San José State University
Department of Computer Science

Ahmad Yazdankhah
ahmad.yazdankhah@sjsu.edu
www.cs.sjsu.edu/~yazdankhah

Pushdown Automata

(Part 2)

Lecture 14

Day 14/31

CS 154

Formal Languages and Computability

Spring 2019

Agenda of Day 14

- Feedback and Solution of Quiz 4 and Quiz +
- Summary of Lecture 13
- Lecture 14: Teaching ...
  - Pushdown Automata (part 2)

Solution and Feedback of Quiz 4 (Out of 22)

Section	Average	High Score	Low Score
01 (TR 3:00 PM)	19.84	22	17
02 (TR 4:30 PM)	18.27	22	10
03 (TR 6:00 PM)	18.9	22	14

Solution and Feedback of Quiz + (Out of 50)

Section	Average	High Score	Low Score
01 (TR 3:00 PM)	45.8	50	38
02 (TR 4:30 PM)	43.53	49	26
03 (TR 6:00 PM)	46.7	50	42

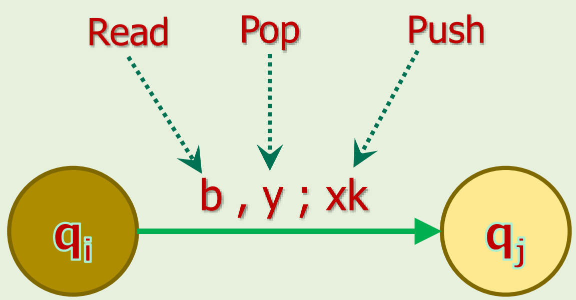
Summary of Lecture 13: We learned ...

PDAs

- NFAs are powerful enough to recognize just regular languages.
- To recognize non-regular languages, we need more powerful machines.
- We noticed that we needed writable memory.
- We added stack to NFAs and ...
- ... introduced PDAs to accept all or at least some of non-regular languages.
- PDA stands for ...

  Pushdown Automata.

- PDAs have both deterministic (DPDA) and nondeterministic (NPDA) versions.
- We talked about the structure of PDAs.



- Condition for transition = ...
  - ... input symbol + top of stack
- We learned how to relax these conditions by λ.

Any question?

Summary of Lecture 13: We learned ...

PDAs

- PDAs halt when ...
  - ... the conditions for the next transition (zero transition) are not satisfied.

$$z \leftrightarrow h$$

- The conditions for a string being accepted by a process ...

$$(h \wedge c \wedge f) \leftrightarrow a$$

- The conditions for a string be rejected by a process ...

$$(\sim h \vee \sim c \vee \sim f) \leftrightarrow \sim a$$

- The content of the stack does not matter for accepting/rejecting.
- The stack alphabet and the input tape alphabet can be totally different.

Any question?

5. PDAs in Action

Design Examples

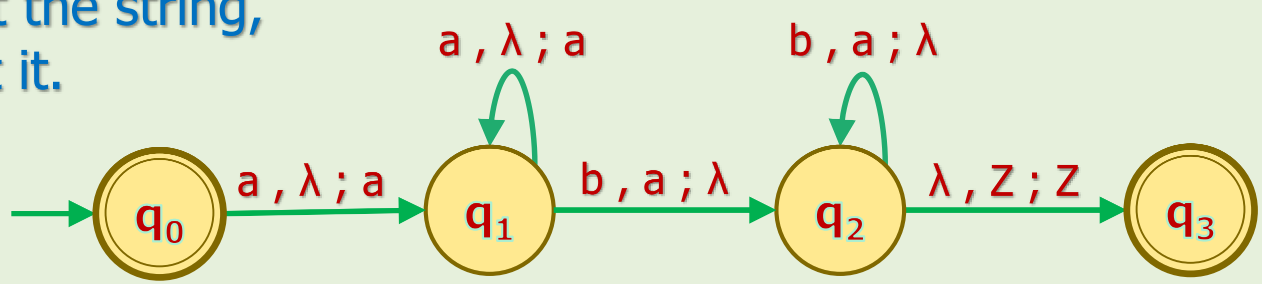
5. PDAs in Action

Example 14

- Design a PDA to accept our famous language $L = \{a^n b^n : n \geq 0\}$.

Solution

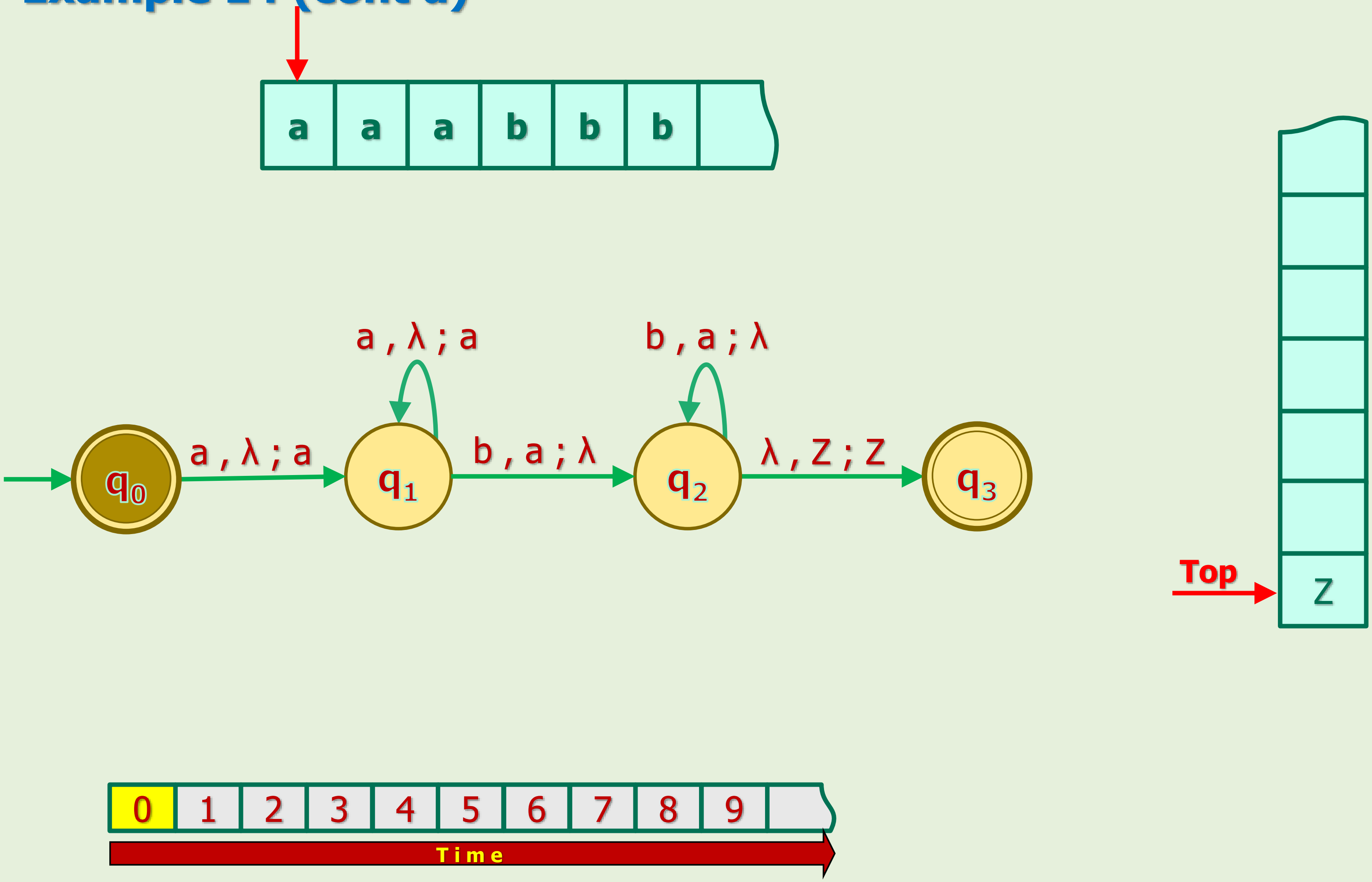
- **Strategy**: read a's and push them in the stack regardless of top of the stack.
- When the first b is sensed, start popping a's to match them with b's.
- Continue popping a's until you are out of b.
- If end of stack is reached, means the number of a's and b's are equal, so, accept the string,
  otherwise, reject it.



- Let's trace this solution for some strings.

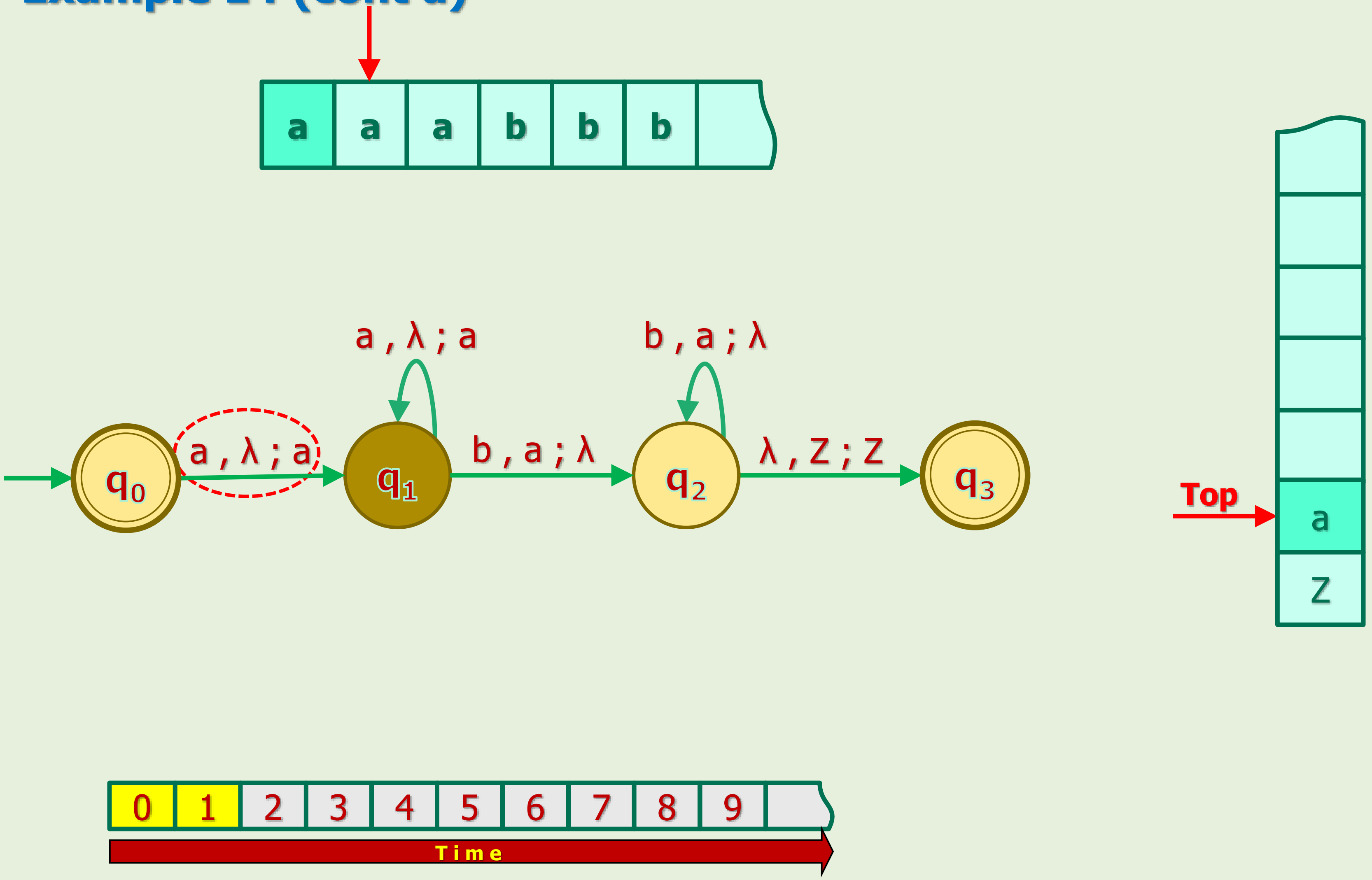
5. PDAs in Action

Example 14 (cont'd)



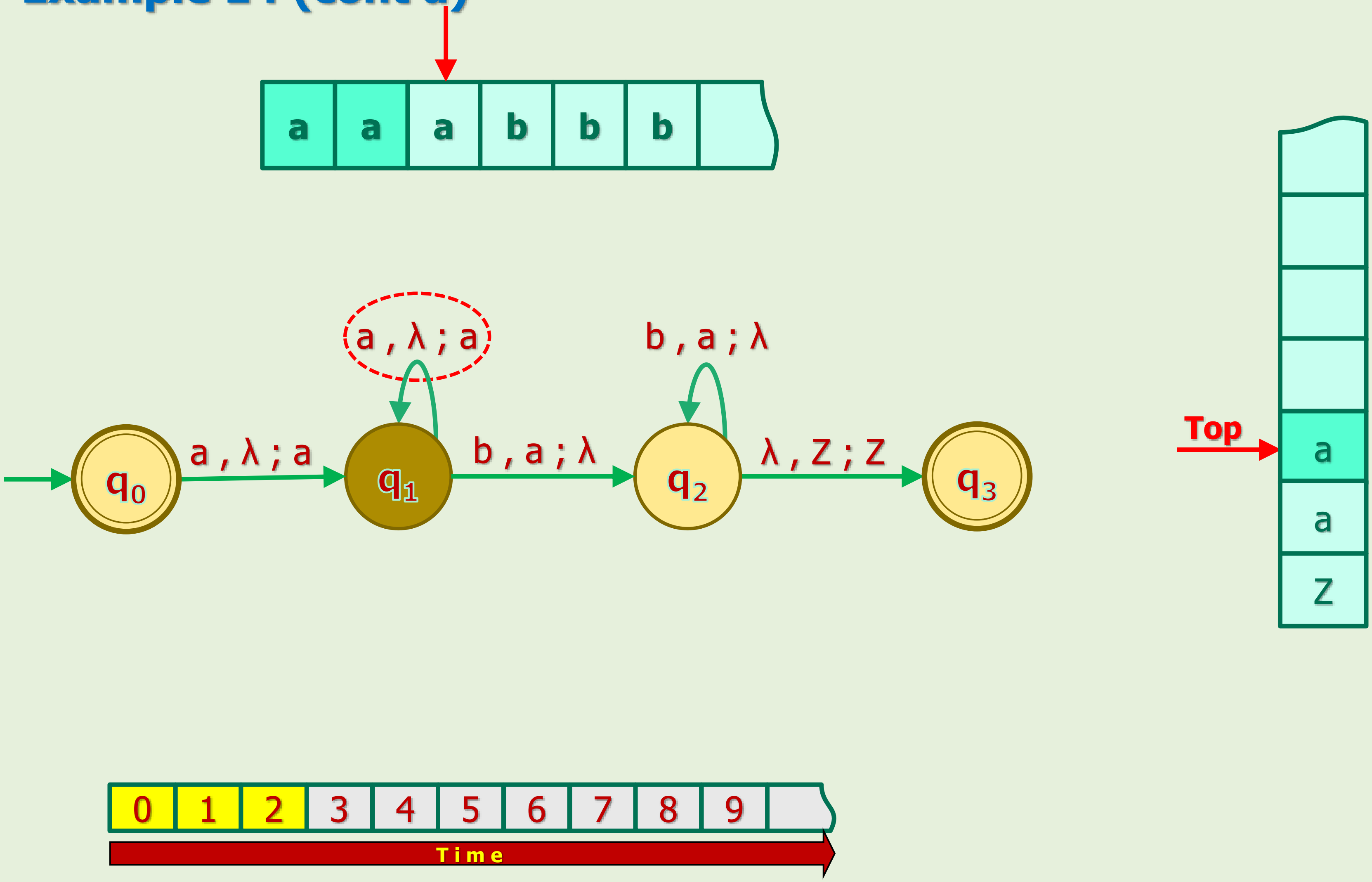
5. PDAs in Action

Example 14 (cont'd)



