$.
- Design a PDA that accepts the language $\{ww^R \mid w \in \Sigma^*\}$.

Examples of PDAs

- $\{wcw^R \mid w \in \{a, b\}*\}$

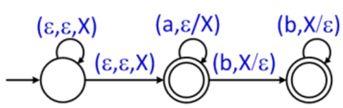




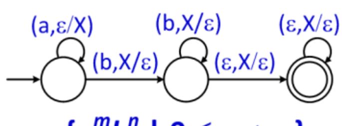
- $\{ww^R \mid w \in \{a, b\}*\}$

Examples of PDAs

- Design a PDA whose language is
  $\{a^m b^n \mid 0 \le m < n\}$.
- Design a PDA whose language is
  $\{a^m b^n \mid 0 \le n < m\}$.



$\{a^m b^n \mid 0 \le m < n\}$



$\{a^m b^n \mid 0 \le n < m\}$

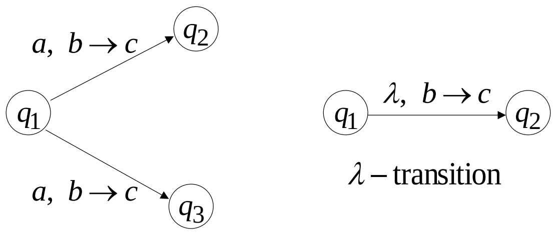
Examples of PDAs

- Design a PDA whose language is $\{a^m b^n c^n d^m \mid m \geq 0, n \geq 0\}$.
- Design a PDA whose language is $\{a^m b^m c^n d^n \mid m \geq 0, n \geq 0\}$.
- Design a PDA whose language is $\{a^m b^n c^p d^q \mid m + n = p + q\}$.
- Design a PDA whose language is $\{a^m b^n c^k \mid m = n \text{ or } m = k\}$.

