San José State University
Department of Computer Science

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

Other Models of TMs

(Part 1)

Lecture 17

Day 18/31

CS 154

Formal Languages and Computability

Fall 2019

Agenda of Day 18

- About Your Term Project
- Solution and Feedback of Quiz 6
- Summary of Lecture 16
- Lecture 17: Teaching ...
  - Other Models of TMs (Part 1)

About Your Term Project

- Your term project has been posted!
- The first assignment of term project has been graded.

- This week is a very good time to ...
  1. Get familiar with each other's capabilities
  2. Get familiar with JFLAP's features
  3. Get familiar with the last year term project that I provided
  4. Start designing the top level structure of your code

- I'll have a quick demo about one of the previous projects later.
- Please read the requirement at least ...

Solution and Feedback of Quiz 6 (Out of 17)

Section	Average	High Score	Low Score
01 (TR 3:00 PM)	14.23	17	10
02 (TR 4:30 PM)	14.67	17	9
03 (TR 6:00 PM)	14.28	17	9

Summary of Lecture 16: We learned ...

TMs

- We observed a new phenomenon that happens in Turing machines ...
  - ... Infinite loops!
- This phenomenon never happened in the previous deterministic machines.
- This is the consequence of ...

  ... having freedom of moving the read-write head to the left or right.
- If a TM is in infinite-loop, the string that is being processed is considered as rejected.

Formal Definition of TMs

- A standard (deterministic) Turing machine M is defined by the septuple:

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

$\square \in \Gamma$ is a special symbol called blank.

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

δ can be total or partial function.

- Sub-rule example:

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

Any Question

Summary of Lecture 16: We learned ...

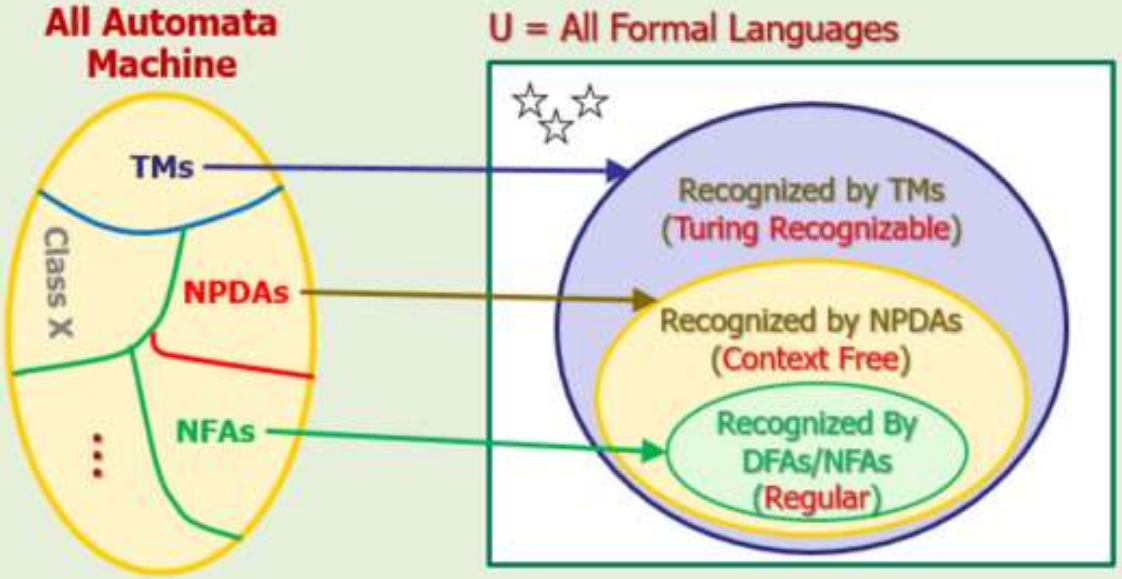
TMs vs NPDAs

- We use simulation to compare TMs and NPDAs.
- We can simulate whatever NPDAs do with TMs.
- But not vice versa!
  - At least we know some languages for which we could not construct NPDAs but could construct TMS
  - such as: $a^n b^n c^n$ and ww
- So, TMs are more powerful than NPDAs.

Any Question

Summary of Lecture 16: We learned ...

Machines and Languages Association



- A language is called Turing-recognizable if ...
- ... there exists a TM that recognize it.

Any Question?

Summary of Lecture 16: We learned ...

Basic Concepts of Computation

- The algorithm for a problem (= language) is ...
  - ... the structure of the TM that solves (= accepts) it.

- The program of a TM is ...
  - ... the transition function of the TM.