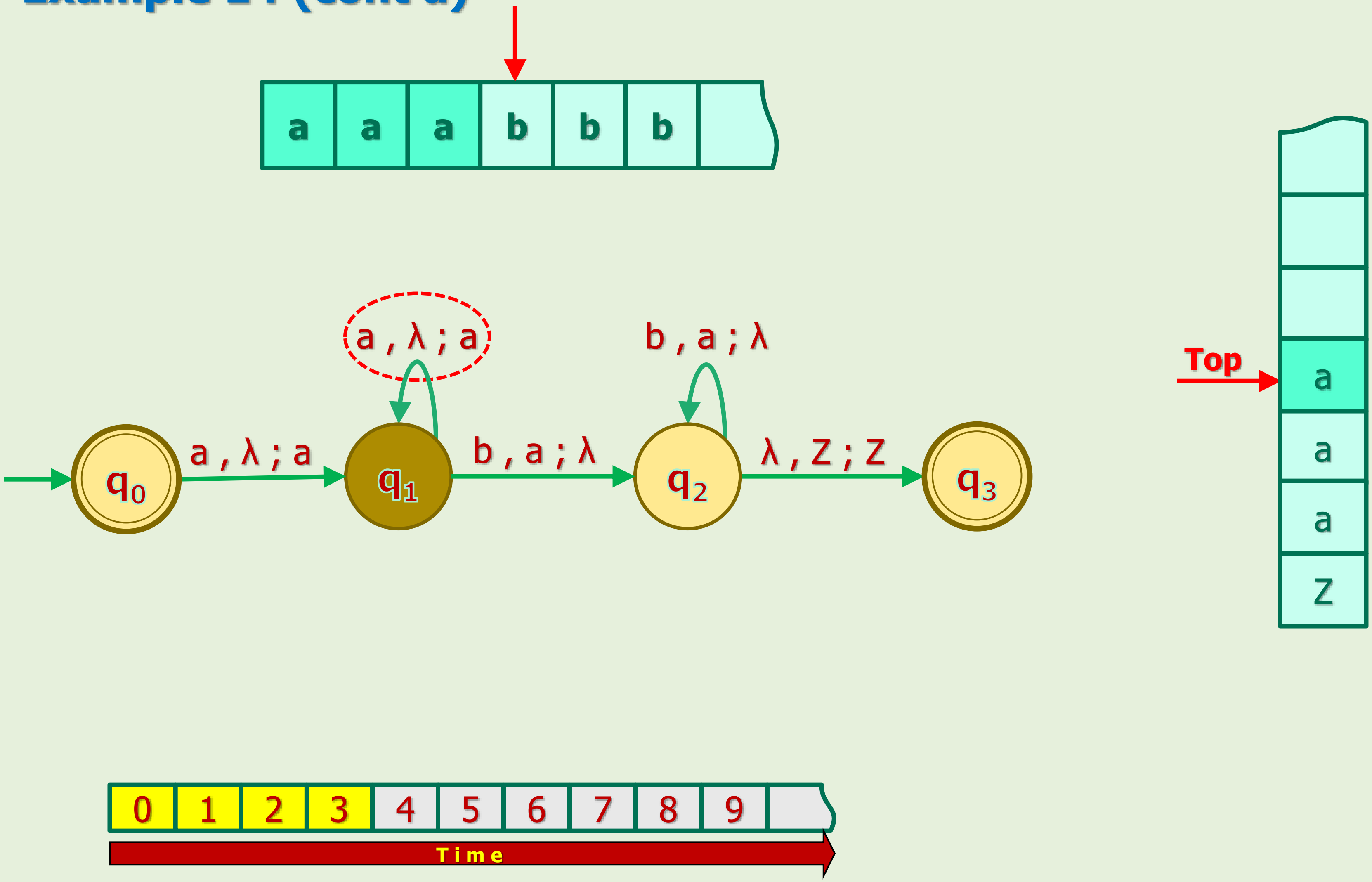
5. PDAs in Action

Example 14 (cont'd)



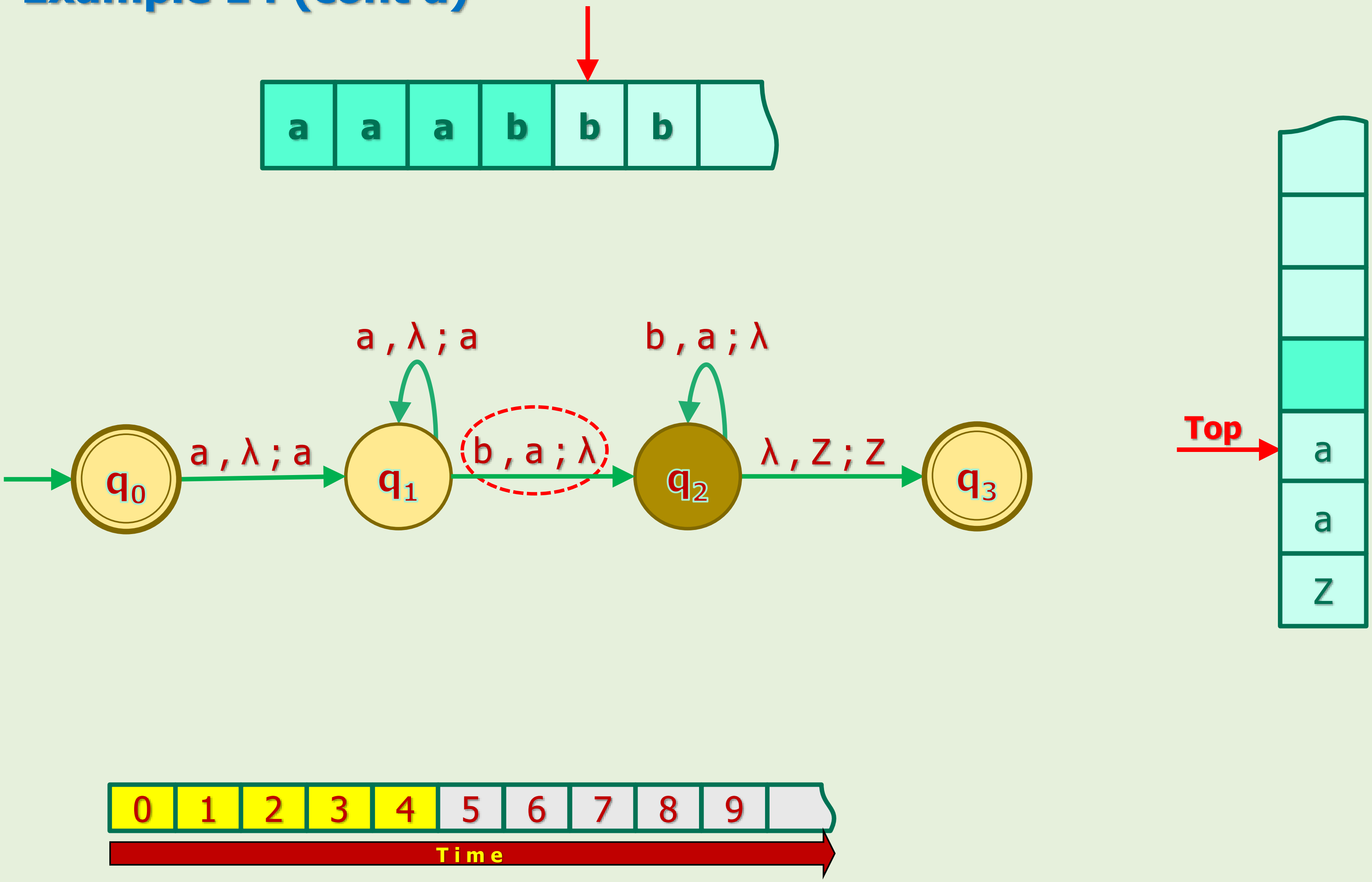
5. PDAs in Action

Example 14 (cont'd)



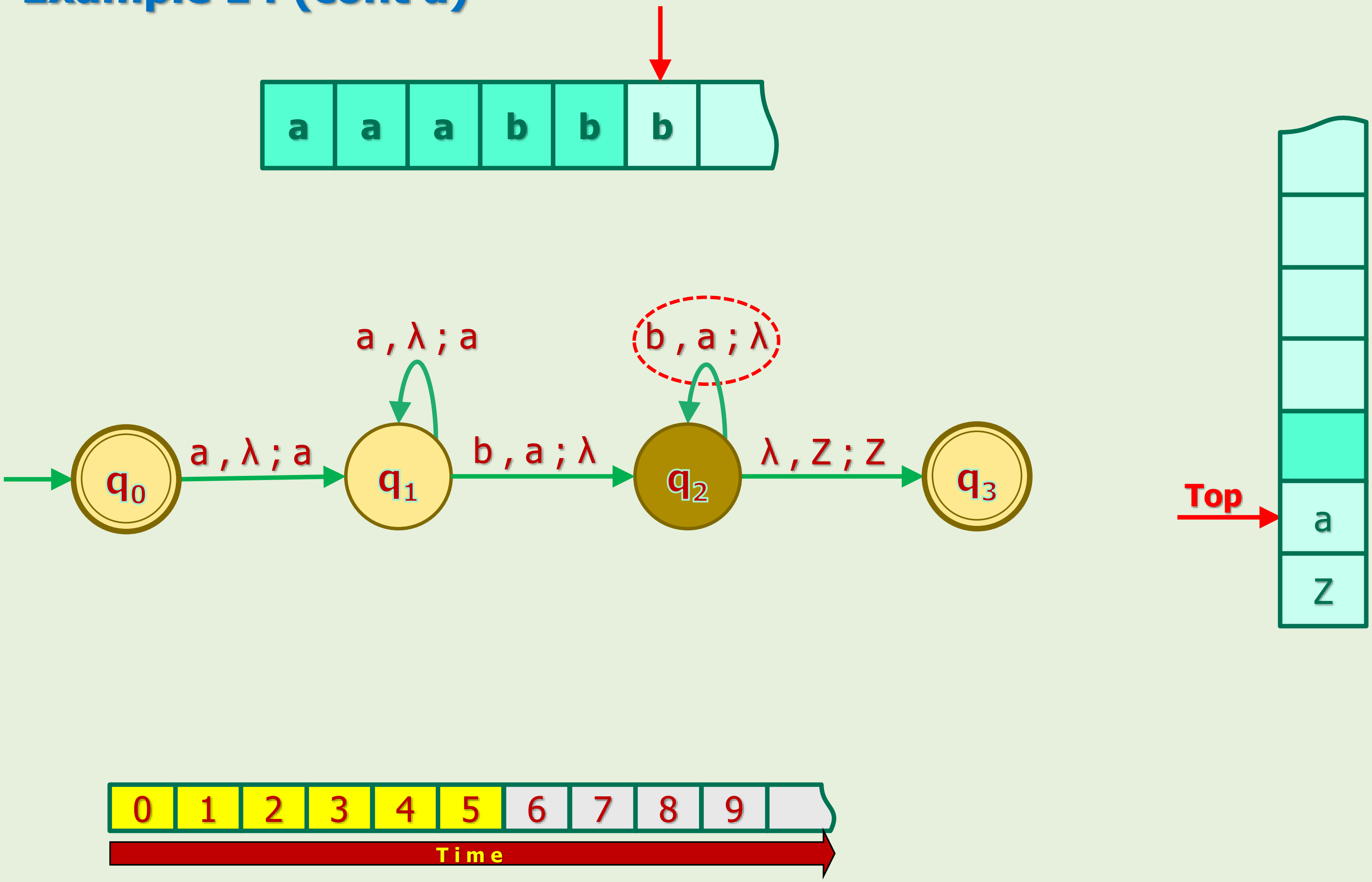
5. PDAs in Action

Example 14 (cont'd)



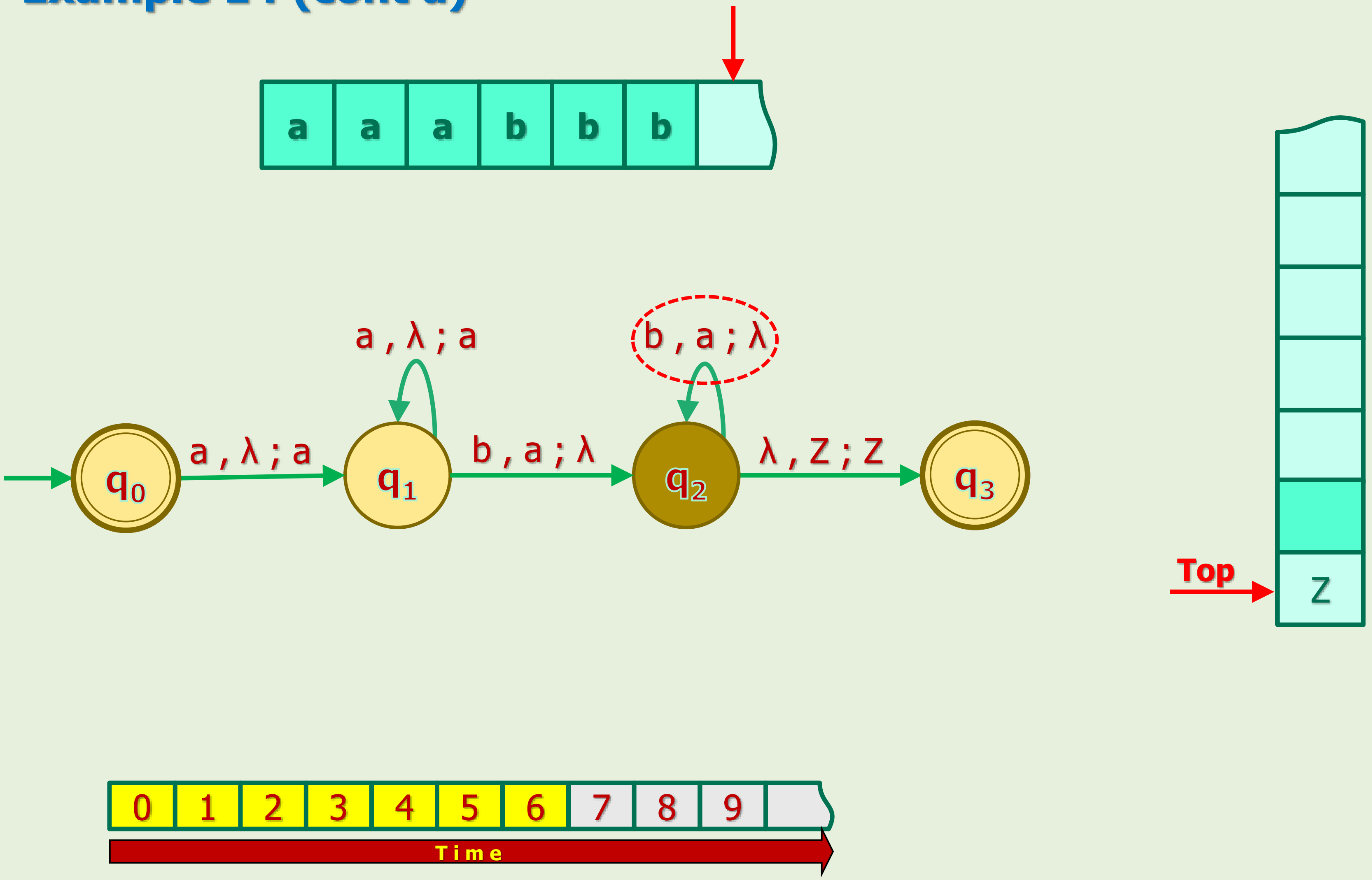
5. PDAs in Action

Example 14 (cont'd)



