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 15/31

CS 154

Formal Languages and Computability

Spring 2018

Agenda of Day 15

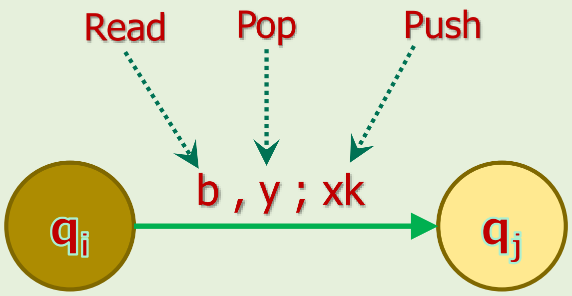
- Summary of Lecture 13
- Quiz 5
- Lecture 14: Teaching ...
  - Pushdown Automata (part 2)

Summary of Lecture 13: We learned ...

NPDAs

- NFAs are strong enough to recognize regular languages.
- To accept non-regular languages, we need more powerful machines.
- We noticed that we needed writable memory.
- We added writable memory in the stack format to NFAs.
- We introduced pushdown automaton to accept all or at least some of those languages.

- We talked about the structure of NPDA's, transitions, ...



- 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 ...

NPDAs

- NPDAs halt when ...
  - ... the conditions for the next transition are not satisfied.

- The conditions for a string be 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.

Any question?

NAME	Alan M. Turing		
SUBJECT	CS 154	TEST NO.	5
DATE	03/15/2018	PERIOD	1 , 2 , 3

TEST RECORD	
PART 1	123
PART 2	
TOTAL	

Your list # goes here!

Quiz 5

Use Scantron

Template for Constructing a New Class of Automata

Repeated

- To construct a new class of automata, we need to respond the following questions:

1. Why do we need a new class of machines? (Justification)
2. Name of the new class
3. Building blocks of the new class
4. How they work
   - 4.1. What is the starting configuration?
   - 4.2. What would happen during a timeframe?
   - 4.3. When would the machines halts?
   - 4.4. How would a string be Accepted/Rejected?
5. The automata in action
6. Formal definition
7. Their power: this class versus previous class
8. What would be the next possible class?

NPDAs in Action

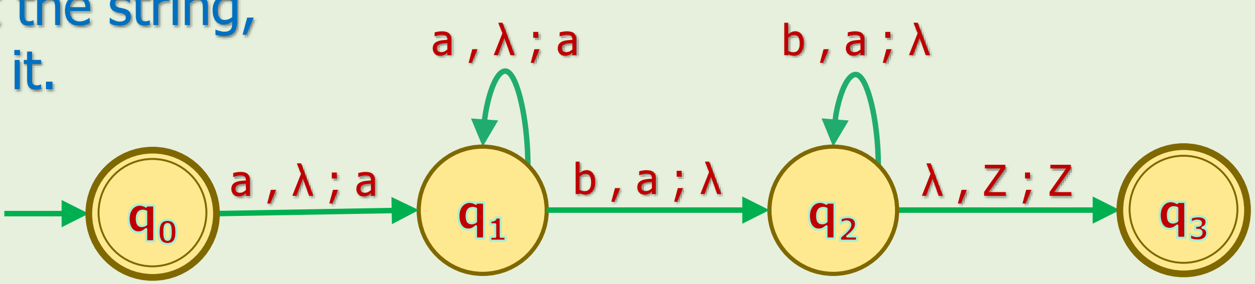
NPDAs in Action

Example 15

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

Solution

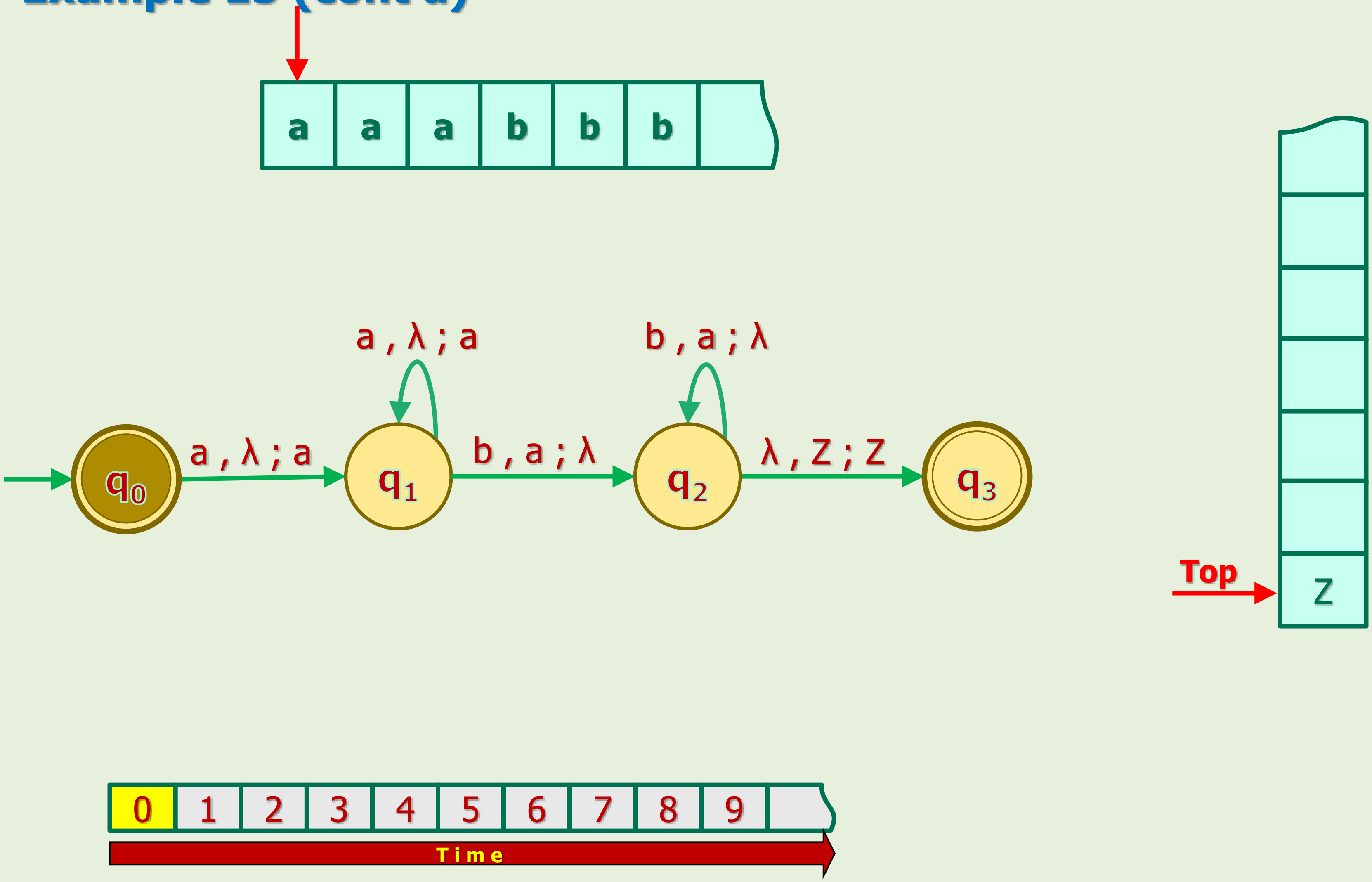
- **Strategy**: read a's and push them in the stack regardless of top of the stack.
- When the first b is sensed, start popping a's to match them with b's.
- Continue popping a's until you are out of b.
- If end of stack is reached, means the number of a's and b's are equal, so, accept the string, otherwise,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



- Let's trace this solution.

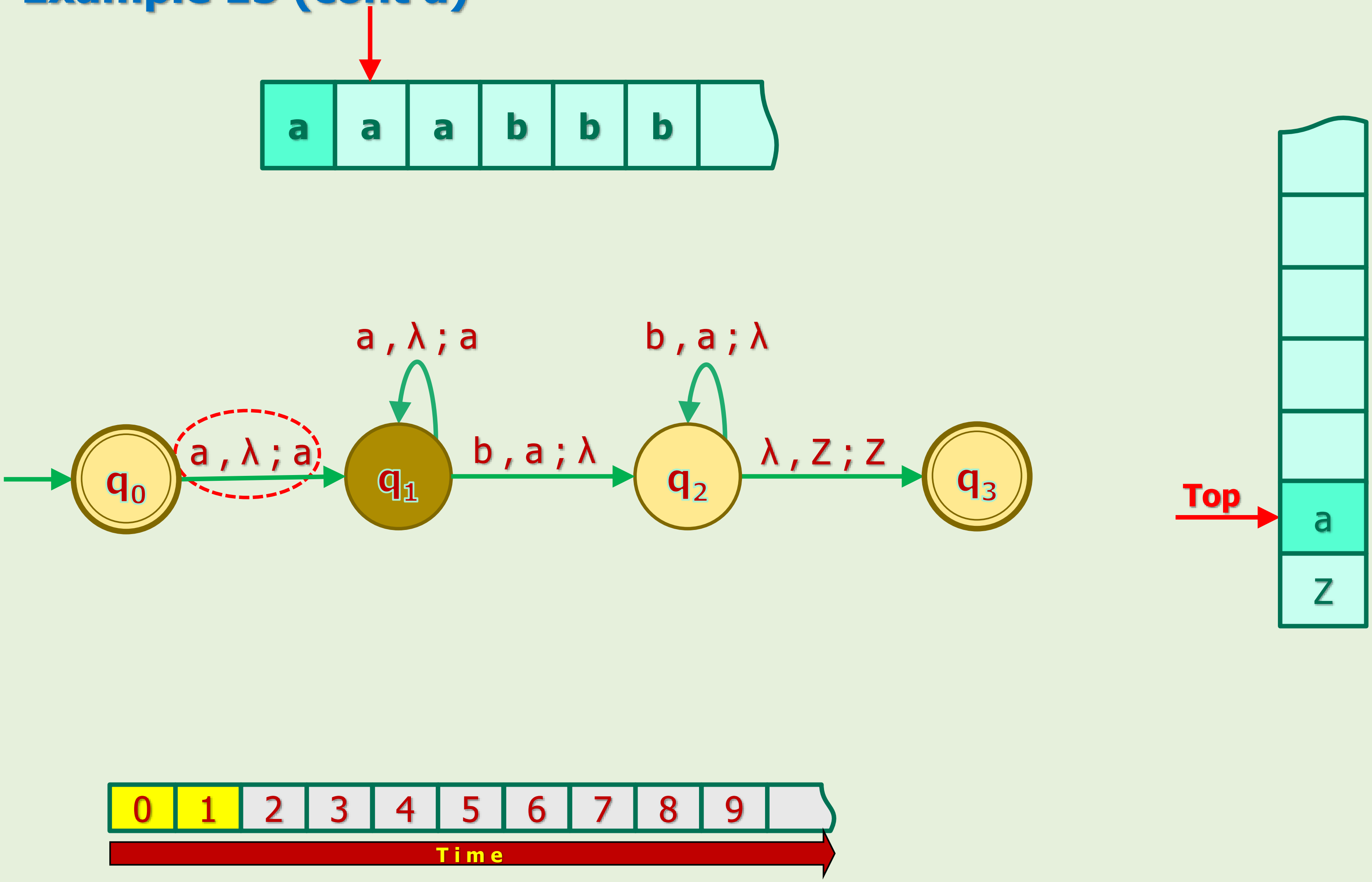
NPDAs in Action

Example 15 (cont'd)