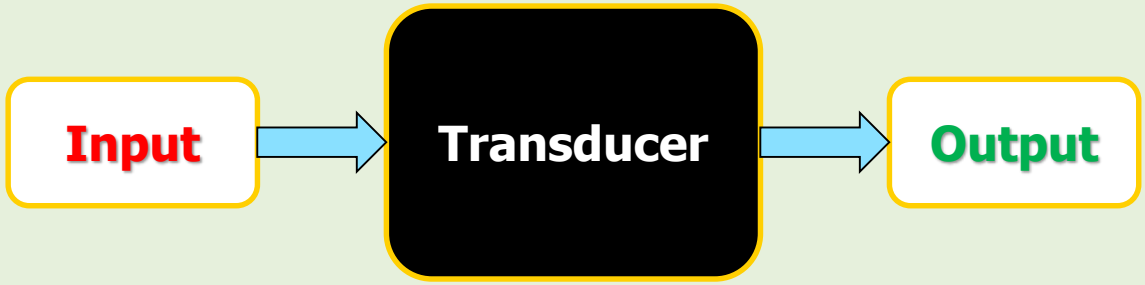
Any Question?

TMs as Transducers

What is Transducer?

- In physics, transducer is a device that converts the variation in a physical quantity into an electrical signal, or vice versa.
  - The variation could be movement, pressure, brightness, or so forth.

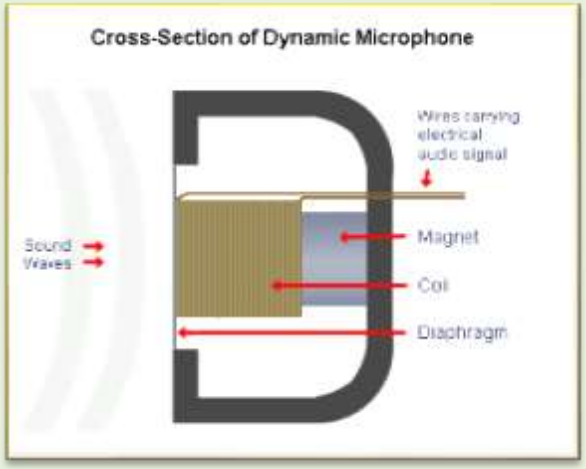
- In a nutshell, transducer is a device that converts an input to an output.



Examples of Transducers

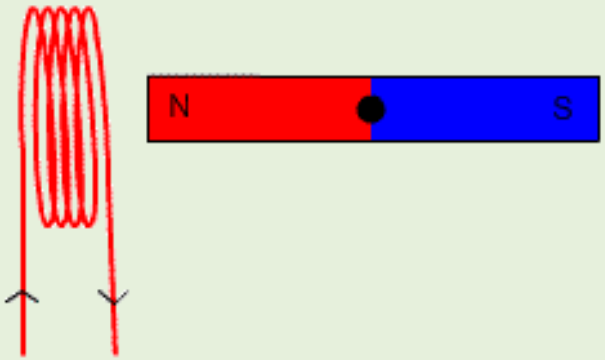
Microphone

- A microphone converts your voice to electrical signals.



Electric Generator

- An electric generator converts movement to electrical power.



Reference

- Bo Krantz Simonsen at da.wikipedia

Mathematical Model of Transducers

- What is the mathematical model for transducers?
- Functions!

- Recall that a function ...
- ... converts (aka maps) an input (a member of its domain) to an output (a member of its range) based on a rule.

TMs as Transducers

- TMs can act like transducers. (= TMs can implement functions.)

- If this is the case, what are the input and output of TMs when they act like transducers?

Input

- All or part of the nonblank symbols on the tape at the initial time.

Output

- All or part of the tape's content when the machine halts.

- In fact, it's designer's responsibility to define the input and output.

How JFLAP Shows TMs' Outputs

- JFLAP can run TMs in two modes:
  - Regular mode [Menu → Input → Multiple Run]
  - Transducer mode [Menu → Input → Multiple Run (Transducer)]

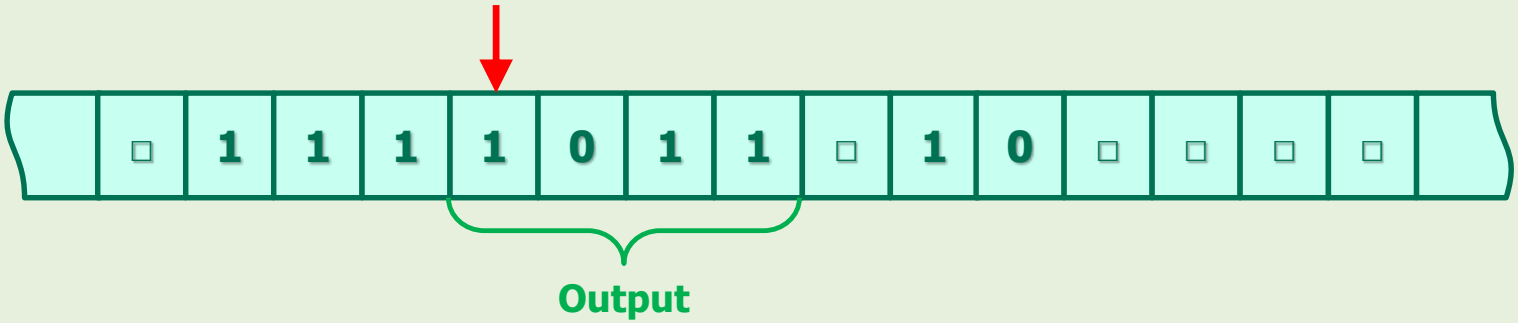
- In regular mode, JFLAP shows "Accept" or "Reject" for accepting/rejecting strings.
- In transducer mode, it shows the output of the computation too.

