San José State University
Department of Computer Science

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

Turing Machines

(Part 2)

Lecture 16

Day 17/31

CS 154

Formal Languages and Computability

Fall 2019

Agenda of Day 17

- Solution and Feedback of Quiz 5
- Summary of Lecture 15
- Quiz 6
- Lecture 15: Teaching ...
  - Turing Machines (Part 1)

Solution and Feedback of Quiz 5 (Out of 30)

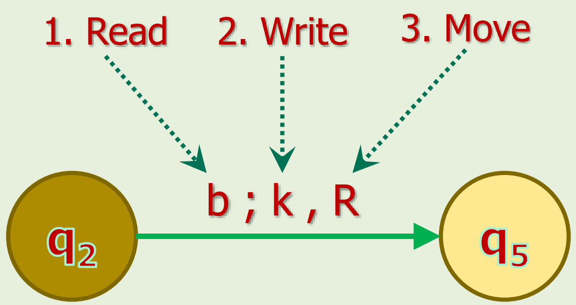
Section	Average	High Score	Low Score
01 (TR 3:00 PM)	26.7	30	22
02 (TR 4:30 PM)	27.37	30	20
03 (TR 6:00 PM)	28.66	30	24

Summary of Lecture 15: We learned ...

Turing Machines (TMs)

- NPDAs are unable to accept some languages like $a^n b^n c^n$ and ww.
- The limitation of NPDAs is ...
  1. ... stack is not so flexible in storing and retrieving data.
  2. ... we lose some data when we access the older data.
- We replaced stack with a kind of RAM and ...
- ... introduced Turing machines (TM) to overcome this limitation.
- Both deterministic and nondeterministic TMs (NTM) exist.

- TMs have 2 main blocks:
  - input / output tape
  - control unit
- Designers have full control on moving the cursor left or right.
- The label on the edges look like this:



Any Question

Summary of Lecture 15: We learned ...

Turing Machines (TMs)

- The transition condition of TMs is ...
  - ... input symbol.
- TMs halt iff ...
  - ... they have zero transition.

$$z \leftrightarrow h$$

- The criteria of accepting strings for previous machines are ...

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

- Consuming all input symbols is meaningless for TMs.

- So, theoretically, the logical representation of accepting strings is

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

- But in practice ...
- it's the responsibility of the TMs designer (you) to define when a string is accepted/rejected.
- In other words, that is the TMs designers responsibility to make sure that the machine halts in an accepting state when all symbols are visited.
- And for rejecting strings is ...