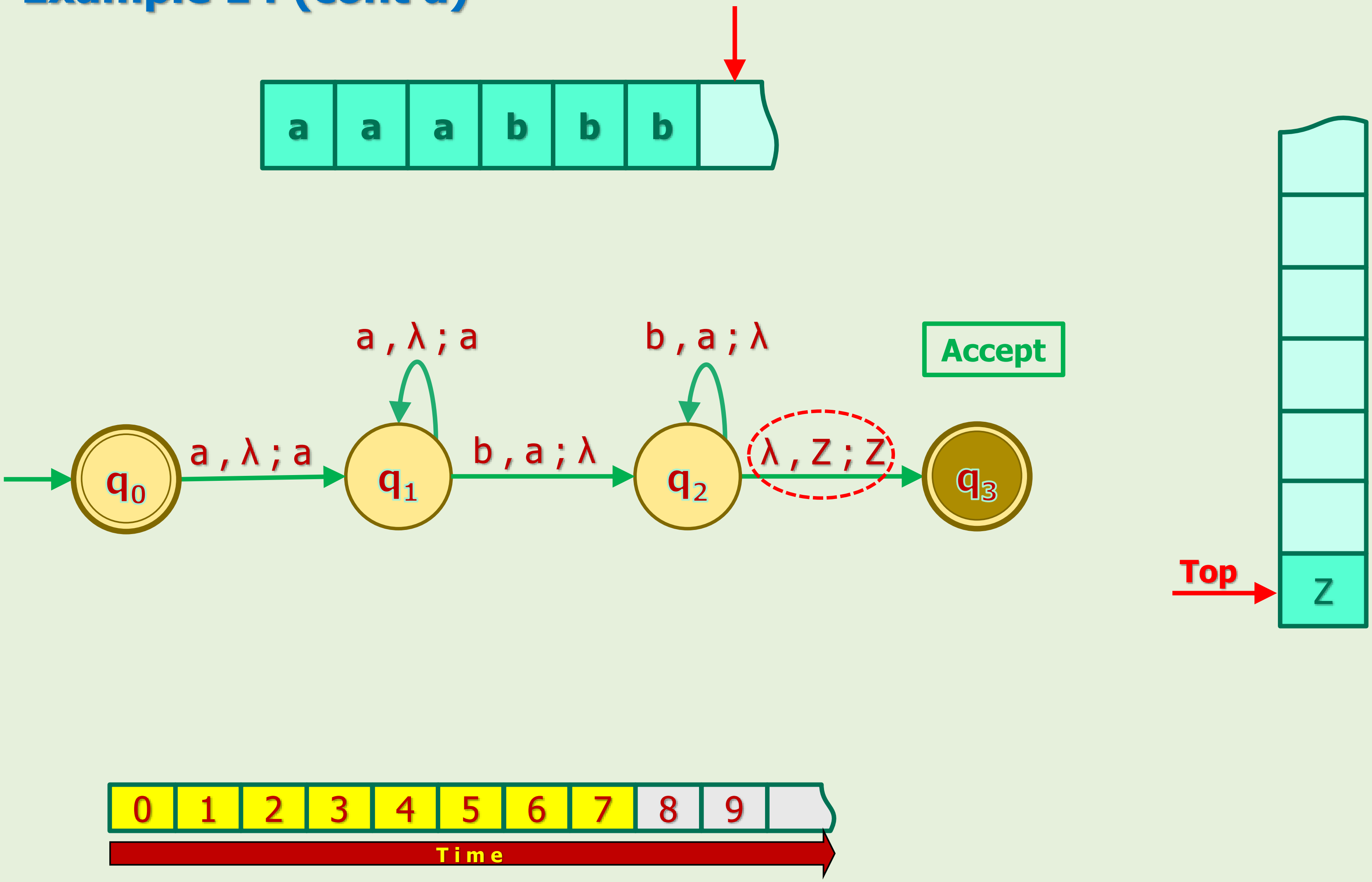
5. PDAs in Action

Example 14 (cont'd)



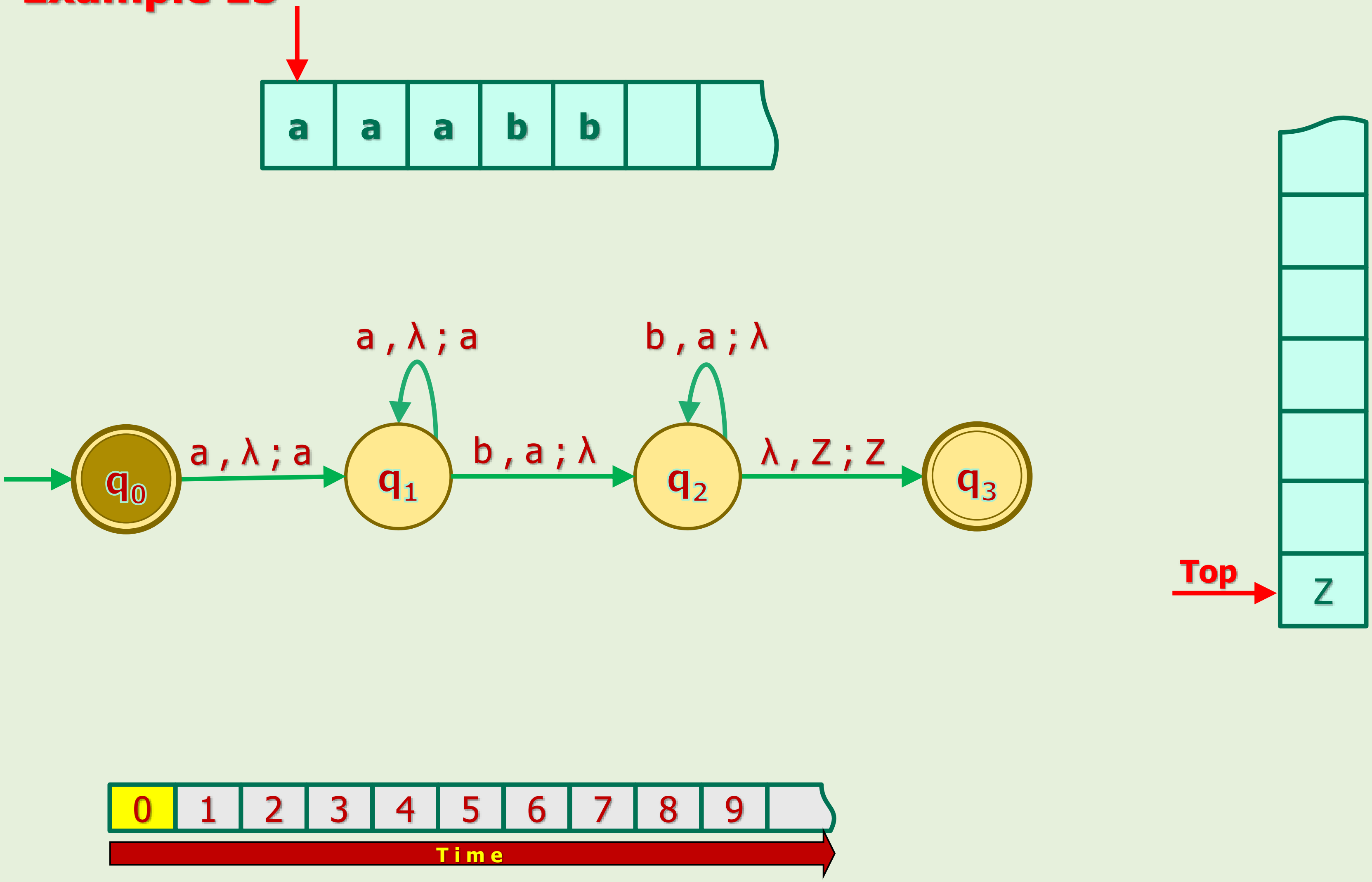
5. PDAs in Action

Example 14 (cont'd)