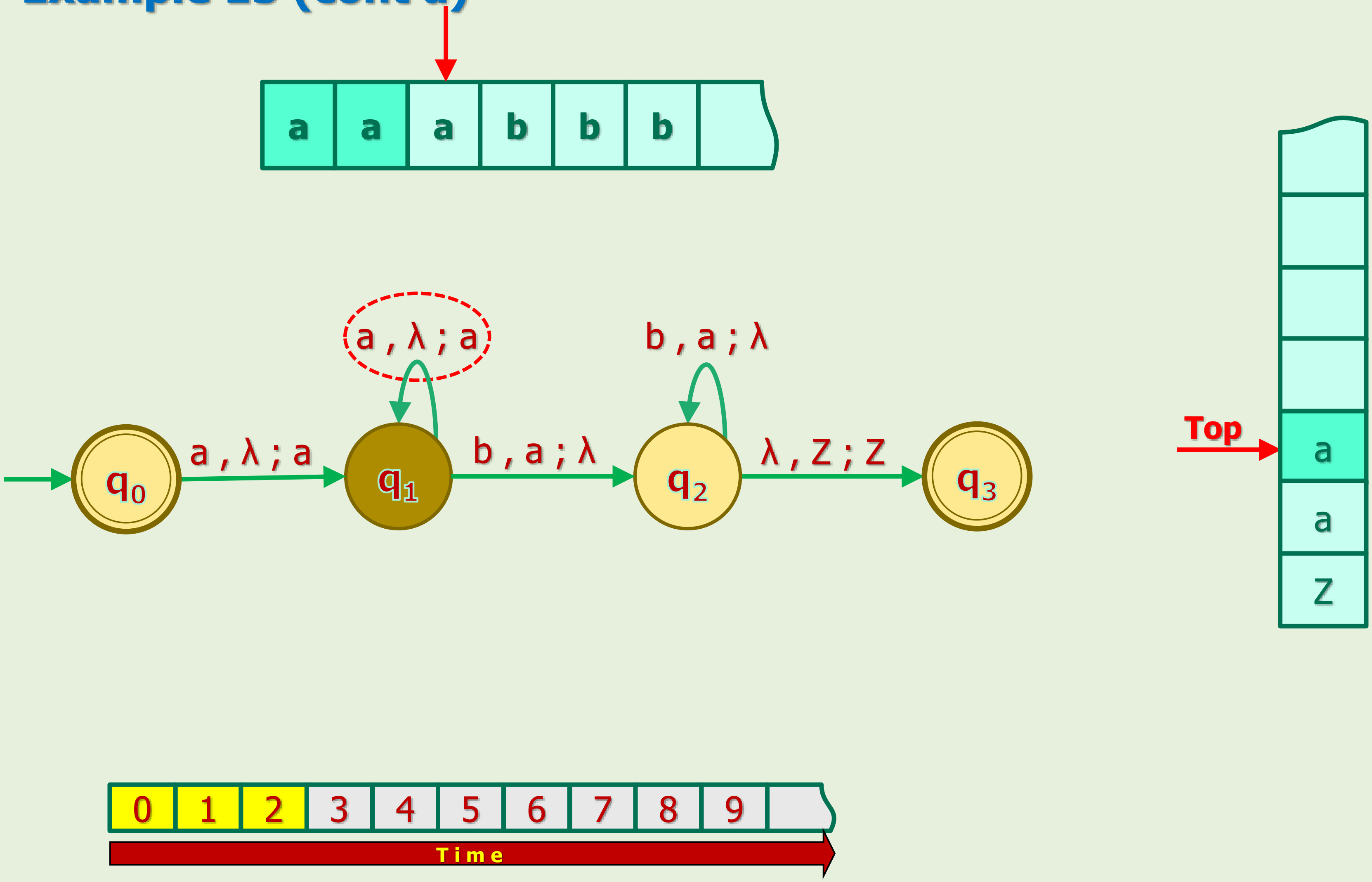
NPDAs in Action

Example 15 (cont'd)



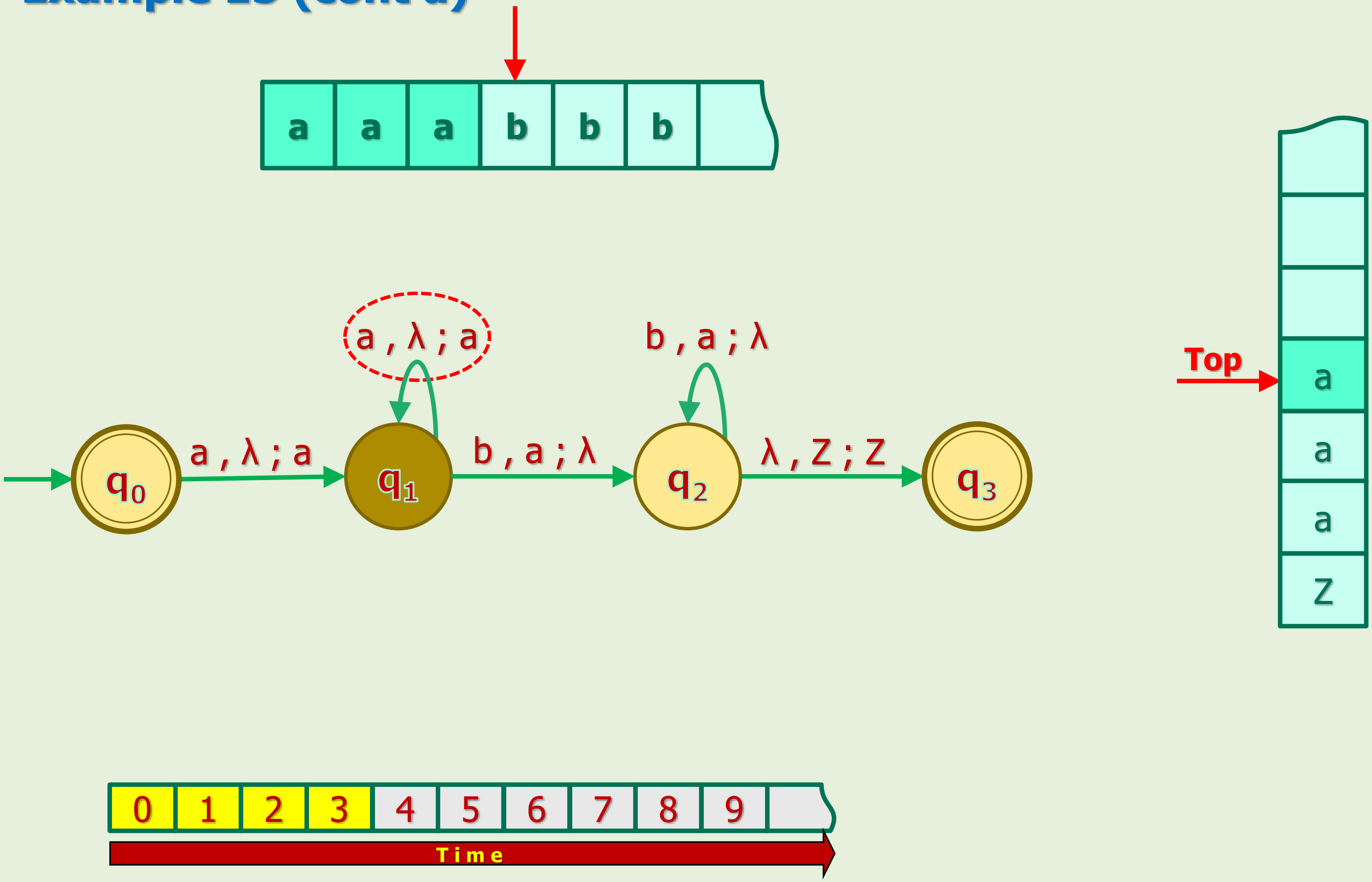
NPDAs in Action

Example 15 (cont'd)



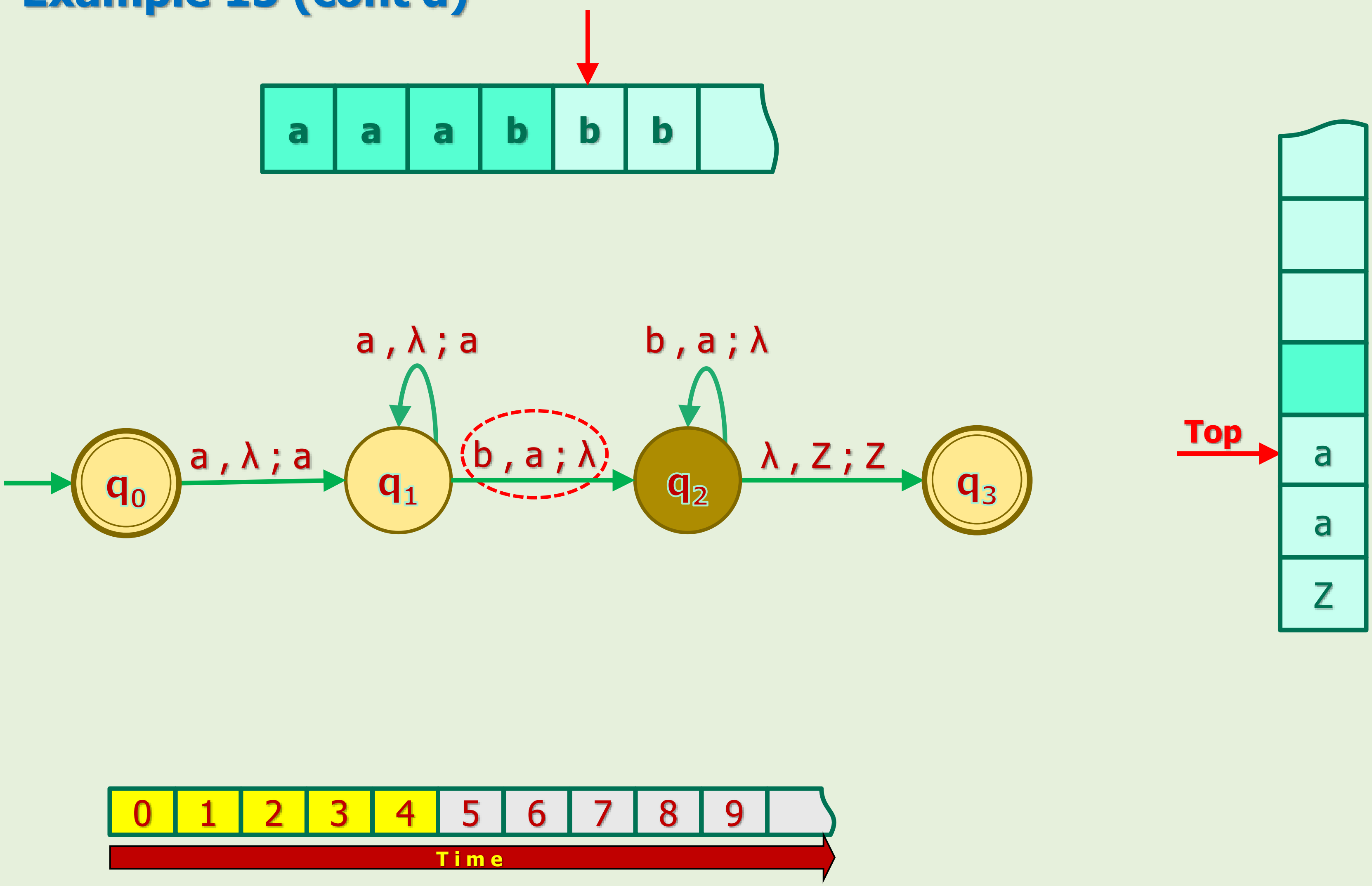
NPDAs in Action

Example 15 (cont'd)



