CHAPTER 3

The Church-Turing Thesis

Turing Machines

FA

PDA

TM

Proposed by Alan Turing in 1936 as a result of studying algorithmic processes by means of a computational model

Similar to FA, but with unlimited and unrestricted memory

Can do everything a real computer can do

Can read and write on the tape

The read-write head can move left and right

The tape is infinite
(Initially it contain an input string and is blank everywhere else)

Formal Definition

- A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_o, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and
    1. Q is the set of states
    2. Σ is the input alphabet not containing the blank symbol B
    3. Γ is the tape alphabet, where $B \in \Gamma$ and $\Sigma \subseteq \Gamma$
    4. $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
    5. $q_o \in Q$ is the start state
    6. $q_{accept} \in Q$ is the accept state (Once a Turing machine enters the accept state, the input string is accepted regardless of the tape content)
    7. $q_{reject} \in Q$ is the reject state, where $q_{reject} \neq q_{accept}$

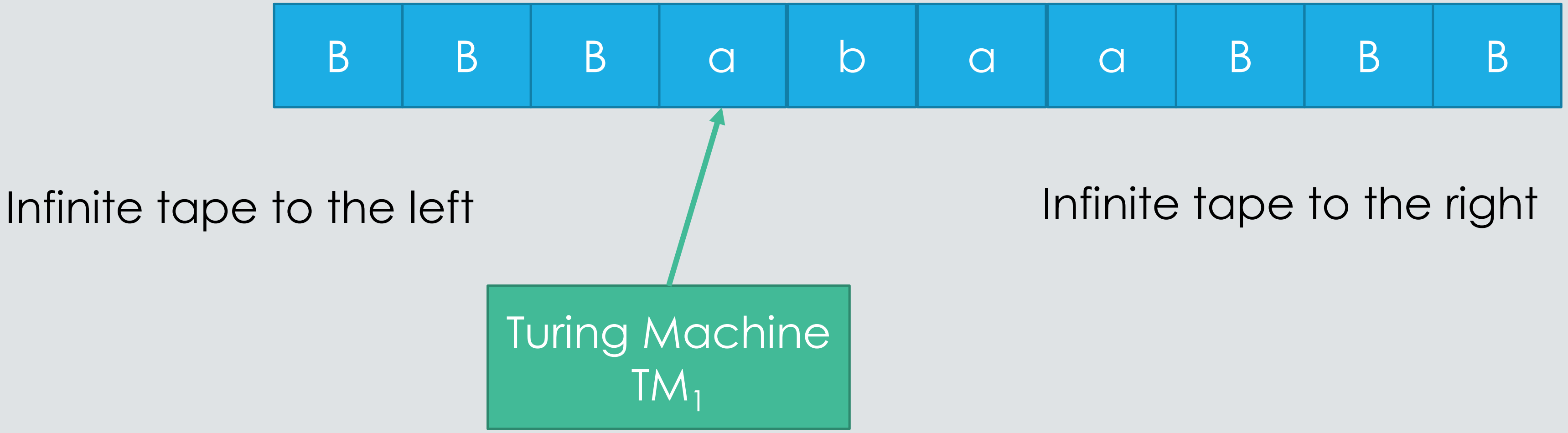
- Observations
    - The TM continues computing until it produces an output, which can be *accept* and *reject* if the TM enter the designated q_{accept} and q_{reject} states, or it can go on forever, never *halting*

Languages

- The language of a TM M is denoted L(M)
- A language is **Turing-decidable** (decidable language or recursive language) if some Turing Machine **decides** it
- A language is **Turing-recognizable** (or recursive enumerable language) if some Turing Machine **recognizes** it

We will come back to these definitions when we discuss power-equivalence!

