Universal Turing Machines

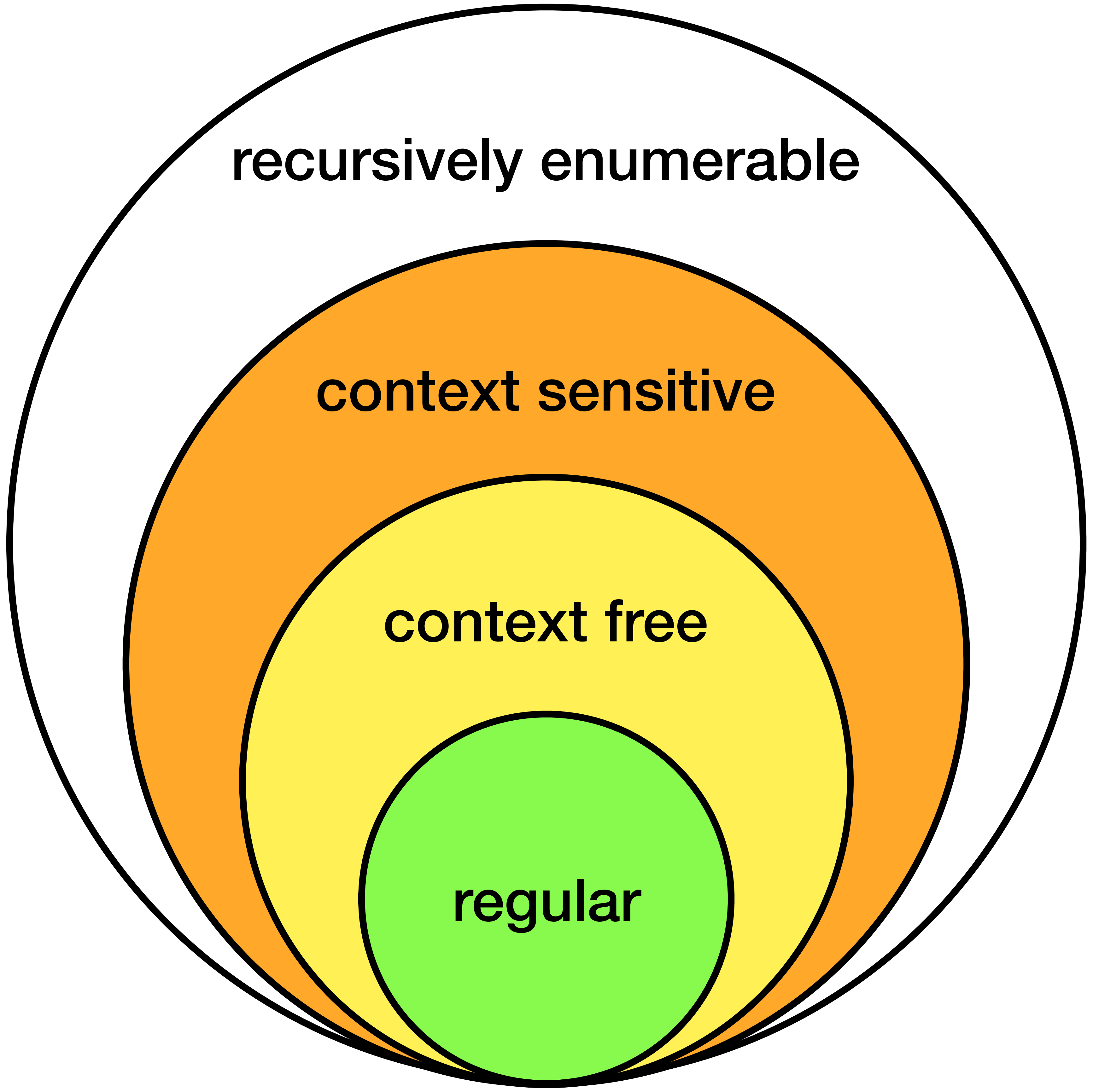
Sides based on material by Kani, Erickson, Chekuri, et. al.

All mistakes are my own! - Ivan Abraham (Fall 2024)

Image by ChatGPT (probably collaborated with DALL-E)

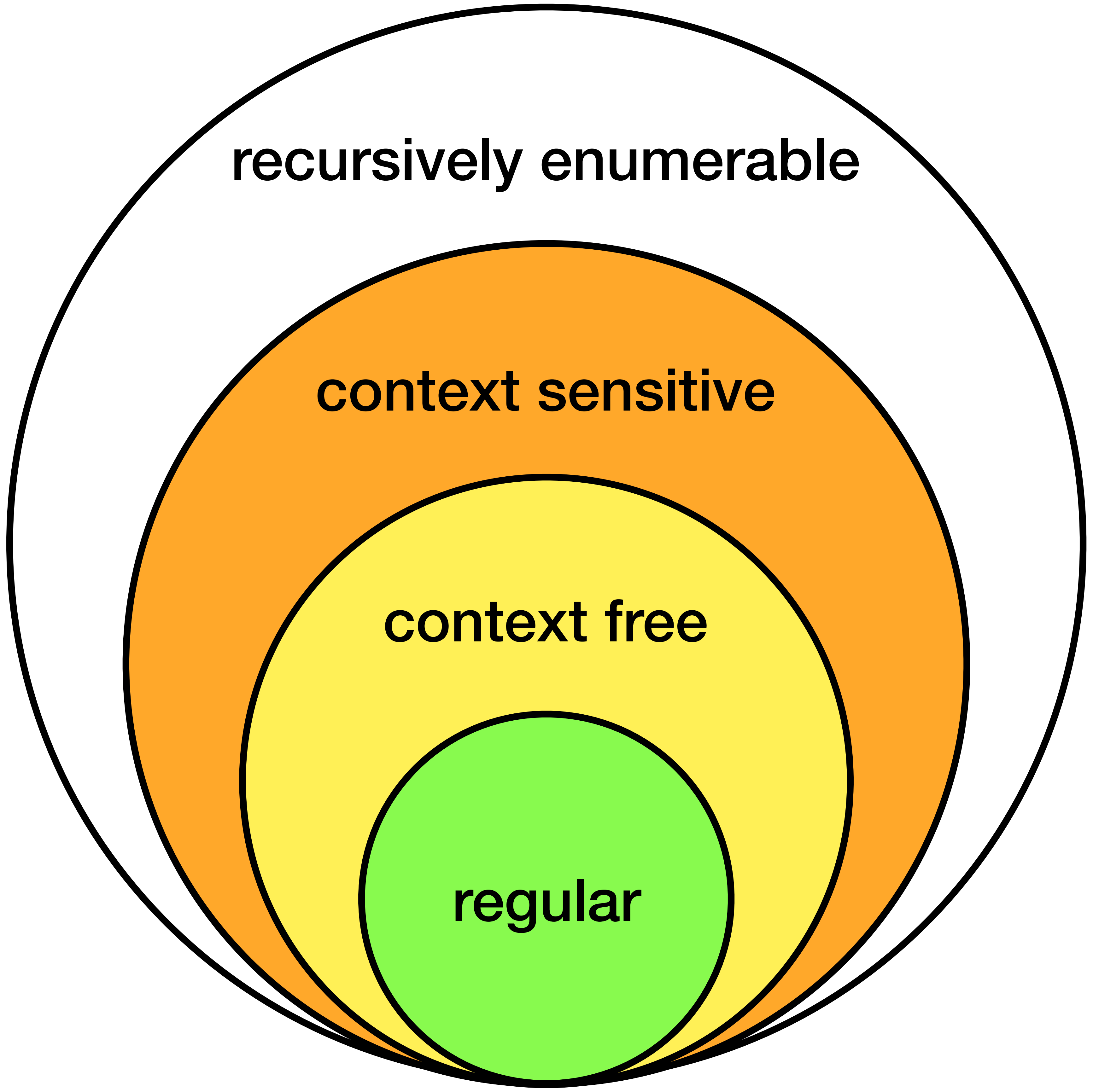
Recap

- We have journeyed in four weeks through different models of computation



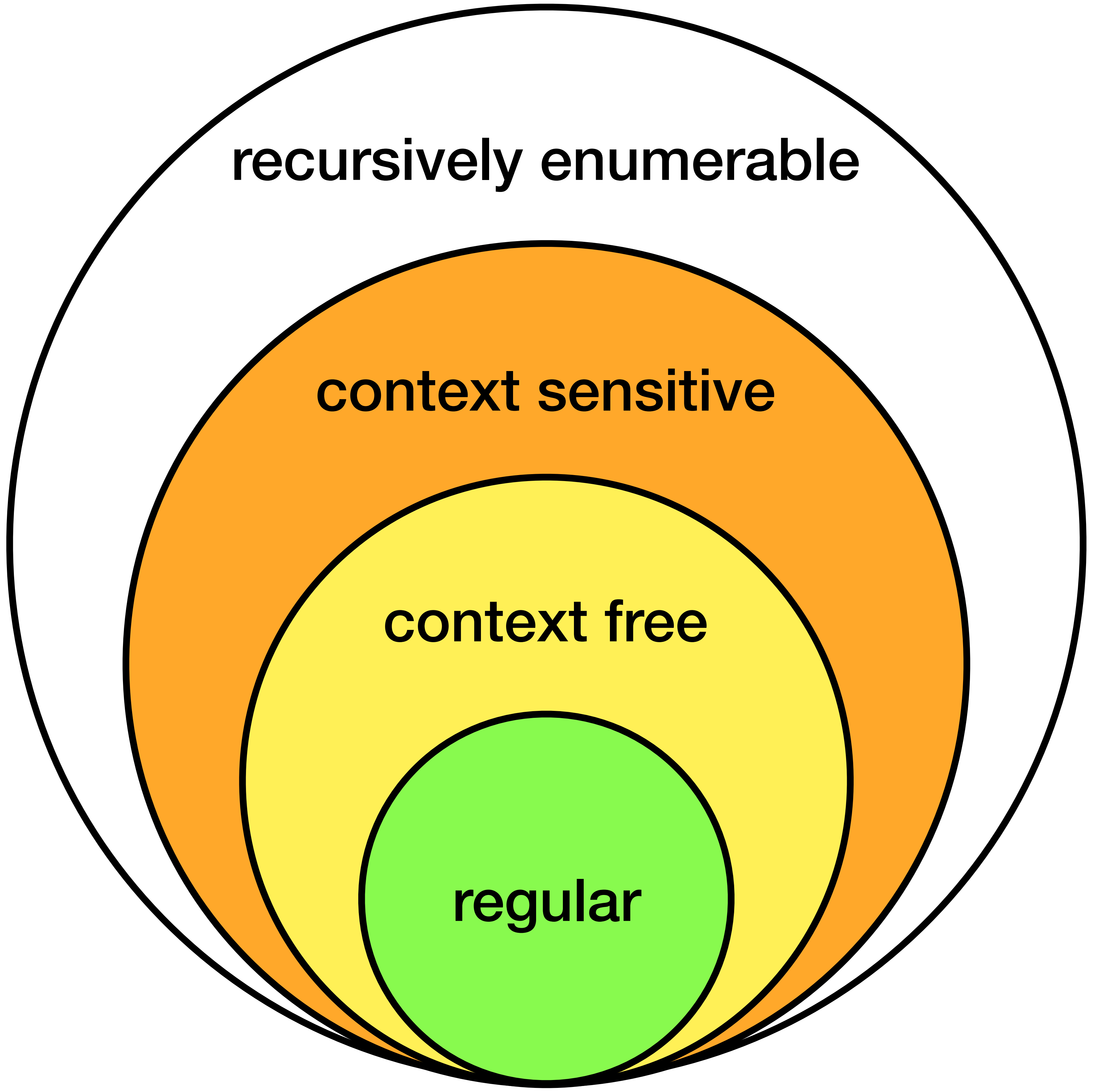
Recap

- We have journeyed in four weeks through different models of computation
- We asked the question last time:



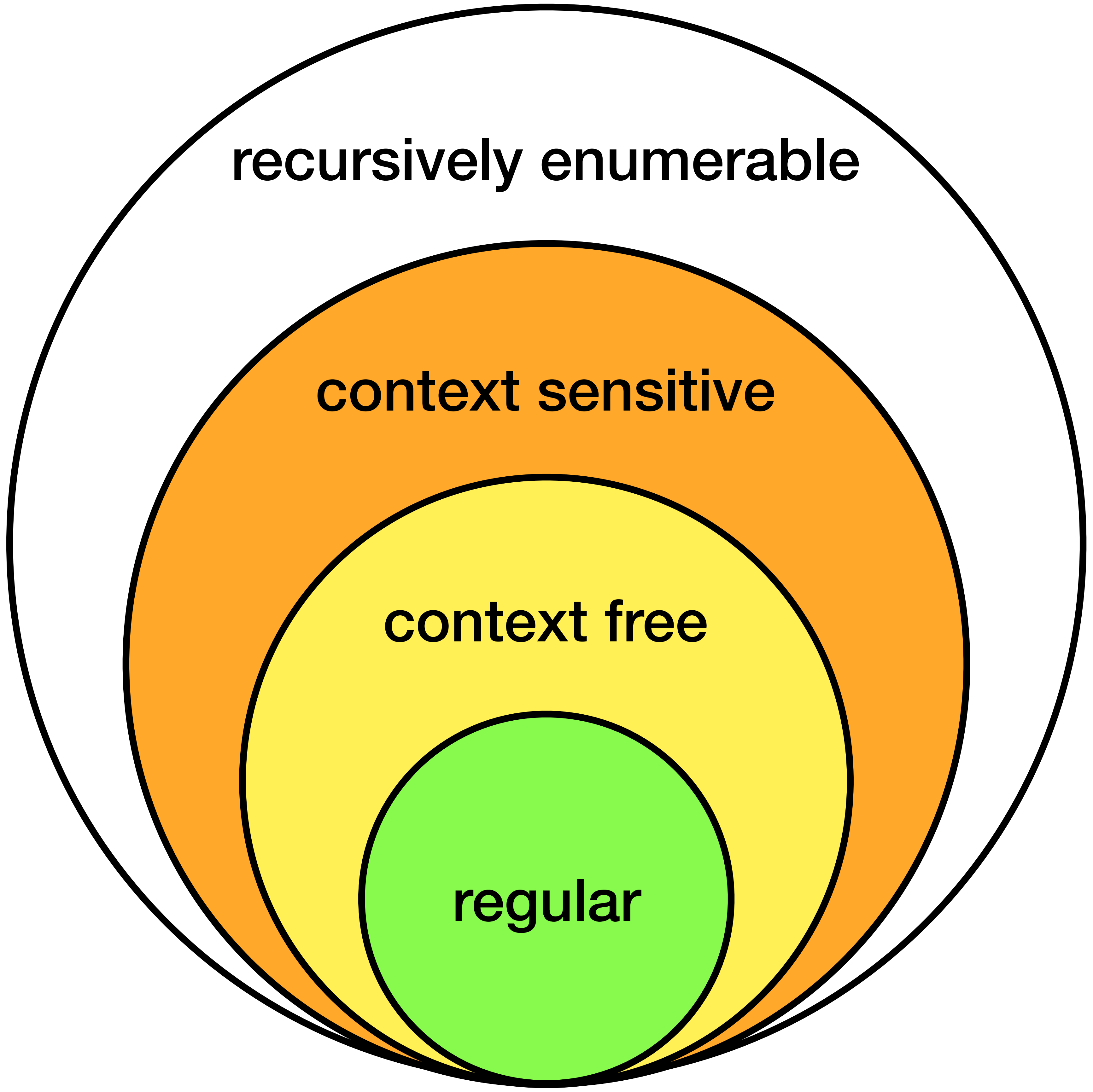
Recap

- We have journeyed in four weeks through different models of computation
- We asked the question last time:
    - What is the most general model of computation we can have, which accepts the largest number of languages.



Recap

- We have journeyed in four weeks through different models of computation
- We asked the question last time:
  - What is the most general model of computation we can have, which accepts the largest number of languages.
- For this we introduced the Turing Machine.



Turing Machine

Recap: High level goals

- Church-Turing Thesis: TMs are the most general computing devices. So far, no “counter-example.”

Turing Machine

Recap: High level goals

- Church-Turing Thesis: TMs are the most general computing devices. So far, no "counter-example."
  - We will look at some possible extensions today and show they amount to the same thing.

Turing Machine

Recap: High level goals

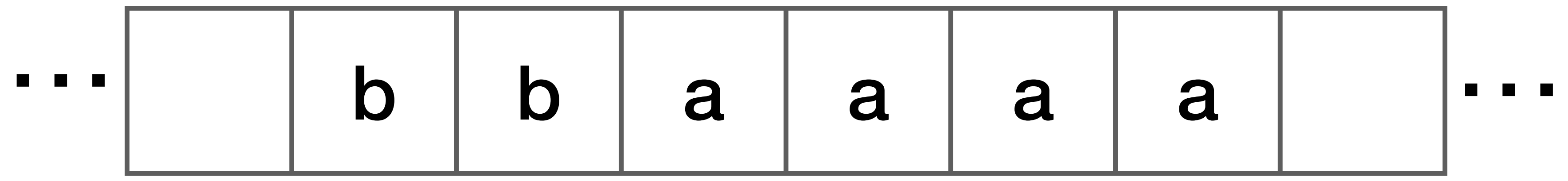
- Church-Turing Thesis: TMs are the most general computing devices. So far, no “counter-example.”
    - We will look at some possible extensions today and show they amount to the same thing.
- Every TM can be represented as a string.

Turing Machine

Recap: High level goals

- Church-Turing Thesis: TMs are the most general computing devices. So far, no "counter-example."
  - We will look at some possible extensions today and show they amount to the same thing.
- Every TM can be represented as a string. → show an example
- The existence of a Universal Turing Machine, which is the model/inspiration for stored program computing. UTM can simulate any TM!

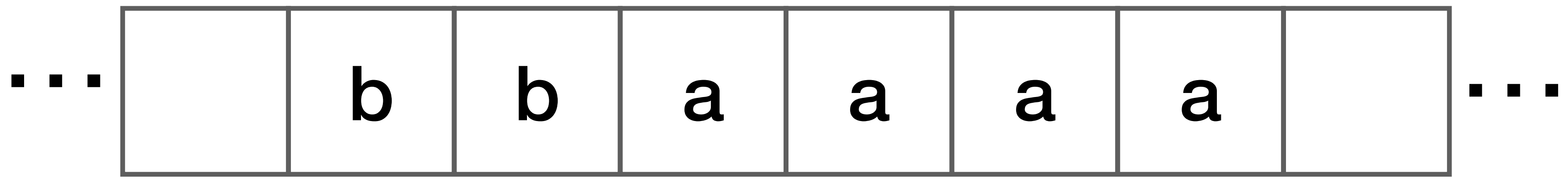
Turing Machine



Input/Output Tape

- Input written on (infinite) one sided tape.

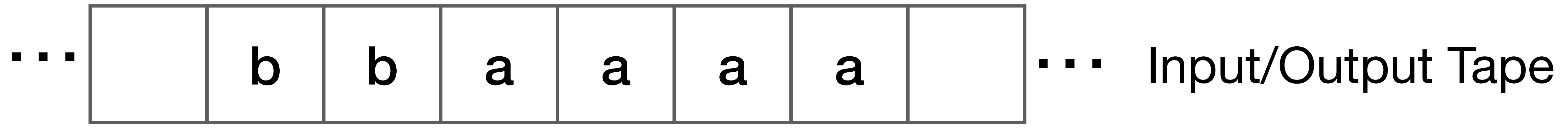
Turing Machine



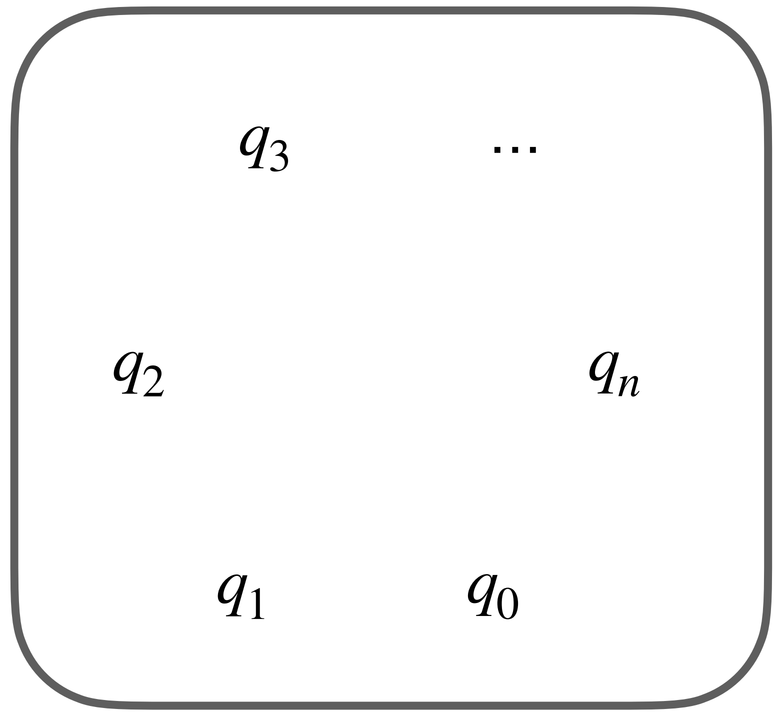
Input/Output Tape

- Input written on (infinite) one sided tape.
- Special blank characters.