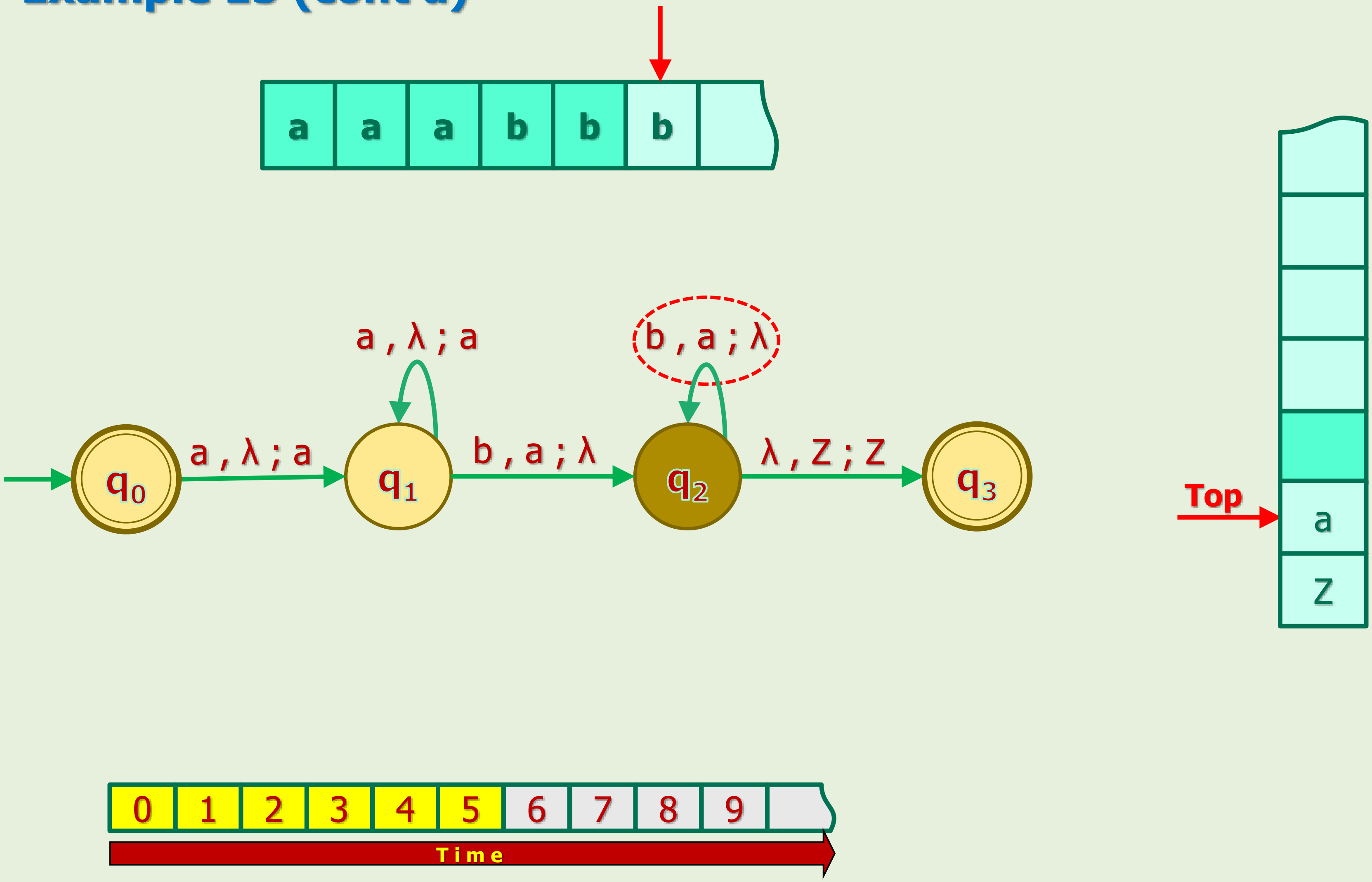
NPDAs in Action

Example 15 (cont'd)



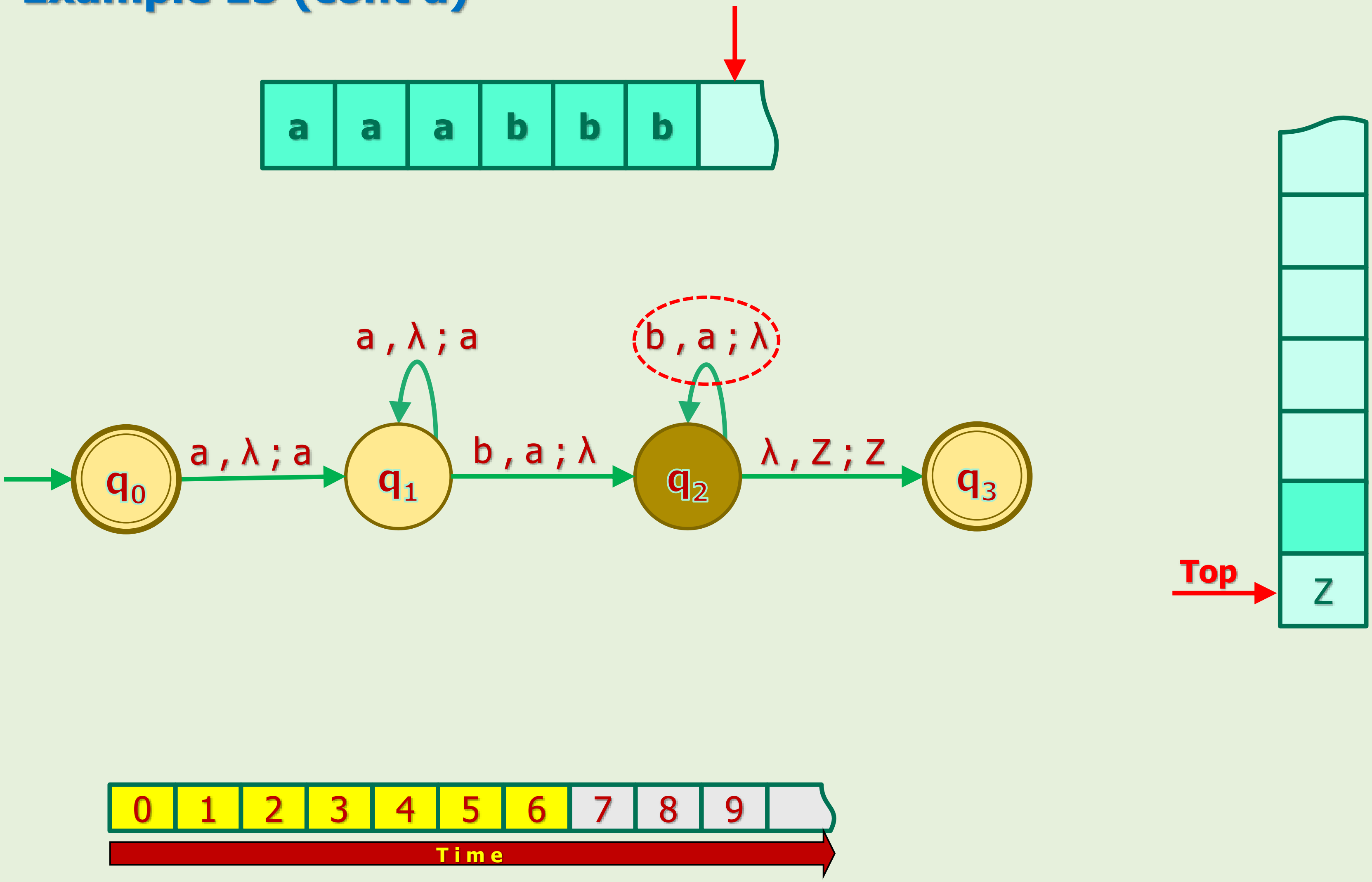
NPDAs in Action

Example 15 (cont'd)



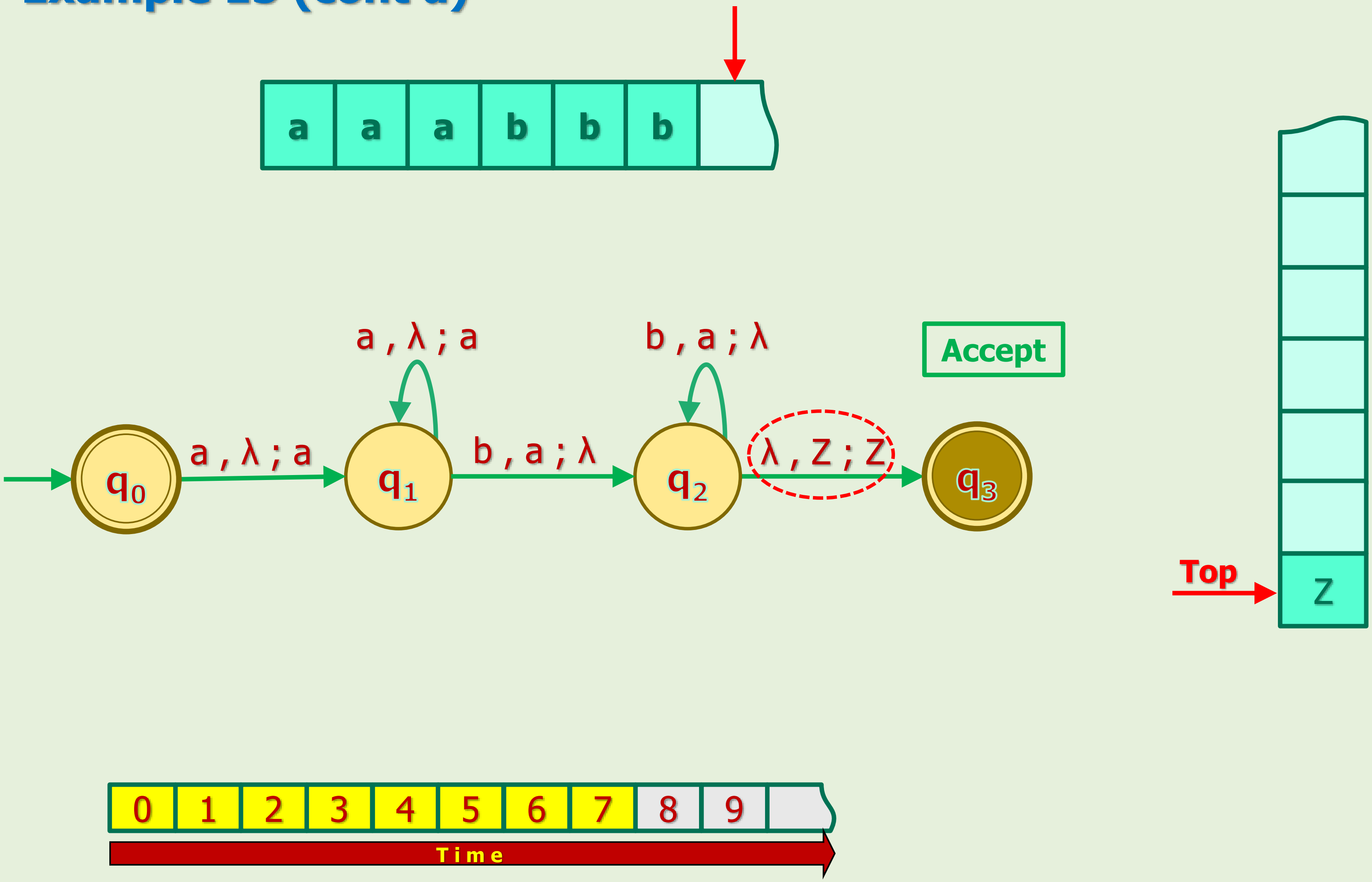
NPDAs in Action

Example 15 (cont'd)