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

Any Question

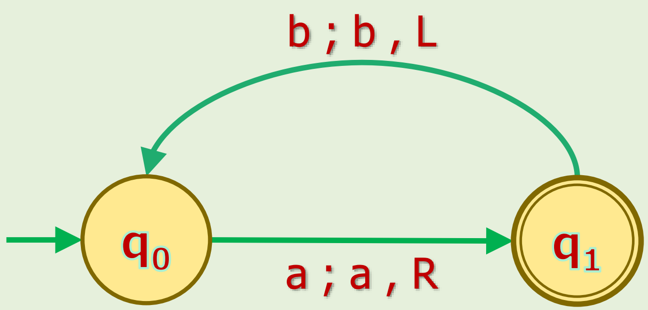
Quiz 6

No Scantron

A Special Phenomenon in TMs

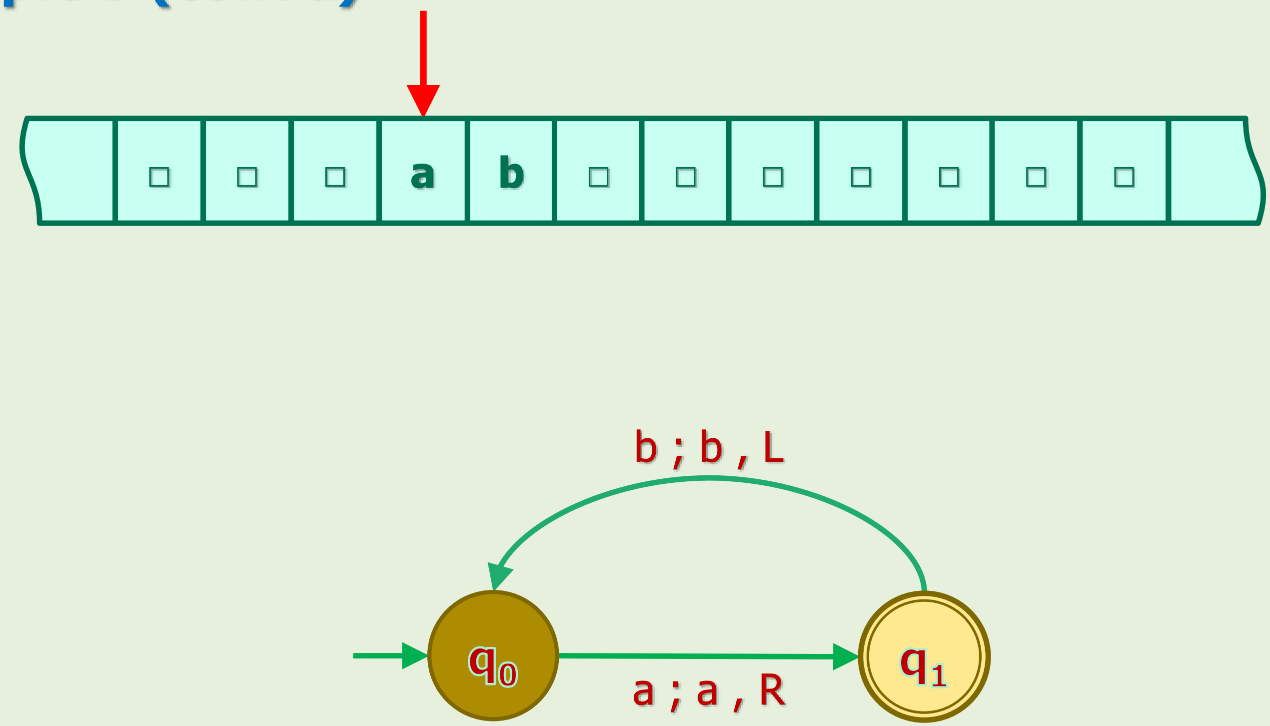
Example 5

- Trace the following TM for the input "ab".



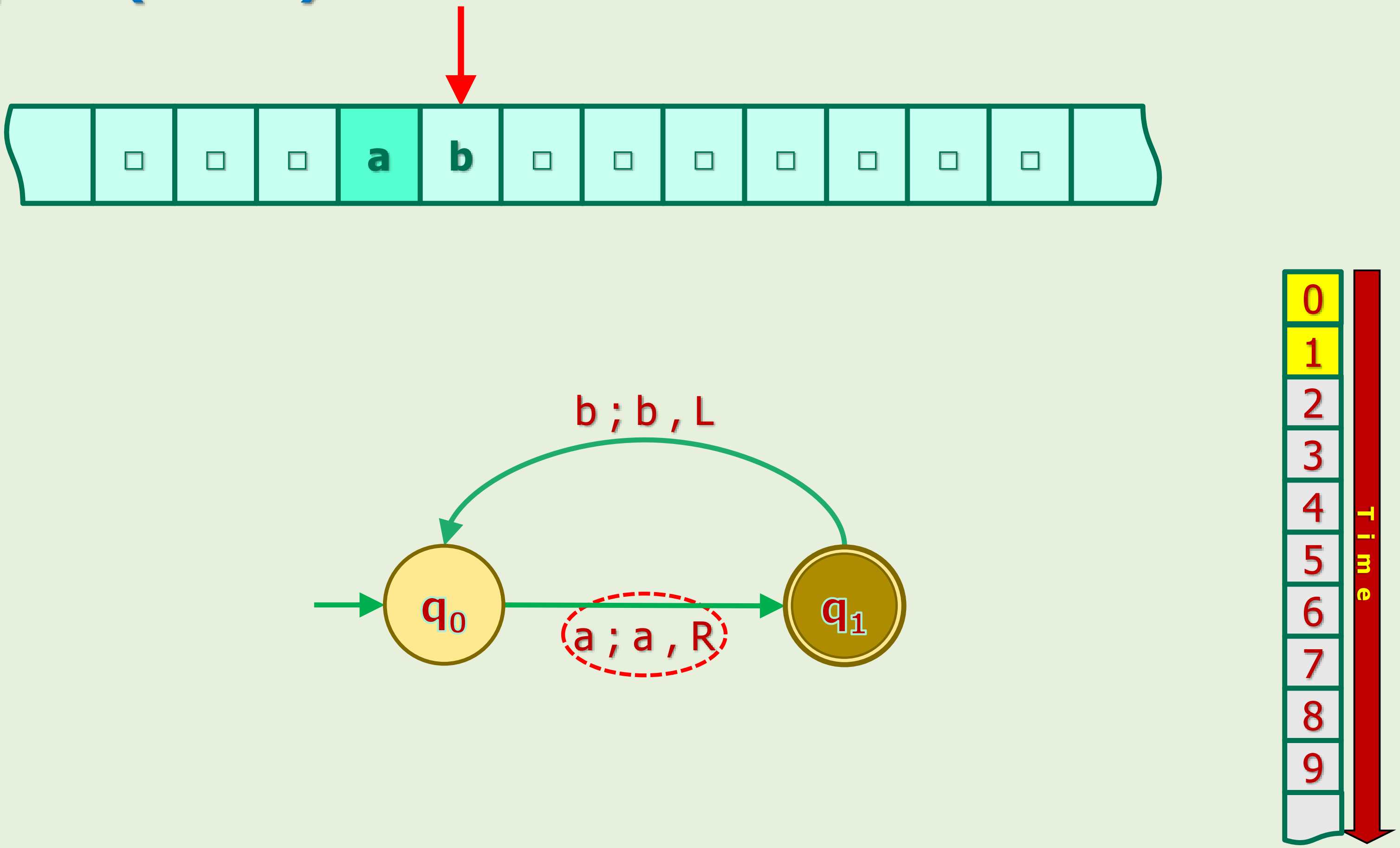
A Special Phenomenon in TMs

Example 5 (cont'd)