- We'll follow how JFLAP shows the output when we run TMs in transducer mode.

- Let's explain it through an example.

How JFLAP Shows TM's Output in Transducer Mode

- Assume our TM halts in an accepting state, and the content of the tape is ...



- ... then, the output would be 1 0 1 1.
- So, the output starts from the symbol at which the cursor is pointing, to the right, until the first blank.
- Note that if the TM does not halt in an accepting-state, JFLAP does not show the output.

- Now, let's get back to TMs' ability to implement functions.

Turing Computable Functions

Definition

- A function is said to be "Turing-computable" (or just "computable") if there exists a TM that implements it.

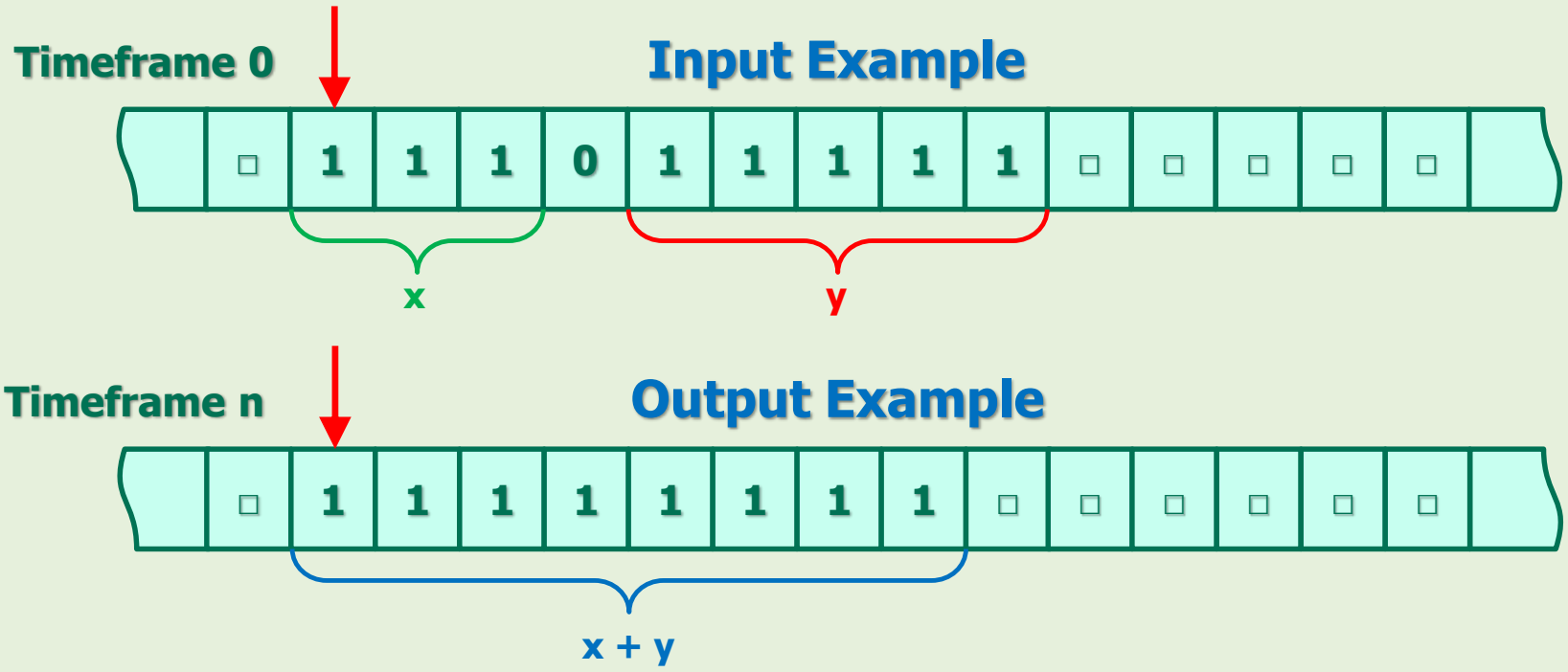
- It has been proved by computer scientists that all common mathematical functions are Turing-computable.