Non-Determinism

PDAs are non-deterministic

Allowed non-deterministic transitions

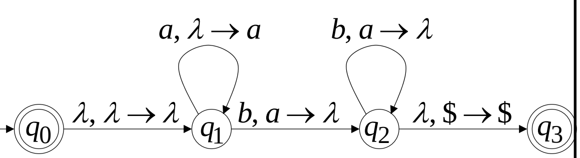


41

Example PDA

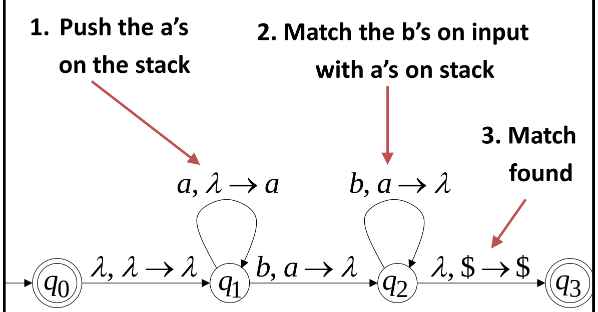
PDA *M*,

$L(M) = \{a^n b^n : n \geq 0\}$



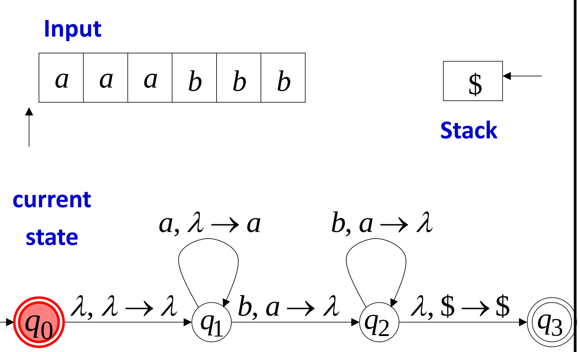
42

Basic Idea: $L(M) = \{a^n b^n : n \geq 0\}$



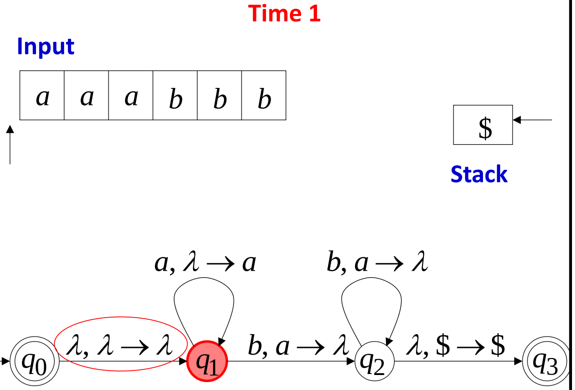
43

Execution Example: Time 0

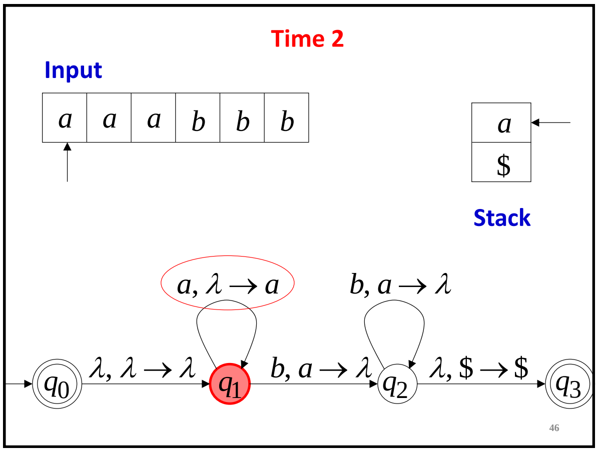


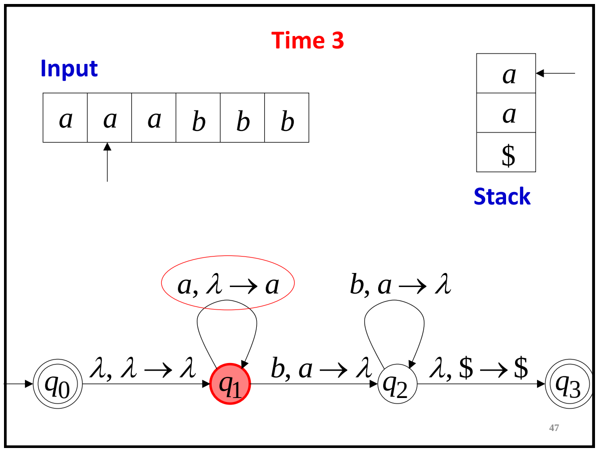
44

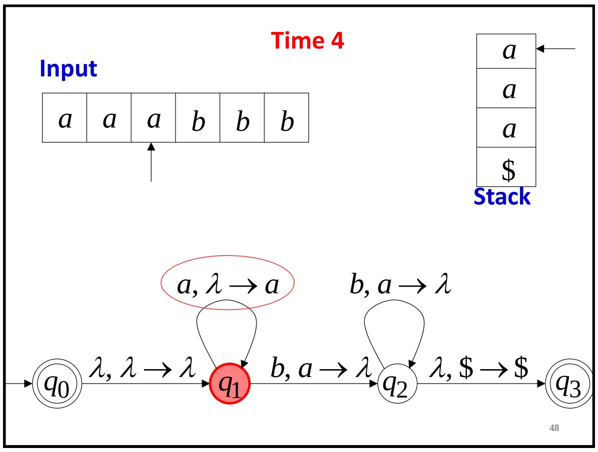
Time 1

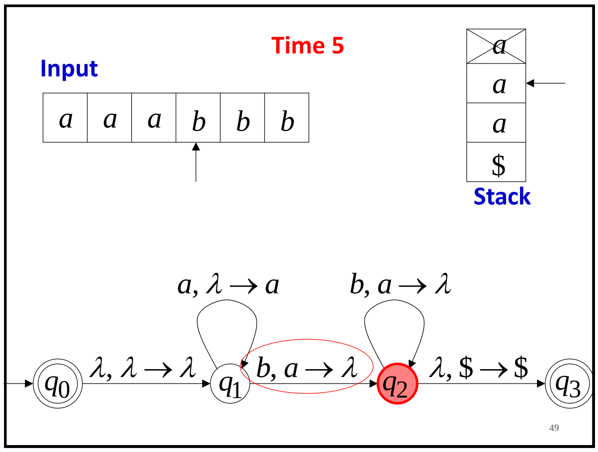


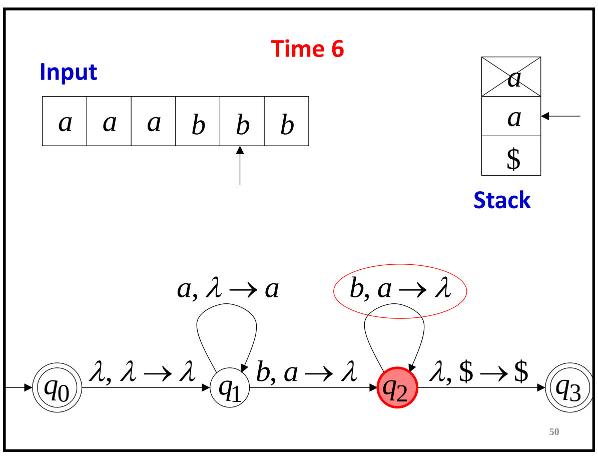
45

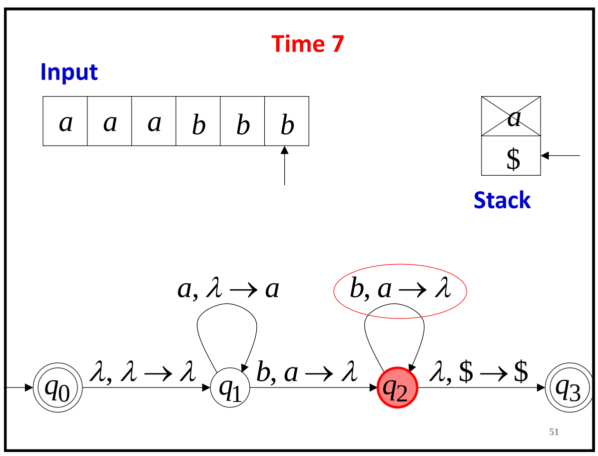

Time 2
Input
a a a b b b
a
$
Stack
a, λ → a
b, a → λ
λ, λ → λ
b, a → λ
λ, $ → $
q0 q1 q2 q3
46

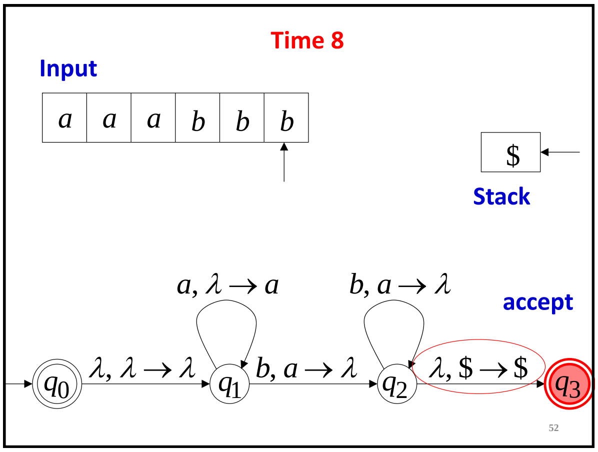

Time 3
Input
a a a b b b
a
a
$
Stack
a, λ → a
b, a → λ
λ, λ → λ
b, a → λ
λ, $ → $
q0 q1 q2 q3
47


Time 4
Input
a a a b b b
a
a
a
$
Stack
a, λ → a
b, a → λ
λ, λ → λ
b, a → λ
λ, $ → $
q0 q1 q2 q3
48


Time 5
Input
a a a b b b
a
a
a
$
Stack
a, λ → a
b, a → λ
λ, λ → λ
b, a → λ
λ, $ → $
q0 q1 q2 q3
49


Time 6
Input
a a a b b b
a
a
$
Stack
a, λ → a
b, a → λ
λ, λ → λ
b, a → λ
λ, $ → $
q0 q1 q2 q3
50

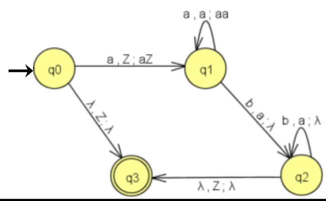

Time 7
Input
a a a b b b
a
$
Stack
a, λ → a
b, a → λ
λ, λ → λ
b, a → λ
λ, $ → $
q0 q1 q2 q3
51



Language $L(M) = \{a^n b^n : n \geq 0\}$

- The computation generated by the input string *aaabbb* is

$[q_0, aaabbb, Z] \vdash [q_1, aabbb, aZ] \vdash [q_1, abbb, aaZ]$
$\vdash [q_1, bbb, aaaZ] \vdash [q_2, bb, aaZ] \vdash [q_2, b, aZ]$
$\vdash [q_2, \varepsilon, Z] \vdash [q_3, \varepsilon, \varepsilon]$



state	string	stack
q_0	aaabbb	Z
q_1	aabbb	aZ
q_1	abbb	aaZ
q_1	bbb	aaaZ
q_2	bb	aaZ
q_2	b	aZ
q_2	λ	Z
q_3	λ	λ

Actions of the Example PDA

Input string *a a a b b b*

q_0

Stack *Z*

Actions of the Example PDA

a a b b b

q_1

a
Z

Actions of the Example PDA

a b b b

q_1

a
a
Z

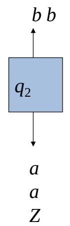
Actions of the Example PDA

b b b

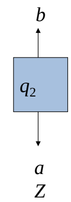
q_1

a
a
a
Z

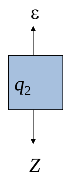
Actions of the Example PDA



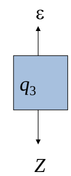
Actions of the Example PDA



Actions of the Example PDA



Actions of the Example PDA



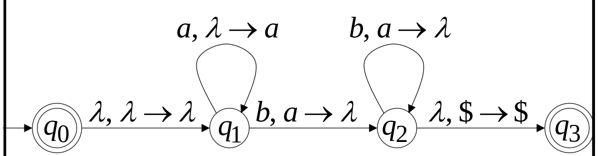
A string is accepted if there is
a computation such that:

All the input is consumed
AND
The last state is an accepting state

At the end of the computation,
we do not care about the stack contents
(the stack can be empty at the last state)