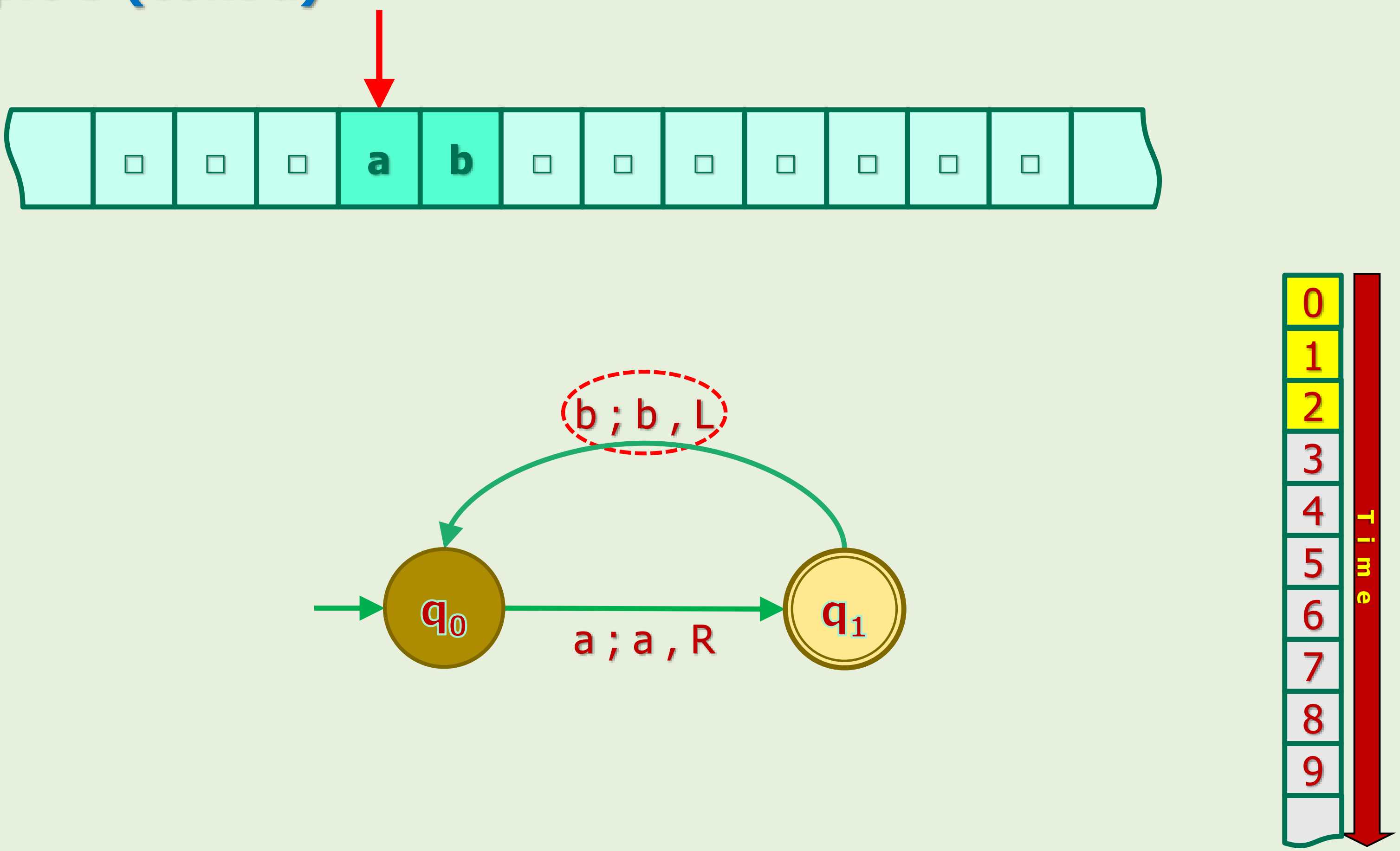
A Special Phenomenon in TMs

Example 5 (cont'd)



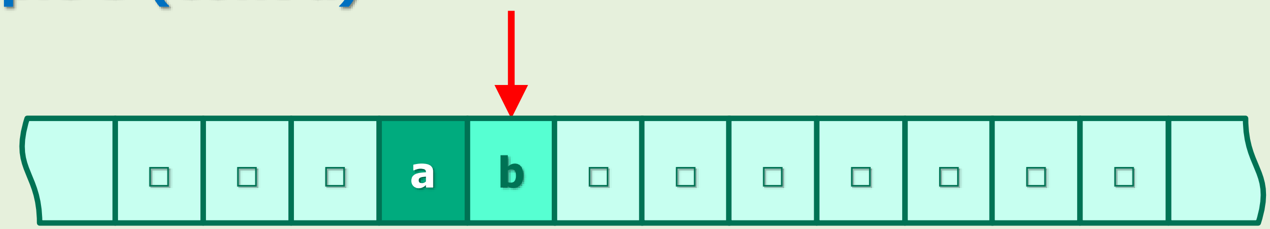
A Special Phenomenon in TMs

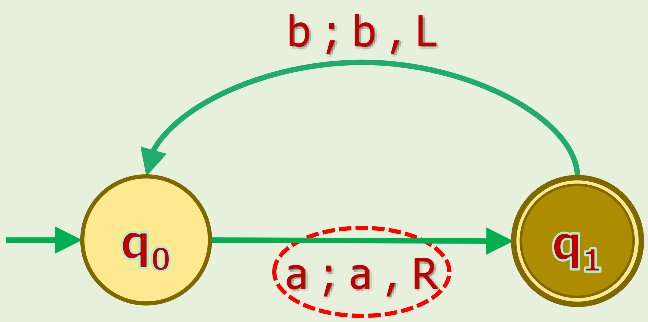
Example 5 (cont'd)