- Let's take some practical examples (basic operations of computers) such as:
  - adding numbers
  - copying strings
  - simple comparisons

TMs as Transducers

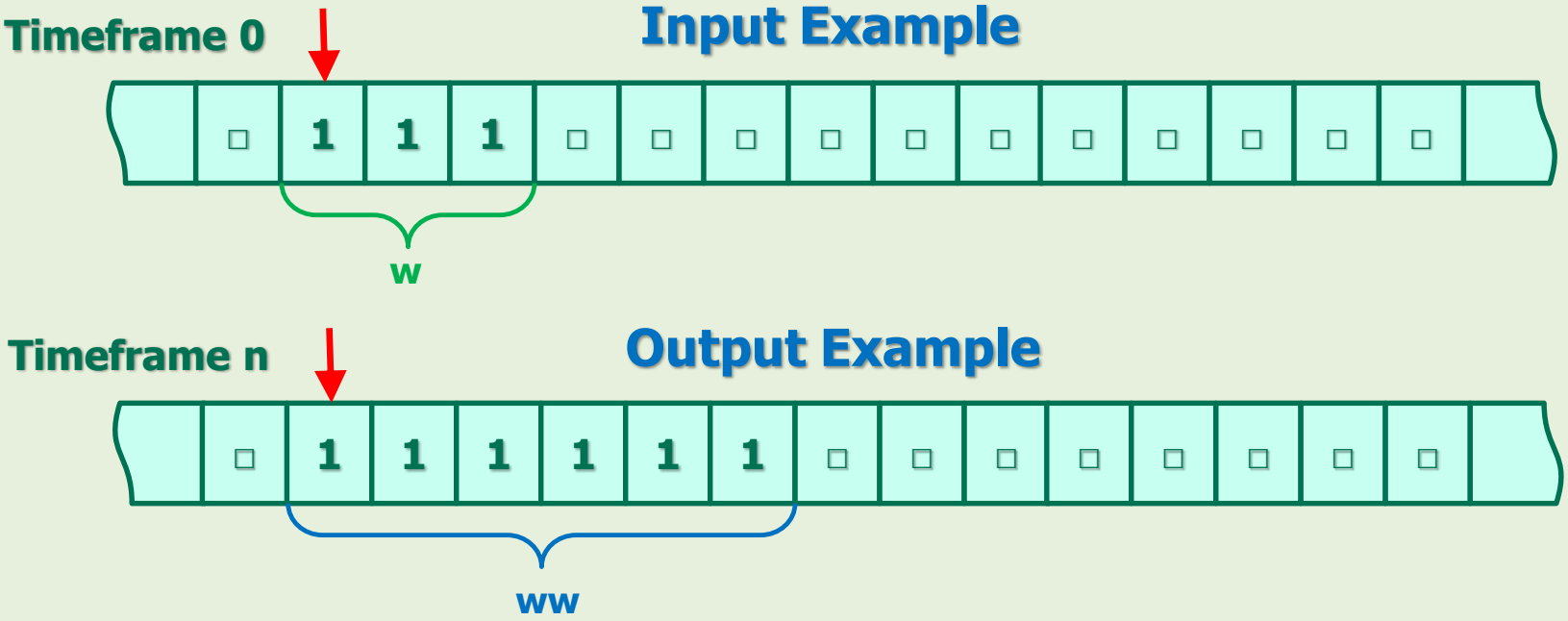
Example 1

- Given two unary numbers x and y, separated by a 0 (zero).
- Design a TM that computes x + y.



Homework

- Given a string $w \in L = \{1\}^+$.
- Design a TM that writes a copy of w at the end of the w.



- At the end of the execution, we'll have "ww" on the tape.

Homework

- Given two unary numbers x and y (x goes first), separated by a 0 (zero).
- Design a TM that halts in an accepting state if $x \geq y$, otherwise it halts in a non-accepting state.

Combining TMs

Combining TMs

- How to combine TMs to solve more complex problems?