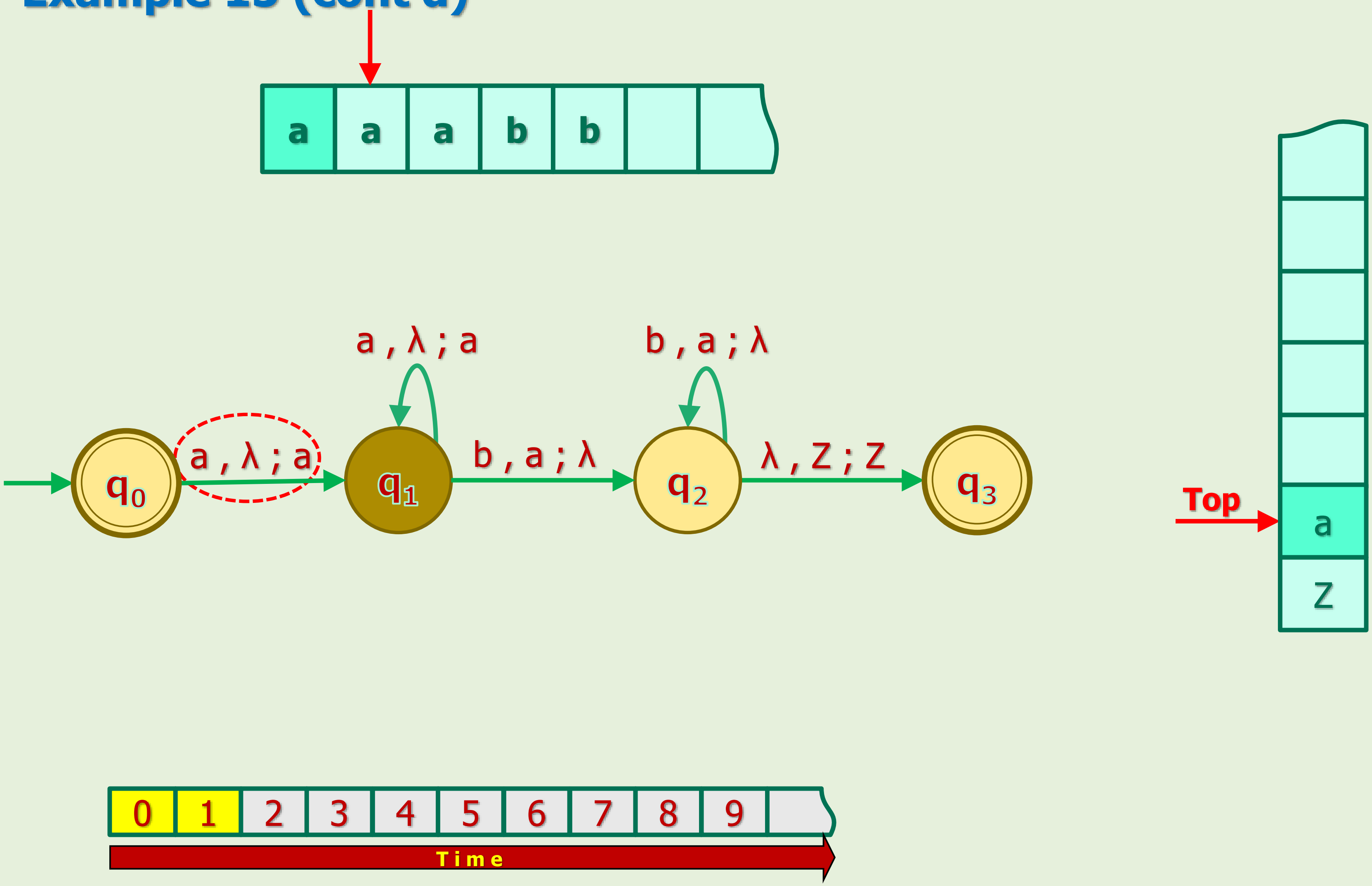
5. PDAs in Action

Example 15



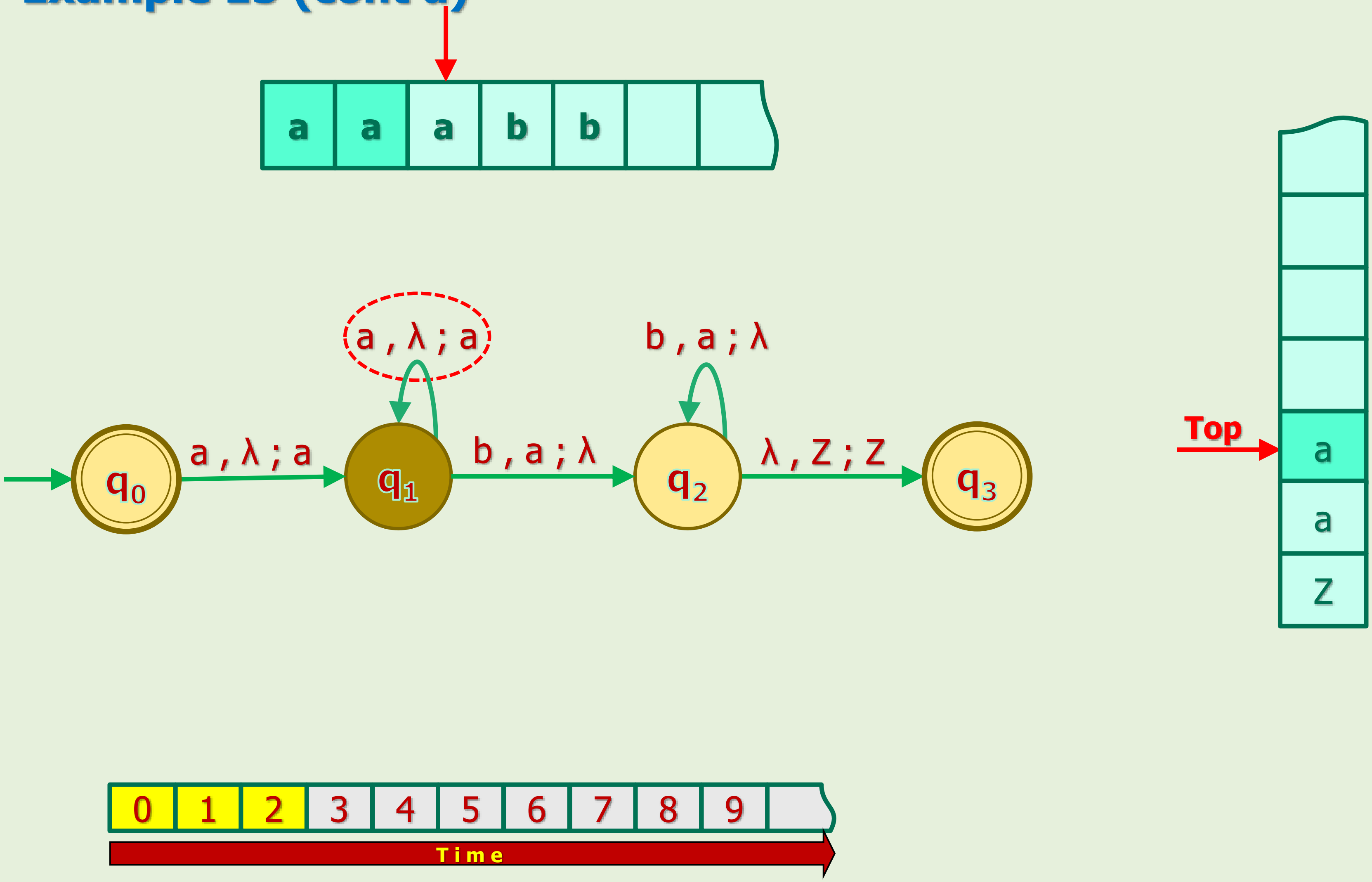
5. PDAs in Action

Example 15 (cont'd)



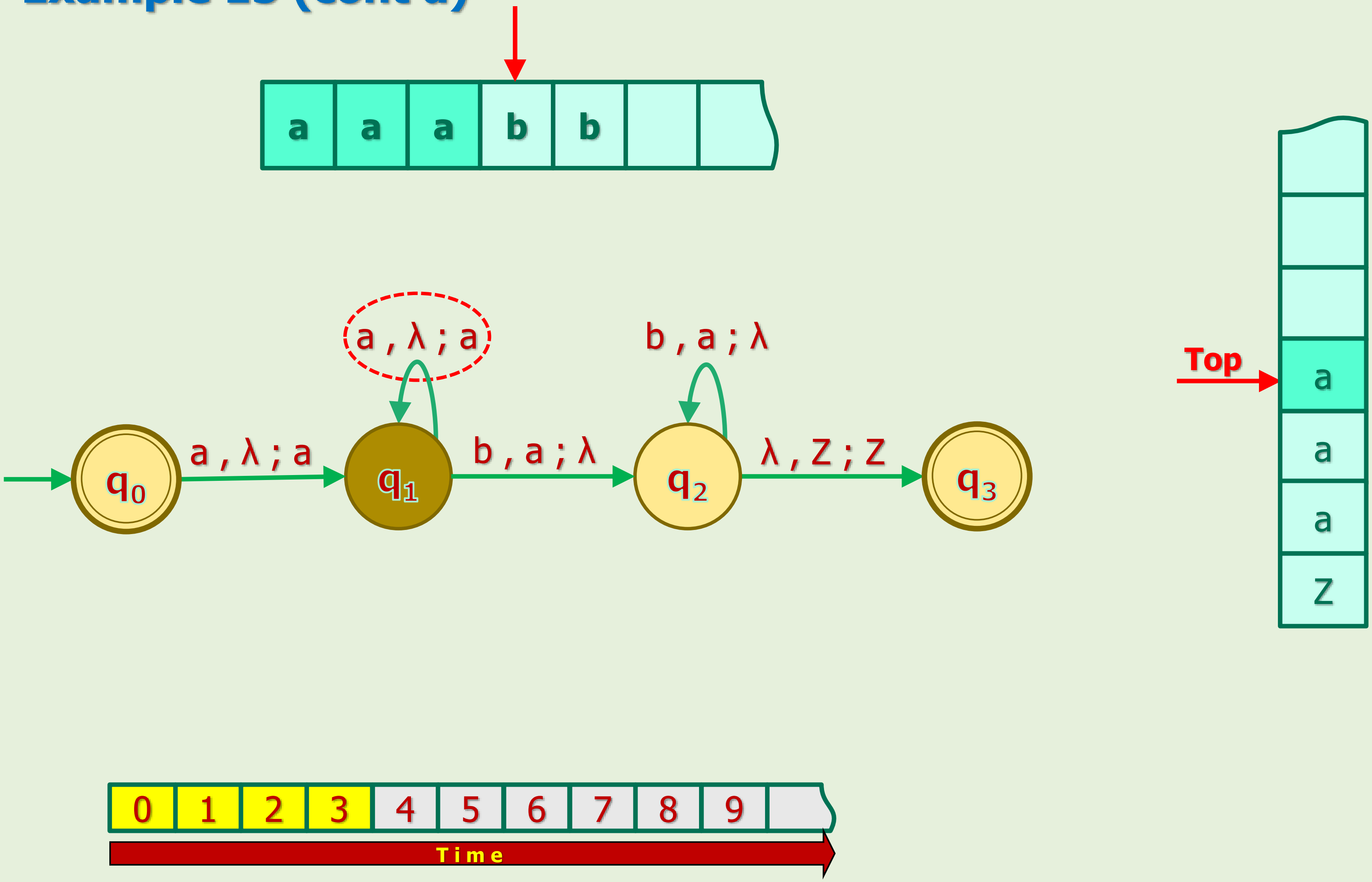
5. PDAs in Action

Example 15 (cont'd)



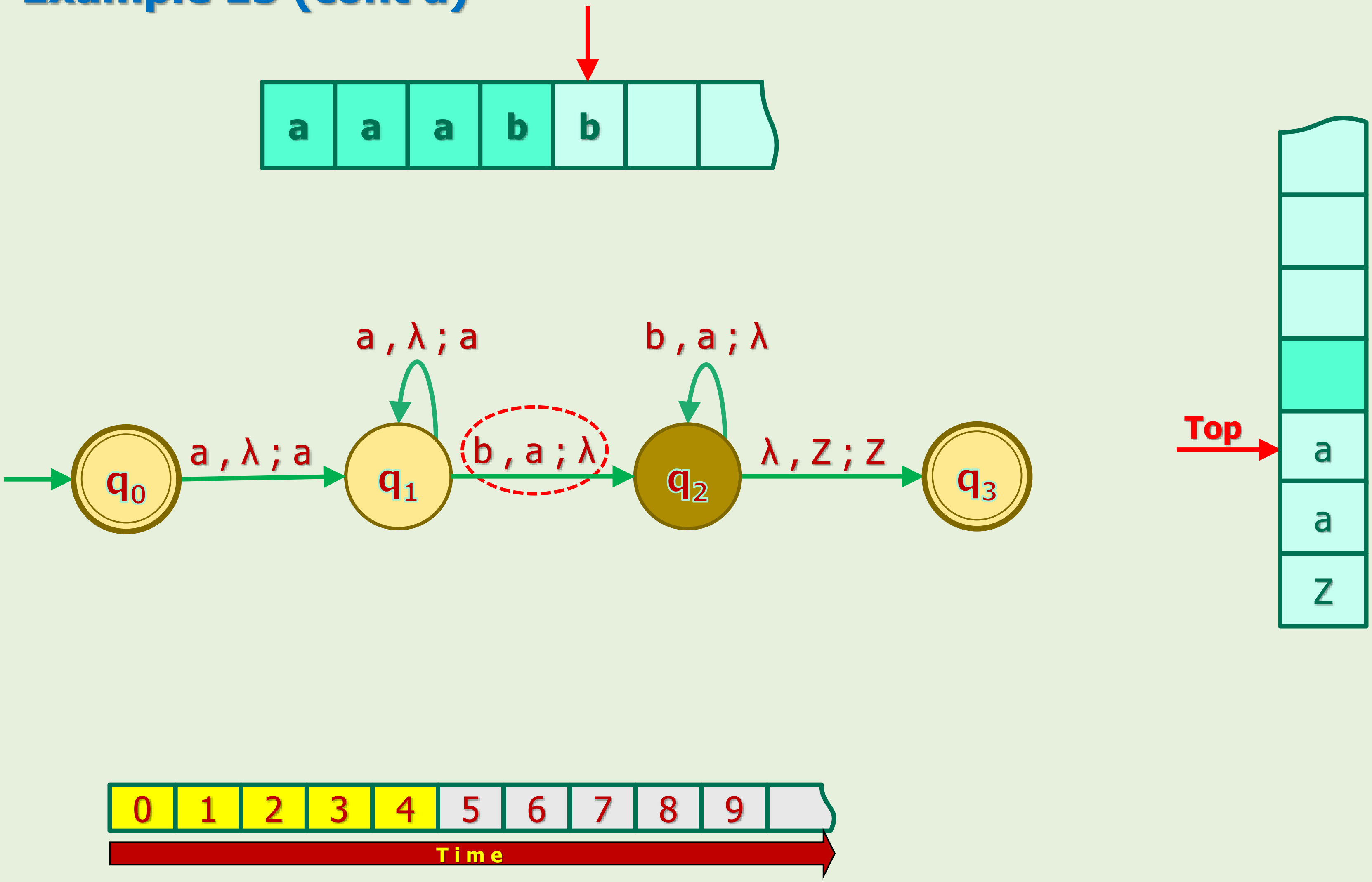
5. PDAs in Action

Example 15 (cont'd)



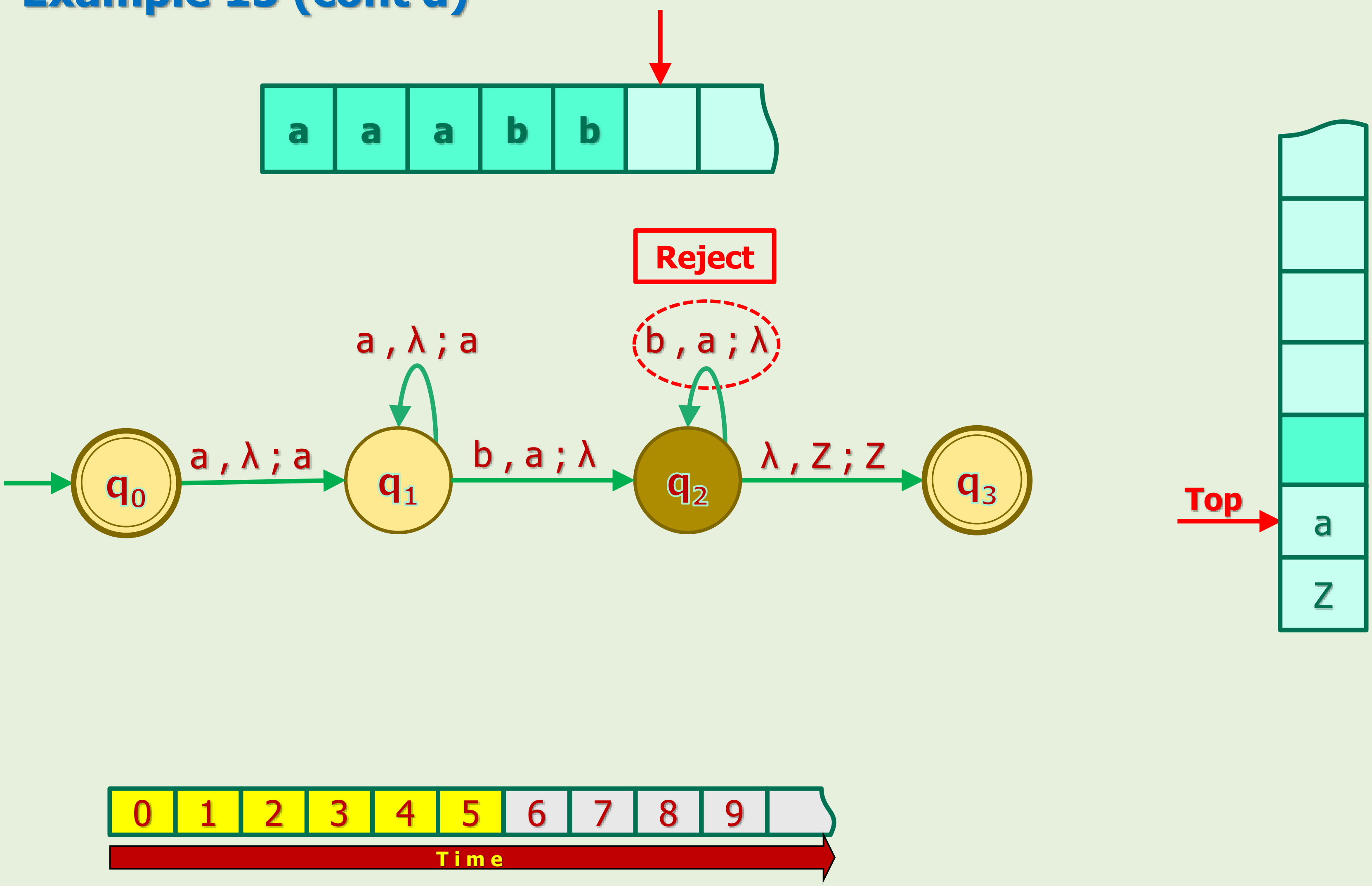
5. PDAs in Action

Example 15 (cont'd)