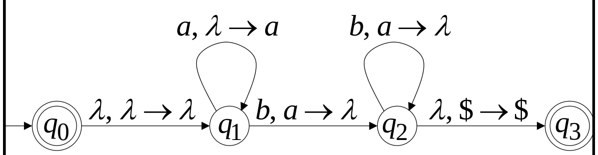
62

The input string ***aaabbb***
is accepted by the PDA:

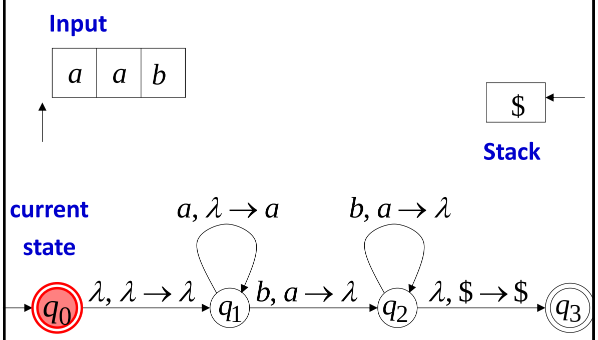


63

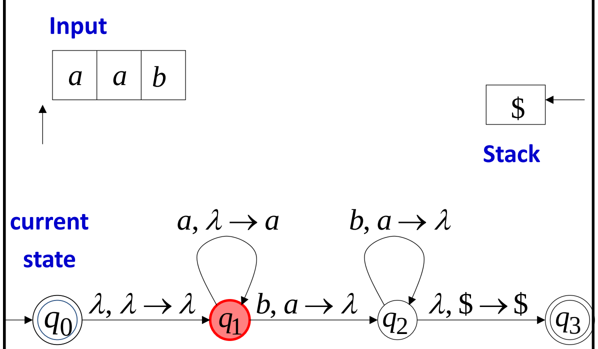
In general, $L = \{a^n b^n : n \geq 0\}$
is the language accepted by the PDA:



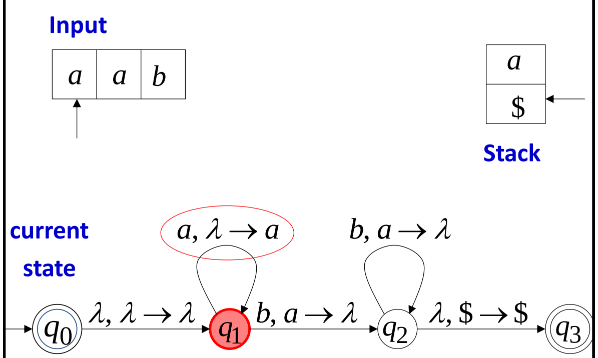
Rejection Example: Time 0



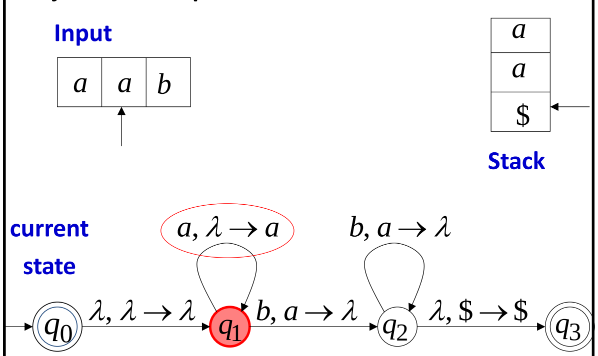
Rejection Example: Time 1



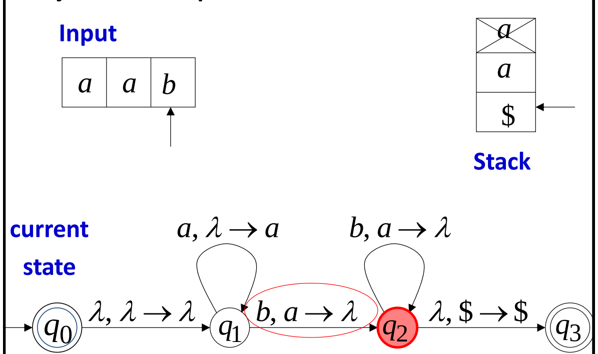
Rejection Example: Time 2



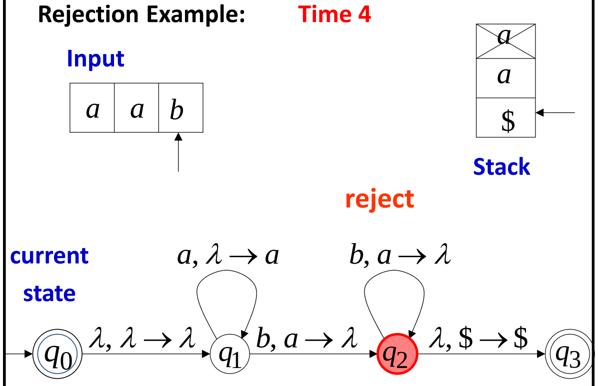
Rejection Example: Time 3



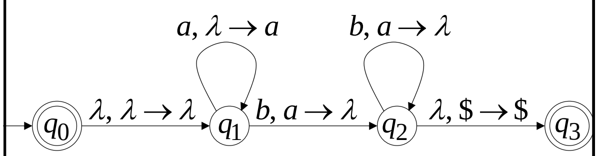
Rejection Example: Time 4



Rejection Example: Time 4



The input string *aab*
is rejected by the PDA:



A string is rejected if there is
no computation such that:

All the input is consumed
AND
The last state is an accept state

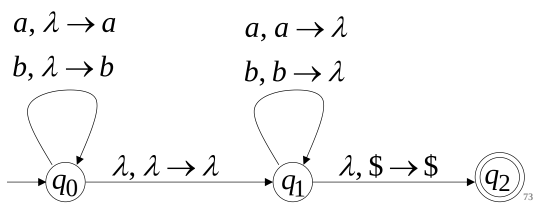
At the end of the computation,
we do not care about the stack contents