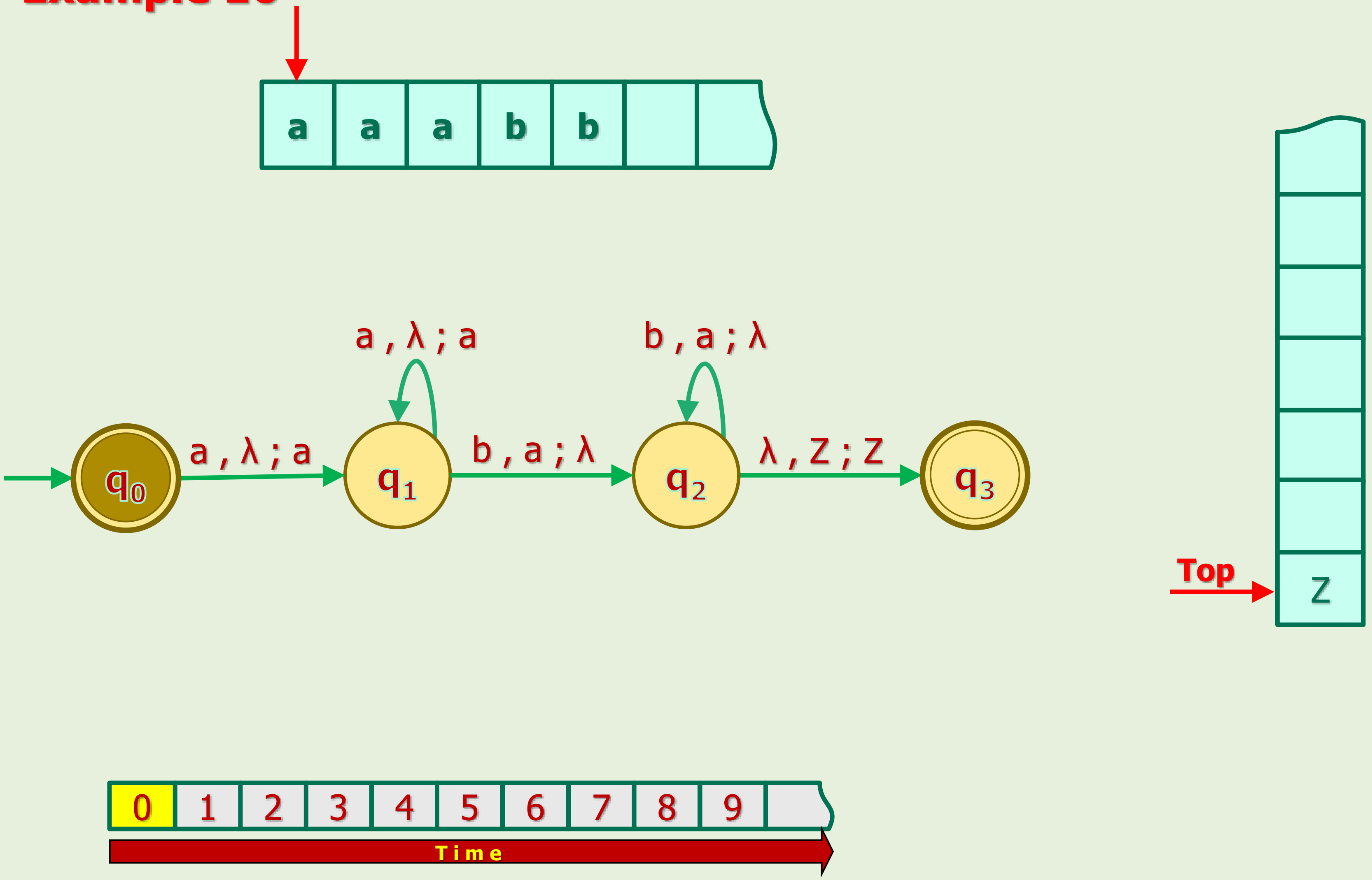
NPDAs in Action

Example 15 (cont'd)



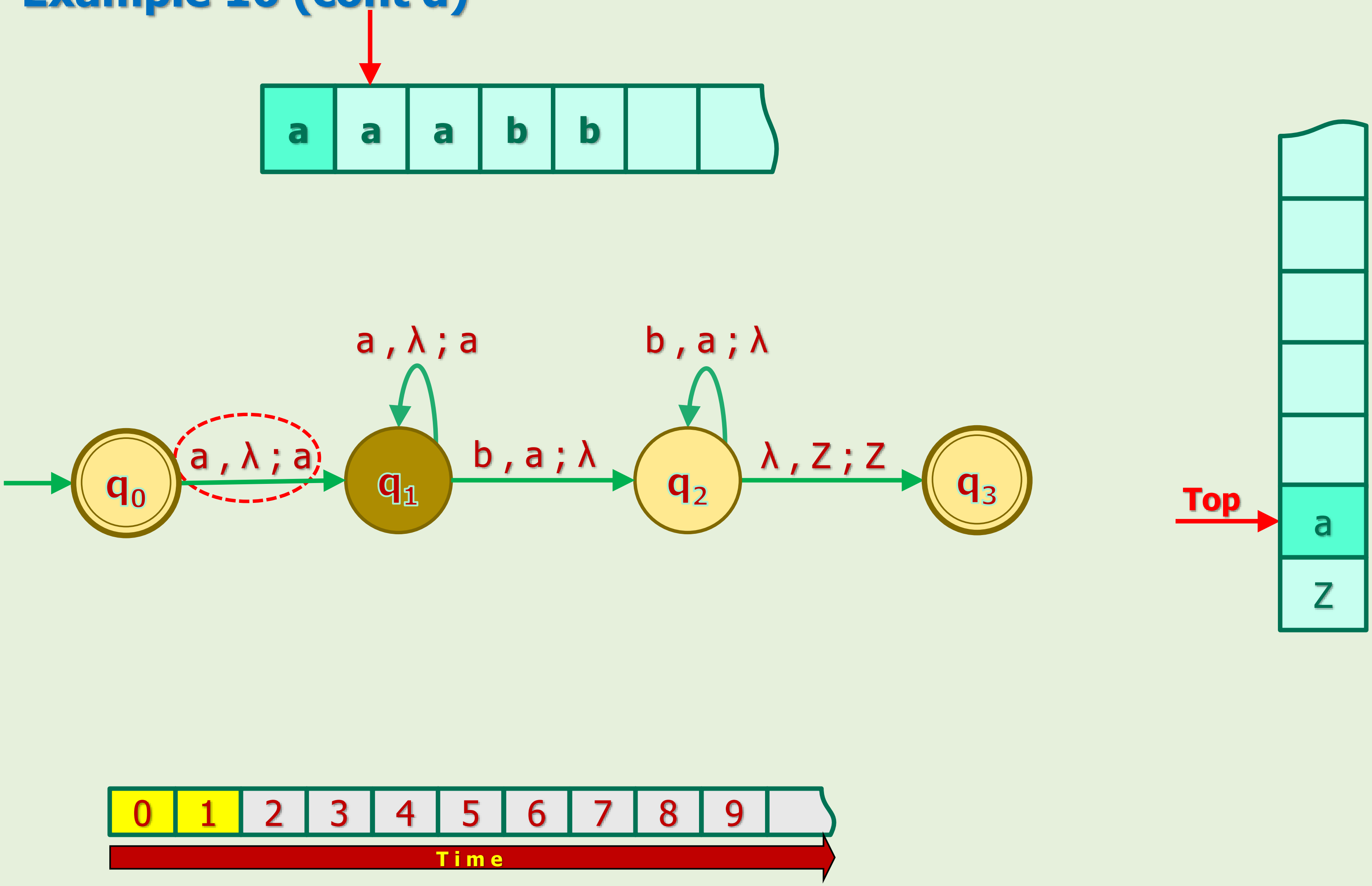
NPDAs in Action

Example 16



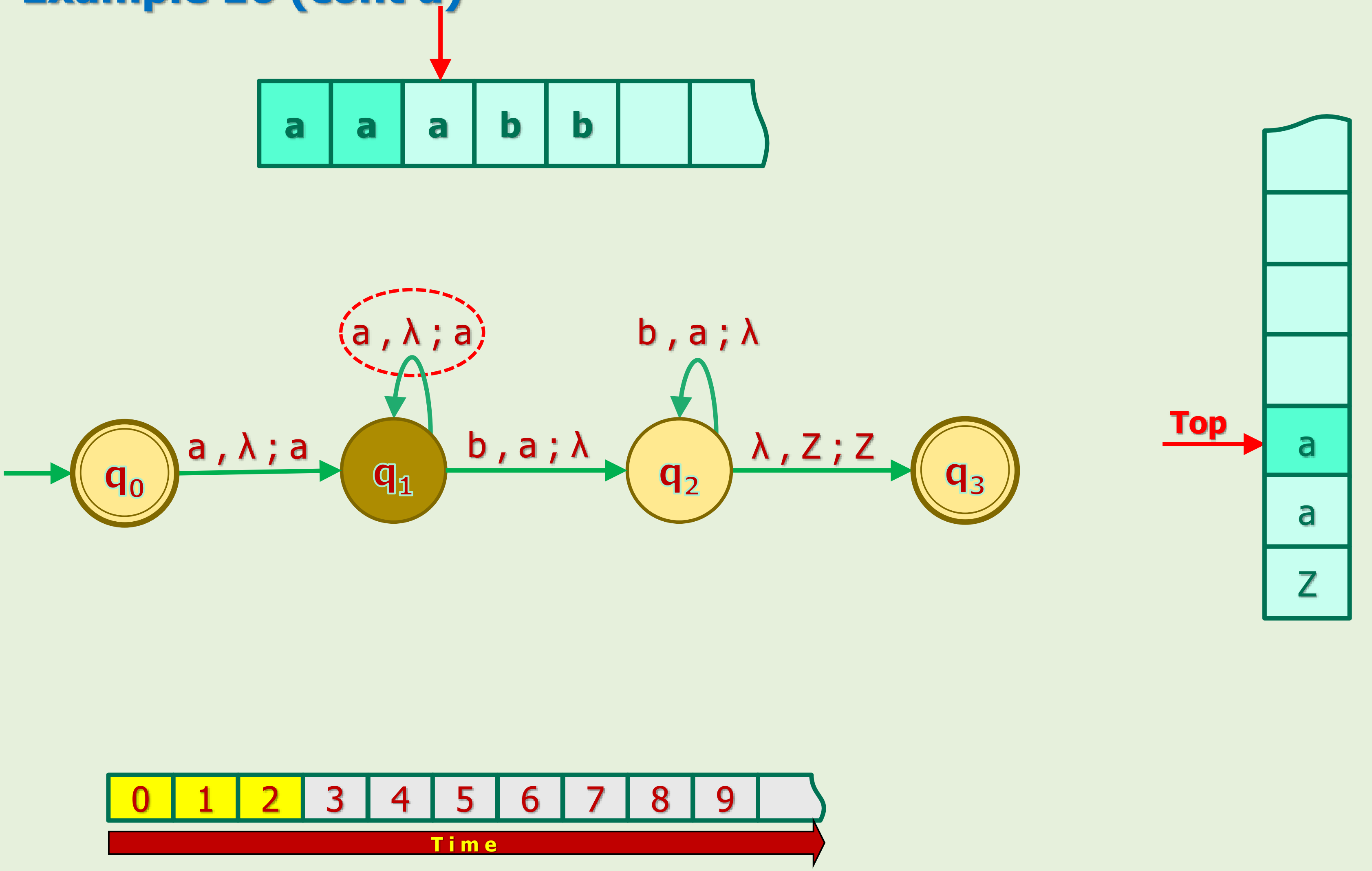
NPDAs in Action

Example 16 (cont'd)