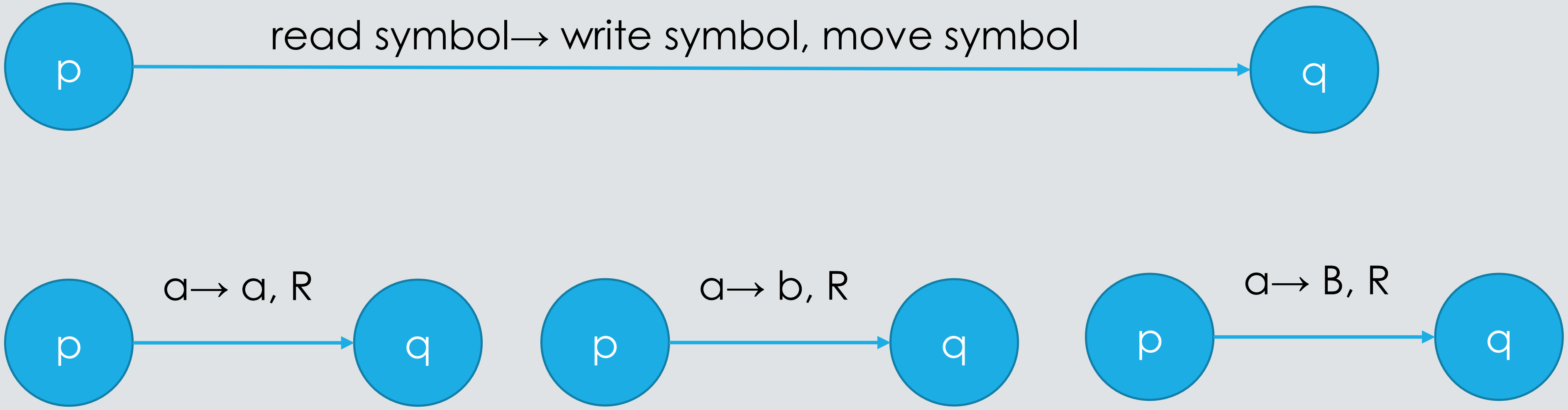
Tape


B
B
B
a
b
a
a
B
B
B
Infinite tape to the left
Infinite tape to the right
Turing Machine
TM1

Operations

- Write
  - *Replaces* a symbol on the tape w/ another symbol & then *shifts* to a new (or current) state
- Move
  - *Moves* the tape head one cell to the right (left, respectively) & then *shift* to a new (or current) state
- Halt
  - *Halts* when the TM encounters a <*state*, *input symbol* > pair for which no transition is defined

Transitions



Configurations

$$u \; q \; v$$

Symbols to the left-hand side of the tape head

Current state

Symbols to the right-hand side of the tape head

- Examples

$q_o w$

Initial configuration

$w \; q_{accept}$

Accepting configuration

$u \; q_x a v$

a→ a, R

$ua \; q_y v$

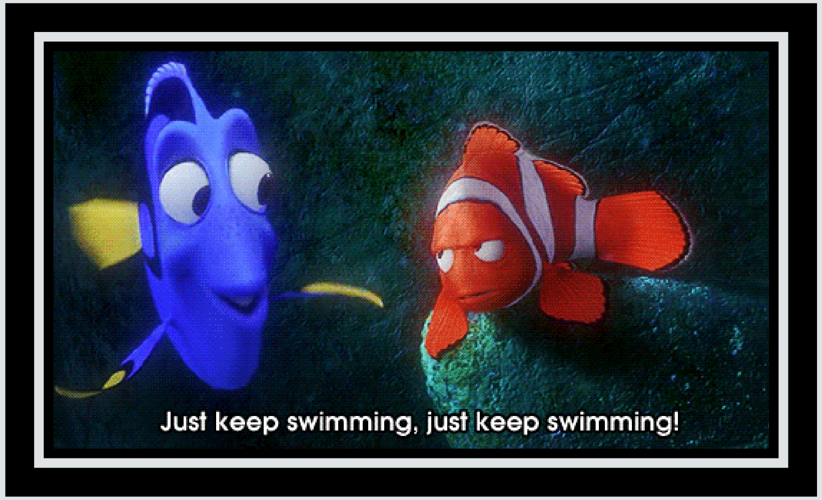
$u \; q_x a v$

a→ b, R

$ub \; q_y v$

A new configuration

Designing a Turing Machine



Come up with a ***high level description*** of how you'll achieve your goal, i.e., creating a TM that accepts valid strings in a given language

Consider if it is easier (or at least possible) to design a ***sub-machine*** for each step of the algorithm (i.e., requirement in the language)

Take a step back and think about the ***requirements*** of the language and how can you keep track of them prior to drawing a TM

Designing a TM

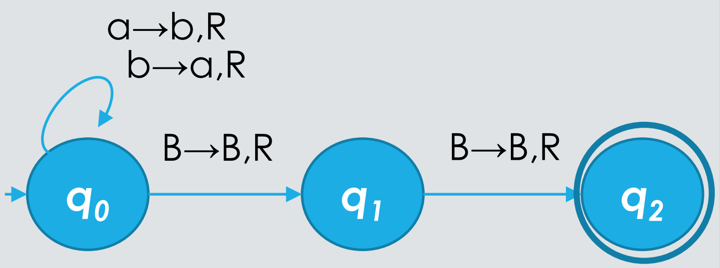
- Requirements
  - Takes q_0wB as an input configuration, where w is a string over {a, b}
  - Transforms a's (in w) into b's and b's into a's
  - Ends on acceptance on the following configuration **w'BBq_2**