72

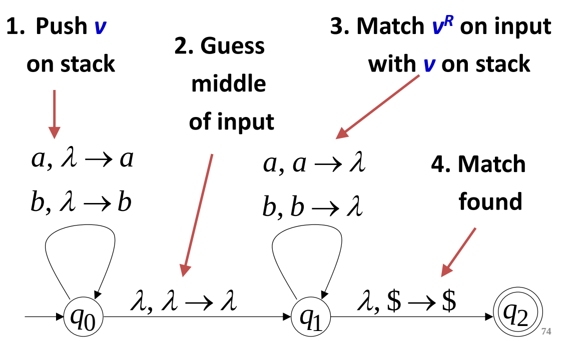
Another PDA example

***L* (*M*) = {vv*R* : *v* ∈ {*a*,*b*}*}**

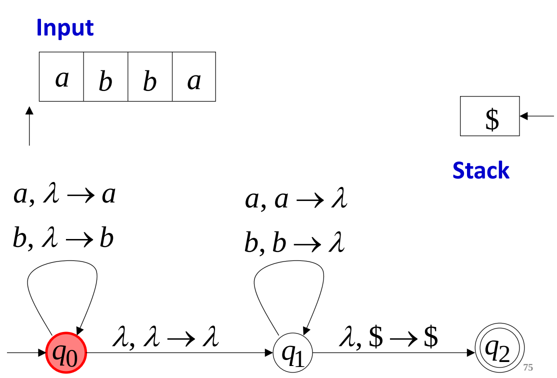
PDA *M*

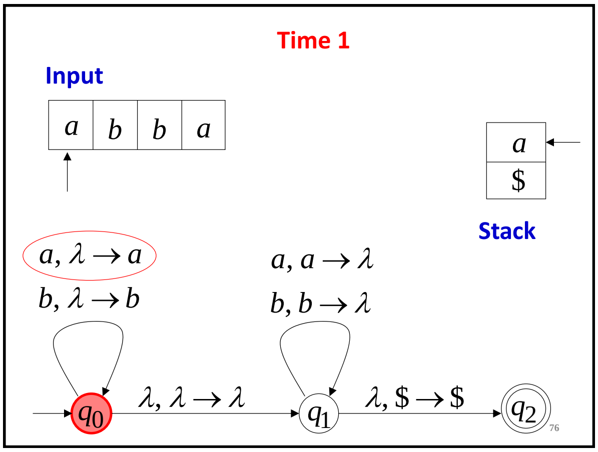


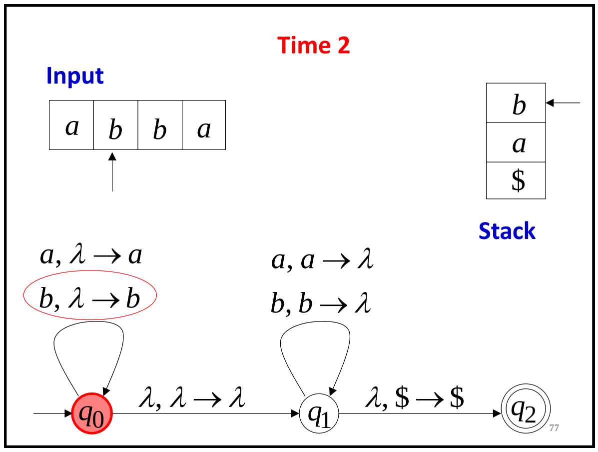
Basic Idea: ***L* (*M*) = {vv*R* : *v* ∈ {*a*,*b*}*}**

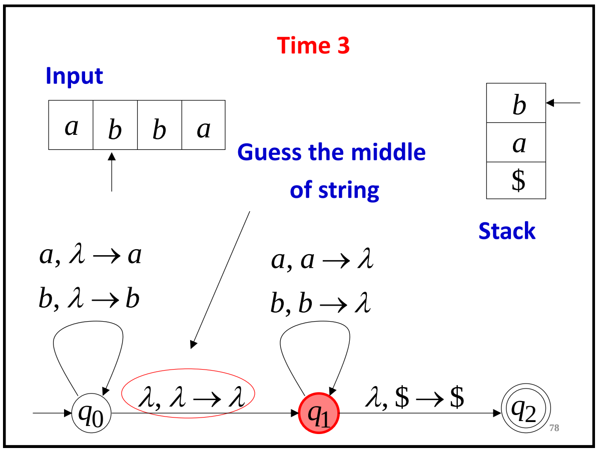


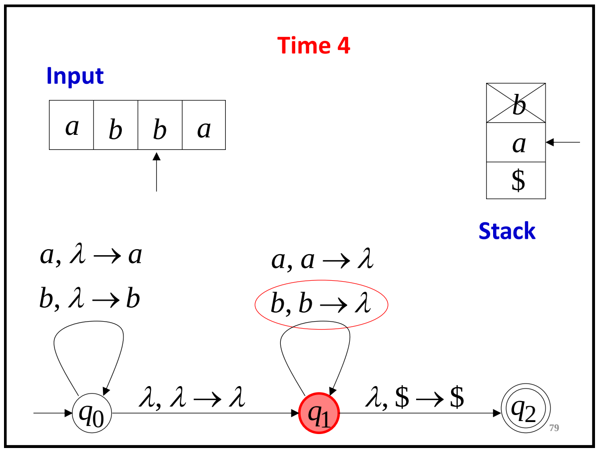
Execution Example: Time 0

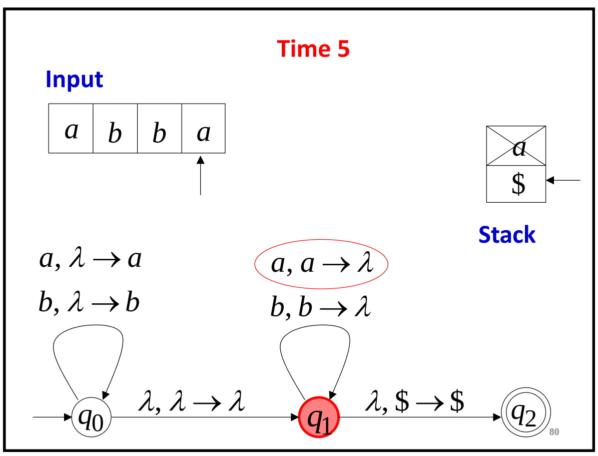


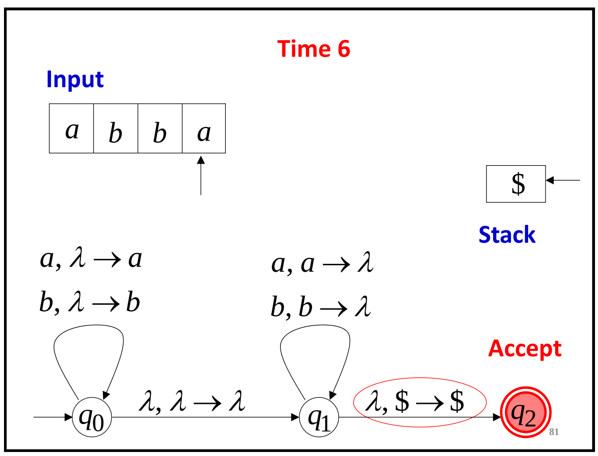

Time 1
Input
a b b a
a
$
Stack
a, λ → a
b, λ → b
a, a → λ
b, b → λ
λ, λ → λ
λ, $ → $
q0
q1
q2
76

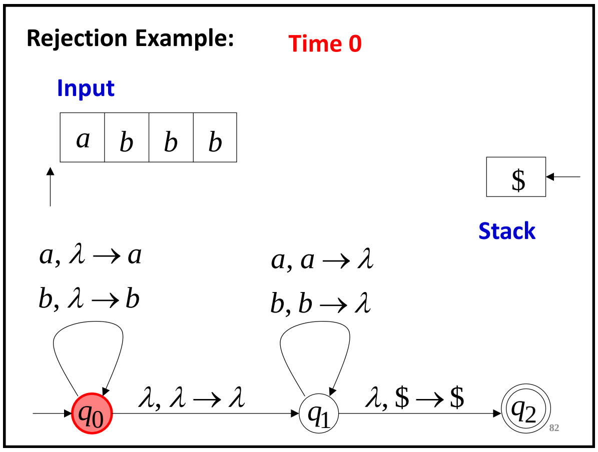

Time 2
Input
a b b a
b
a
$
Stack
a, λ → a
b, λ → b
a, a → λ
b, b → λ
λ, λ → λ
λ, $ → $
q0
q1
q2
77