A Special Phenomenon in TMs

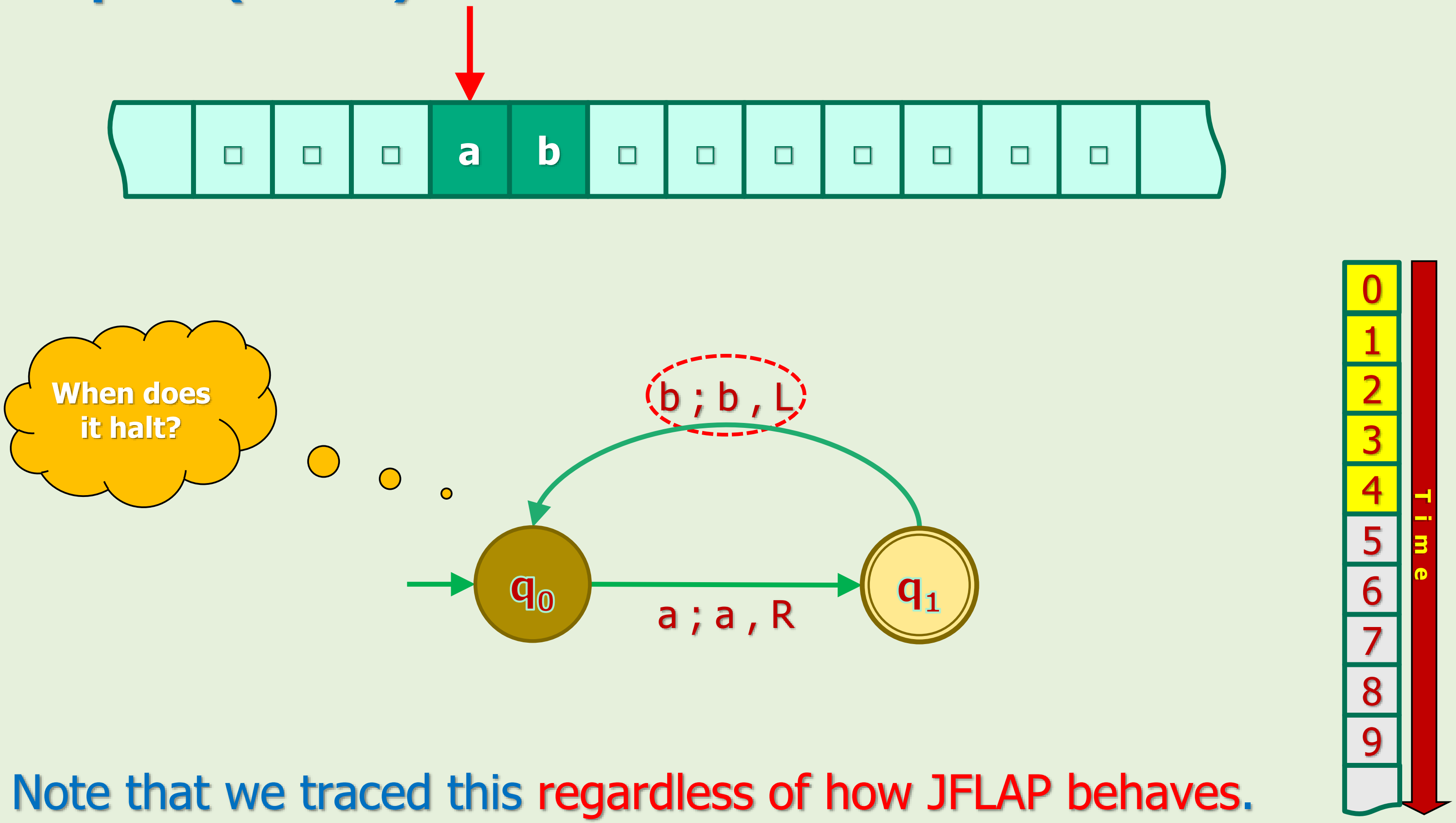
Example 5 (cont'd)





A Special Phenomenon in TMs

Example 5 (cont'd)



- Note that we traced this regardless of how JFLAP behaves.
- Our setting for JFLAP will stop in timeframe 1!

What was that phenomenon?

- The TM never halts.
- In other words, in some situations,

  A TM can fall into an "infinite loop".

- This phenomenon ...
- ... never happened in the previous DETERMINISTIC machines.

- What is the reason?
  - This is the consequence of ...

    ... having freedom of moving the cursor to the left or right.

A Side Note About Rejecting String

- Based on the rejection logic:

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

- If we can prove somehow that
  the machine falls into an infinite loop, then ...

- ... the string, that is being processed, is considered as rejected.
- ... because $\sim h \equiv$ True.

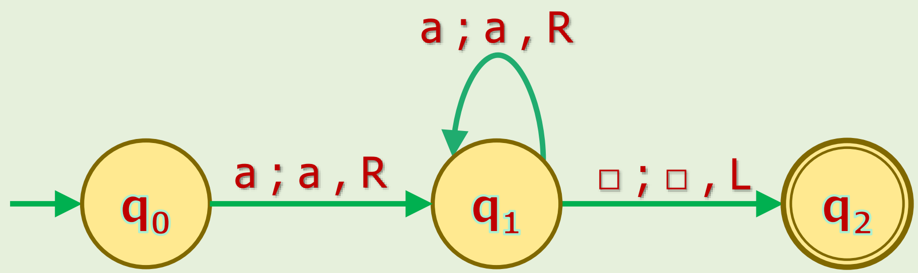
5. TMs in Action

Design Examples

TMs Design Examples

Example 6

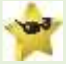
- Design a TM to accept $L = \{a^n : n \geq 1\}$ over $\Sigma = \{a, b\}$.
- Note that TMs usually don't like λ!