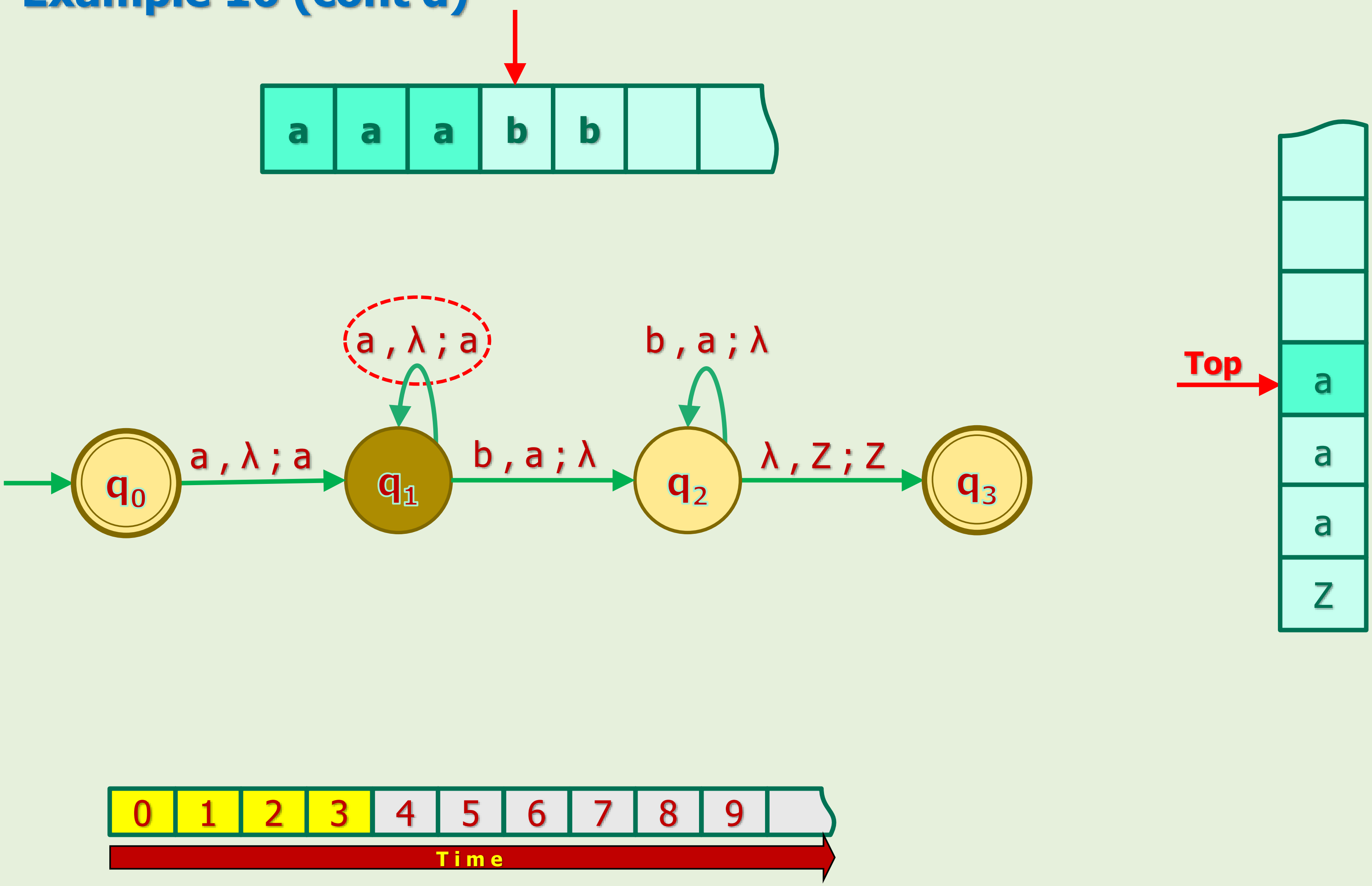
NPDAs in Action

Example 16 (cont'd)



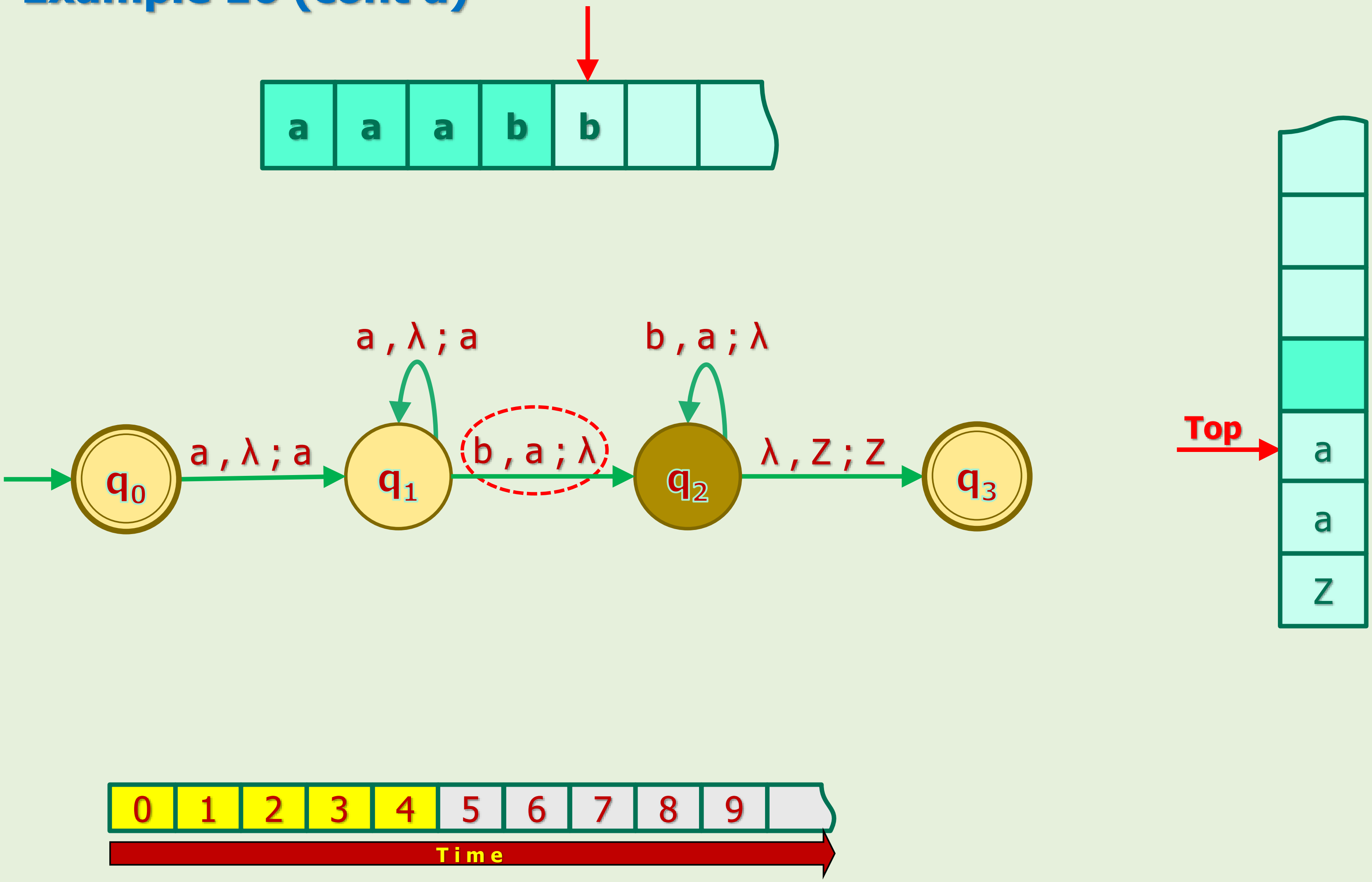
NPDAs in Action

Example 16 (cont'd)



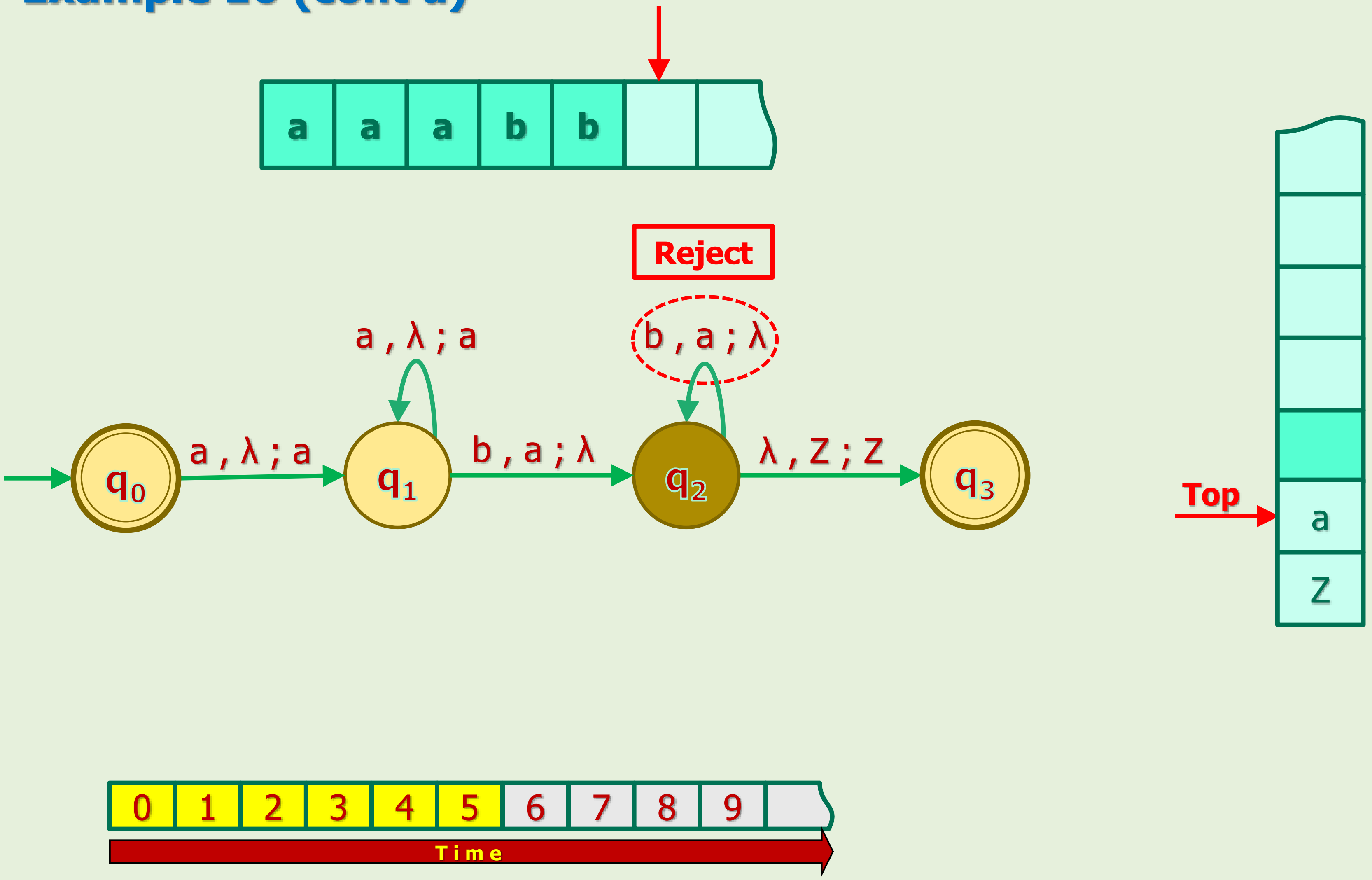
NPDAs in Action

Example 16 (cont'd)