Turing Machine

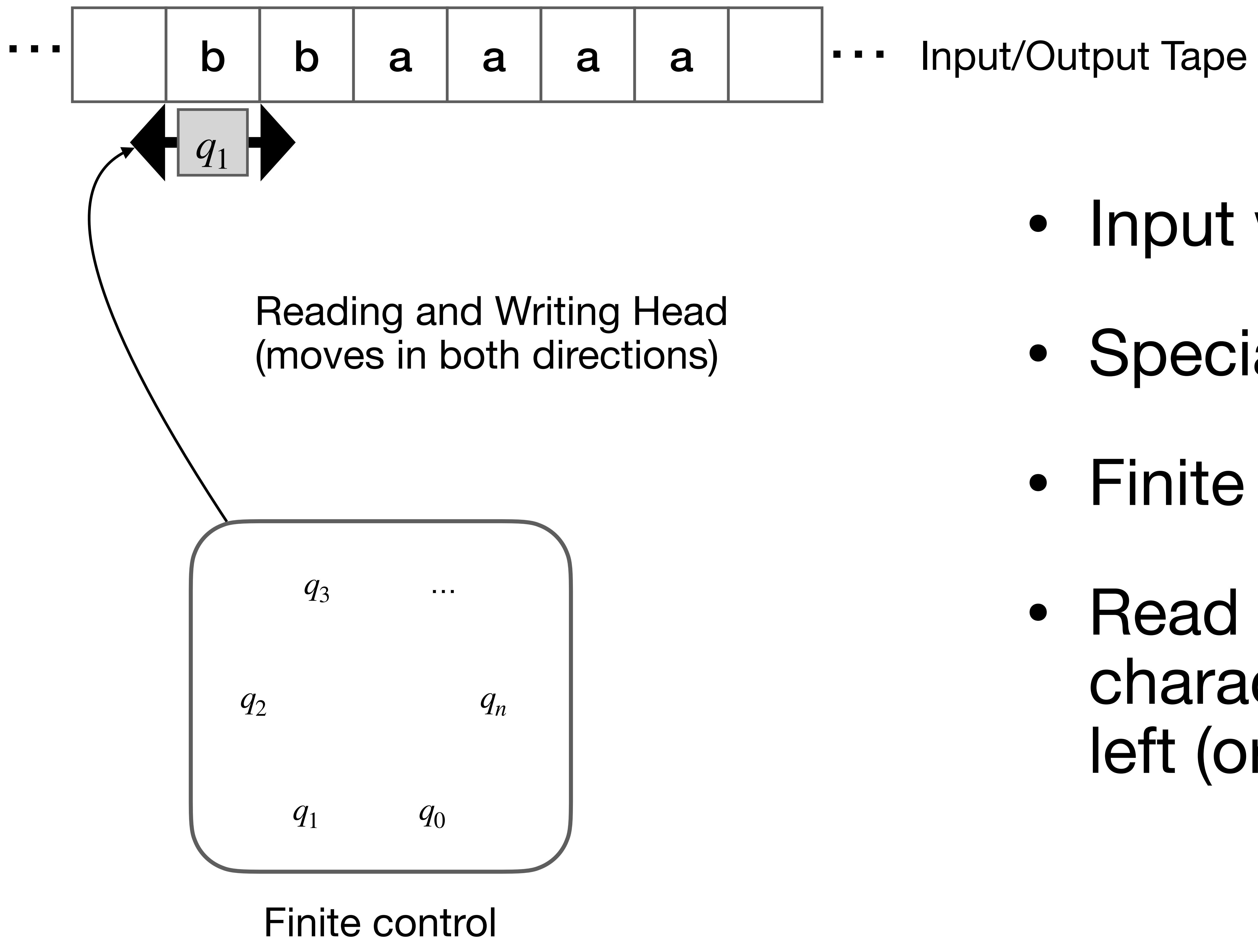


- Input written on (infinite) one sided tape.
- Special blank characters.
- Finite state control (similar to DFA).



Finite control

Turing Machine



- Input written on (infinite) one sided tape.
- Special blank characters.
- Finite state control (similar to DFA).
- Read character under head, write character out, move the head right or left (or stay).

Turing Machine

Transition function

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

From state q, on reading a:

- go to state p
- write b
- move head *Left*

Turing Machine

Transition function

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

current state

symbol scanned

new state

symbol to write

Direction to move on tape

From state q, on reading a:

- go to state p
- write b
- move head *Left*

Turing Machine

Transition function

$$\delta : \underbrace{Q}_{\text{current state}} \times \underbrace{\Gamma}_{\text{symbol scanned}} \to \underbrace{Q}_{\text{new state}} \times \underbrace{\Gamma}_{\text{symbol to write}} \times \underbrace{\{L, R, S\}}_{\text{Direction to move on tape}}$$

From state q, on reading a:

- go to state p
- write b
- move head *Left*

$$\longrightarrow \quad \delta(q, a) = (p, b, L)$$

Turing Machine

Transition function

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

current state

symbol scanned

new state

symbol to write

Direction to move on tape

From state q, on reading a:

- go to state p
- write b
- move head *Left*

$$\delta(q, a) = (p, b, L)$$

- Missing transitions lead to hell state = “Blue screen of death” = “Machine crashes.

Turing Machine variants

Equivalent Turing Machines

Several variations of a Turing machine:

- Standard Turing machine (single infinite tape)
- Multi-track tapes
- Bi-infinite tape
- Multiple heads
- Multiple heads and tapes

Turing machine variants

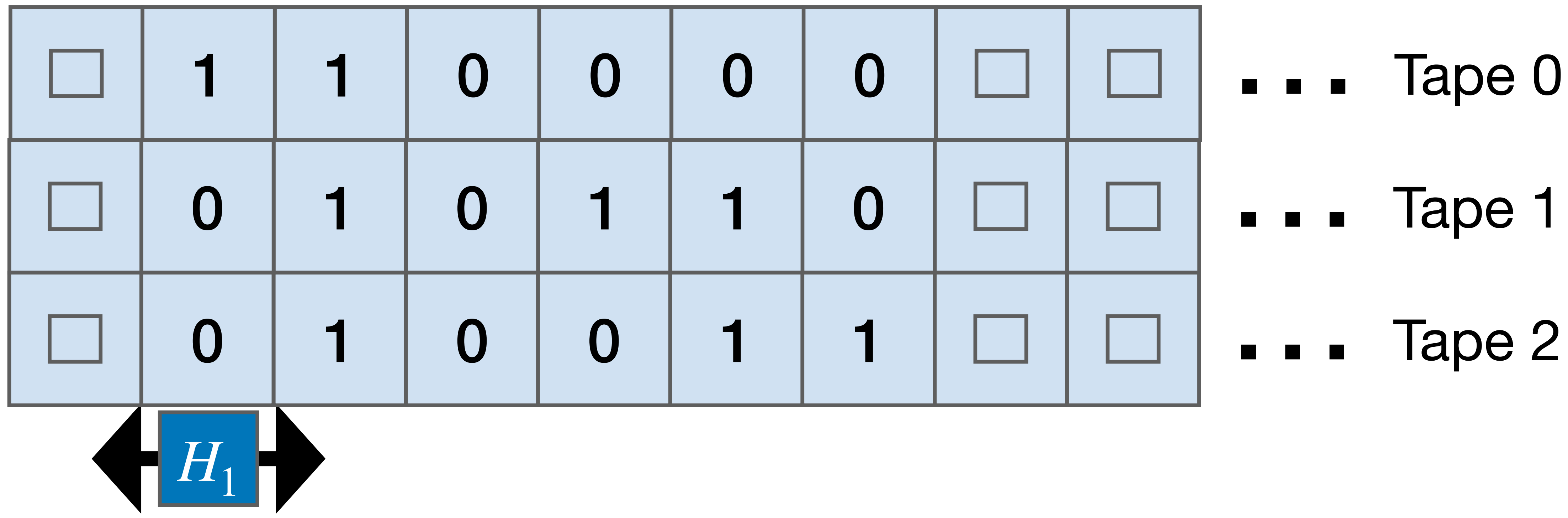
Multi-track Tapes

Suppose we have a TM with multiple tracks:

Turing machine variants

Multi-track Tapes

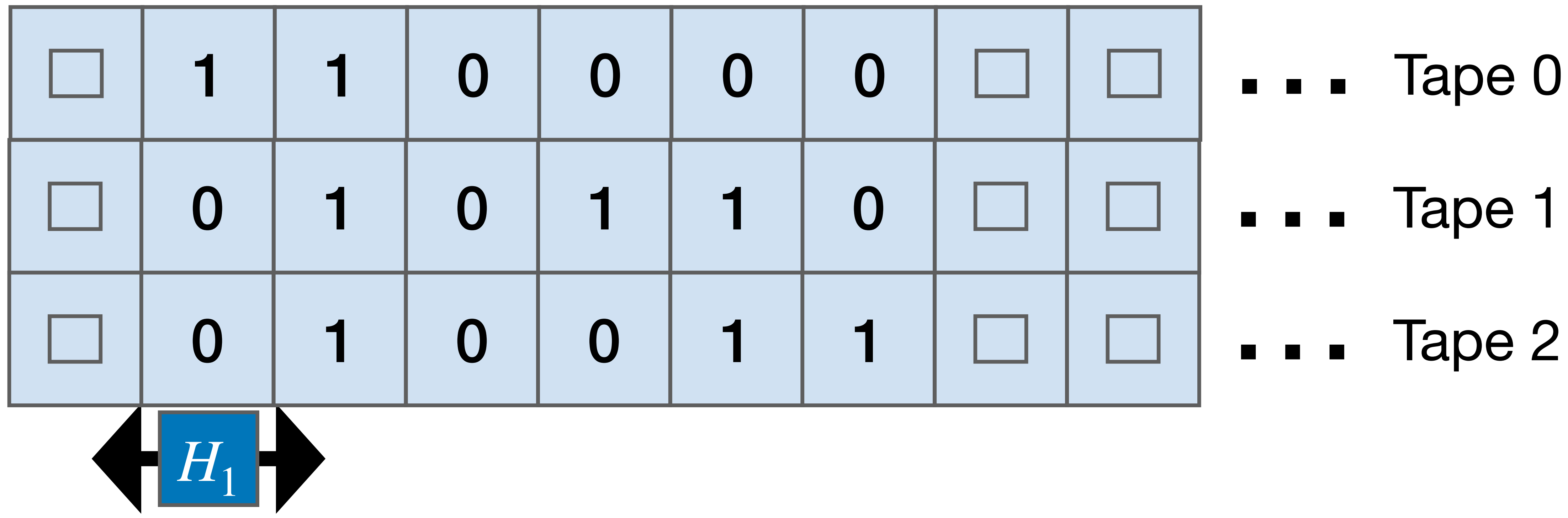
Suppose we have a TM with multiple tracks:



Turing machine variants

Multi-track Tapes

Suppose we have a TM with multiple tracks:



New transition function: $\delta : Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \rightarrow Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \{L, R, S\}$