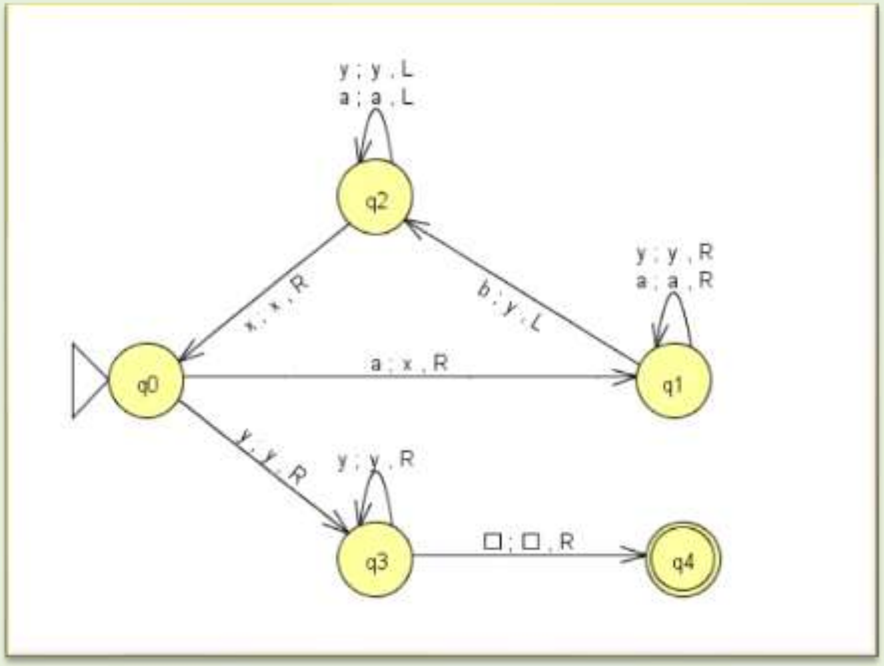


TMs Design Examples

Example 7

- Design a TM to accept our famous language $L = \{a^n b^n : n \geq 1\}$ over $\Sigma = \{a, b\}$.





Solution

- **Strategy**: For every a's, you should find one 'b'. So, we read the first 'a' and mark it as read by replacing it with 'x'. Then we go right to find a corresponding 'b' and mark it as 'y'.
  We continue this process until we don't have any a's.
  The string is accepted if there is no 'b' either.

Homework: TM Design

- Design a TM for the following languages:

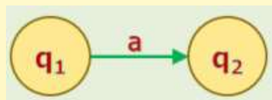
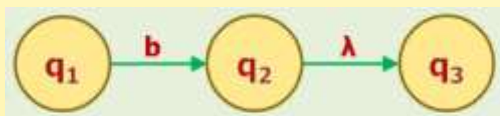
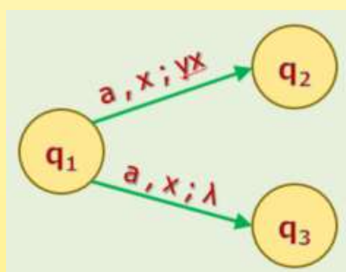
1. $L = \{w \in \{a, b\}^+\}$
2. $L = \{w \in \{a, b\}^+ : |w| = 2k, k \geq 1\}$
3. $L = \{w \in \{a, b\}^+ : |w| = 2k+1, k \geq 0\}$
4. $L = \{1^{2k} : k \geq 1\}$ over $\Sigma = \{1\}$
5. $L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$ //number of a's = number of b's
6. $L = \{w \in \{a, b\}^+ : n_a(w) = n_b(w)\}$ //number of a's = number of b's
7. $L = \{a^n b^n c^n : n \geq 1\}$
8. $L = \{a^n b^m c^{nm} : n \geq 1, m \geq 1\}$
9. $L = \{w\#w : w \in \{a, b\}^+\}$
10. $L = \{b^n a w : n \geq 0, |w| = 2k+1, k \geq 0, w \in \{a, b\}^+\}$
11. $L = \{ww : w \in \{a, b\}^+\}$

6. Definitions

Transition Function of TMs

- In this section, we are going to formally (mathematically) define the TMs.
- As usual, the transition function is the important part of this definition.
- Because we are familiar with most other items of the definition.
- So, let's take some examples on transition functions.
- And try to figure out what the transition functions look like.

Transition Function: DFAs, NFAs, NPDAs, TMs

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

TMs Transition Function Examples

Example 9

- Write the sub-rule of the following transition.