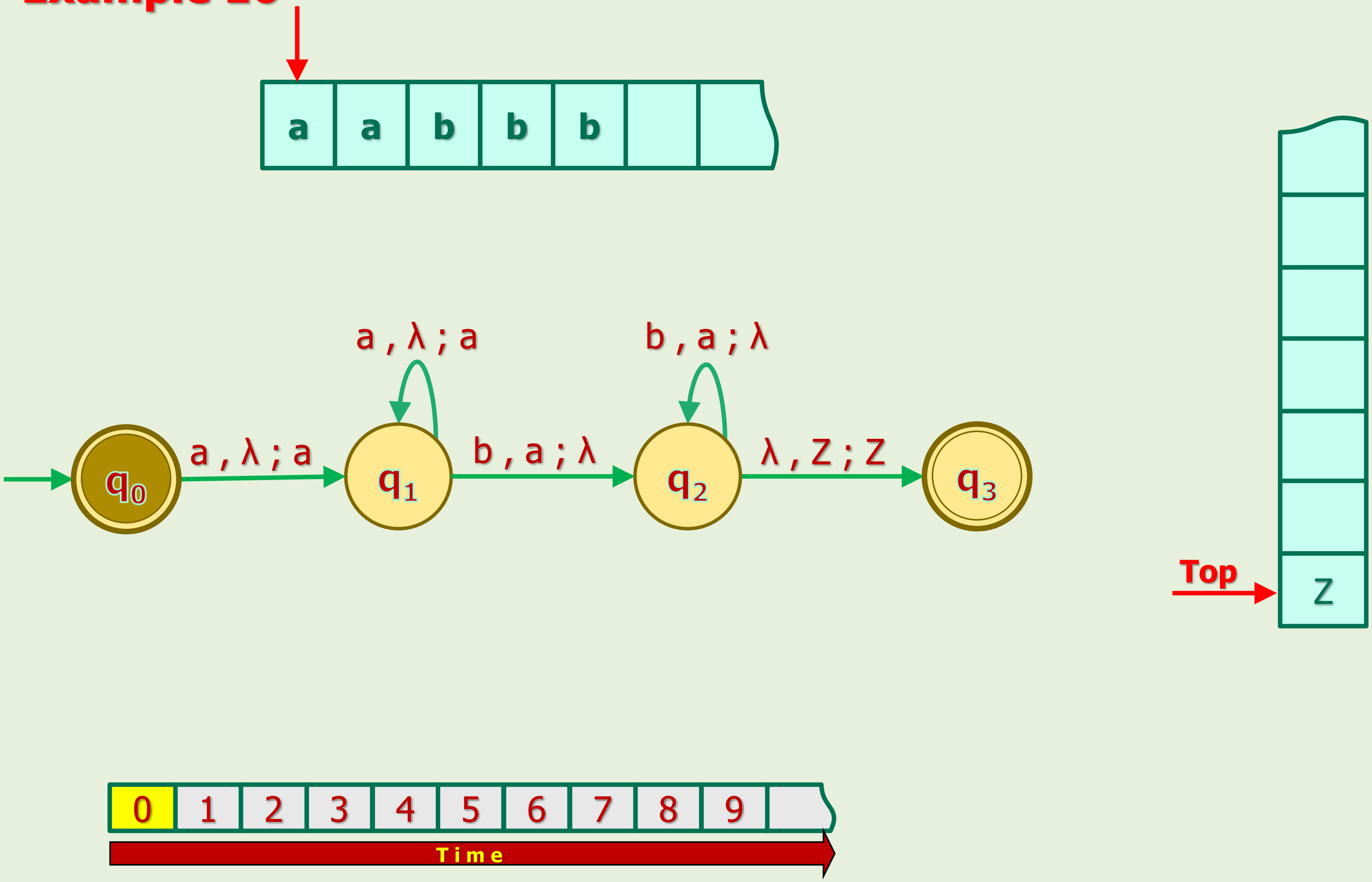
5. PDAs in Action

Example 15 (cont'd)



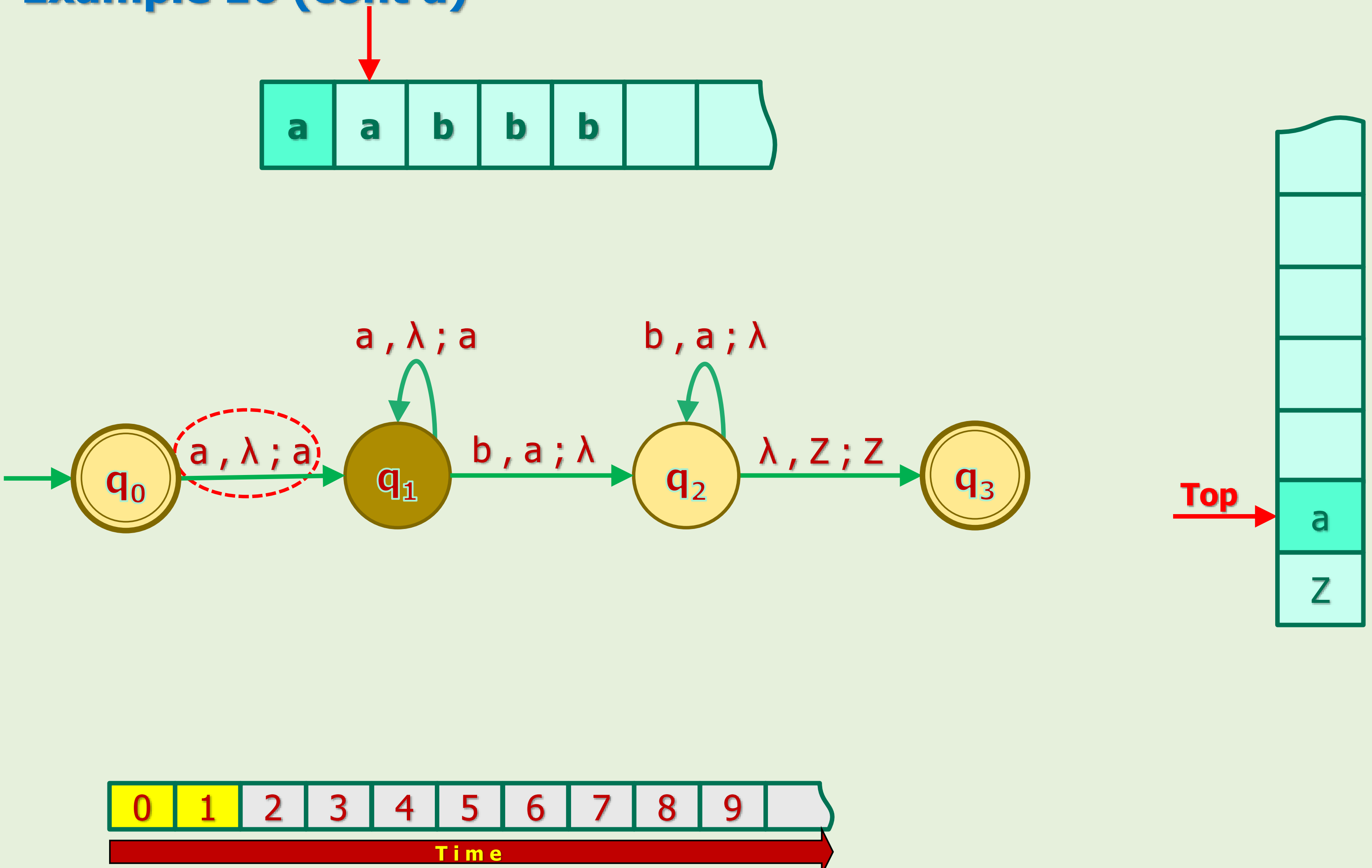
5. PDAs in Action

Example 16



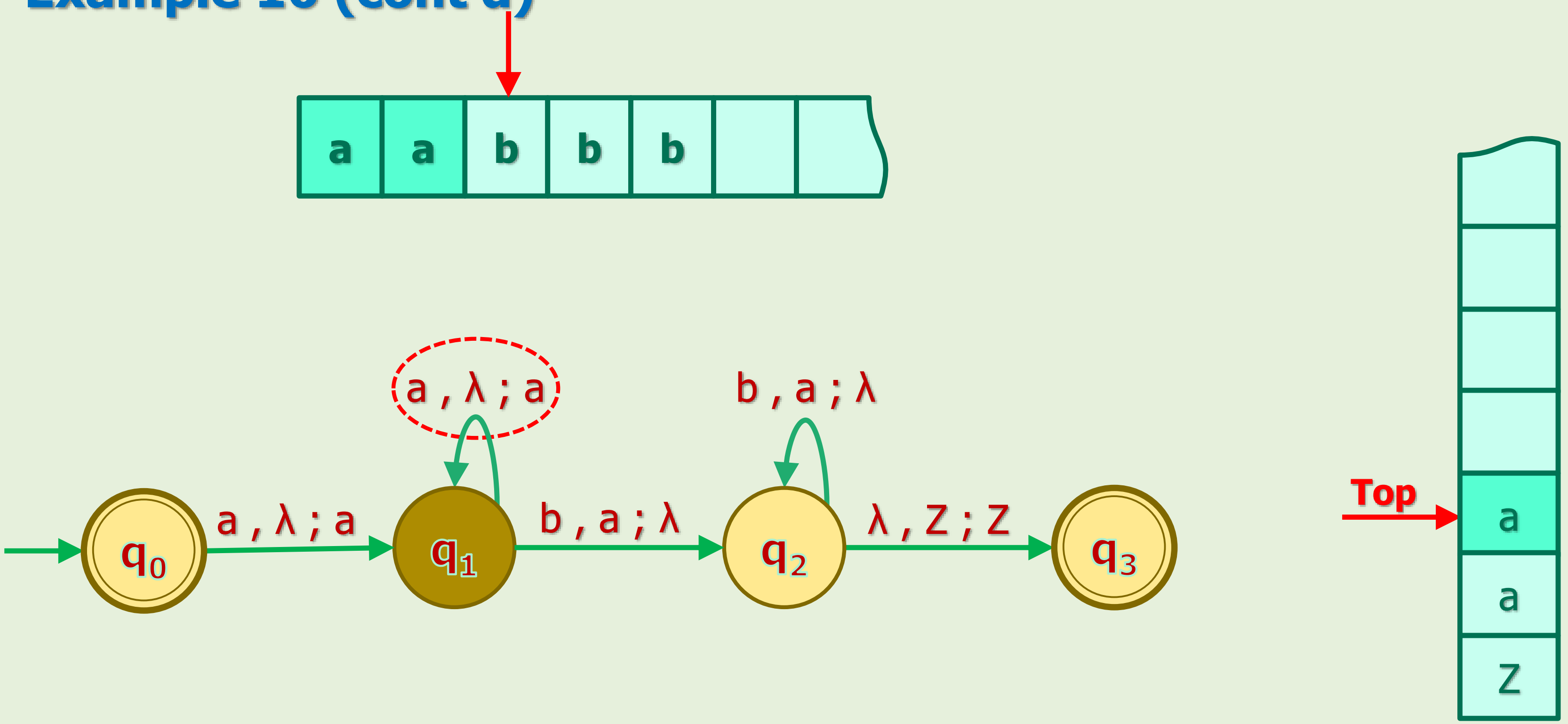
5. PDAs in Action

Example 16 (cont'd)



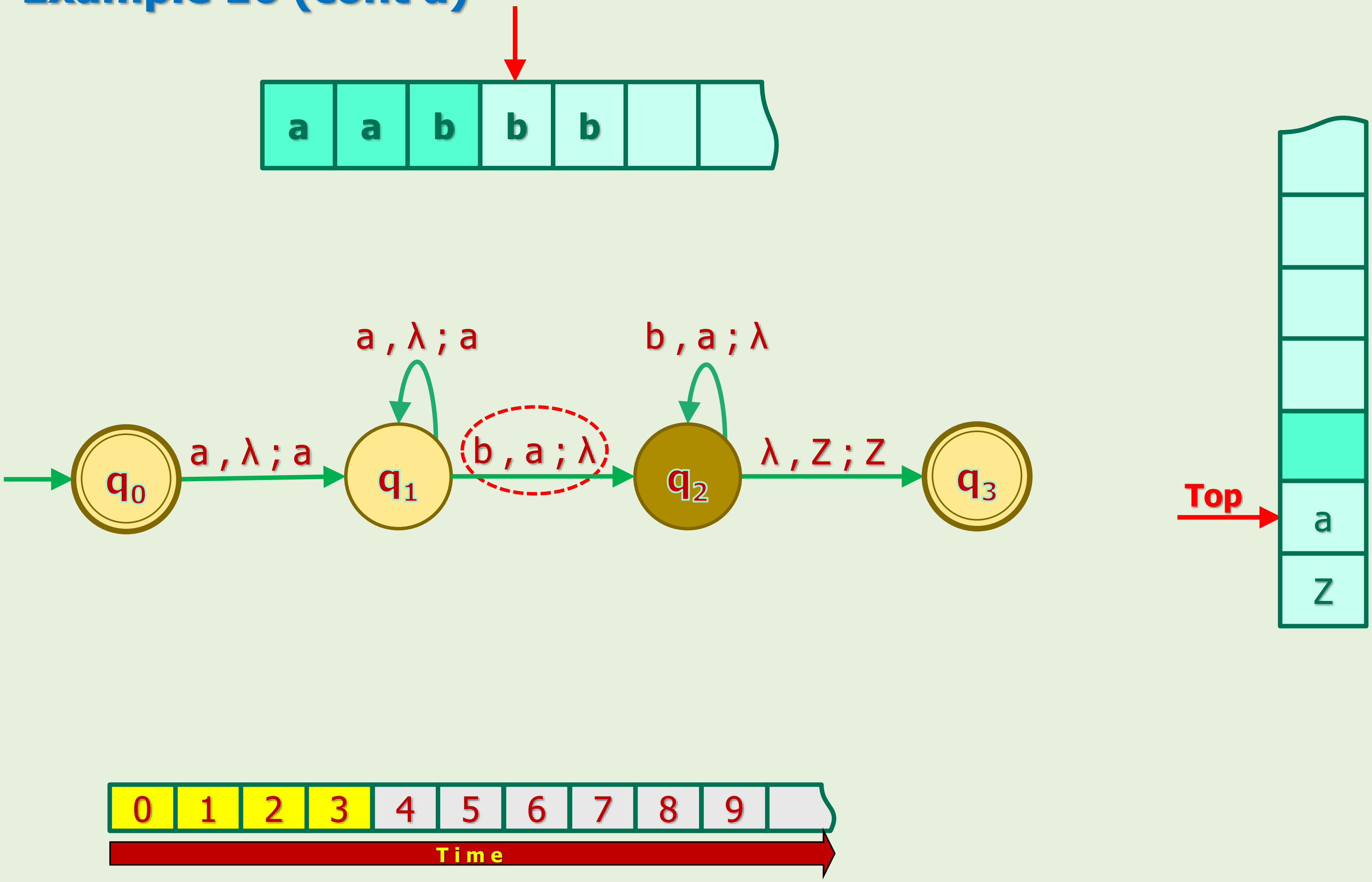
5. PDAs in Action

Example 16 (cont'd)