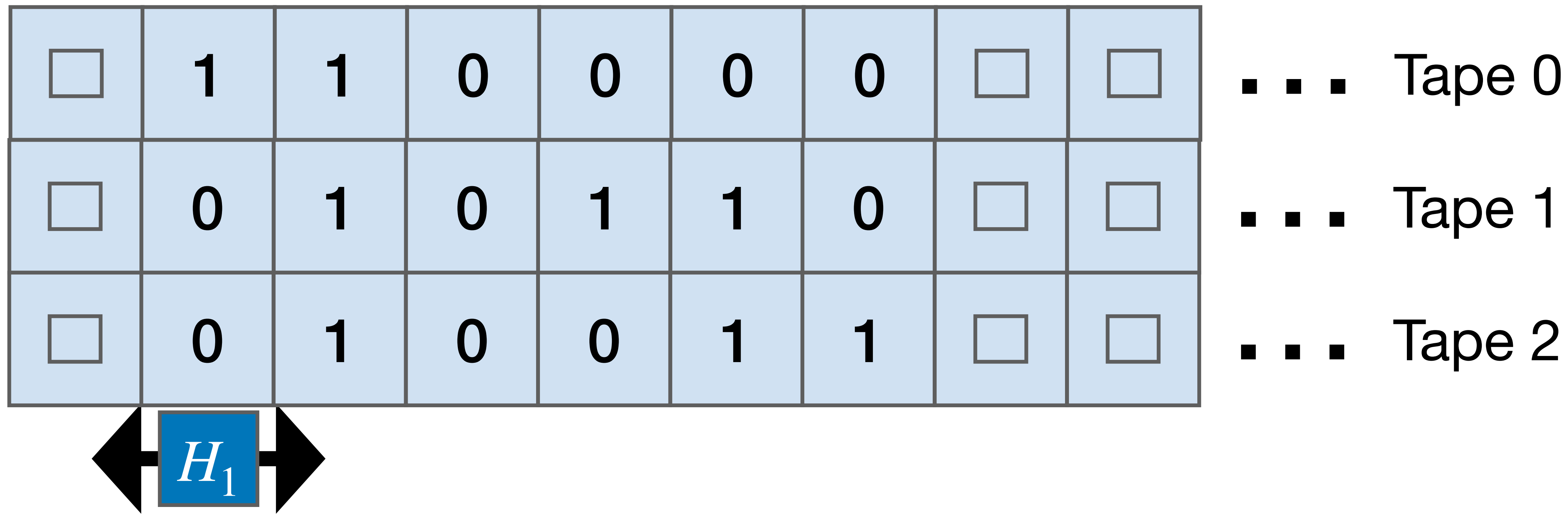
one head

each tape can have its own alphabet

Turing machine variants

Multi-track Tapes

Suppose we have a TM with multiple tracks:



New transition function: $\delta : Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \to Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \{L, R, S\}$

Is there an equivalent single-track TM?

Turing machine variants

Multi-track Tapes

Suppose we have a TM with multiple tracks:

□	1	1	0	0	0	0	□	□	...	Tape 0
□	0	1	0	1	1	0	□	□	...	Tape 1
□	0	1	0	0	1	1	□	□	...	Tape 2

H_1

1 bit / slot
//
//

New transition function: $\delta : Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \rightarrow Q \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \{L, R, S\}$

Is there an equivalent single-track TM?

□	4	7	0	2	3	1	□	□	...	Input/ Output Tape

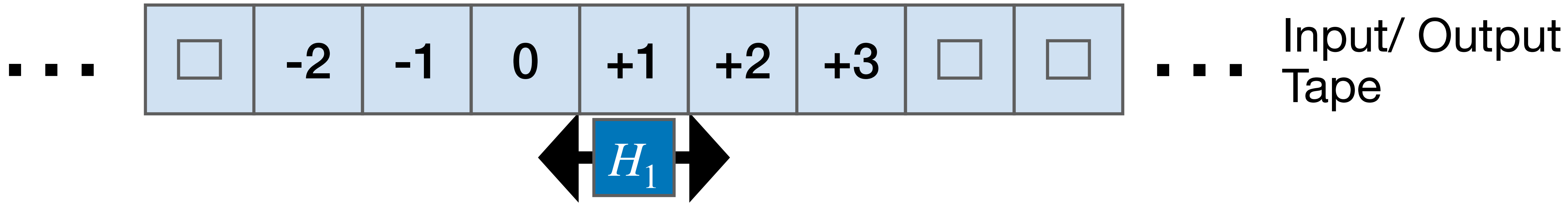
H_1

3 bits / slot

Turing machine variants

Infinite Bi-directional Tape

Suppose we have a TM with tape that is bi-infinite:

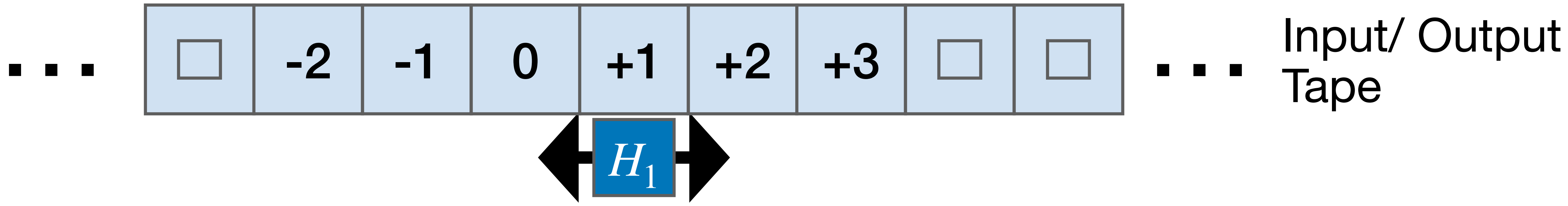


Is there an equivalent one-sided TM?

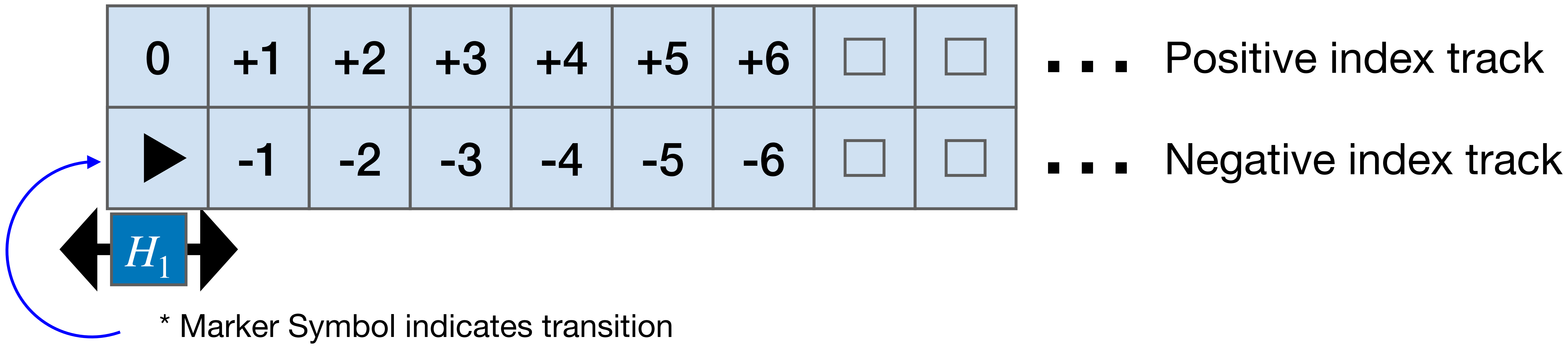
Turing machine variants

Infinite Bi-directional Tape

Suppose we have a TM with tape that is bi-infinite:



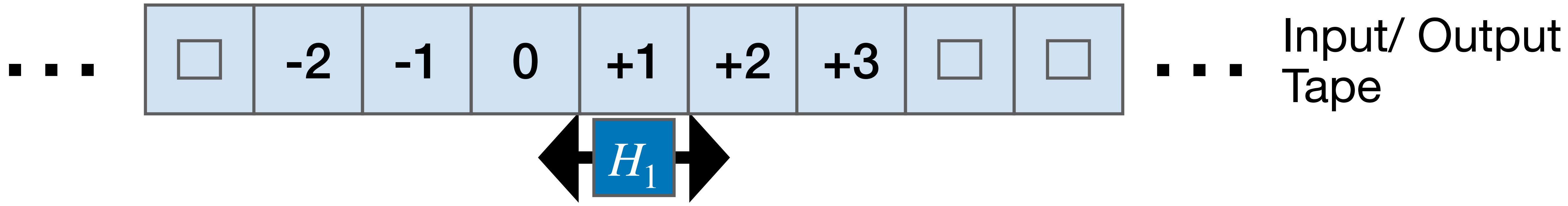
Is there an equivalent one-sided TM?



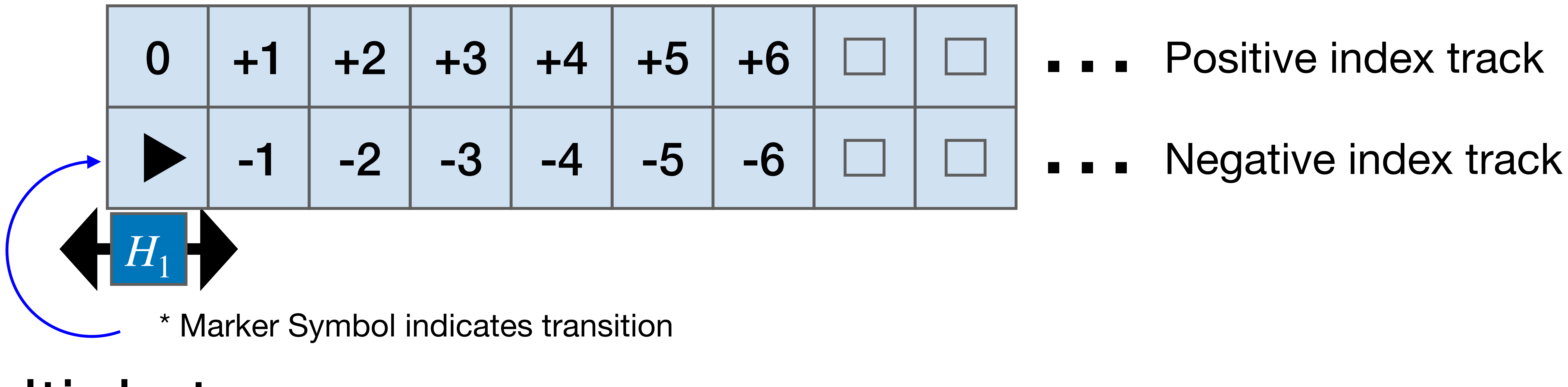
Turing machine variants

Infinite Bi-directional Tape

Suppose we have a TM with tape that is bi-infinite:



Is there an equivalent one-sided TM?