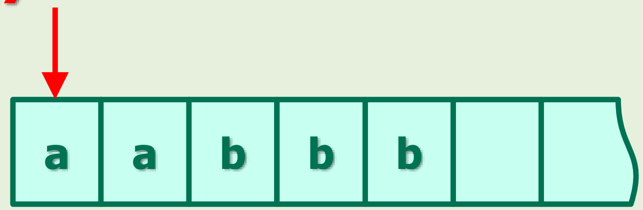
NPDAs in Action

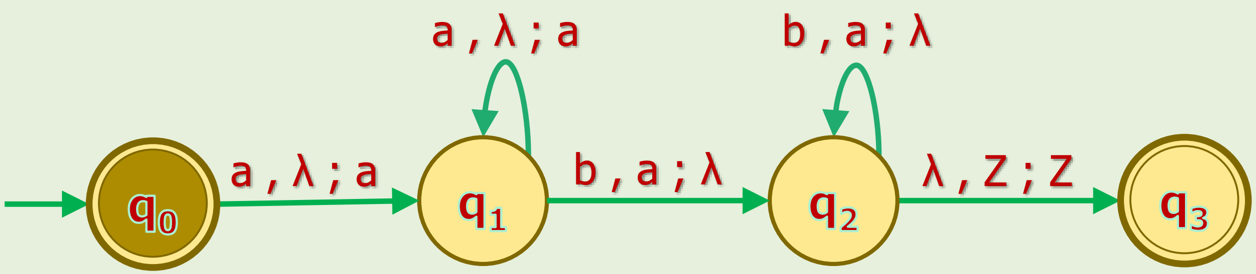
Example 16 (cont'd)

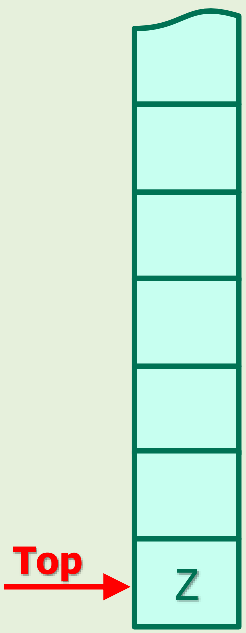


NPDAs in Action

Example 17

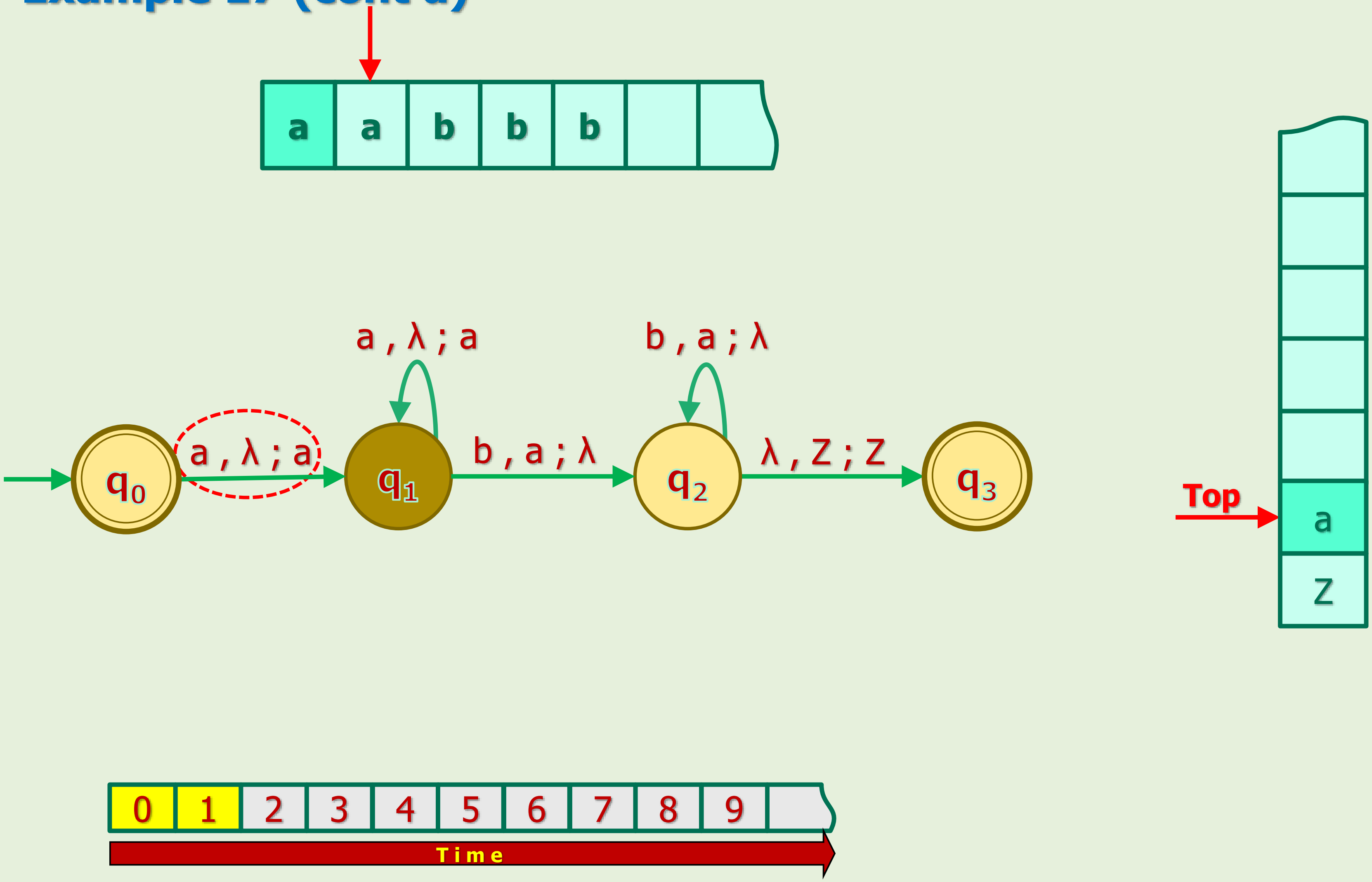






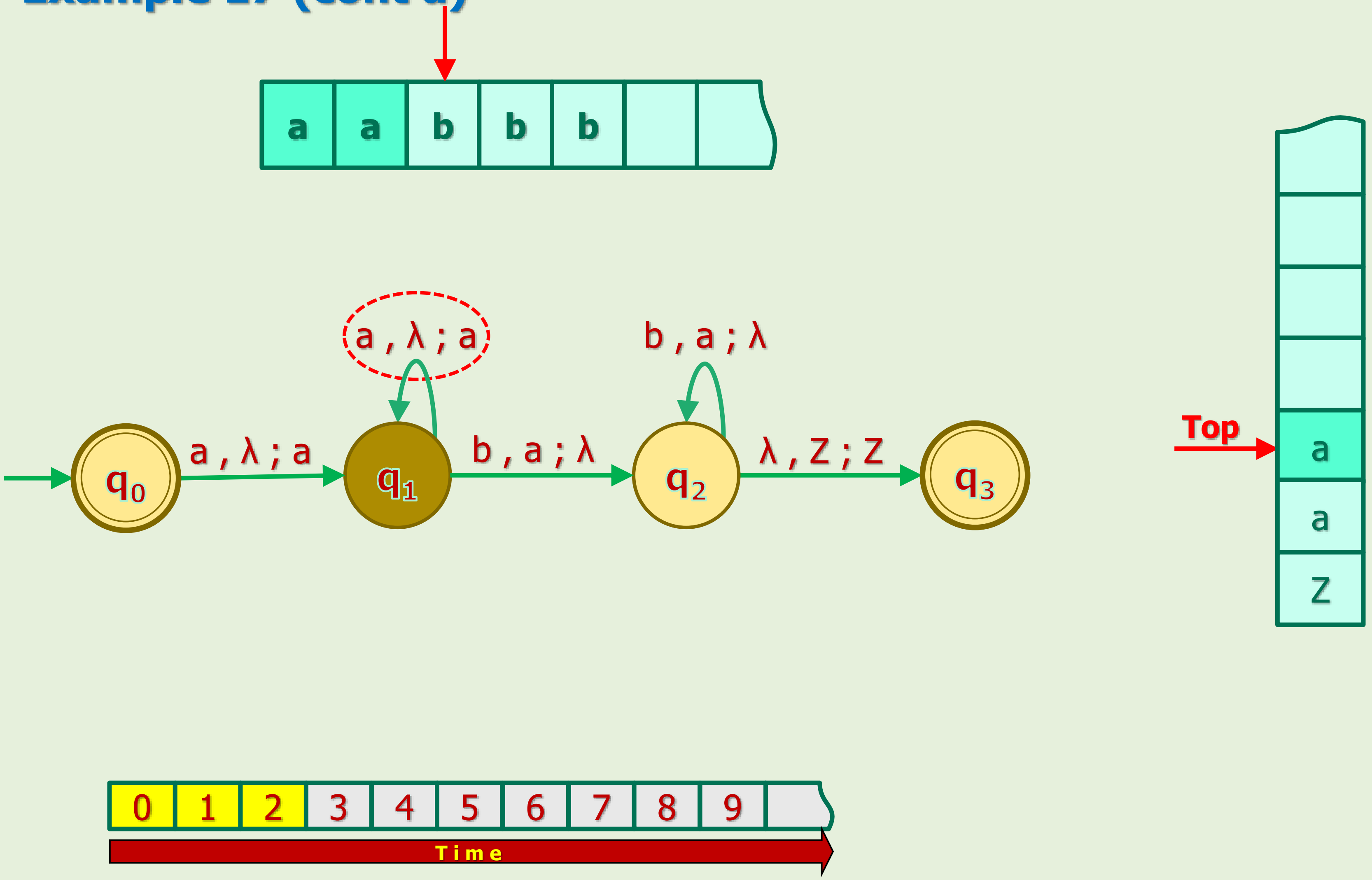
NPDAs in Action

Example 17 (cont'd)