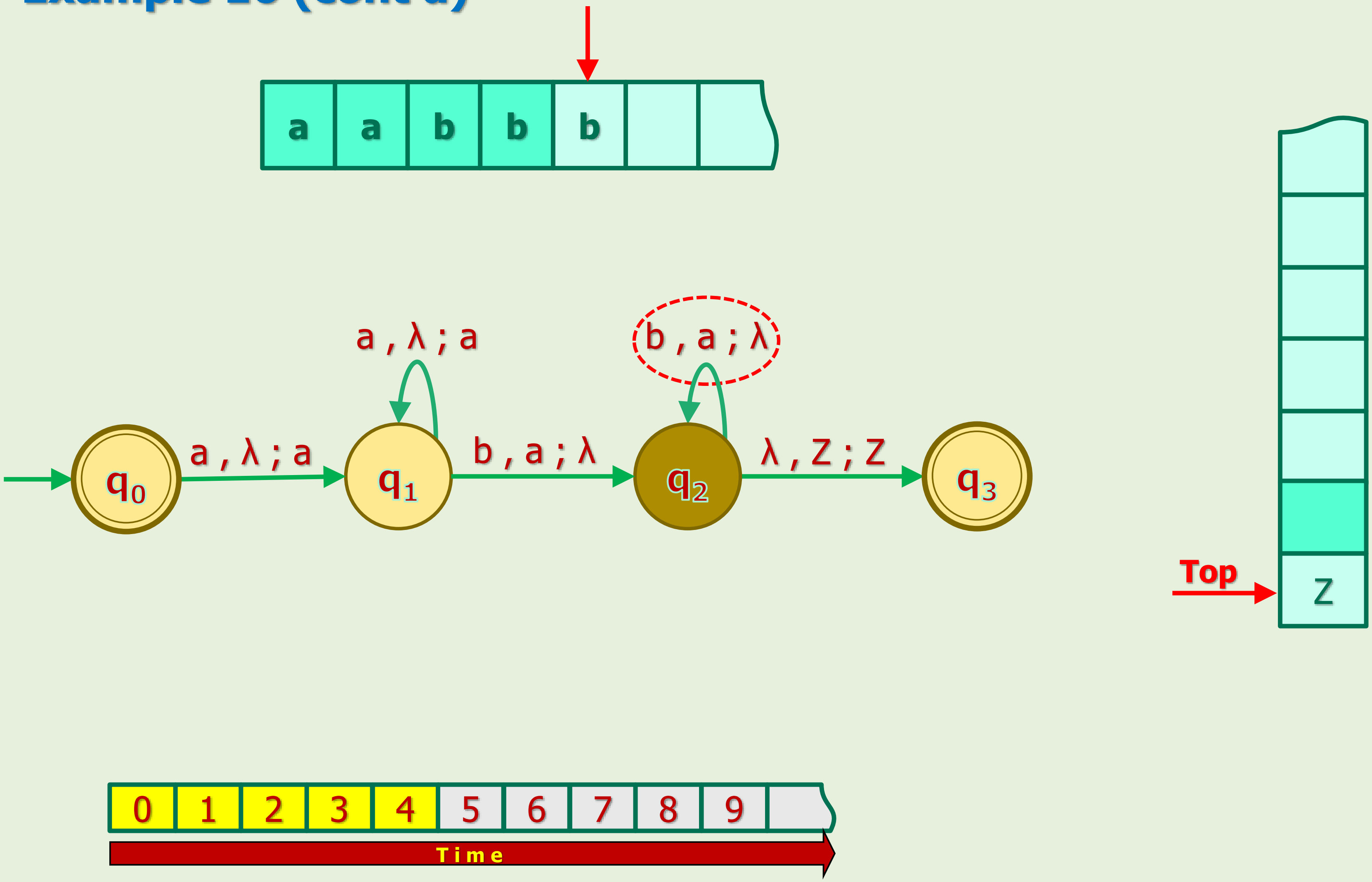
5. PDAs in Action

Example 16 (cont'd)



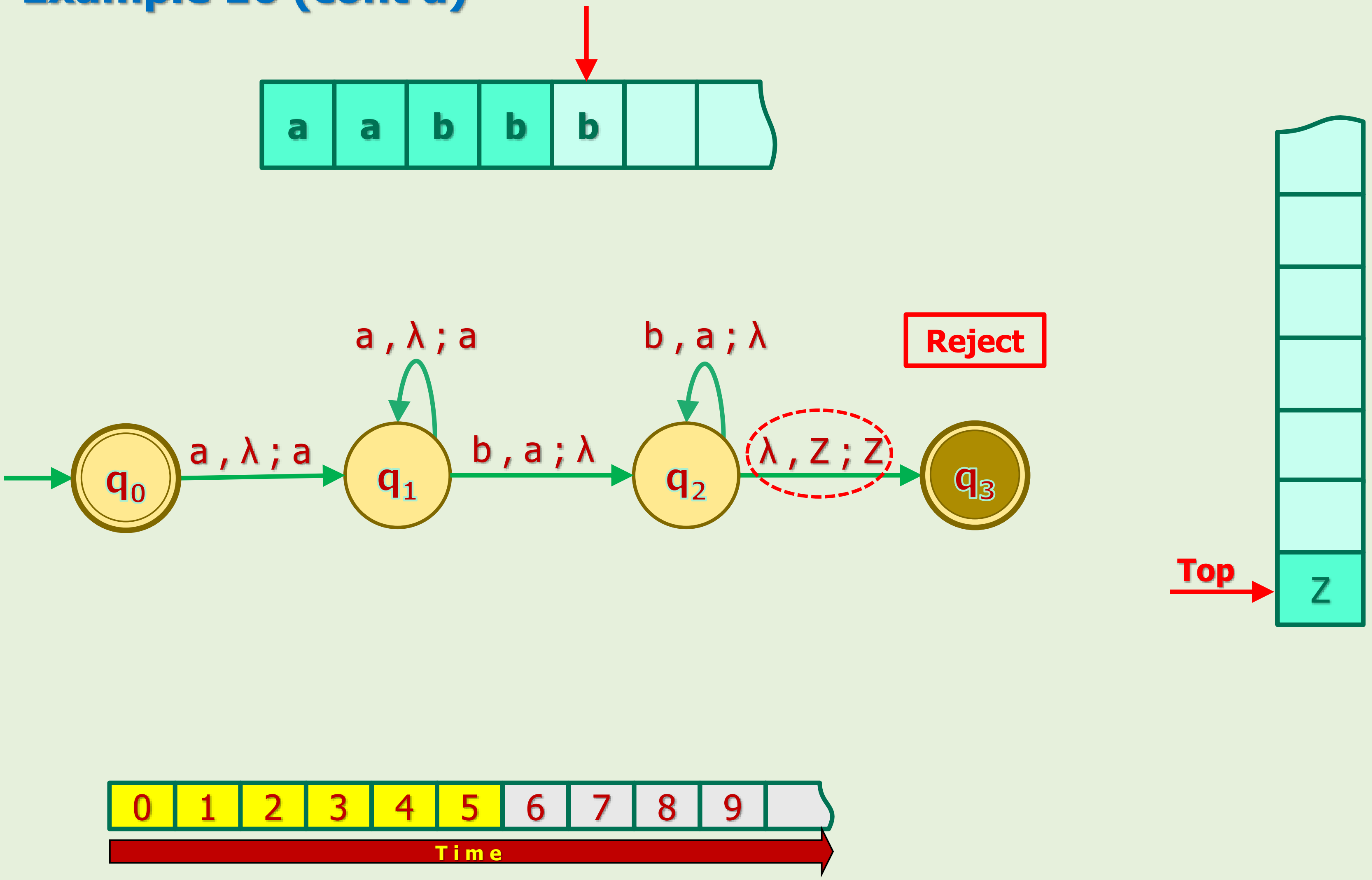
5. PDAs in Action

Example 16 (cont'd)

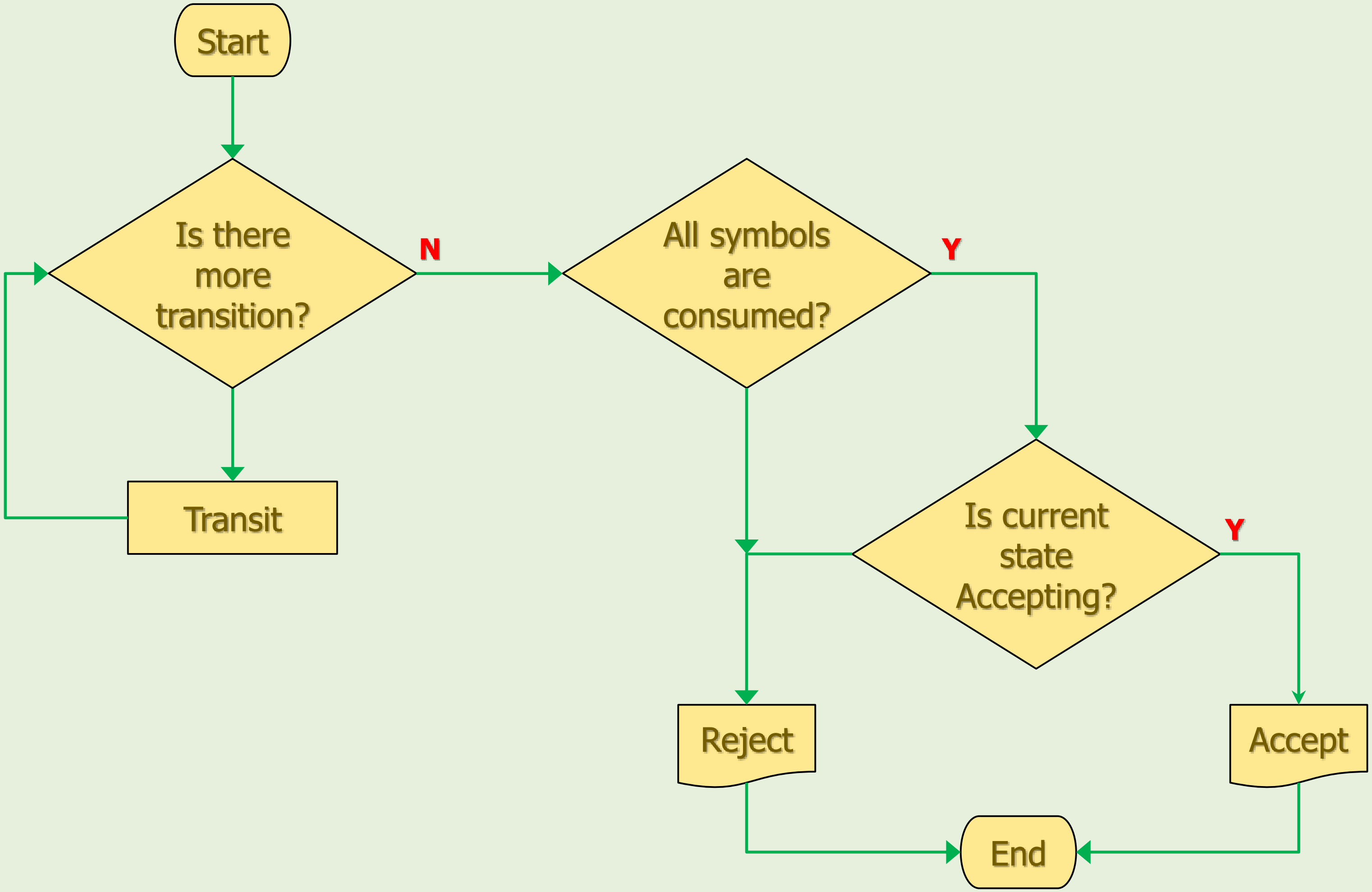


5. PDAs in Action

Example 16 (cont'd)



DPDAs Operation Flowchart



Nondeterministic PDAs (NPDAs)

Nondeterministic PDAs

Determinism:
During any timeframe, there is no more than one transition.

Any violation of this makes a machine nondeterministic.

Recap

- What could be those violations?
  - λ-transition
  - When δ is multifunction
- Let's explain each one in detail!

λ-Transitions

λ-transition in automata theory:
The machine may unconditionally transit.

If we put λ in the condition places, we make a λ-transition.

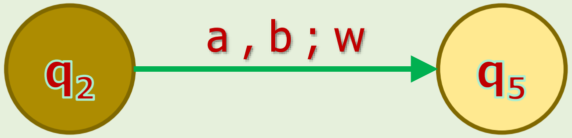
Recap

- This is our knowledge so far:

Automata Class	Transition Condition
DFA/NFA	Input Symbol
NPDA	Input Symbol + Top of stack

NPDAs λ-Transitions

- For example, in the following transition, conditions for transition are:



input symbol = 'a'
AND
top of the stack = 'b'

- So, if we put λ in the conditions places, we make a λ-transition.

NPDAs λ-Transitions

Definition

- For NPDAs, a transition is called λ-transition iff both input and pop parts of the label are λ.

Note

- If the machine initiates a new process starting from q_j, then after replicating the current configuration, it should initialize the stack by pushing w.

NPDAs λ-Transitions

- Note that w is a string and can be λ.

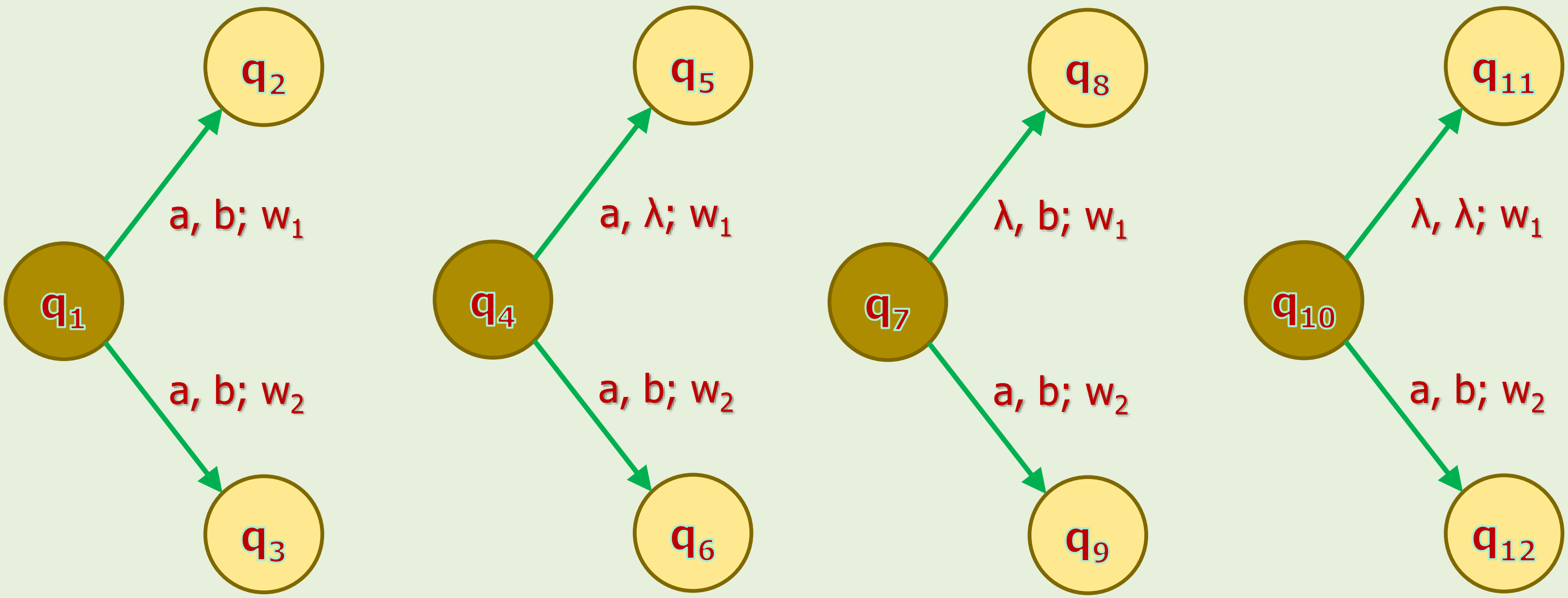
- So, the λ-transition that is usually used in practice is:

- In this case, the machine does not need to do anything if it transits to q_j.