- Tape contains input string, tape head is at the beginning of the tape
- Repeat until a "B" symbol is found:
  - Read "a" and write "b" or read "b" and write "a"
- Read "B", write "B", and mover Right
- Read "B", write "B", and mover Right
- Reach acceptance state

Design description

δ	B	a	b
q_0	$q_{1,\ B,\ R}$	$q_{0,\ b,\ R}$	$q_{0,\ a,\ R}$
q_1	$q_{2,\ B,\ R}$		
q_2			

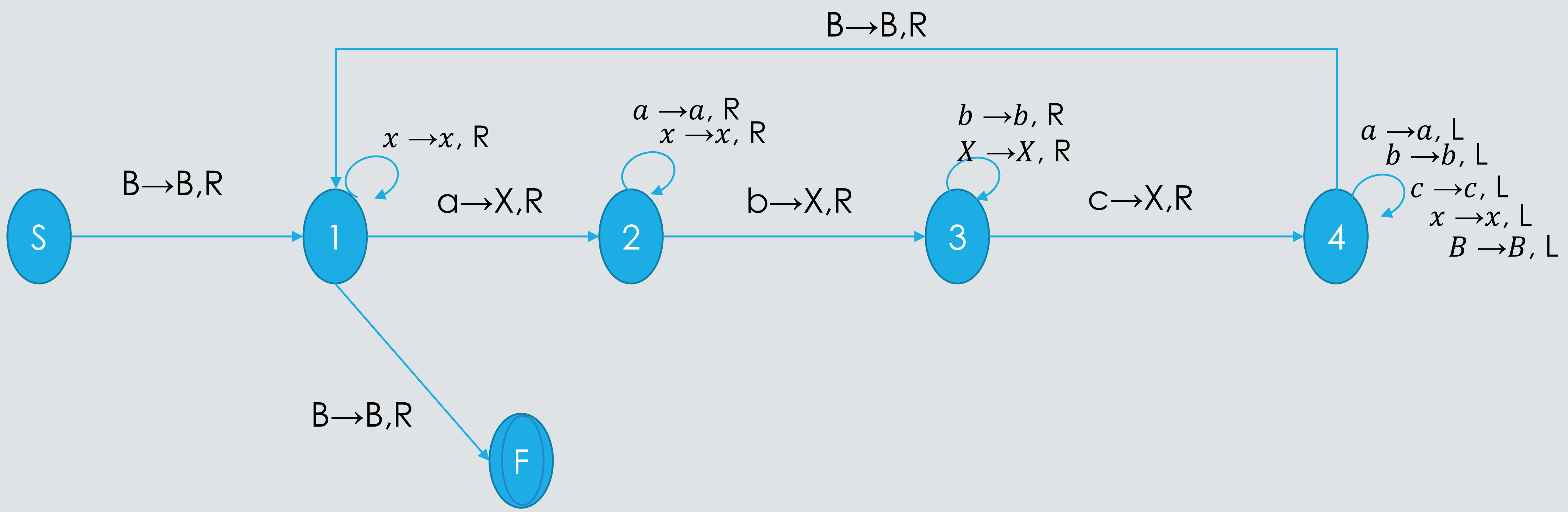
Transition diagram



State diagram

Example 1

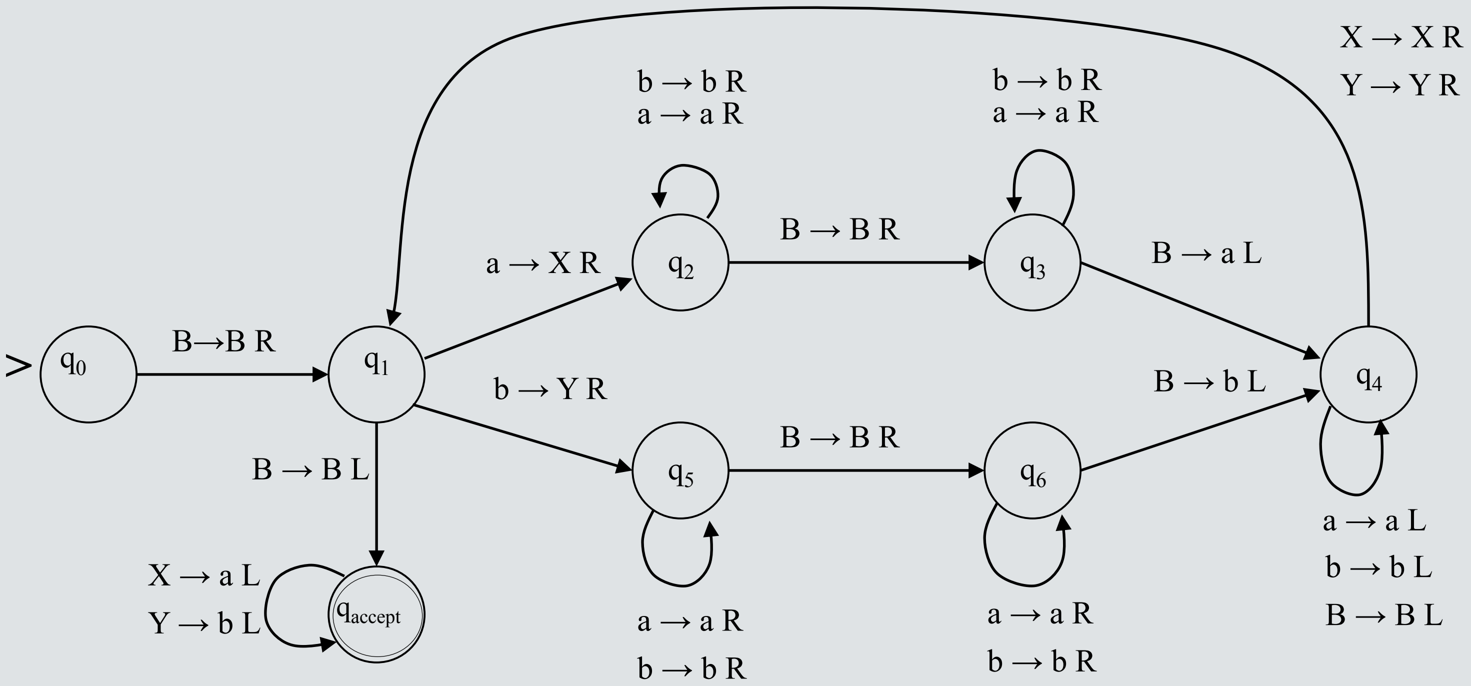
- TM that accepts language L={$a^n b^n c^n$ | $n \geq 0$}



	FA	PDA	TM
Example	a*b*	a^nb^n, ww^r	$a^nb^nc^n$
Storage	Finite tape	Finite tape and infinite stack	Infinite tape