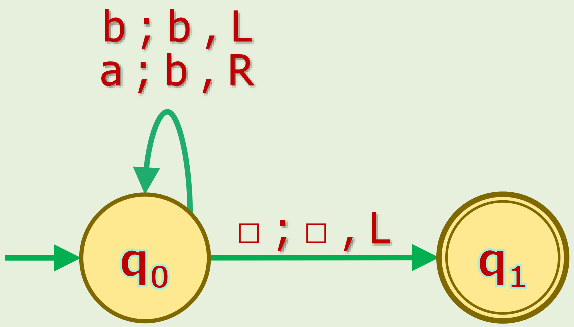
Solution

$\delta\ (q_1\ ,\ a) = (q_2\ ,\ b\ ,\ R)$

TMs Transition Function Examples

Example 10

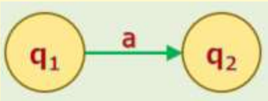
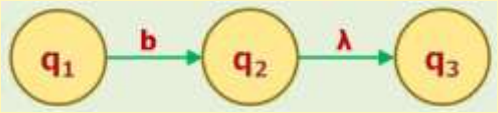
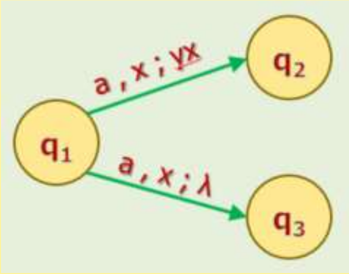
- Write the δ of the following transition graph.



Solution

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

Transition Function: DFAs, NFAs, NPDAs, TMs

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		$\delta\ (q_1\ , a\ , x) = \{(q_2\ , yx), (q_3\ , \lambda)\}$ $\delta: Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \rightarrow 2^{Q \times \Gamma^*}$
TMs		$\delta\ (q_1\ , a) = (q_2\ , b\ , R)$ $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

6. Formal Definition of TMs

- A standard TM M is defined by the septuple (7-tuple):

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

- 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 tape alphabet.
  - δ is called transition function and is defined as:

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

δ can be total or partial function.

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

6. Formal Definition of TMs

$$M = (\overbrace{Q}^{\text{Set of States}}, \underbrace{\Sigma}_{\text{Input Alphabet}}, \overbrace{\Gamma}^{\text{Tape Alphabet}}, \underbrace{\delta}_{\text{Transition Function}}, \overbrace{q_0}^{\text{Initial State}}, \underbrace{\square}_{\text{Blank Symbol}}, \overbrace{F}^{\text{Set of Final States}})$$

6. Formal Definition of TMs: Notes

1. $\Sigma \subseteq (\Gamma - \{\square\})$
   - The input string cannot contain blank symbol.
2. There is no relationship between determinism and δ being total function.

   The following table clearly depicts this fact.

Class	Transition Function Type	Type of Machine
DFAs	Total	Deterministic
NFAs	Total	Nondeterministic
DPDAs	Partial or Total	Deterministic
NPDAs	Total	Nondeterministic
TMs	Partial or Total	Deterministic

7. TMs vs NPDAs

Can TMs Simulate NPDAs?

- Let's assume that we've constructed an NPDA for an arbitrary language L.
- Can we always construct a TM for L?
- Recall that to answer this question for previous machines (i.e. DFAs, NFAS, NPDAs),
  we used the "formal definition conversion" technique .
- For this case, we cannot do that ...
  - ...because their formal definitions are very different.
- Therefore, we'll be using real "simulation".
- The answer would be: Yes! How?

Can TMs Simulate NPDAs?

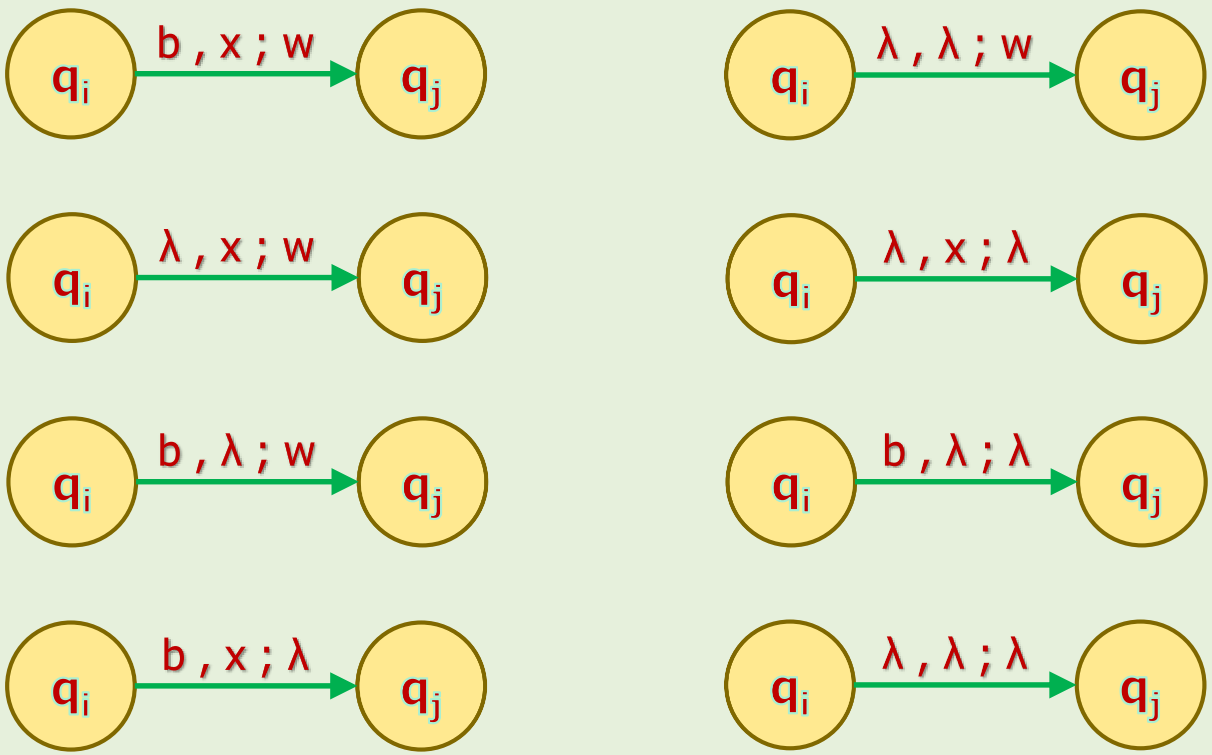
- Let M be an NPDA for the language L.
- We want to simulate M by an equivalent TM called M' such that:

$$L(M) = L(M')$$

- M has some transitions and we should be able to simulate all of them by TM.
- Let's list all kind of transitions that an NPDA can have.
- If we can simulate them by TMs, then we'd be able to simulate any NPDAs by TMs