NPDAs: Multifunction Examples

Example 17

- Are the following transitions violations for determinism?

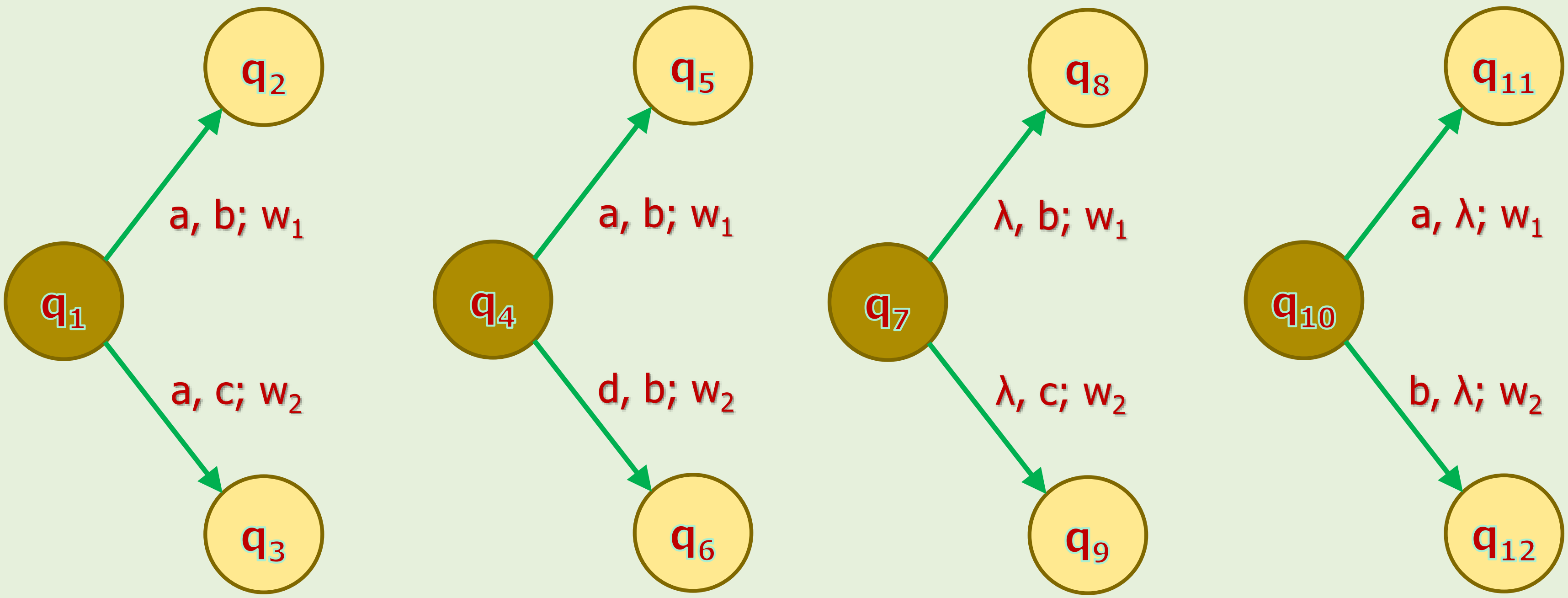


Yes Yes Yes Yes

NPDAs: Multifunction Examples

Example 17 (cont'd)

- Are the following transitions violations for determinism?

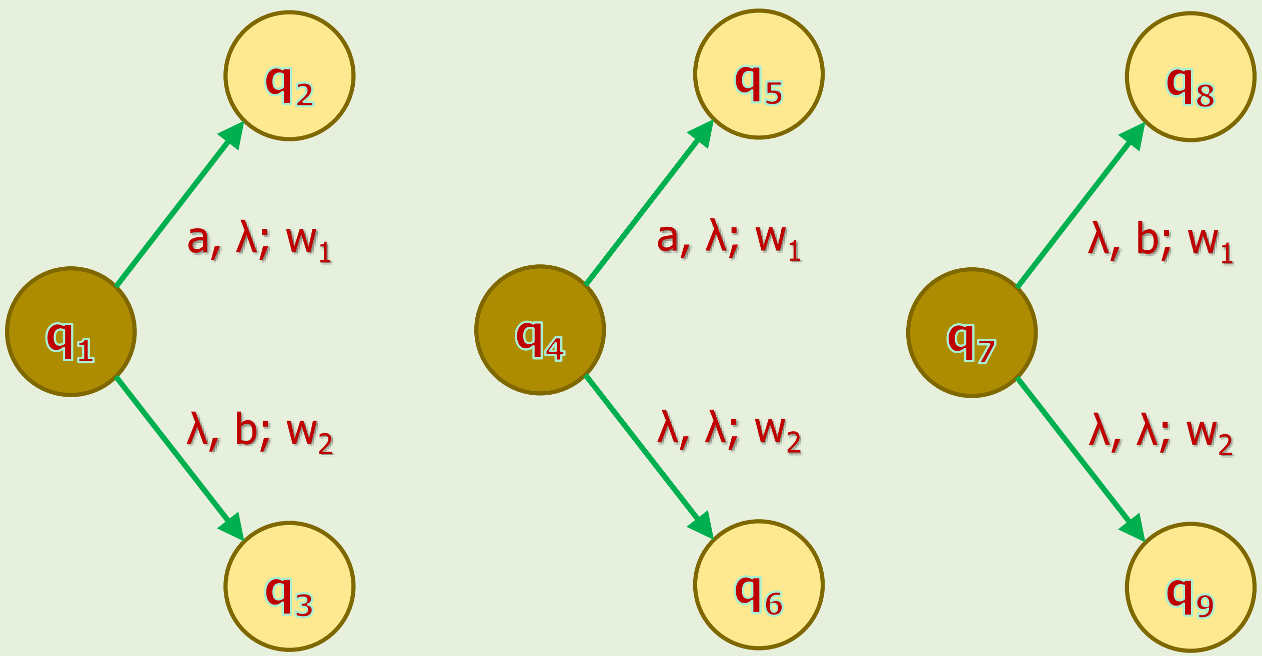


No No No No

NPDAs: Multifunction Examples

Example 17 (cont'd)

- Are the following transitions violations for determinism?



Yes Yes Yes

How NPDAs Behave If They Have Multiple Choices

- We already know that:

All types of nondeterministic machines start
parallel processing when they have multiple choices.

- In other words, for every possible choice,
they create a new process and every process independently continues processing the string.

- The procedure of initiating new processes is exactly the same as NFAs.

How NPDAs Behave If They Have Multiple Choices

Procedure of Initiating New Processes

1. It replicates its entire structure
   (transition graph + input tape + stack)
2. It initializes the new process with the current configuration.
3. The new process independently continues processing
   the rest of the input string.

- The only thing we need to know is:

  What info do we need for the configuration?

NPDAs' Configuration

1. Current state of the transition graph
2. Input string + Position of the read-head
3. The stack and its content

4.4. How NPDAs Accept/Reject Strings

- We already know a process of NPDAs accepts a string iff:

$$(h \wedge c \wedge f) \leftrightarrow a$$

- But how about if we have multiple processes?
- The rule is the same as NFAs':

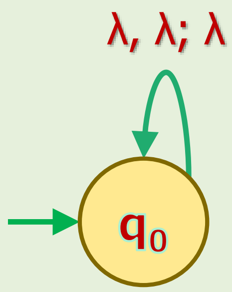
Overall, NPDAs accept a string iff at least one process recognize it.

- And for rejection:

Overall, NPDAs reject a string iff all processes reject it.

An Interesting NPDA!

- Consider the following NPDA:



- What does it do when we input a string?
- Check your answer with JFLAP.

Homework: PDAs Design

- Design a PDA for each of the following languages:
  1. $L = \{a^n b^{2n} : n \geq 0\}$ over $\Sigma = \{a, b\}$
  2. $L = \{a^n b^m c^{n+m} : n \geq 1, m \geq 1\}$ over $\Sigma = \{a, b, c\}$
  3. $L = \{w : n_a(w) > n_b(w)\}$ //number of a's > number of b's
  4. $L = \{ww^R : w \in \{a, b\}^*\}$
  5. $L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$ //number of a's = number of b's
  6. $L = \{1^n+1^m=1^{n+m} : n \geq 1, m \geq 1\}$ over $\Sigma = \{1, +, =\}$ (Unary addition)

6. Definitions

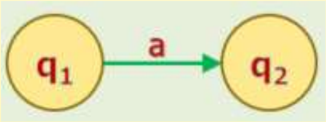
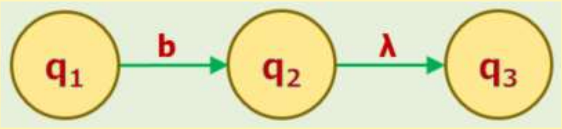
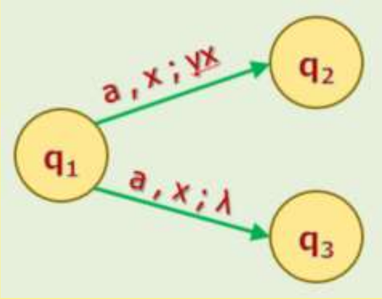
NPDAs Transition Function

- In this section, we are going to formally (mathematically) define the NPDAs.
- What is the important part of this definition?
- As usual, the transition function.

- So, let's take some examples on transition functions.
- And try to figure out what the transition functions look like.

- Note that NPDAs' definition is more general than DPDAs'.
- In other words, we can use NPDAs' definition to describe DPDAs.

Transition Function: DFAs, NFAs, NPDAs

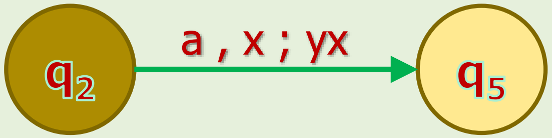
Class	Transition	Sub-Rule Example Transition Function
DFAs		$\delta\ (q_1, a) = q_2$ $\delta : Q \times \Sigma \rightarrow Q$
NFAs		$\delta\ (q_1, b) = \{q_2, q_3\}$ $\delta\ (q_2, a) = \{\ \}$ $\delta : Q \times \Sigma \rightarrow 2^Q$
NPDAs		$\delta\ (q_1\ , a\ , x) = ???$ δ: ???

NPDAs Transition Function: Examples

Example 18

- Write the sub-rule of the following transition.

Solution



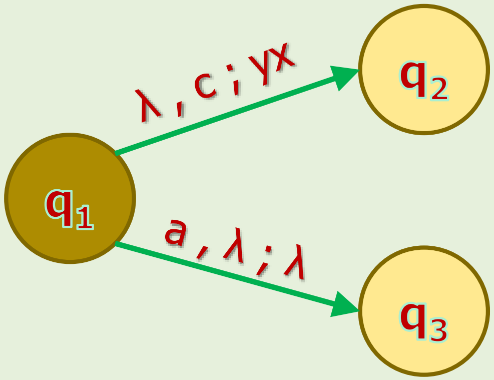
$\delta\,(q_2\,,\,a\,,\,x) = \{(q_5\,,\,yx)\}$

$\delta\,(q_2\,,\,b\,,\,x) = \{\ \}$

Example 19

- Write the sub-rules of the following transition.

Solution



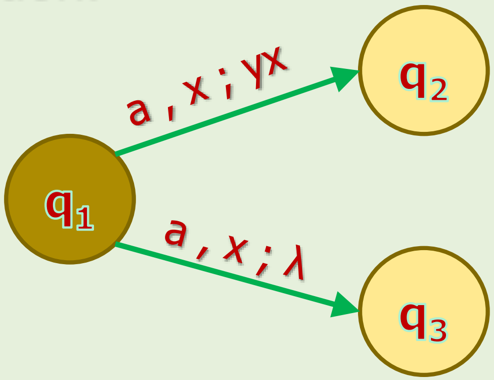
$\delta\,(q_1, \lambda, c) = \{(q_2, yx)\}$

$\delta\,(q_1, a, \lambda) = \{(q_3, \lambda)\}$

NPDAs Transition Function: Examples

Example 20

- Write the sub-rule of the following transition.

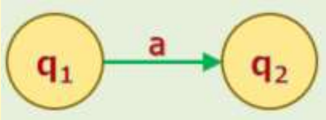
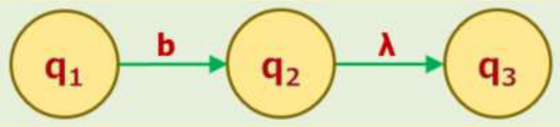
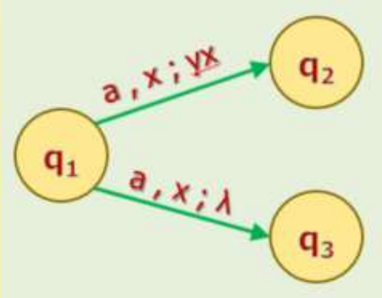


Solution

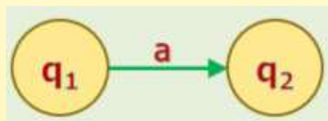
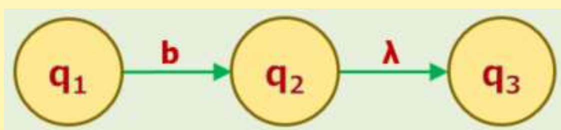
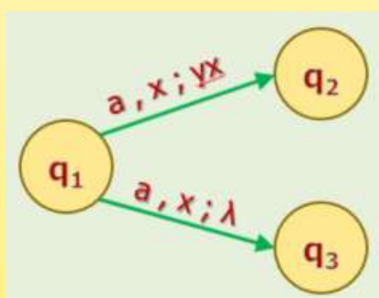
- The condition for transitions for both edges are the same.
- Therefore, we need only one sub-rule.