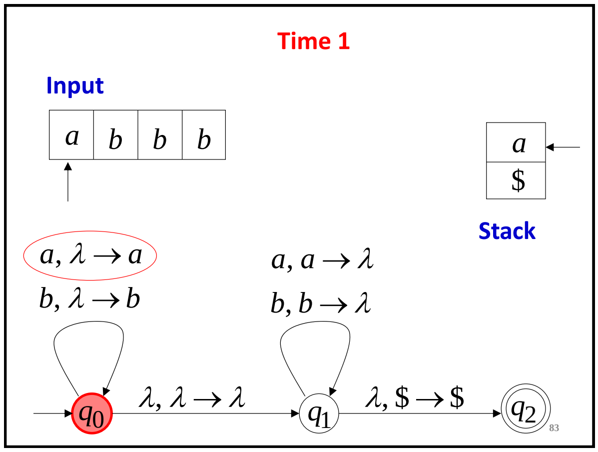

Time 3
Input
a b b a
Guess the middle
of string
b
a
$
Stack
a, λ → a
b, λ → b
a, a → λ
b, b → λ
λ, λ → λ
λ, $ → $
q0
q1
q2
78

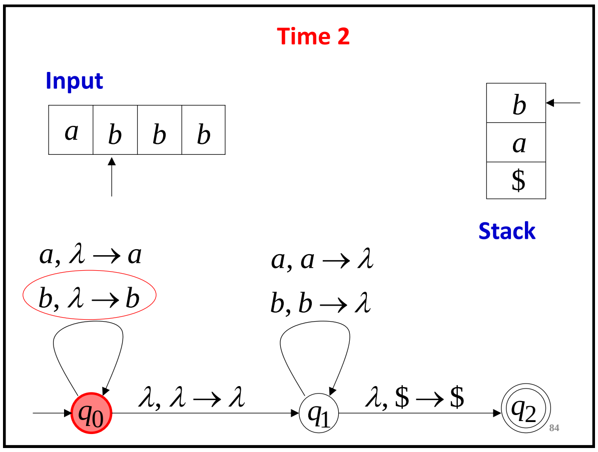

Time 4
Input
a b b a
b
a
$
Stack
a, λ → a
b, λ → b
a, a → λ
b, b → λ
λ, λ → λ
λ, $ → $
q0
q1
q2
79

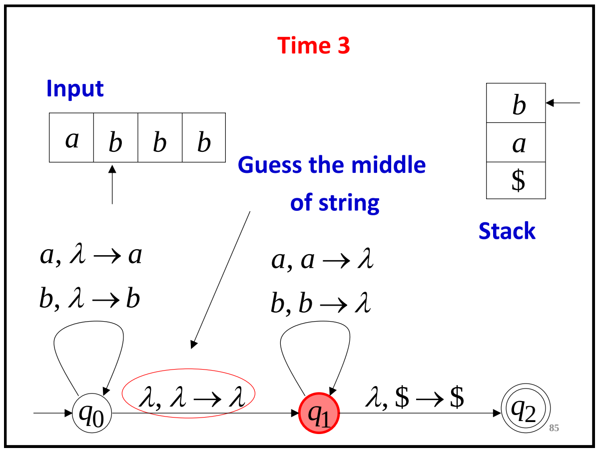

Time 5
Input
a b b a
a
$
Stack
a, λ → a
b, λ → b
a, a → λ
b, b → λ
λ, λ → λ
λ, $ → $
q0
q1
q2
80

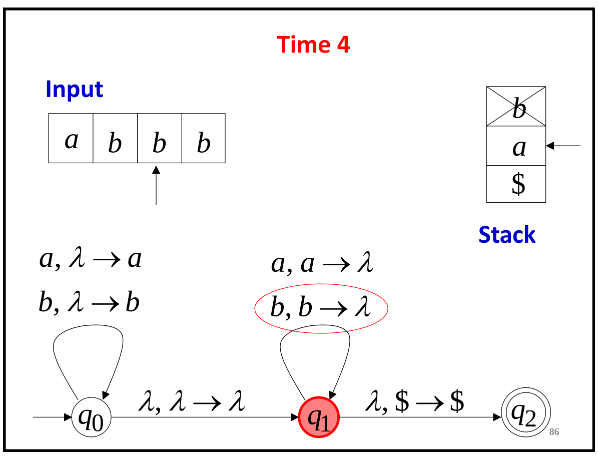

Time 6
Input
a b b a
$
Stack
a, λ → a
b, λ → b
a, a → λ
b, b → λ
Accept
λ, λ → λ
λ, $ → $
q0
q1
q2
81

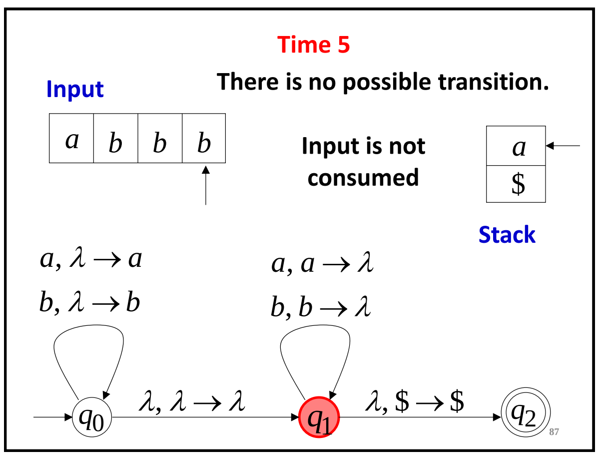

Rejection Example:
Time 0
Input
a b b b
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0 q_1 q_2
82

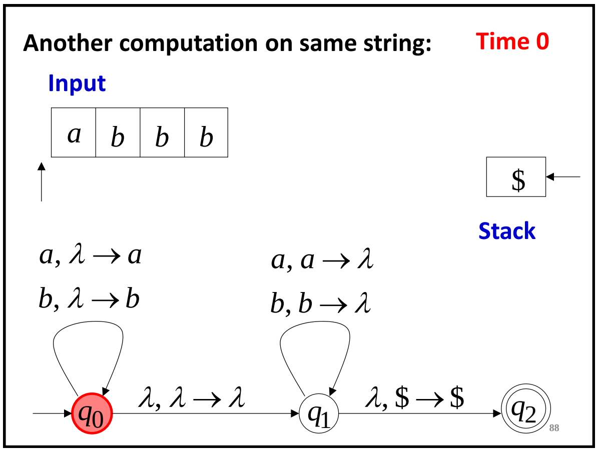

Time 1
Input
a b b b
a
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0 q_1 q_2
83

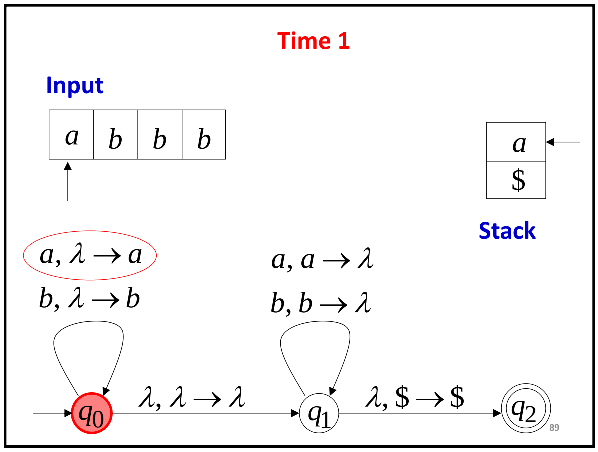

Time 2
Input
a b b b
b
a
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0 q_1 q_2
84