Tape Operations	Read only	Read only	Read and write
Tape Head	Move right	Move right	Move left or right
Accepts	Stops in a final state after reading an input string		Enters the accept state
Rejects	Stops in a non-final state after reading a string or no possible transitions to take		Enters reject state (no more transitions to take) or loops

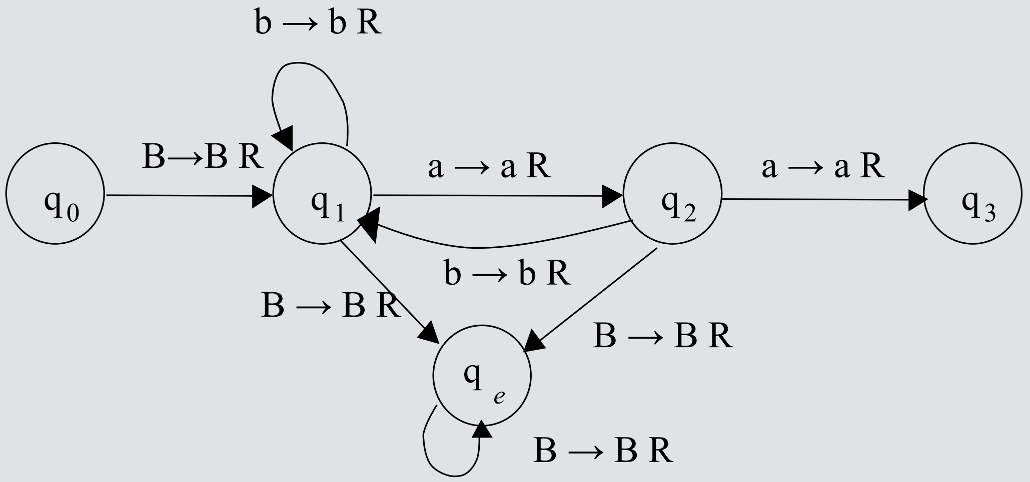
Example 2

- Consider a TM *M* that produces a copy of input string over {a, b} with input *BuB* and terminates with tape *BuBuB*, where u ∈ (a ∪ b)*, e.g., *BabB* yields *BabBabB*
  - How does the computation of *BabB* using M looks like?