Example 2

- Given two unary numbers x and y (x goes first), separated by a **0** (zero).
- Design a TM to compute the following function:

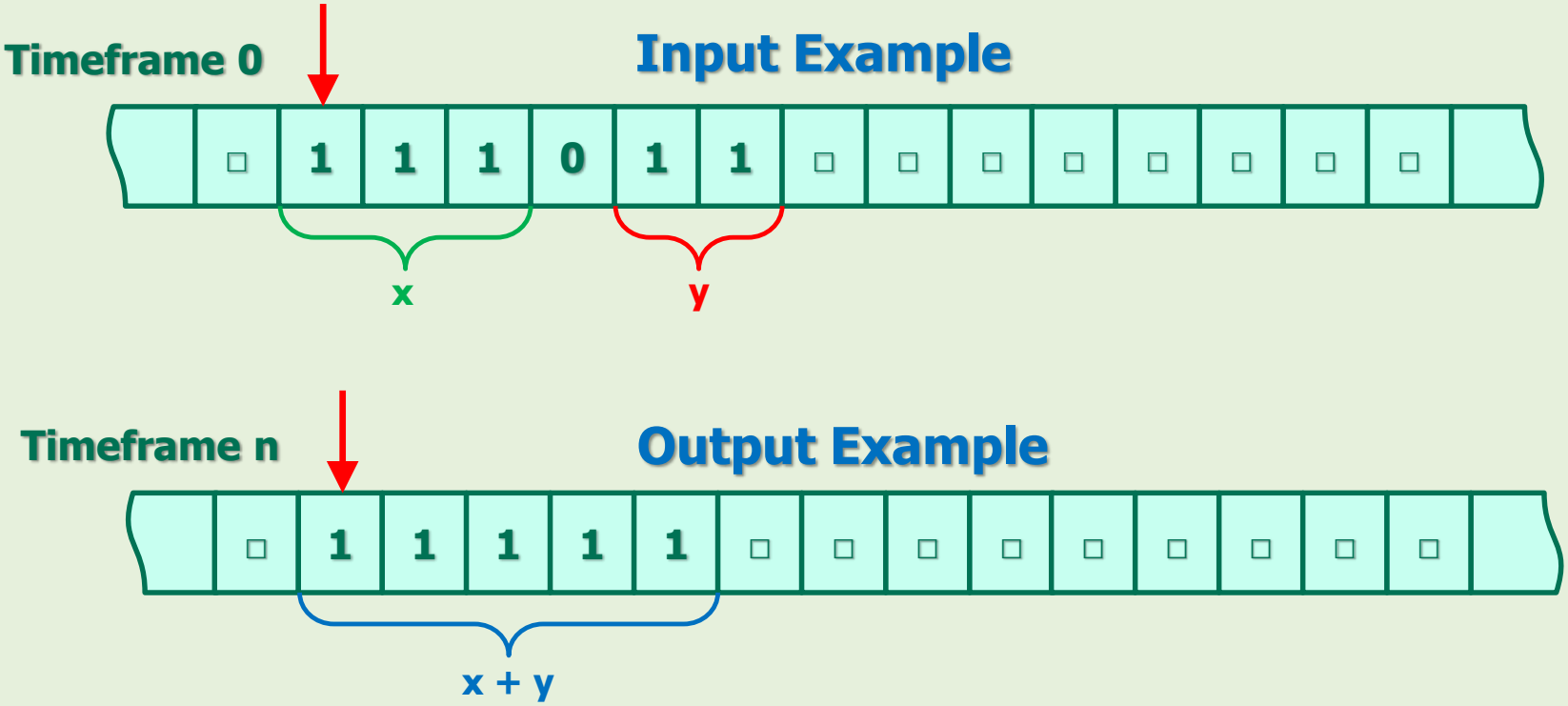
$$f(x, y) = \begin{cases} x + y & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

- Let's visualize what would be the output in different situations ...

TM as Transducer

Example 2: When x ≥ y

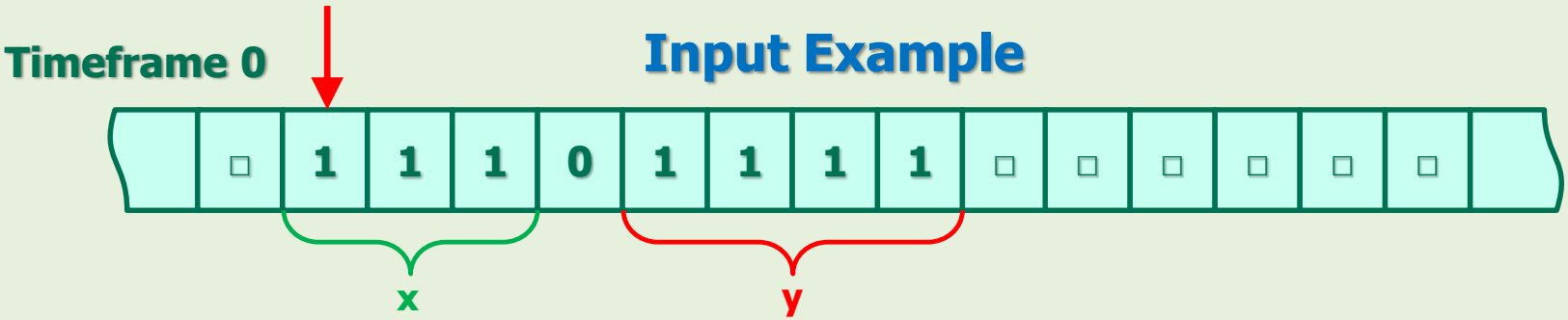
$$f(x, y) = \begin{cases} x + y & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