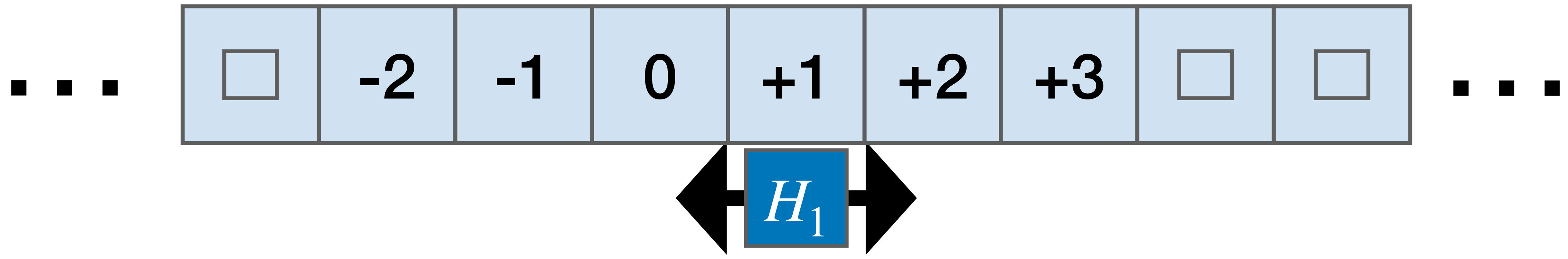


Can model as multiple tapes.

Turing machine variants

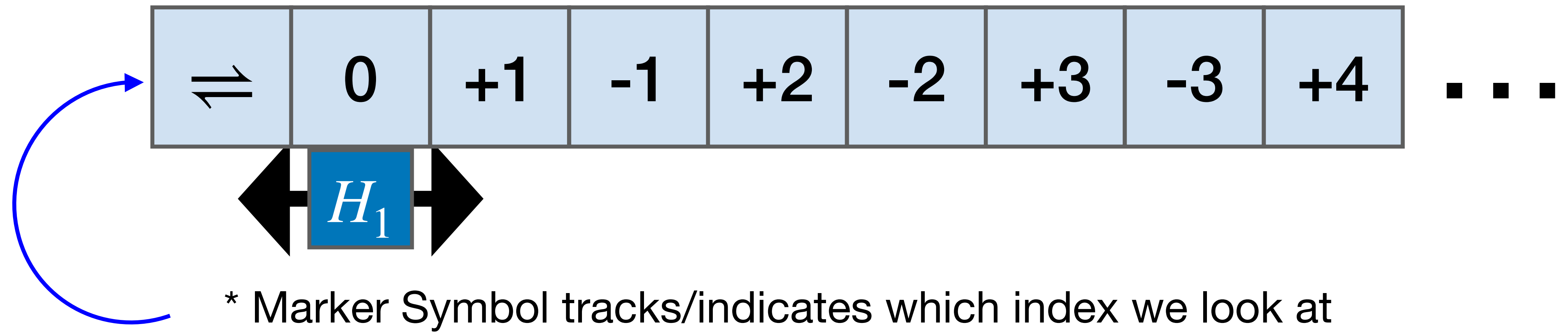
Infinite Bi-directional Tape

Suppose we have a TM with tape that is bi-infinite:



Input/ Output Tape

Is there an equivalent one-sided TM?



Input/ Output Tape

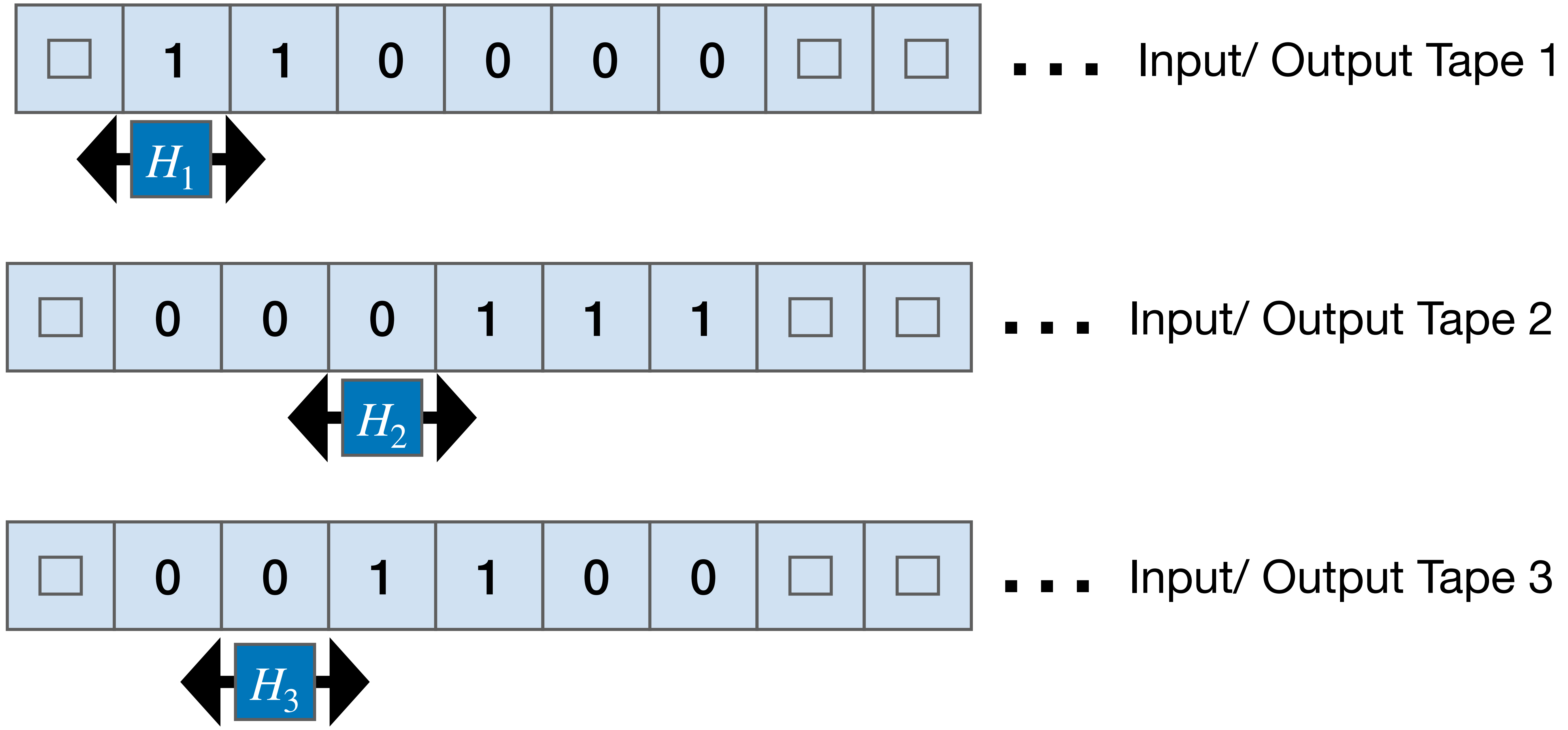
* Marker Symbol tracks/indicates which index we look at

Or as single tape interleaved with positive and negative indexes.

Turing machine variants

Multiple Read/Write Heads

What is the transition function for a TM with multiple heads and multiple tapes:



$$\delta : \mathbb{Q} \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \to \mathbb{Q} \times \Gamma_1 \times \Gamma_2 \times \Gamma_3 \times \{L, R\}^3$$

$\mathbb{Q} = Q_1 \times Q_2 \times Q_3$

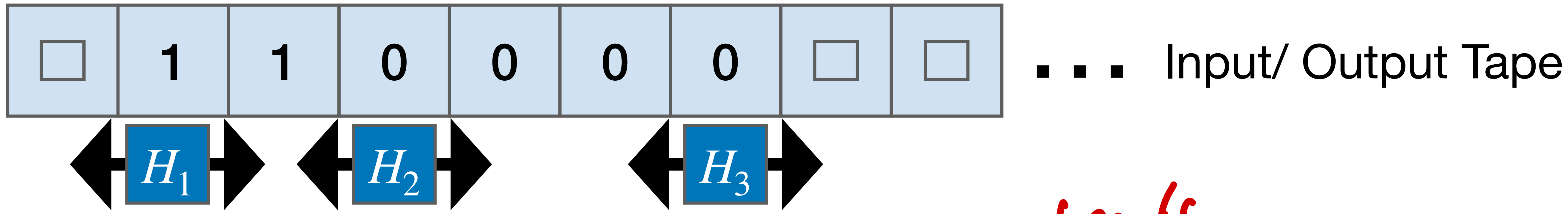
↳ can have different alphabet

↳ each head can move independently

Turing machine variants

Multiple Read/Write Heads

Suppose we have a TM with multiple heads:



heads can still move independently

$$\delta : Q \times \Gamma^3 \to Q \times \left(\Gamma \times \{L, R\}\right)^3$$