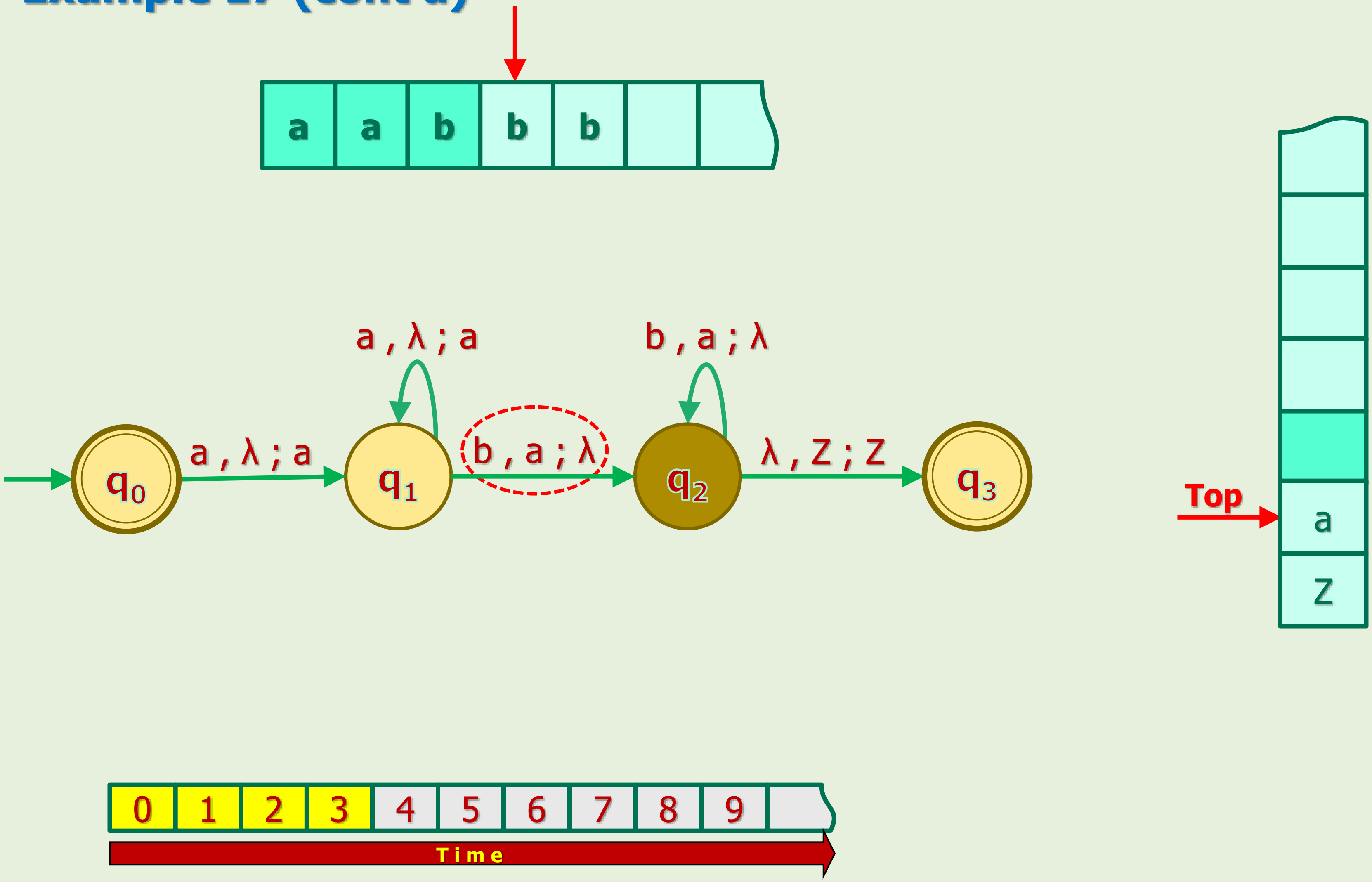
NPDAs in Action

Example 17 (cont'd)



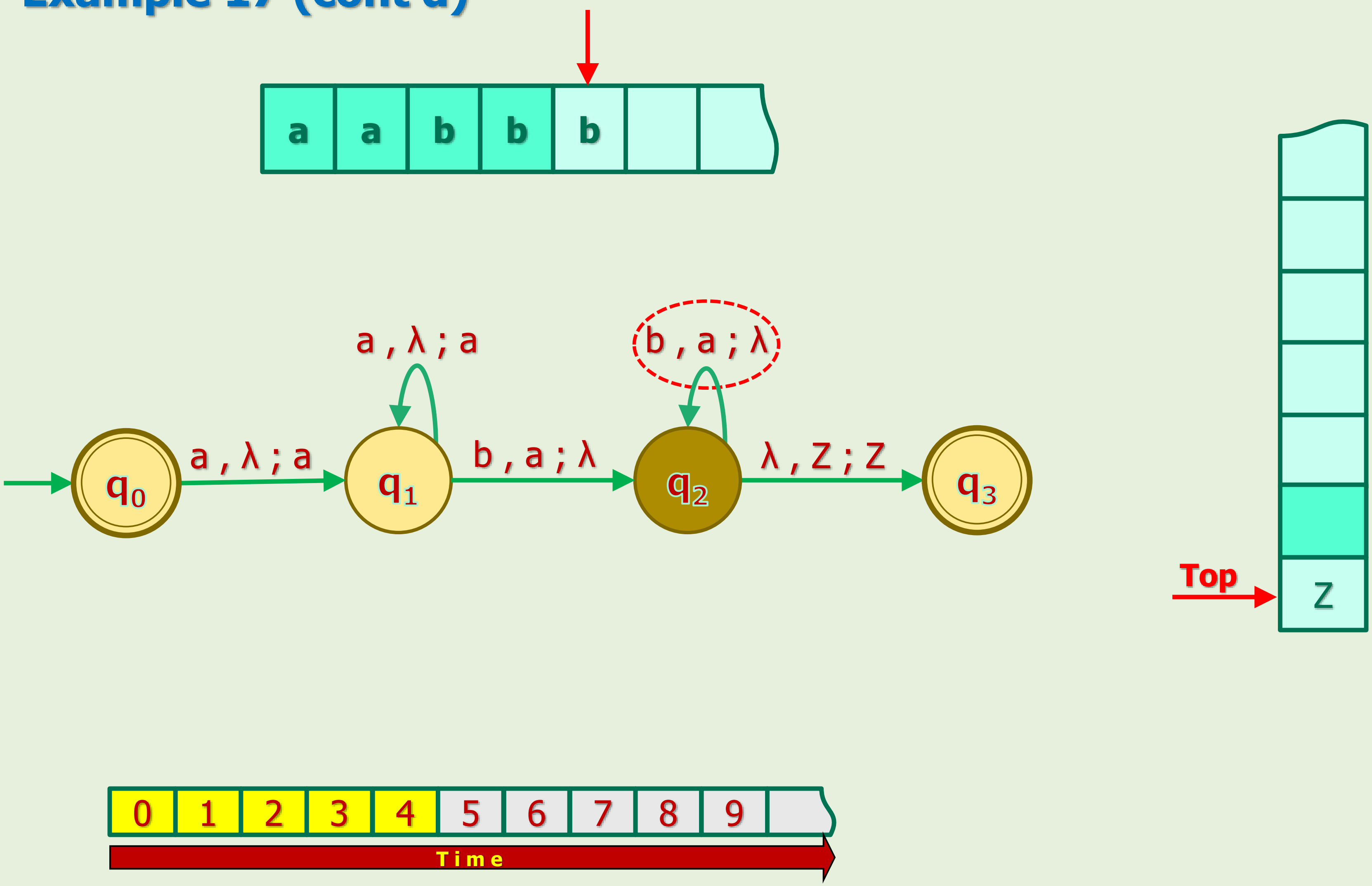
NPDAs in Action

Example 17 (cont'd)



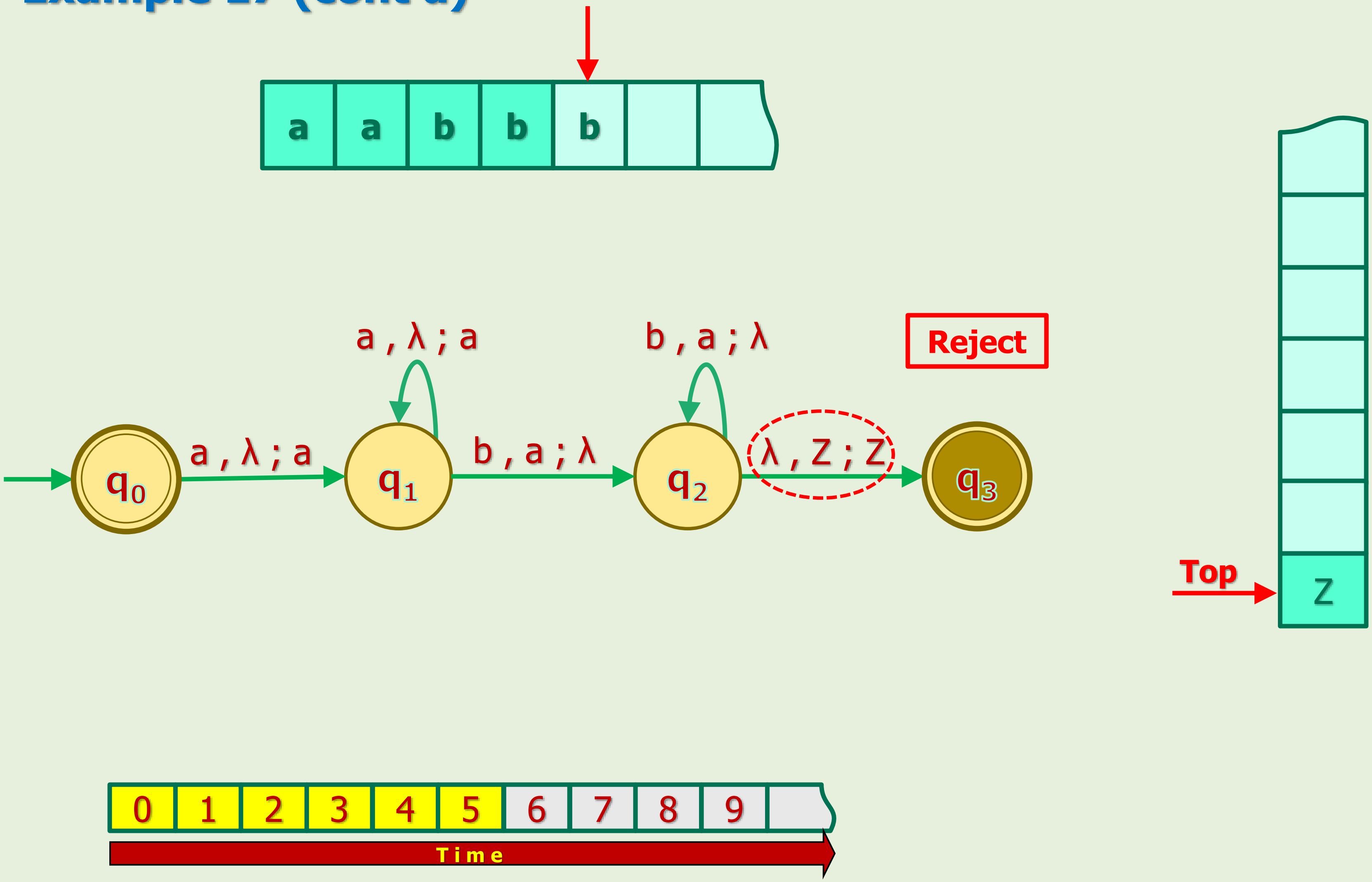
NPDAs in Action

Example 17 (cont'd)



NPDAs in Action

Example 17 (cont'd)



λ-Transitions

λ-Transitions

Recap

- λ-transition in automata theory means:

  The machine may "unconditionally" transit.