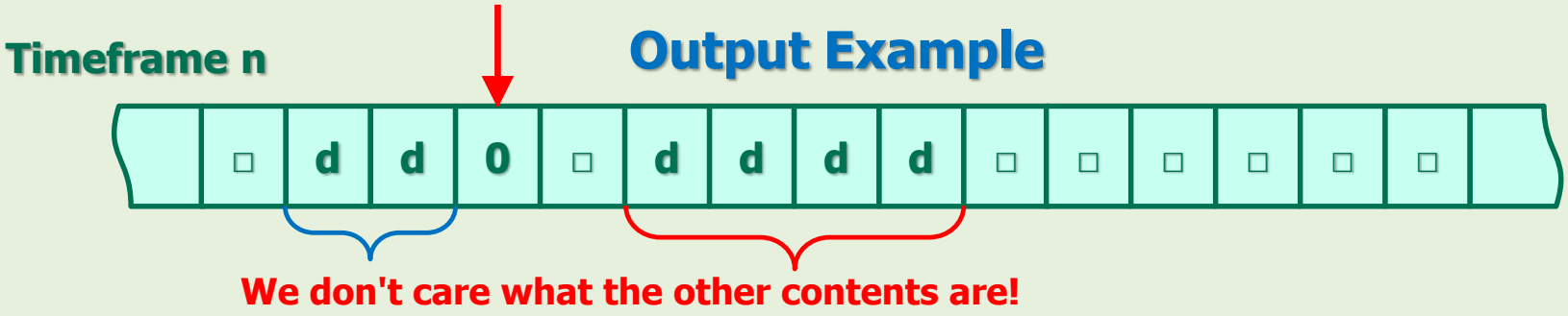


TM as Transducer

Example 2: When x < y

$$f(x, y) = \begin{cases} x + y & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

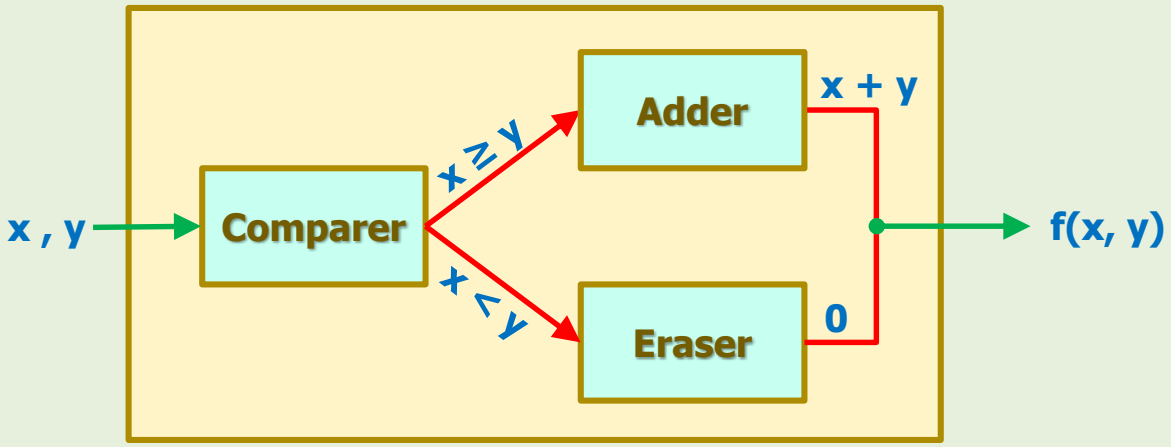




Combining TMs

Example 2 (cont'd)

- To implement this function, we would need the following routines:
  - a comparer
  - an adder
  - an eraser
- A comparer compares x and y ...
  - If $x \geq y$, then it activates the adder to compute x + y.
  - Else (i.e.: $x < y$), it activates the eraser to show zero.



Combining TMs

Example 2 (cont'd)

- This example shows that we can break any complex problem into simpler routines and implement each routine with a TM.

- Do the rest of this example as homework!

Homework

1. Given two unary numbers x and y, separated by a 0 (zero).
   Design a TM to compute x * y.
2. Given a binary number x.
   Design a TM to compute x + 1. (Binary Incrementor)
3. Use your binary incrementor as a subroutine to design
   a Unary-to-Binary converter.
4. Compare n binary numbers as $b_1 < b_2 < \ldots < b_n$ and make sure that they are sorted ascending.