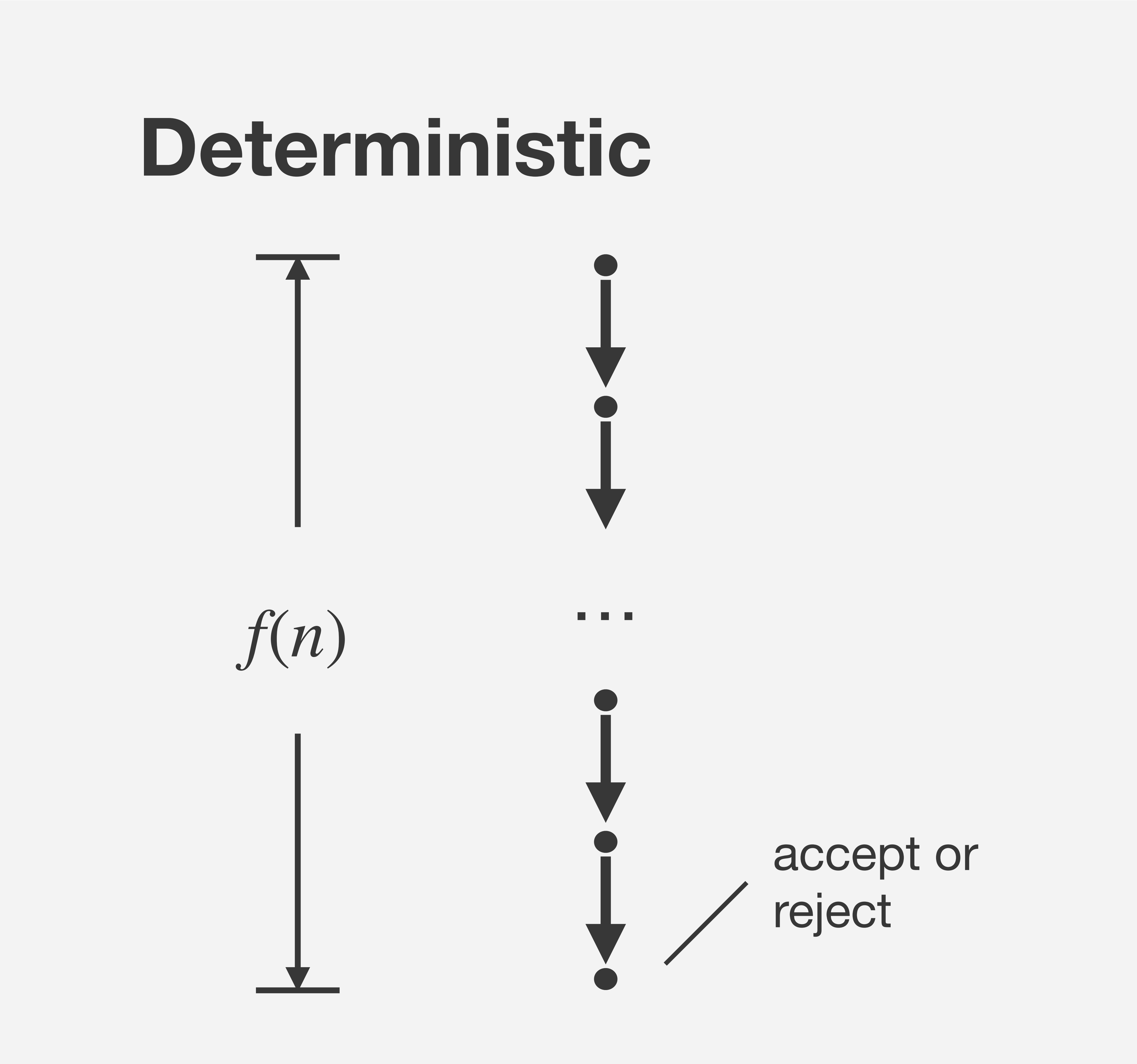
same alphabet

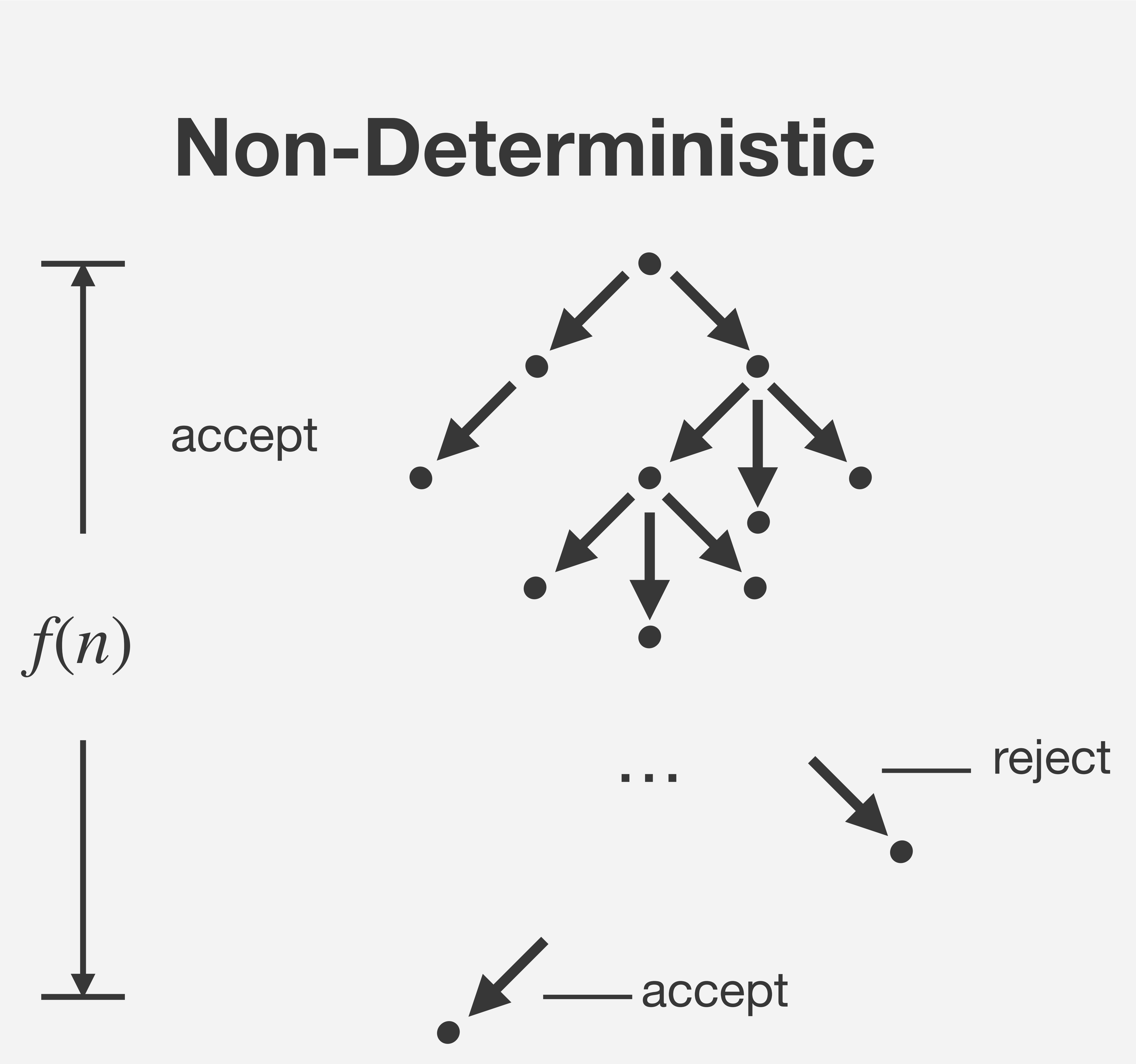
What does the transition function for the equivalent nominal TM look like?

Determinism in Turing Machines

Determinism in Turing Machines

Deterministic vs Non-Deterministic





Power of NTM vs. DTM?

A DTM can simulate a NTM in the following ways:

- **Multiplicity of configuration of states**
    1. Have the store multiple configurations of the NTM.

Power of NTM vs. DTM?

A DTM can simulate a NTM in the following ways:

- **Multiplicity of configuration of states**
    1. Have the store multiple configurations of the NTM.
    2. At every timestep, process each configuration. Add configurations to the set if multiple paths exist.

Power of NTM vs. DTM?

A DTM can simulate a NTM in the following ways:

- **Multiplicity of configuration of states**
    1. Have the store multiple configurations of the NTM.
    2. At every timestep, process each configuration. Add configurations to the set if multiple paths exist.
- **Multiple Tapes** - Can simulate NTM with 3-tape DTM:

Power of NTM vs. DTM?

A DTM can simulate a NTM in the following ways:

- **Multiplicity of configuration of states**
    1. Have the store multiple configurations of the NTM.
    2. At every timestep, process each configuration. Add configurations to the set if multiple paths exist.
- **Multiple Tapes** - Can simulate NTM with 3-tape DTM:
    1. First tape holds original input

Power of NTM vs. DTM?

A DTM can simulate a NTM in the following ways:

- **Multiplicity of configuration of states**
    1. Have the store multiple configurations of the NTM.
    2. At every timestep, process each configuration. Add configurations to the set if multiple paths exist.
- **Multiple Tapes** - Can simulate NTM with 3-tape DTM:
    1. First tape holds original input
    2. Second used to simulate a particular computation of NTM

Power of NTM vs. DTM?

A DTM can simulate a NTM in the following ways:

- **Multiplicity of configuration of states**
    1. Have the store multiple configurations of the NTM.
    2. At every timestep, process each configuration. Add configurations to the set if multiple paths exist.
- **Multiple Tapes** - Can simulate NTM with 3-tape DTM:
    1. First tape holds original input
    2. Second used to simulate a particular computation of NTM
    3. Third tape encodes path in NTM computation tree.

Universal Turing Machine

Introduction

A single Turing Machine M_u that can compute anything computable!

Universal Turing Machine

Introduction

A single Turing Machine M_u that can compute anything computable!

Takes as input:

Universal Turing Machine

Introduction

A single Turing Machine M_u that can compute anything computable!

Takes as input:

- the description of some other TM M

Universal Turing Machine

Introduction

A single Turing Machine M_u that can compute anything computable!

Takes as input:

- the description of some other TM M
- data w for M to run on

Universal Turing Machine

Introduction

A single Turing Machine M_u that can compute anything computable!

Takes as input:

- the description of some other TM M
- data w for M to run on

Outputs:

Universal Turing Machine
Introduction

A single Turing Machine M_u that can compute anything computable!

Takes as input:

- the description of some other TM M
- data w for M to run on

Outputs:

- results of running $M(w)$

Universal Turing Machine

Some notation

M: Turing machine

$\langle M \rangle$: a string uniquely describing M (we will see that it can be thought of as a number)

w: An input string.

$\langle M, w \rangle$: A unique string encoding both M and input w.

$$L(M_u) = \{\langle M, w \rangle \ M \text{ is a TM and } M \text{ accepts } w\}$$

Universal Turing Machine

Introduction

We want to construct a Turing machine such that:

language of the universal Turing machine

$$L\left(M_u\right) = \{\langle M, w\rangle \ \mid M \text{ accepts } w\}$$

M_u is a stored-program computer. It reads $\langle M, w\rangle$, parses it as a program M and data w, and executes M on data w.

Universal Turing Machine

Introduction

We want to construct a Turing machine such that:

$$L\left(M_u\right) = \{\langle M, w\rangle \ \mid M \text{ accepts } w\}$$

M_u is a stored-program computer. It reads $\langle M, w\rangle$, parses it as a program M and data w, and executes M on data w.

In other words, M_u simulates the run of M on w.

Universal Turing Machine
Coding of TMs

Lemma: If a language L over alphabet $\{0,1\}$ is accepted by some $TM\ M$, then there is a one-tape *TM* M' that accepts L, such that

Universal Turing Machine

Coding of TMs

Lemma: If a language L over alphabet $\{0,1\}$ is accepted by some $TM\ M$, then there is a one-tape *TM* M' that accepts L, such that

- $\Gamma = \{0,1,B\}$

Universal Turing Machine

Coding of TMs

Lemma: If a language L over alphabet $\{0,1\}$ is accepted by some $TM\ M$, then there is a one-tape *TM* M' that accepts L, such that

- $\Gamma = \{0,1,B\}$
- states numbered $1,\ldots,k$

Universal Turing Machine

Coding of TMs

Lemma: If a language L over alphabet $\{0,1\}$ is accepted by some $TM\ M$, then there is a one-tape *TM* M' that accepts L, such that

- $\Gamma = \{0,1,B\}$
- states numbered $1,...,k$
- q_1 is a unique start state

Universal Turing Machine

Coding of TMs

Lemma: If a language L over alphabet $\{0,1\}$ is accepted by some $TM\ M$, then there is a one-tape *TM* M' that accepts L, such that

- $\Gamma = \{0,1,B\}$
- states numbered $1,...,k$
- q_1 is a unique start state
- q_2 is a unique halt/accept state

Universal Turing Machine

Coding of TMs

Lemma: If a language L over alphabet $\{0,1\}$ is accepted by some *TM* M, then there is a one-tape *TM* M' that accepts L, such that

- $\Gamma = \{0,1,B\}$ → augment tape alphabet w/ blank
- states numbered $1,\ldots,k$ → k can be large
- q_1 is a unique start state
- q_2 is a unique halt/accept state
- q_3 is a unique halt/reject state

can think of it as a "normalization"

typos fixed

Universal Turing Machine

Coding of TMs

Lemma: If a language L over alphabet $\{0,1\}$ is accepted by some *TM* M, then there is a one-tape *TM* M' that accepts L, such that

- $\Gamma = \{0,1,B\}$
- states numbered $1,\ldots,k$
- q_1 is a unique start state
- q_2 is a unique halt/accept state
- q_3 is a unique halt/reject state

can be inferred

Note: To represent a TM, we need only list its set of transitions - everything else is implicit by the above.