Can TMs Simulate NPDAs?

NPDAs All Possible Transitions



Can TMs Simulate NPDAs?

- We just show the simulation of one transition.
- And we leave the rest for the readers as exercise.

- I put the following file in Canvas for your reference:

  Canvas → Files → Misc

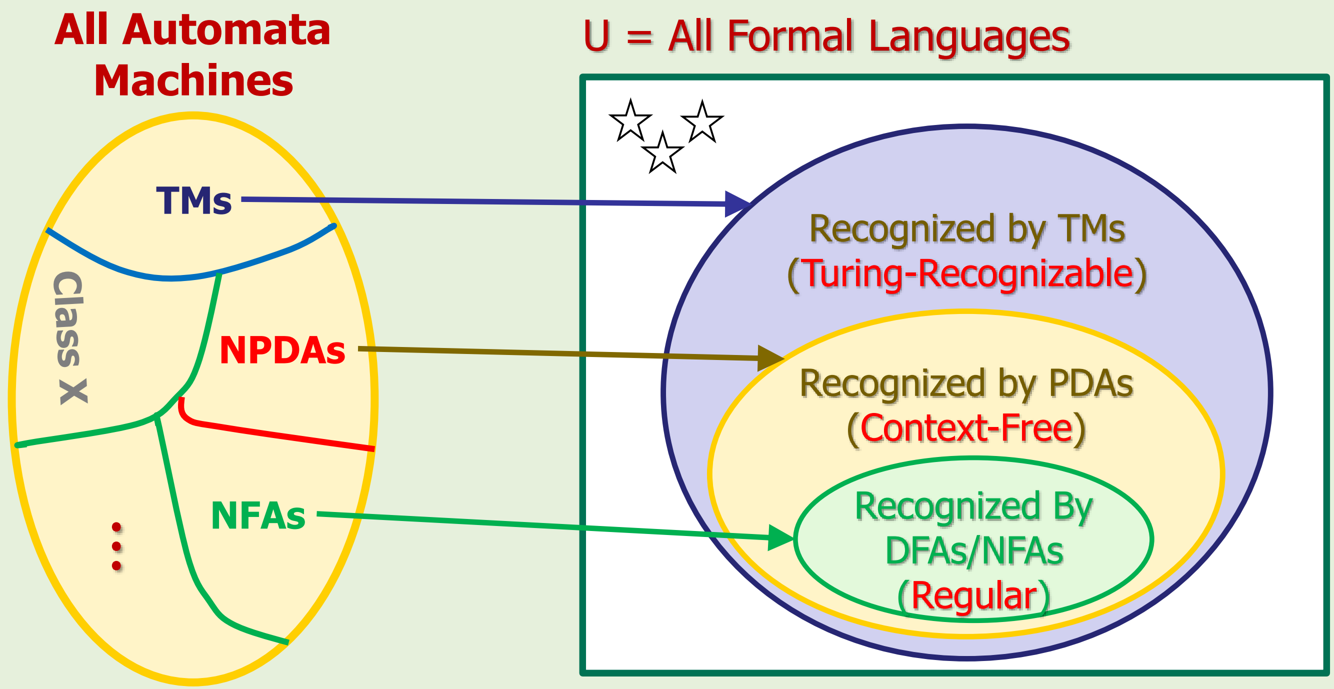
  CS154-Ahmad Y-NPDAs-Transition-Simulation.pdf

- That'd be a good experience for your term project too.

Can NPDAs Simulate TMs?

- Let's assume that we've constructed a TM for an arbitrary language L.
- Can we always construct an NPDA for L?
- No! Why?
- At least, we know the following languages for which we can construct TMs but it is impossible to construct NPDAs.
  - $L = \{a^n b^n c^n : n \geq 1\}$
  - $L = \{ww : w \in \Sigma^*\}$
- Let's summarize our knowledge and figure out what would be the next step.