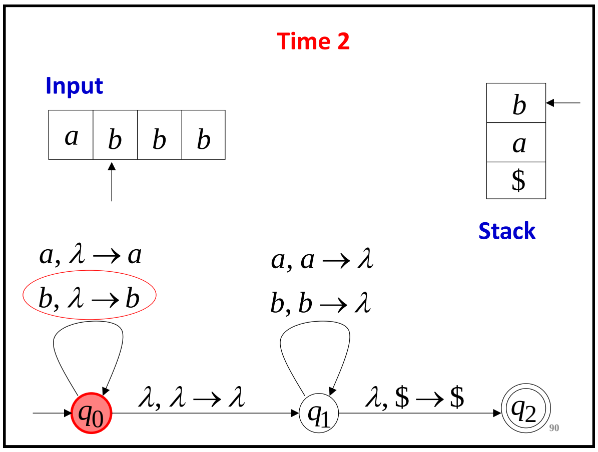

Time 3
Input
a b b b
Guess the middle
of string
b
a
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0 q_1 q_2
85

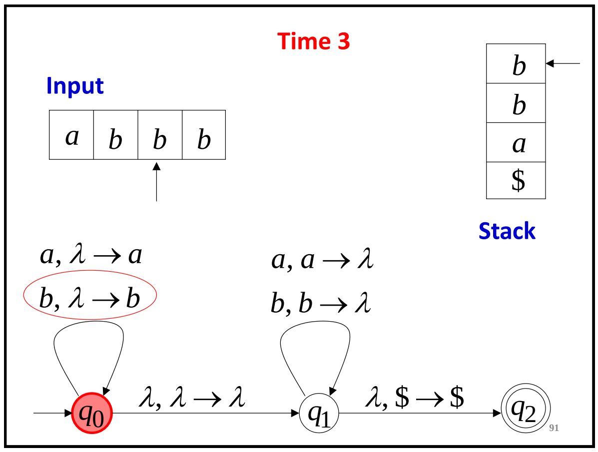

Time 4
Input
a b b b
b
a
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0 q_1 q_2
86

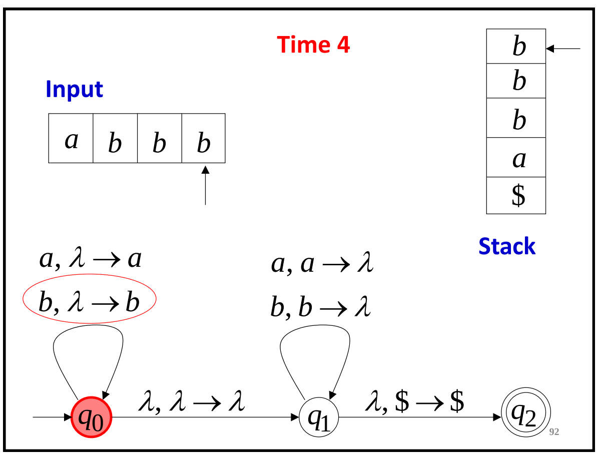

Time 5
There is no possible transition.
Input
a b b b
Input is not
consumed
a
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0 q_1 q_2
87

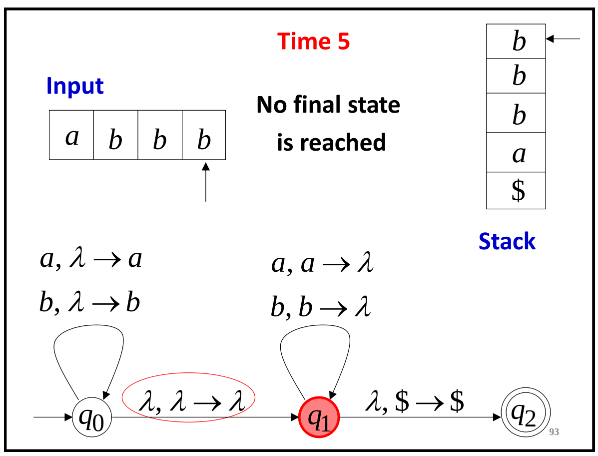

Another computation on same string:
Time 0
Input
a b b b
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2
88


Time 1
Input
a b b b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2
89


Time 2
Input
a b b b
b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2
90

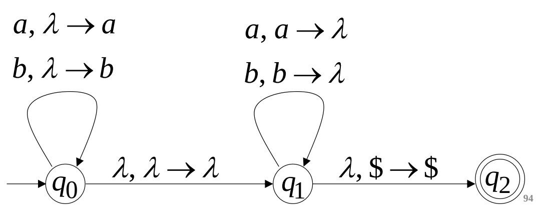

Time 3
Input
a b b b
b
b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2
91


Time 4
Input
a b b b
b
b
b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2
92


Time 5
Input
a b b b
No final state
is reached
b
b
b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2
93

There is no computation that accepts string ***abbb***

abbb $\notin L(M)$

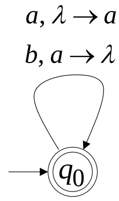


94

Another PDA example

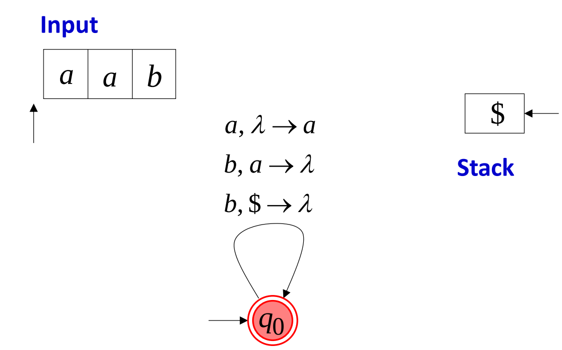
$L(M) = \{w \in \{a,b\}^*:$ in every prefix v, $n_a(v) \geq n_b(v)\}$