Listing Transition

- $\Gamma = \{0,1,B\}$
- states numbered $1,\ldots,k$
- q_1 is a unique start state
- q_2 is a unique halt/accept state
- q_3 is a unique halt/reject state

- Use the following order:

$$\delta\left(q_1,0\right), \delta\left(q_1,1\right), \delta\left(q_1,B\right), \delta\left(q_2,0\right), \delta\left(q_2,1\right), \delta\left(q_2,B\right), \ldots$$

Listing Transition

- $\Gamma = \{0,1,B\}$
- states numbered $1,\ldots,k$
- q_1 is a unique start state
- q_2 is a unique halt/accept state
- q_3 is a unique halt/reject state

- Use the following order:

$$\delta\left(q_1,0\right), \delta\left(q_1,1\right), \delta\left(q_1, B\right), \delta\left(q_2,0\right), \delta\left(q_2,1\right), \delta\left(q_2, B\right), \ldots$$

$$\ldots \delta\left(q_k,0\right), \delta\left(q_k,1\right), \delta\left(q_k, B\right)$$

Listing Transition

- $\Gamma = \{0,1,B\}$
- states numbered $1,...,k$
- q_1 is a unique start state
- q_2 is a unique halt/accept state
- q_3 is a unique halt/reject state

- Use the following order:

$$\delta\left(q_1,0\right), \delta\left(q_1,1\right), \delta\left(q_1,B\right), \delta\left(q_2,0\right), \delta\left(q_2,1\right), \delta\left(q_2,B\right), \ldots$$

$$\ldots\delta\left(q_k,0\right), \delta\left(q_k,1\right), \delta\left(q_k,B\right)$$

- Use the following encoding:

Listing Transition

- $\Gamma = \{0,1,B\}$
- states numbered $1,\ldots,k$
- q_1 is a unique start state
- q_2 is a unique halt/accept state
- q_3 is a unique halt/reject state

- Use the following order:

$$\delta\left(q_1,0\right), \delta\left(q_1,1\right), \delta\left(q_1, B\right), \delta\left(q_2,0\right), \delta\left(q_2,1\right), \delta\left(q_2, B\right), \ldots$$

$$\ldots\delta\left(q_k,0\right), \delta\left(q_k,1\right), \delta\left(q_k, B\right)$$

- Use the following encoding:

$$111\ t_1\ 11\ t_2\ 11\ t_3\ 11\ \ldots\ 11\ t_k\ 111$$

triple ones mark beginning and ending of TM encoding

Listing Transition

- $\Gamma = \{0,1,B\}$
- states numbered $1,\ldots,k$
- q_1 is a unique start state
- q_2 is a unique halt/accept state
- q_3 is a unique halt/reject state

- Use the following order:

$$\delta\left(q_1,0\right),\delta\left(q_1,1\right),\delta\left(q_1,B\right),\delta\left(q_2,0\right),\delta\left(q_2,1\right),\delta\left(q_2,B\right),\ldots$$

$$\ldots\delta\left(q_k,0\right),\delta\left(q_k,1\right),\delta\left(q_k,B\right)$$

- Use the following encoding:

double ones demarcate transition encodings

$$111\ t_1\ 11\ t_2\ 11\ t_3\ 11\ \ldots\ 11\ t_k\ 111$$

where t_i is the encoding of transition i as given on the next slide.

Encoding a transition

Recall transition looks like $\delta(q, a) = (p, b, L)$. So, encode as

$$< \text{state} > 1 < \text{input} > 1 < \text{new state} > 1 < \text{new-symbol} > 1 < \text{direction} >$$

Encoding a transition

Recall transition looks like $\delta(q, a) = (p, b, L)$. So, encode as

$$< \text{state} > 1 < \text{input} > 1 < \text{new state} > 1 < \text{new-symbol} > 1 < \text{direction} >$$

where

Encoding a transition

Recall transition looks like $\delta(q, a) = (p, b, L)$. So, encode as

$$< \text{state} > 1 < \text{input} > 1 < \text{new state} > 1 < \text{new-symbol} > 1 < \text{direction} >$$

where

- state q_i represented by 0^i

Encoding a transition

Recall transition looks like $\delta(q, a) = (p, b, L)$. So, encode as

$$< \text{state} > 1 < \text{input} > 1 < \text{new state} > 1 < \text{new-symbol} > 1 < \text{direction} >$$

where

- state q_i represented by 0^i
- $0,1,B$ represented by $0,\ 00,\ 000$

Encoding a transition

Recall transition looks like $\delta(q, a) = (p, b, L)$. So, encode as

$$< \text{state} > 1 < \text{input} > 1 < \text{new state} > 1 < \text{new-symbol} > 1 < \text{direction} >$$

where

- state q_i represented by 0^i
- $0,1,B$ represented by $0,\ 00,\ 000$
- L, R, S represented by $0,\ 00,\ 000$

Encoding a transition

Recall transition looks like $\delta(q, a) = (p, b, L)$. So, encode as

$$< \text{state} > 1 < \text{input} > 1 < \text{new state} > 1 < \text{new-symbol} > 1 < \text{direction} >$$

where

- state q_i represented by 0^i
- $0,1,B$ represented by $0,\ 00,\ 000$ respectively
- L, R, S represented by $0,\ 00,\ 000$ respectively

$$\delta\left(q_3,1\right) = \left(q_4,0,R\right) \text{ represented by } \underbrace{000}_{q_3}\ 1\ \underbrace{00}_{1}\ 1\ \underbrace{0000}_{q_4}\ 1\ \underbrace{0}_{0}\ 1\ \underbrace{00}_{R}$$

Example

Typical TM code:

$$111010100001001001101001000000101011 \ldots\ldots 11 \ldots\ldots\ldots 11 \ldots\ldots\ldots 111$$

Example

Typical TM code:

$1110101000010010011010010000010101 1 \ldots\ldots 11 \ldots\ldots\ldots 11 \ldots\ldots\ldots 111$

- Begins, ends with 111

Example

Typical TM code:

$$111010100001001001101001000001010 11 \ldots\ldots 11 \ldots\ldots\ldots 11 \ldots\ldots\ldots 111$$

- Begins, ends with 111
- Transitions separated by 11

Example

Typical TM code:

$$111010100001001001101001000001010 11 \ldots\ldots 11 \ldots\ldots\ldots 11 \ldots\ldots\ldots 111$$

- Begins, ends with 111
- Transitions separated by 11
- Fields within transition separated by 1

Example

Typical TM code:

$$111010100001001001101001000001010 11 \ldots\ldots 11 \ldots\ldots\ldots 11 \ldots\ldots\ldots 111$$

- Begins, ends with 111
- Transitions separated by 11
- Fields within transition separated by 1
- Individual fields represented by 0s

TMs are (binary) numbers

- Every TM is encoded by a unique element of $\mathbb{N}$

TMs are (binary) numbers

- Every TM is encoded by a unique element of $\mathbb{N}$
- Convention: elements of $\mathbb{N}$ that do not correspond to any TM encoding represent the “null TM” that accepts nothing.

TMs are (binary) numbers

- Every TM is encoded by a unique element of $\mathbb{N}$
- Convention: elements of $\mathbb{N}$ that do not correspond to any TM encoding represent the "null TM" that accepts nothing.
- Thus, every TM is a number, and vice versa

TMs are (binary) numbers

- Every TM is encoded by a unique element of $\mathbb{N}$
- Convention: elements of $\mathbb{N}$ that do not correspond to any TM encoding represent the “null TM” that accepts nothing.
- Thus, every TM is a number, and vice versa
- Let $< M >$ mean the number that encodes M

TMs are (binary) numbers

- Every TM is encoded by a unique element of $\mathbb{N}$
- Convention: elements of $\mathbb{N}$ that do not correspond to any TM encoding represent the "null TM" that accepts nothing.
- Thus, every TM is a number, and vice versa
- Let $< M >$ mean the number that encodes M
- Conversely, let M_n be the TM with encoding n.

How M_u works

Configuration

Three tapes

How M_u works
Configuration

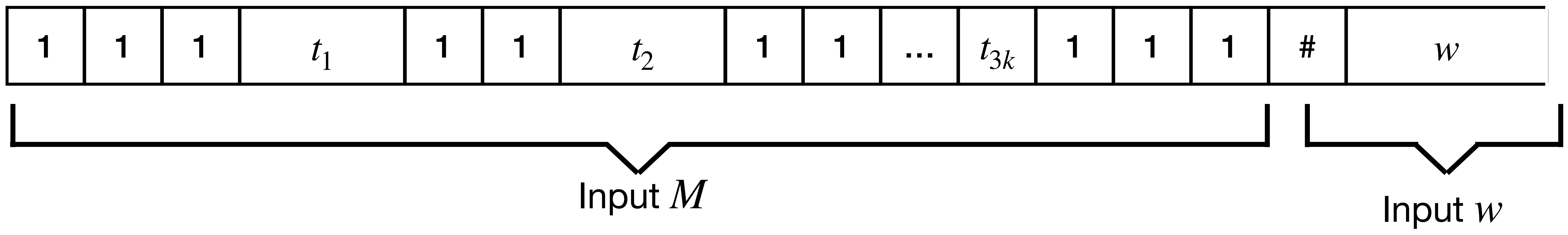
Three tapes

- Tape 1: holds input M and w demarcated with #; never changes

How M_u works
Configuration

Three tapes

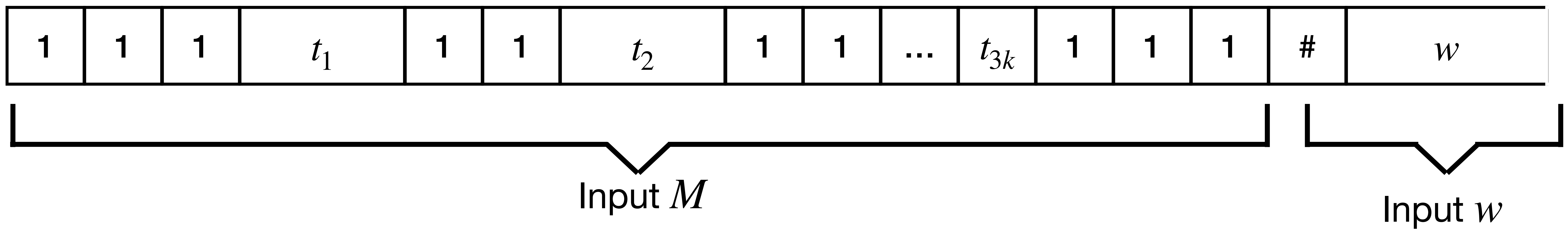
- Tape 1: holds input M and w demarcated with #; never changes