Other Models of TMs

Introduction

- We saw how adding memory could increase the computing power.
- Now the question is ...
- ... to make more powerful machines, can we add more memory, or other kind of memories?
- For example:
  - two or more stacks,
  - a queue and a stack,
  - two or more RAMs
  - or ...

- How far can we go?
- What would be the most powerful automata?

Objective

- We are going to add some extra capabilities to the standard TMs.

- By any changes, we introduce a new class of automata.
- Then, the very first question would be:
  - Is this new class more powerful than the standard TMs?

- We've already saw how we compare the classes of automata by simulation.

TMs with Stay-Option

TMs with Stay-Option

- In standard TMs, the cursor can move left or right.
- Now, we add a new movement, ...

  Stay (or no movement) denoted by S

- Everything else would remain exactly the same as standard TMs.

Formal Definition

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

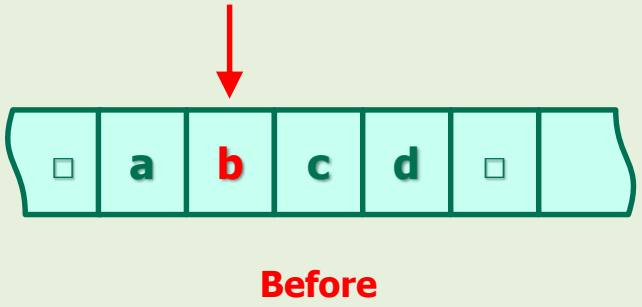
- Where:
  - ... (same as standard TM)
  - δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$

TMs with Stay-Option

Example 3

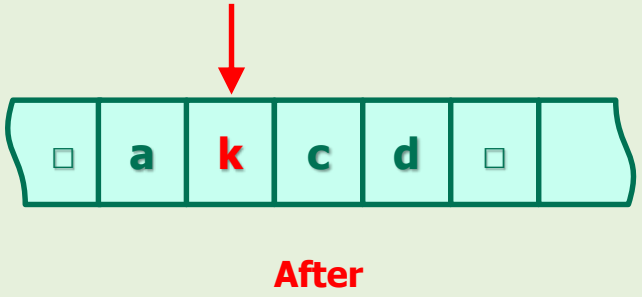
- The following transition is using stay option.

$\delta(q_2, b) = (q_5, k, S)$



Note

- To use this option in JFLAP, you'd need to activate it in the JFLAP preferences.



Is this new class more powerful than standard TMs?

Theorem

- TMs with stay-option class is equivalent to standard TMs class.
- We need to prove two things:
  1. TMs with stay-option simulate standard TMs.
  2. Standard TMs simulate TMs with stay-option.

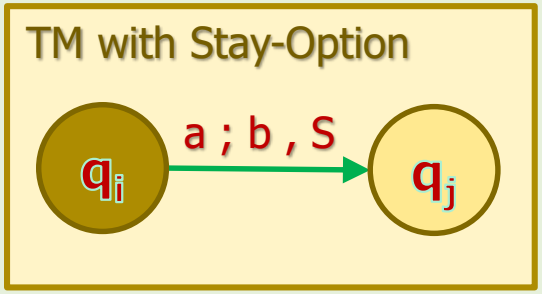
Proof of 1

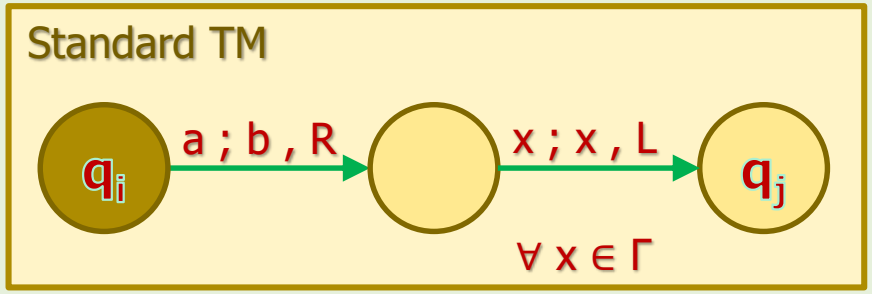
- Let's assume we've constructed a standard TM for an arbitrary language L.
- Can we always construct a TM with stay-option for L?
- Yes, just don't use "S" option!

Is this new class more powerful than standard TMs?