- To understand the λ-transition in any automata class, we need to know what the "transition condition" is.

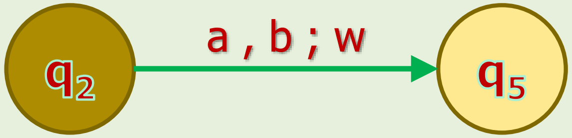
- This is our knowledge so far:

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

λ-Transitions

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

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



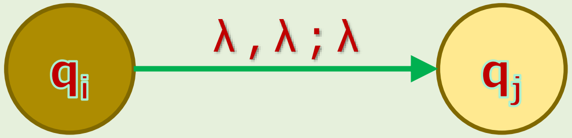
- So, if we put λ in the conditions places, we make a λ-transition.

λ-Transitions

Definition

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

- Note that w is a string and can be λ.
- So, one possible λ-transition that is used extensively is:



NPDAs' Behavior for λ-Transitions

- Since they may or may not transit, they would have multiple choices.

- We already know that in these cases, machines would check all possibilities by "parallel processing".
  - In other words, for every possible choice, they create a new independent process and every process independently continues processing the string.

- NPDAs' configuration = state + rest of input + stack

Homework

- Design an NPDA 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. $L = \{ww^R : w \in \{a, b\}^*\}$
  4. $L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$ //number of a's and b's are equal
  5. $L = \{1^n+1^m=1^{n+m} : n \geq 1, m \geq 1\}$ over $\Sigma = \{1, +, =\}$ (Unary addition)

Definitions

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:

$$\delta: Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow \text{a finite subset of } Q \times \Gamma^*$$

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.

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

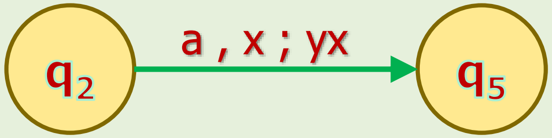
Transition Function

Stack Start Symbol

NPDAs Transition Function Examples

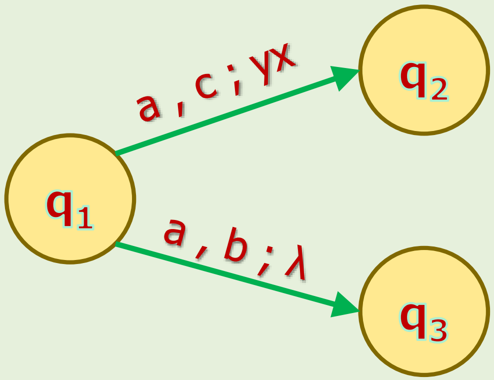
Example 19

- Write the sub-rule of the following transition.
- $\delta(q_2, a, x) = \{(q_5, yx)\}$



Example 20

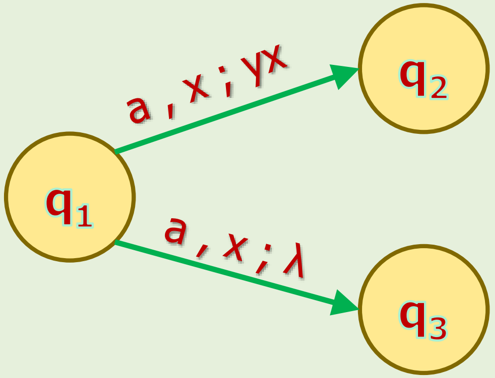
- Write the sub-rules of the following transition.
- $\delta(q_1, a, c) = \{(q_2, yx)\}$
- $\delta(q_1, a, b) = \{(q_3, \lambda)\}$



NPDAs Transition Function Examples

Example 21

- Write the sub-rule of the following transition.
- $\delta\,(q_1\,,\,a\,,\,x) = \{(q_2\,,\,yx), (q_3\,,\,\lambda)\}$



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.

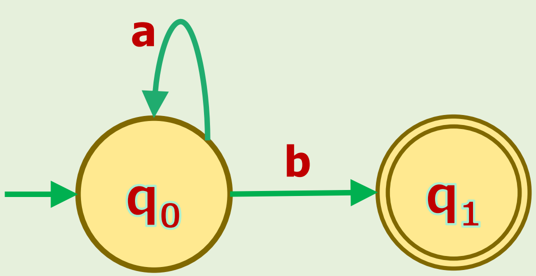
Can NPDAs Do Whatever NFAs Can Do?

Example 22

- Convert the following NFA's definition to an NPDA's.
- q_0 is the initial state, and q_1 is the final state.

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

NFA NPDA



- Just convert the δ.
- The reset items are the same.

Any NFA Can be Converted to NPDA

	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.

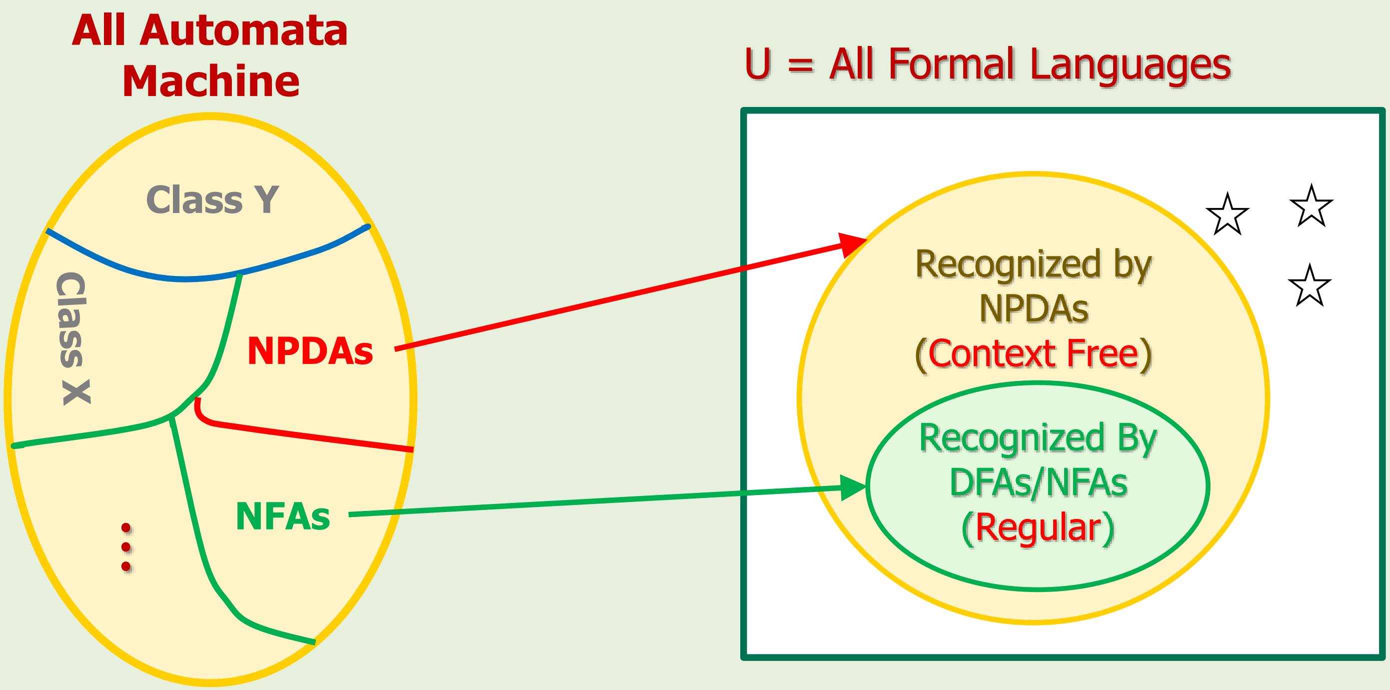
Can NFAs Do Whatever NPDAs Can Do?

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

- No! Why?
- We know at least the following languages for which we constructed NPDAs but it was 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 NPDAs recognize.
  - We'll explain later what the "context free" languages are.

NPDAs Power

- Design an NPDA for each of the following languages:
- $L = \{a^n b^n c^n : n \geq 1\}$ over $\Sigma = \{a, b, c\}$
- $L = \{ww : w \in \Sigma^*\}$ over $\Sigma = \{a, b\}$

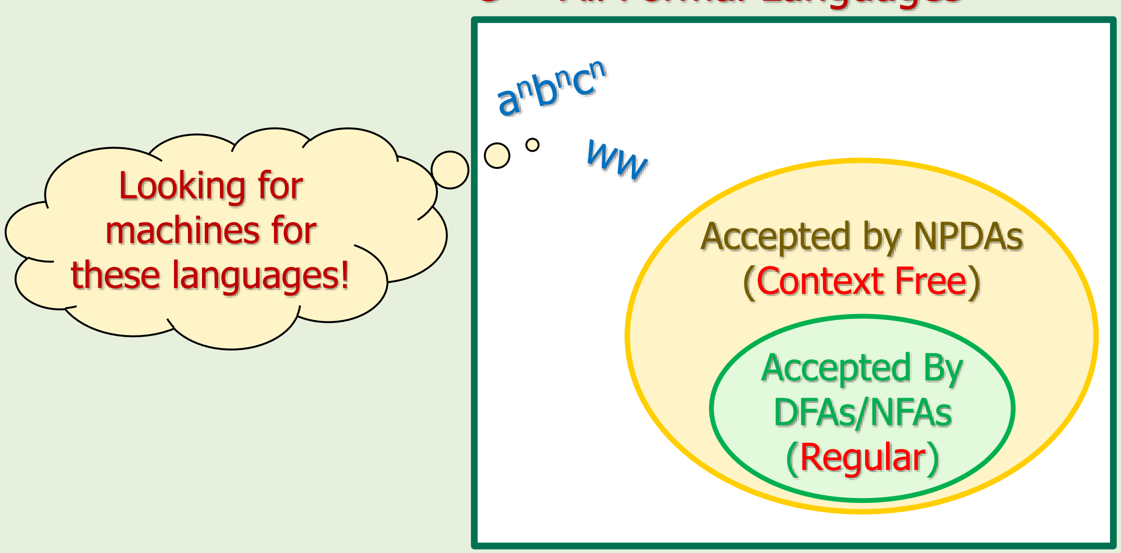
- After some struggling, you realize that you cannot construct such machines.

- 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.

What is the Next Step?

- NFAs/DFAs recognize regular languages.
- NPDAs recognize some non-regular languages called "context-free".
  - We'll see the meaning of context-free later.
- The next step is to define a new class of machines that recognizes all or part of the remaining non-regular languages.

U = All Formal Languages



Project (Optional)

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

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