Machines and Languages Association

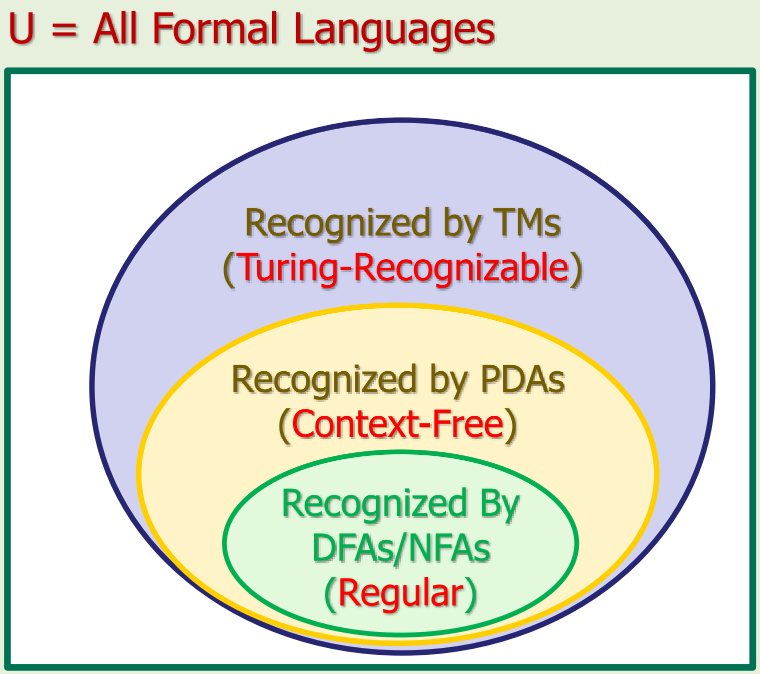


- The set of languages that NPDAs recognize is a proper subset of the set of languages that TMs recognize.
- So, TMs are more powerful than NPDAs.

Turing-Recognizable Languages

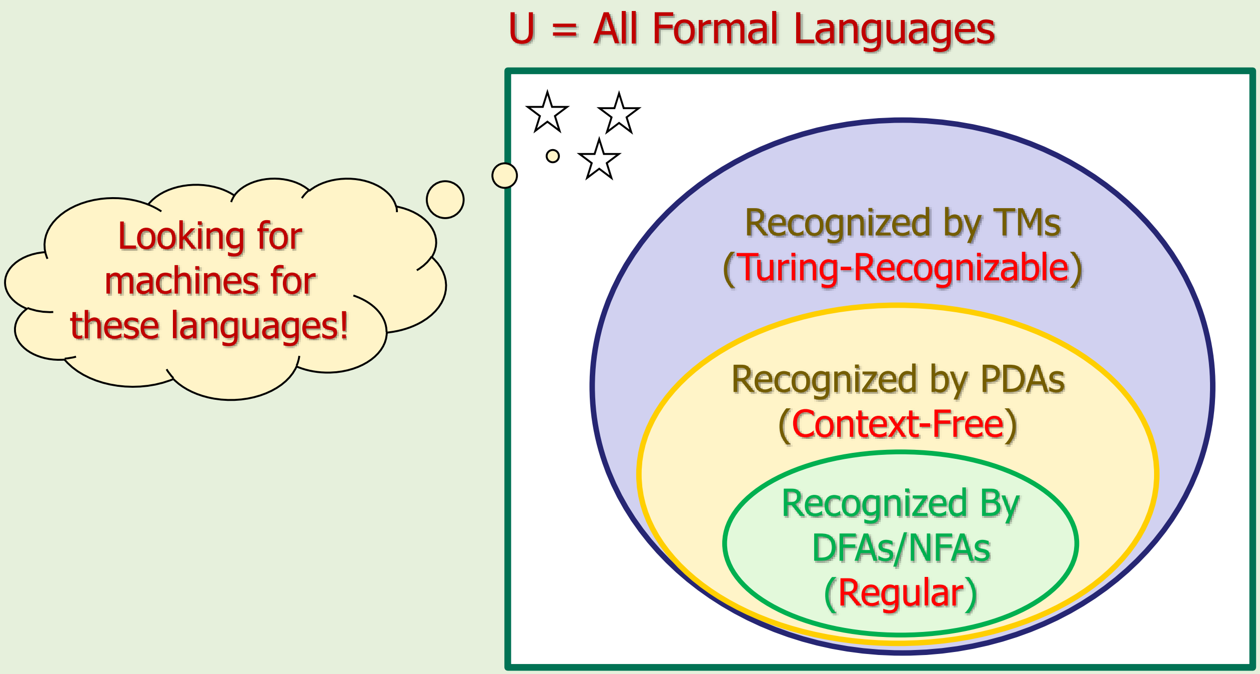
Definition

- A languages is called "Turing-recognizable" if there exists a TM that accepts (aka recognizes) it.



8. What is the Next Step?

- There are still languages that are not Turing-recognizable!
- First, we need to find at least one of them, then we'll think about constructing a new class of automata!



Basic Concepts of Computation

Definition of Algorithm

Definition

- An algorithm for a problem L (= language) is equivalent to design a TM that solves it (= accept the language).

- In other words, we define the TM's transition graph as the "algorithm" for solving that problem.

Definition of Program

- A sub-rule defines how a machine acts in one transition for a specific state.
- Based on "logic of transition" that we mentioned before, a sub rule is an "IF-THEN" statement for a specific state.
- Therefore, the transition function of a TM, contains a set of "IF-THEN"s.
- This set can be called "program".

Definition

- The transition function of a TM is its "program".

- Based on this definition, TMs programming follows "functional programming paradigm".

Nice Videos

1. Turing machines explained visually
   https://www.youtube.com/watch?v=-ZS_zFg4w5k
2. A Turing machine – Overview
   https://www.youtube.com/watch?v=E3keLeMwfHY

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
4. Wikimedia Commons, https://commons.wikimedia.org/wiki/Category:Animations_of_machinery
5. https://en.wikipedia.org/wiki/Turing_Award
6. https://en.wikipedia.org/wiki/Alan_Turing
7. https://www.turing.org.uk/