Proof of 2

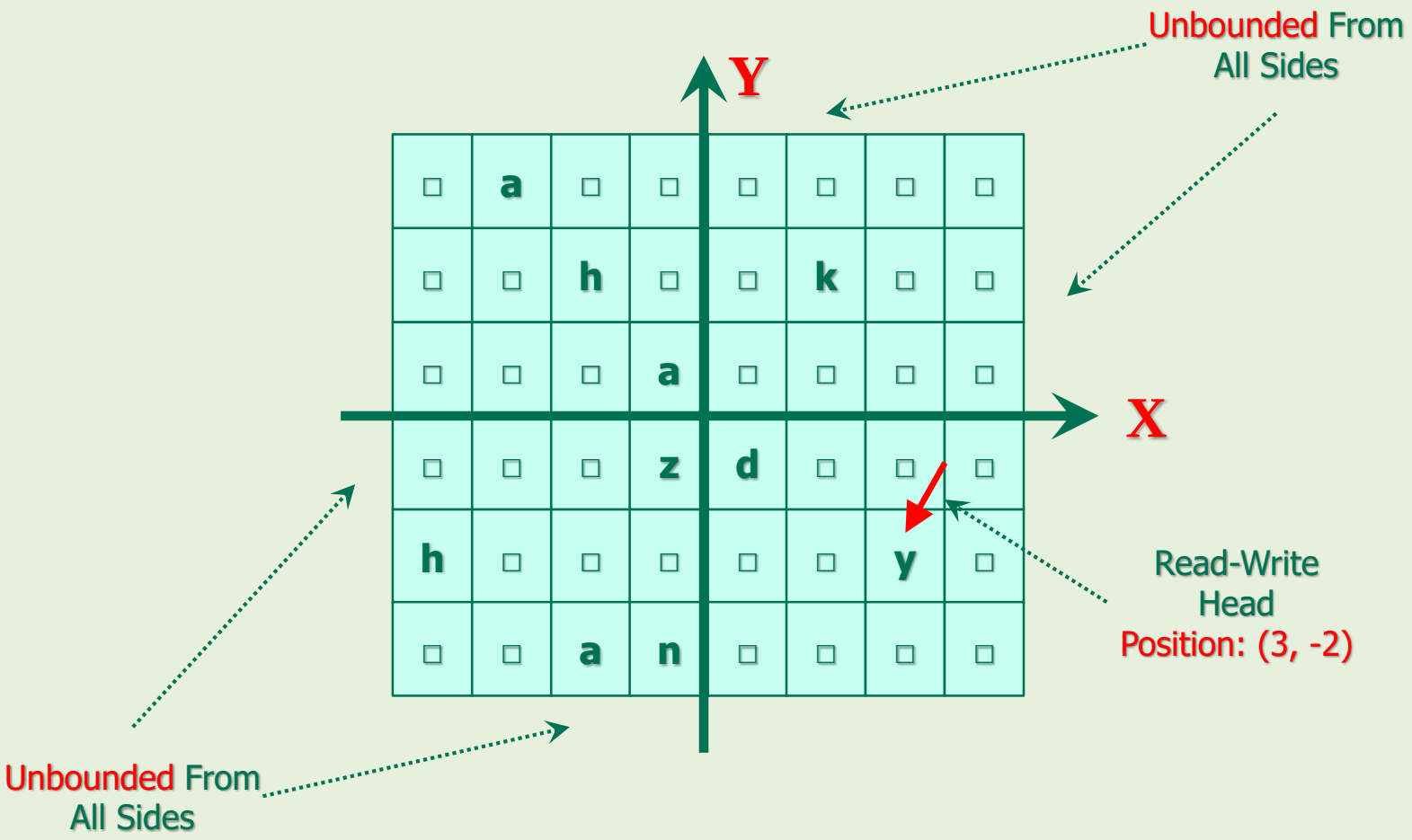
- Let's assume we've constructed a TM with stay-option for an arbitrary language L.
- Can we construct a standard TM for L?
- The only difference is those transitions that use "S" option.
- So, we just need to simulate "no move" for those transitions.
- That can be simulated by two moves of the head: "left, then right" (or "right, then left") as the following figures show.





Multidimensional-Tape TMs

Multidimensional-Tape TMs



Multidimensional-Tape TMs

Formal Definition

- A TM with multidimensional-tape M is defined by the septuple:

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

- Where:
  - ... (same as standard TM elements)

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

- The new movements: U=Up and D=Down

Is this new class more powerful than standard TMs?

Theorem

- The TMs with multidimensional-tape class is equivalent to standard TMs class.
- We need to prove two things:
  1. Multidimensional-tape TMs simulate standard TMs.
  2. Standard TMs simulate multidimensional-tape TMs.

Proof

1. Multidimensional-tape TMs simulate standard TMs.

- This step is trivial because if we just use one row of the tape, then we have standard TM.

2. Standard TMs simulate multidimensional-tape TMs.

- Prove this as exercise!

References

1. Linz, Peter, "An Introduction to Formal Languages and Automata, 5th ed.," Jones & Bartlett Learning, LLC, Canada, 2012
   ISBN: 978-1-4496-1552-9
2. Michael Sipser, "Introduction to the Theory of Computation, 3rd ed.," CENGAGE Learning, United States, 2013
   ISBN-13: 978-1133187790