PDA M



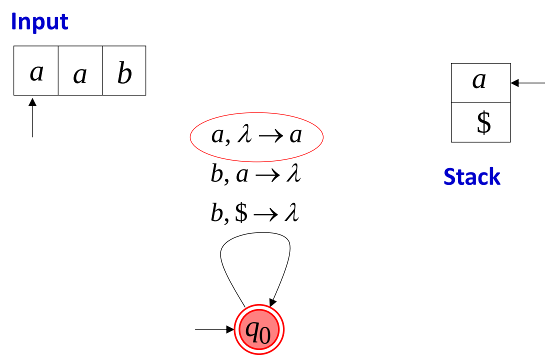
95

Execution Example: Time 0



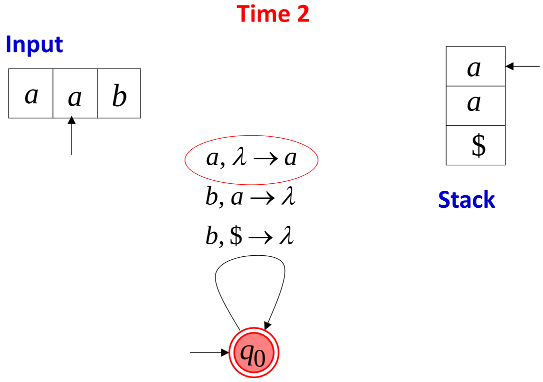
96

Time 1



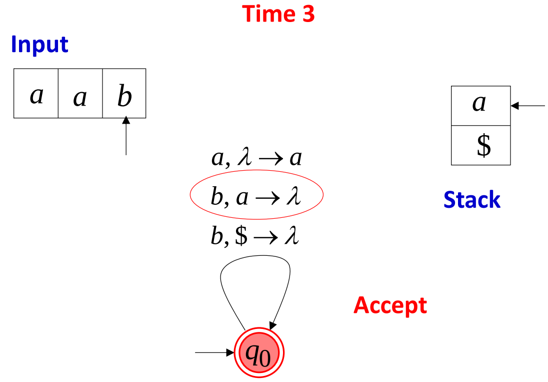
97

Time 2

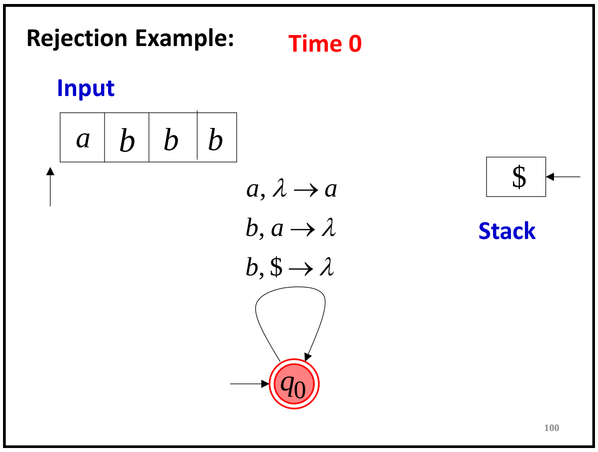


98

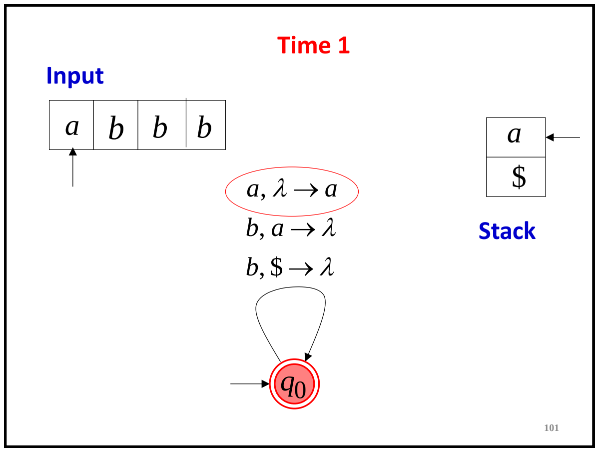
Time 3



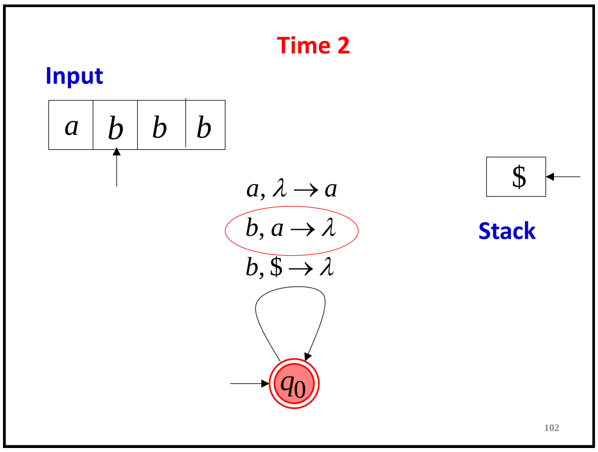
99



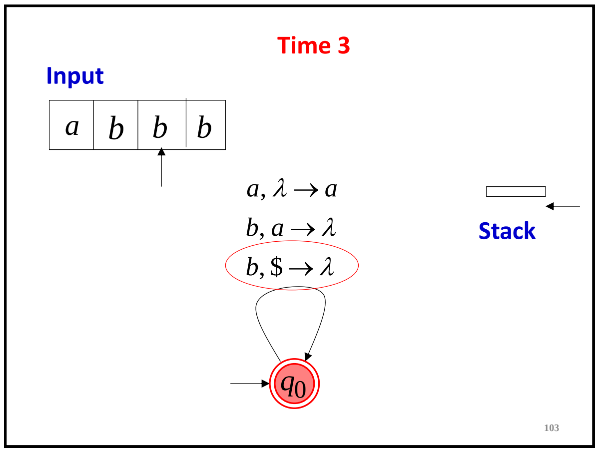
Rejection Example:
Time 0
Input
a b b b
$a, \lambda \to a$
$b, a \to \lambda$
$b, \$ \to \lambda$
q_0
$
Stack
100



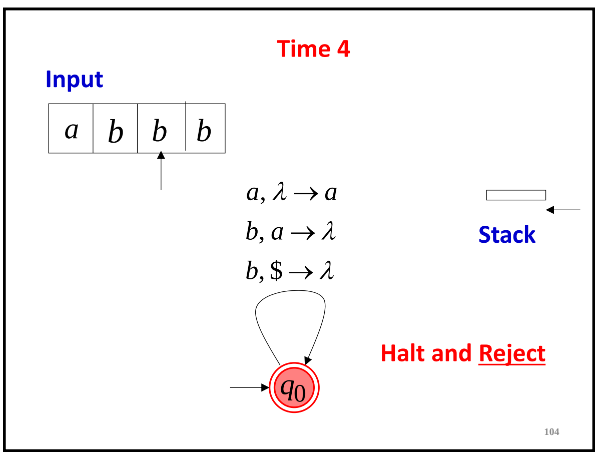
Time 1
Input
a b b b
$a, \lambda \to a$
$b, a \to \lambda$
$b, \$ \to \lambda$
q_0
a
$
Stack
101



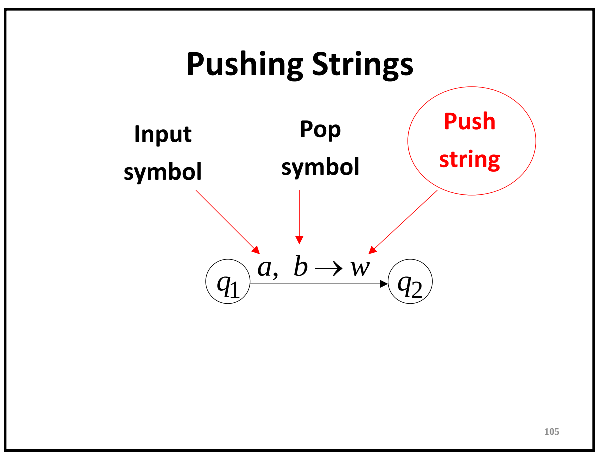
Time 2
Input
a b b b
$a, \lambda \to a$
$b, a \to \lambda$
$b, \$ \to \lambda$
q_0
$
Stack
102