How M_u works
Configuration

Three tapes

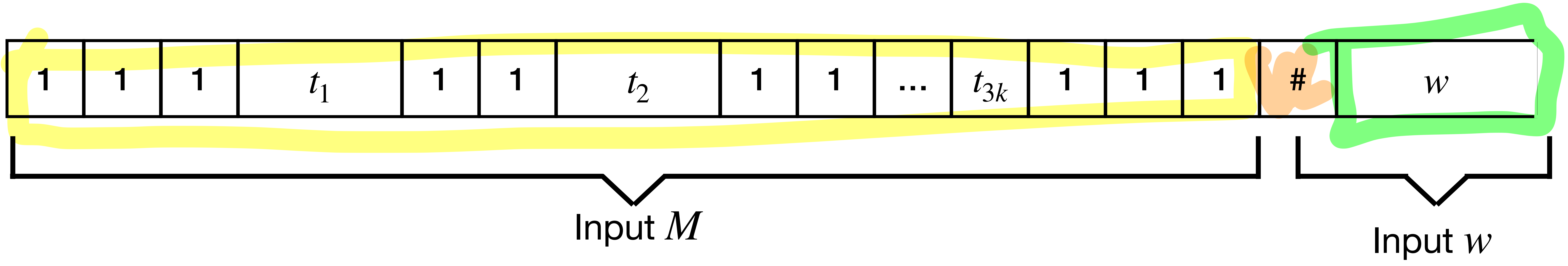
- Tape 1: holds input M and w demarcated with #; never changes
- Tape 2: simulates M's single tape



How M_u works
Configuration

Three tapes

- Tape 1: holds input M and w demarcated with #; never changes
- Tape 2: simulates M's single tape
- Tape 3: holds M's current state



Universal Turing Machine

How M_u works: Phase 1 (validate)

- Check if Tape 1 holds a valid TM by examining $< M >$

Universal Turing Machine

How M_u works: Phase 1 (validate)

- Check if Tape 1 holds a valid TM by examining $< M >$
  - There should be no more than three consecutive ones.

Universal Turing Machine

How M_u works: Phase 1 (validate)

- Check if Tape 1 holds a valid TM by examining $< M >$
  - There should be no more than three consecutive ones.
  - The beginning and ending must be enclosed in 111's.

Universal Turing Machine

How M_u works: Phase 1 (validate)

- Check if Tape 1 holds a valid TM by examining $< M >$
  - There should be no more than three consecutive ones.
  - The beginning and ending must be enclosed in 111's.
  - Substring $110^i|0^j1$ does not appear twice.

prevents two different transitions from the same state on the same input

Universal Turing Machine

How M_u works: Phase 1 (validate)

- Check if Tape 1 holds a valid TM by examining $< M >$
  - There should be no more than three consecutive ones.
  - The beginning and ending must be enclosed in 111's.
  - Substring $110^i | 0^j 1$ does not appear twice.
  - Appropriate number of zeros and ones between 1's demarcating transition code

Universal Turing Machine

How M_u works: Phase 1 (validate)

- Check if Tape 1 holds a valid TM by examining $< M >$
    - There should be no more than three consecutive ones.
    - The beginning and ending must be enclosed in 111's.
    - Substring $110^i \mid 0^j 1$ does not appear twice.
    - Appropriate number of zeros and ones between 1's demarcating transition code

$$11000010100000100001...$$

Universal Turing Machine

How M_u works: Phase 1 (validate)

- Check if Tape 1 holds a valid TM by examining $< M >$
  - There should be no more than three consecutive ones.
  - The beginning and ending must be enclosed in 111's.
  - Substring $110^i | 0^j 1$ does not appear twice.
  - Appropriate number of zeros and ones between 1's demarcating transition code

$$11000010100000100001...$$

not part of tape alphabet

  - Etc.

Universal Turing Machine

How M_u works - Phase 2 (initialize)

1110101000010010011010010000010101 1 ...... 111 # 100110	Tape 1

Code for M

Universal Turing Machine

How M_u works - Phase 2 (initialize)

- Copy w to Tape 2

$111010100001001001101001000001010 11 \ldots\ldots 111 \# 100110$	Tape 1

Code for M

Universal Turing Machine

How M_u works - Phase 2 (initialize)

- Copy w to Tape 2

111010100001001001101001000001010 11 . . . . . . 111 # 100110 Tape 1

Code for M

$\$\underline{1}00110$ Tape 2

Current contents of M's tape

Universal Turing Machine

How M_u works - Phase 2 (initialize)

- Copy w to Tape 2
- Write 0 on Tape 3 indicating it is in the start state

$$1110101000010010011010010000010101 1 \ldots\ldots 111 \# 100110$$ Tape 1

Code for M

$$\$\underline{1}00110$$ Tape 2

Current contents of M's tape

Universal Turing Machine

How M_u works - Phase 2 (initialize)

- Copy w to Tape 2
- Write 0 on Tape 3 indicating it is in the start state

$11101010000100100110100100000101011 \ldots\ldots 111 \ \# \ 100110$	Tape 1
Code for M	
$\$\underline{1}00110$	Tape 2
Current contents of M's tape	
$\$\underline{0}$	Tape 3
Current state of M	

Universal Turing Machine

How M_u works - Phase 2 (initialize)

- Copy w to Tape 2
- Write 0 on Tape 3 indicating it is in the start state
- If at any time, Tape 3 holds 00 (or 000), then halt and accept (or reject)

$11101010000100100110100100000101011\ldots\ldots111\ \#\ 100110$	Tape 1
Code for M	
$\$\underline{1}00110$	Tape 2
Current contents of M's tape	
$\$\underline{0}$	Tape 3
Current state of M	

$<$ state $>$ 1 $<$ input $>$ 1 $<$ new state $>$ 1 $<$ new-symbol $>$ 1 $<$ direction $>$

Universal Turing Machine

How M_u works - Phase 3 (simulation)

- Repeatedly simulate the steps of M

11101010000100100110100100000101011 . . . . . . 111 # 100110	Tape 1
Code for M	
\$$\underline{1}$00110	Tape 2
Current contents of M's tape	
\$$\underline{0}$	Tape 3
Current state of M	

< state > 1 < input > 1 < new state > 1 < new-symbol > 1 < direction >

Universal Turing Machine

How M_u works - Phase 3 (simulation)

- Repeatedly simulate the steps of M
- Example: If tape 3 holds 0^i and tape 2 is scanning 1, then search for substring 110^i1001 on tape 1.

$11101010000100100110100100000101011 \ldots\ldots 111 \# 100110$ Tape 1

Code for M

$\$\underline{1}00110$ Tape 2

Current contents of M's tape

$\$\underline{0}$ Tape 3

Current state of M

Universal Turing Machine

How M_u works - Phase 3 (simulation)

- Repeatedly simulate the steps of M
- Example: If tape 3 holds 0^i and tape 2 is scanning 1, then search for substring 110^i1001 on tape 1.

$11101010000100100110100100000101011 \ldots\ldots 111 \# 100110$ Tape 1

Code for M

$\$\underline{1}00110$ Tape 2

Current contents of M's tape

$\$\underline{0}$ Tape 3

Current state of M

Universal Turing Machine

How M_u works - Phase 3 (simulation)

- Repeatedly simulate the steps of M
- Example: If tape 3 holds 0^i and tape 2 is scanning 1, then search for substring 110^i1001 on tape 1.

if $i = 1$

$11101010000100100110100100000101011 \ldots\ldots 111 \# 100110$ Tape 1

Code for M

$\$\underline{1}00110$ Tape 2

Current contents of M's tape

$\$\underline{0}$ Tape 3

Current state of M

Universal Turing Machine

How M_u works - Phase 3 (simulation)

- Repeatedly simulate the steps of M
- Example: If tape 3 holds 0^i and tape 2 is scanning 1, then search for substring 110^i1001 on tape 1.

What to do next

if $i = 1$

111010100001001001101001000001010 11 . . . . . . 111 # 100110 Tape 1

Code for M

$\$\underline{1}00110$ Tape 2

Current contents of M's tape

$\$\underline{0}$ Tape 3

Current state of M

< state > 1 < input > 1 < new state > 1 < new-symbol > 1 < direction >

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

111010100001001001101001000001010 11 . . . . . . 111 # 100110

Code for M

< state > 1 < input > 1 < new state > 1 < new-symbol > 1 < direction >

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

move tape 2 head to the right

11101010000100100110100100000101011 . . . . . . 111 # 100110

Code for M

write a 0 under tape 2's head

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

move tape 2 head to the right

11101010000100100110100100000101011 . . . . . . 111 # 100110

Code for M

\$0$\underline{0}$0110

Current contents of M's tape

write a 0 under tape 2's head

< state > 1 < input > 1 < new state > 1 < new-symbol > 1 < direction >

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

move tape 2 head to the right

1110101000010010011010010000010101011 . . . . . . 111 # 100110

Code for M

copy new state 00000 to tape 3

$\$0\underline{0}0110$

Current contents of M's tape

write a 0 under tape 2's head

< state > 1 < input > 1 < new state > 1 < new-symbol > 1 < direction >

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

move tape 2 head to the right

1110101000001001001101001100000101011 . . . . . . 111 # 100110

Code for M

copy new state 00000 to tape 3

0\underline{0}$0110

Current contents of M's tape

write a 0 under tape 2's head

$\underline{0}0000

Current state of M

< state > 1 < input > 1 < new state > 1 < new-symbol > 1 < direction >

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

- Check if 00 or 000 is on tape 3; if so, halt and accept or reject

move tape 2 head to the right

11101010000100100110100100000101011 . . . . . . 111 # 100110

Code for M

copy new state 00000 to tape 3

$000110

Current contents of M's tape

write a 0 under tape 2's head

$00000

Current state of M

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

- Check if 00 or 000 is on tape 3; if so, halt and accept or reject
- Otherwise, simulate the next move by searching for pattern. In this example, the next pattern = 1100000101

move tape 2 head to the right

11101010000100100110100100000101011 . . . . . . 111 # 100110

Code for M

copy new state 00000 to tape 3

$\$0\underline{0}0110$

Current contents of M's tape

write a 0 under tape 2's head

$\$\underline{0}0000$

Current state of M

< state > 1 < input > 1 < new state > 1 < new-symbol > 1 < direction >

Universal Turing Machine

How M_u works - Phase 3 - (simulation, after a single move)

- Check if 00 or 000 is on tape 3; if so, halt and accept or reject
- Otherwise, simulate the next move by searching for pattern. In this example, the next pattern = 1100000101

move tape 2 head to the right

11101010000100100110100100000101011 . . . . 111 # 100110

Code for M

copy new state 00000 to tape 3

$000110

Current contents of M's tape

write a 0 under tape 2's head

$00000

Current state of M

Keeps repeating …

Examples

https://rosettacode.org/wiki/Universal_Turing_machine#Python

https://pastebin.com/raw/JqZGrddK