  $\delta\ (q_1\ , a\ , x) = \{(q_2\ , yx), (q_3\ , \lambda)\}$

Transition Function: DFAs, NFAs, NPDAs

Class	Transition	Sub-Rule Example Transition Function
DFAs		$\delta\ (q_1, a) = q_2$ $\delta : Q \times \Sigma \rightarrow Q$
NFAs		$\delta\ (q_1, b) = \{q_2, q_3\}$ $\delta\ (q_2, a) = \{\ \}$ $\delta : Q \times \Sigma \rightarrow 2^Q$
NPDAs		$\delta\ (q_1\ , a\ , x) = \{(q_2\ , yx), (q_3\ , \lambda)\}$ δ: ???

Transition Function: DFAs, NFAs, NPDAs

Class	Transition	Sub-Rule Example Transition Function
DFAs		$\delta\ (q_1, a) = q_2$ $\delta : Q \times \Sigma \rightarrow Q$
NFAs		$\delta\ (q_1, b) = \{q_2, q_3\}$ $\delta\ (q_2, a) = \{\ \}$ $\delta : Q \times \Sigma \rightarrow 2^Q$
NPDAs		$\delta\ (q_1 , a , x) = \{(q_2 , yx), (q_3 , \lambda)\}$ $\delta: Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \rightarrow 2^{Q \times \Gamma^*}$

6. Formal Definition of NPDAs

- An NPDA M is defined by the septuple (7-tuple):

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

- Where:
  - Q is a finite and nonempty set of states of the transition graph.
  - Σ is a finite and nonempty set of symbols called input alphabet.
  - Γ is a finite and nonempty set of symbols called stack alphabet.
  - δ is called transition function and is defined as:

$$\delta: Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \rightarrow 2^{Q \times \Gamma^*}$$

δ is total function.

  - $q_0 \in Q$ is the initial state of the transition graph.
  - $Z \in \Gamma$ is a special symbol called stack start symbol.
  - $F \subseteq Q$ is the set of accepting states of the transition graph.

6. Formal Definition of NPDAs

Set of States | Stack Alphabet | Initial State | Set of Final States

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

Input Alphabet | Transition Function | Stack Start Symbol

7. NPDAs vs NFAs

Can NPDAs Do Whatever NFAs Can Do?

- Let's assume that we've constructed an NFA for an arbitrary language L.
- Can we always construct an NPDA for L?

- Yes! Why?
- We should prove that we can always convert an NFA's definition to an NPDA's definition.

- Let's show this through an example first.

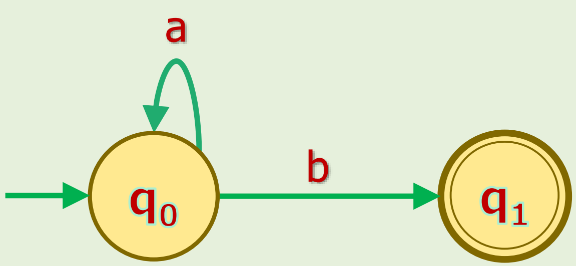
Can NPDAs Do Whatever NFAs Can Do?

Example 21

- Convert the following NFA to an NPDA.

$$\delta: \begin{cases} \delta(q_0, a) = \{q_0\} \\ \delta(q_0, b) = \{q_1\} \end{cases}$$

?



NFA

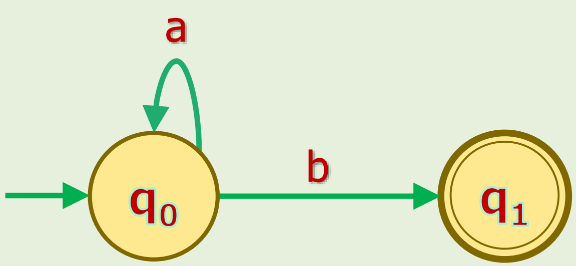
NPDA

Can NPDAs Do Whatever NFAs Can Do?

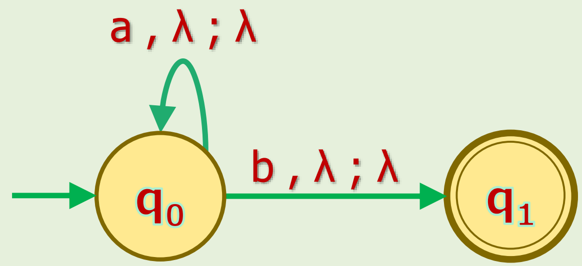
Example 21 (cont'd)

- Convert the following NFA to an NPDA.

$$\delta: \begin{cases} \delta(q_0, a) = \{q_0\} \\ \delta(q_0, b) = \{q_1\} \end{cases} \quad \Rightarrow \quad \delta: \begin{cases} \delta(q_0, a, \lambda) = \{(q_0, \lambda)\} \\ \delta(q_0, b, \lambda) = \{(q_1, \lambda)\} \end{cases}$$



NFA



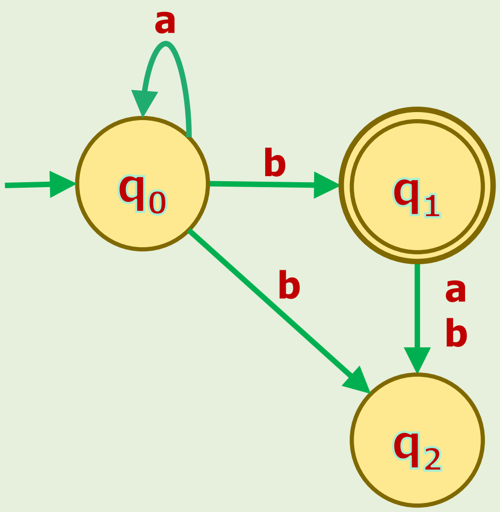
NPDA

Homework

- Convert the following NFA to an NPDA.

$$\delta: \begin{cases} \delta(q_0, a) = \{q_0\} \\ \delta(q_0, b) = \{q_1, q_2\} \\ \delta(q_1, a) = \{q_2\} \\ \delta(q_1, b) = \{q_2\} \end{cases}$$

?



NFA

NPDA

NFAs Can be Converted to NPDAs

	NFA	NPDA
States	$Q = \{q_0, q_1, q_2\}$	$Q = \{q_0, q_1, q_2\}$
Alphabet	$\Sigma = \{a, b\}$	$\Sigma = \{a, b\}$
Stack alphabet	N/A	$\Gamma = \{Z\}$
Sub-rule	$\delta\ (q_i, a) = \{q_j\}$	$\delta\ (q_i\ , x\ , \lambda) = \{(q_j\ , \lambda)\}$
Initial state	q_0	q_0
Stack start symbol	N/A	Z
Final states	$F = \{q_1\}$	$F = \{q_1\}$

Can NPDAs Do Whatever NFAs Can Do?

- As the previous example showed, there is a simple algorithm to convert an NFA to an NPDA.

Algorithm: Converting NFAs' Formal Definition to NPDAs'

- Change all NFAs' sub-rules to NPDAs format by adding λ in the pop and push parts. i.e.:

$$\delta\,(q_i\,,\,x) = \{q_j\,,\,q_{j+1}\,,\,\dots\,,\,q_{j+n}\}$$

changes to

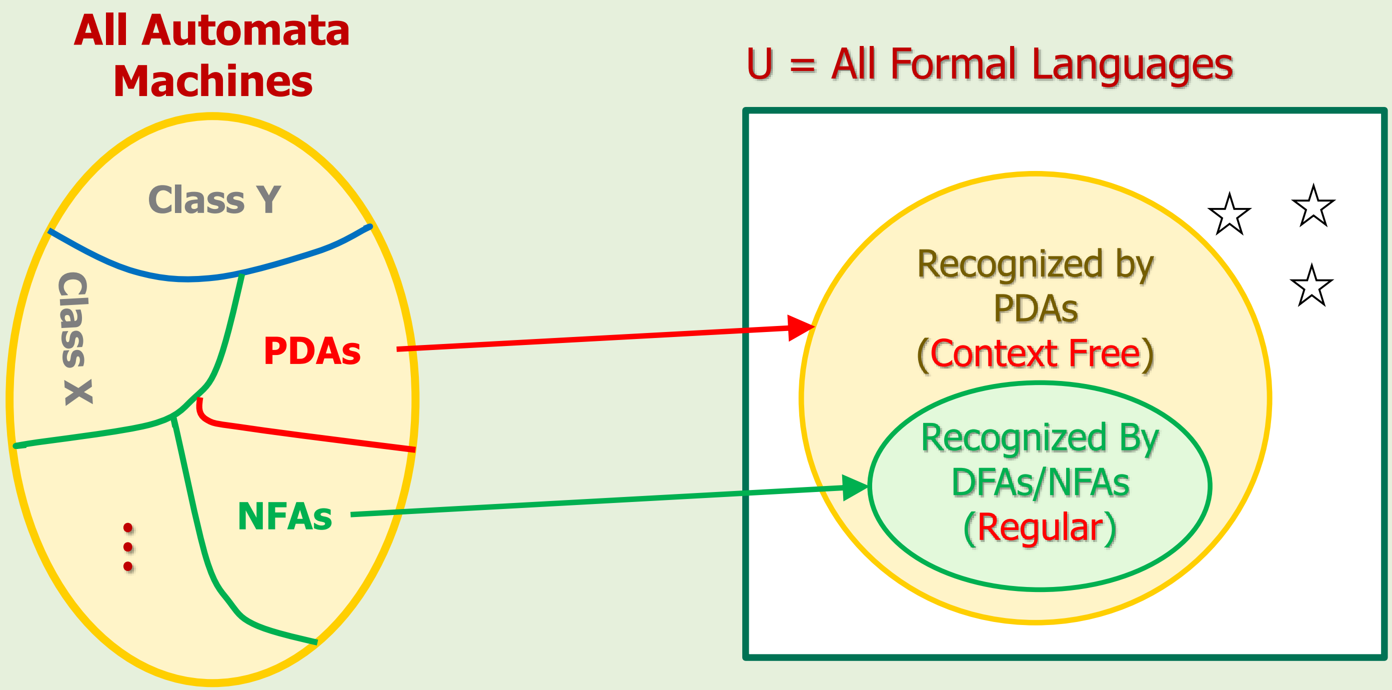
$$\delta\,(q_i\,,\,x\,,\,\lambda) = \{(q_j\,,\,\lambda),\,(q_{j+1}\,,\,\lambda),\,\dots\,,\,(q_{j+n}\,,\,\lambda)\}$$

- Set $\Gamma = \{Z\}$.
- Set the stack start symbol as Z.
- The rest of the definitions, (i.e. Q, Σ, q_0, F) are the same.

Can NFAs Do Whatever NPDAs Can Do?

- Let's assume that we've constructed an NPDA for an arbitrary language L.
- Can we always construct an NFA for L?
- No! Why?
- There is no way to simulate the stack operations by NFAs!
- It means, there is no way to simulate a read/write memory like stack with a read-only memory.
- Moreover, we know at least the following languages for which we can construct NPDAs but it is impossible to construct NFAs.
  – $L = \{a^n b^n : n \geq 0\}$
  – $L = \{ww^R : w \in \Sigma^*\}$
- Let's summarize our knowledge and figure out what would be the next step.

Machines and Languages Association



- The set of languages that NFAs recognize is a proper subset of the set of languages that PDAs recognize.
  - We'll explain later what the "context free" meaning is.
- So, PDAs are more powerful than NFAs.

PDAs for More Complex Languages

- Design a PDA for the following language:
- $L = \{a^n b^n c^n : n \geq 1\}$ over $\Sigma = \{a, b, c\}$

Solution

- Struggling?!
- After some struggling, you realize that you cannot construct any NPDA for these language! Why?

- You'd get the same problem if you try to construct a PDA for the following language:
- $L = \{ww : w \in \Sigma^*\}$ over $\Sigma = \{a, b\}$

PDAs for More Complex Languages

- The reason is ...

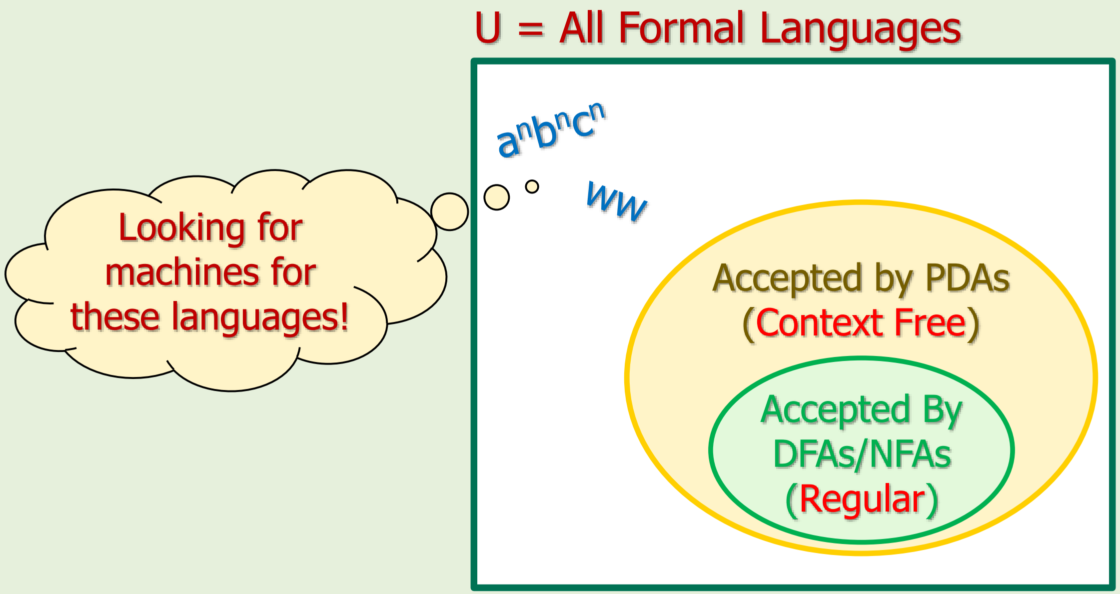
  ... we need more control on the memory.

  ... stack is not so flexible in storing and manipulating data.

  ... if you access the older data, you'd lose newer data.

8. What is the Next Step?

- The next step is to define a new class of machines that recognizes all or part of the remaining non-regular languages.



Project (Optional)

- Design a new class of machines like PDAs but use "Queue" for the memory.
- Pick a name for your machine.
- Discuss what kind of languages it can recognize.
- Compare its power with PDAs'.

References

1. Linz, Peter, "An Introduction to Formal Languages and Automata, 5th ed.," Jones & Bartlett Learning, LLC, Canada, 2012
2. Kenneth H. Rosen, "Discrete Mathematics and Its Applications, 7th ed.," McGraw Hill, New York, United States, 2012
3. Michael Sipser, "Introduction to the Theory of Computation, 3rd ed.," CENGAGE Learning, United States, 2013
   ISBN-13: 978-1133187790