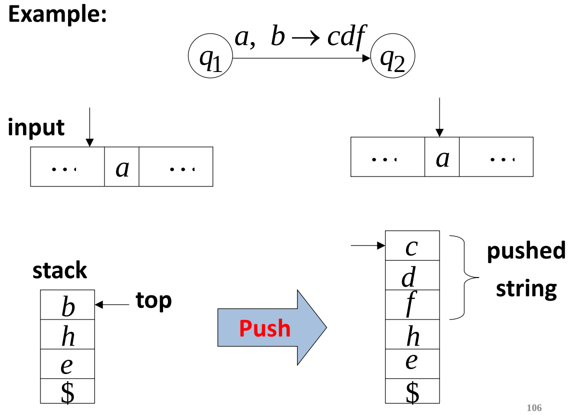
Time 3
Input
a b b b
$a, \lambda \to a$
$b, a \to \lambda$
$b, \$ \to \lambda$
q_0
Stack
103



Time 4
Input
a b b b
$a, \lambda \to a$
$b, a \to \lambda$
$b, \$ \to \lambda$
q_0
Stack
Halt and Reject
104



Pushing Strings
Input symbol
Pop symbol
Push string
q_1
$a, b \to w$
q_2
105



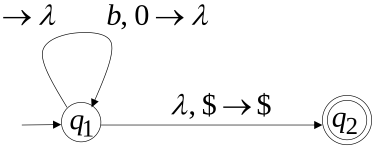
106

Another PDA example

$L(M) = \{w \in \{a,b\}^*: n_a(w) = n_b(w)\}$

PDA M

$a, \$ \to 0\$$ $\quad b, \$ \to 1\$$
$a, 0 \to 00$ $\quad b, 1 \to 11$
$a, 1 \to \lambda$ $\quad b, 0 \to \lambda$



107

Execution Example: Time 0

Input

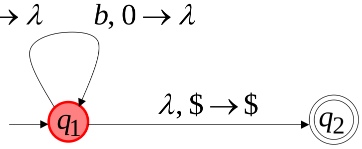
a	b	b	b	a	a

$\$$

Stack

$a, \$ \to 0\$$ $\quad b, \$ \to 1\$$
$a, 0 \to 00$ $\quad b, 1 \to 11$
$a, 1 \to \lambda$ $\quad b, 0 \to \lambda$

current state



108

Time 1

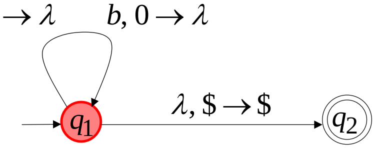
Input

a	b	b	b	a	a

0
$\$$

Stack

$a, \$ \to 0\$$ $\quad b, \$ \to 1\$$
$a, 0 \to 00$ $\quad b, 1 \to 11$
$a, 1 \to \lambda$ $\quad b, 0 \to \lambda$



109

Time 3

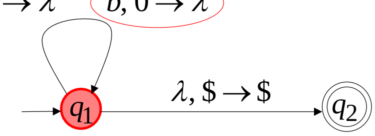
Input

a	b	b	b	a	a

$\cancel{0}$
$\$$

Stack

$a, \$ \to 0\$$ $\quad b, \$ \to 1\$$
$a, 0 \to 00$ $\quad b, 1 \to 11$
$a, 1 \to \lambda$ $\quad b, 0 \to \lambda$



110

Time 4

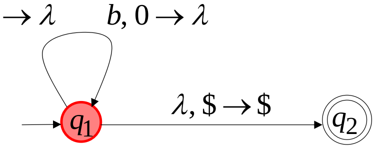
Input

a	b	b	b	a	a

1
$\$$

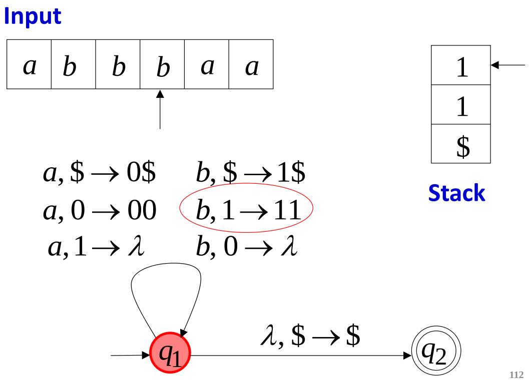
Stack

$a, \$ \to 0\$$ $\quad b, \$ \to 1\$$
$a, 0 \to 00$ $\quad b, 1 \to 11$
$a, 1 \to \lambda$ $\quad b, 0 \to \lambda$

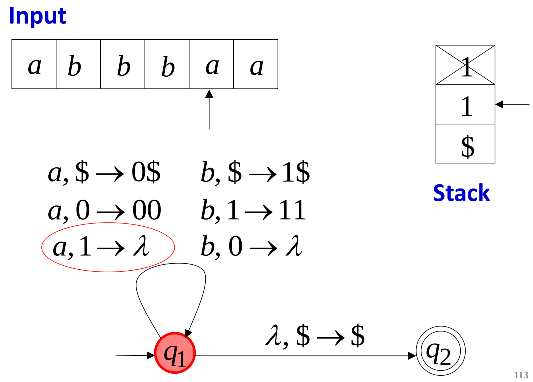


111

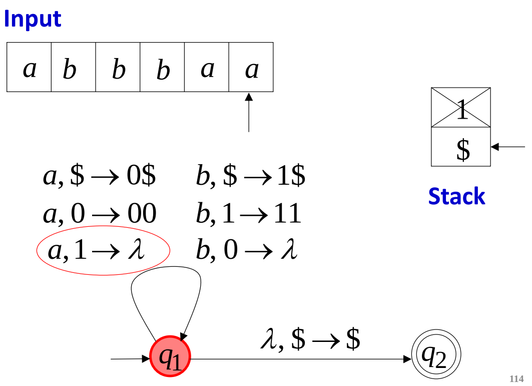
Time 5



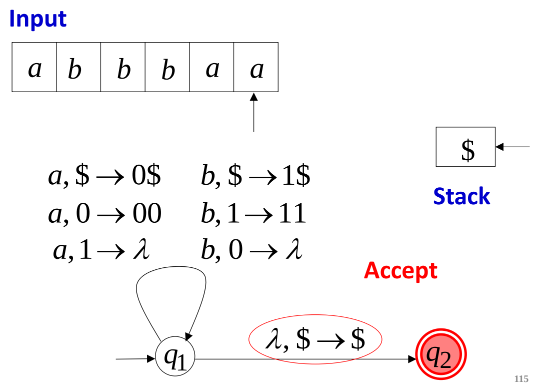
Time 6



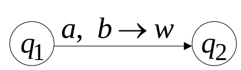
Time 7



Time 8

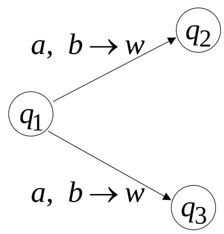


Formalities for PDAs



Transition function:

$$\delta(q_1, a, b) = \{(q_2, w)\}$$



Transition function:

$$\delta(q_1, a, b) = \{(q_2, w),\ (q_3, w)\}$$

118

Formal Definition

Pushdown Automaton (PDA)

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$$

States

Input alphabet

Stack alphabet

Transition function

Initial state

Stack start symbol

Final states

119

Instantaneous Description

$$(q,\ u,\ s)$$

Current state

Remaining input

Current stack contents

120