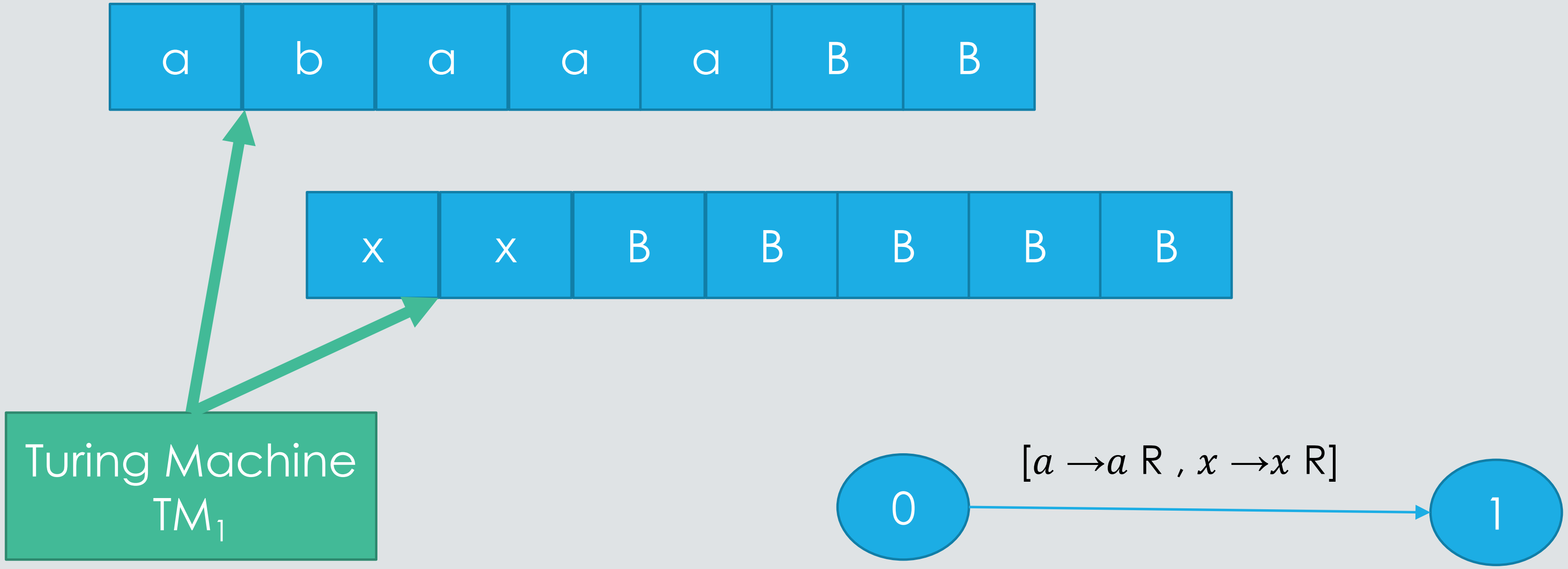
Variants of TM

- Acceptance by Halting
  - Equivalent (in power) to an ordinary TM that accepts by final state
- Example : A TM that accepts $(a \cup b)^*aa(a \cup b)^*$ by halting



Variants of TM

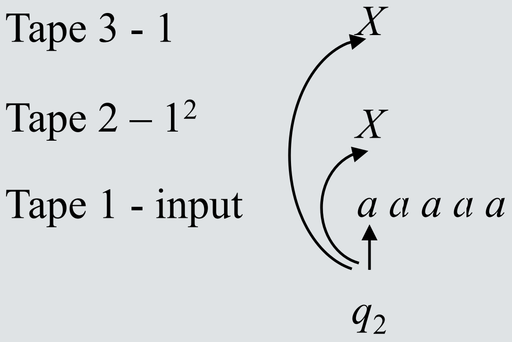
- Multitape TM
  - Ordinary TM with more than one tape
  - Equivalent in power to ordinary TM



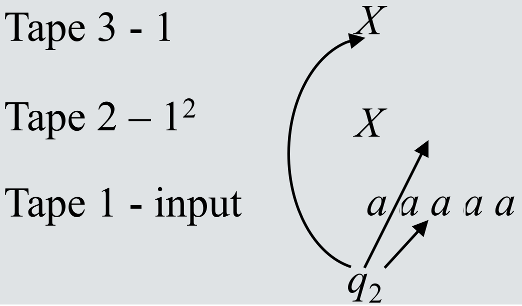
Multitape TM

- Example: TM for L = {a^k | k is a perfect square }
  - Tape 1 holds the input string, a string of a's
  - Tape 2 holds a string of X's whose length is a perfect square
  - Tape 3 holds a string of *X*'s whose length is √ |*S*|, where *S* is the string on Tape 2

(i)



(ii)

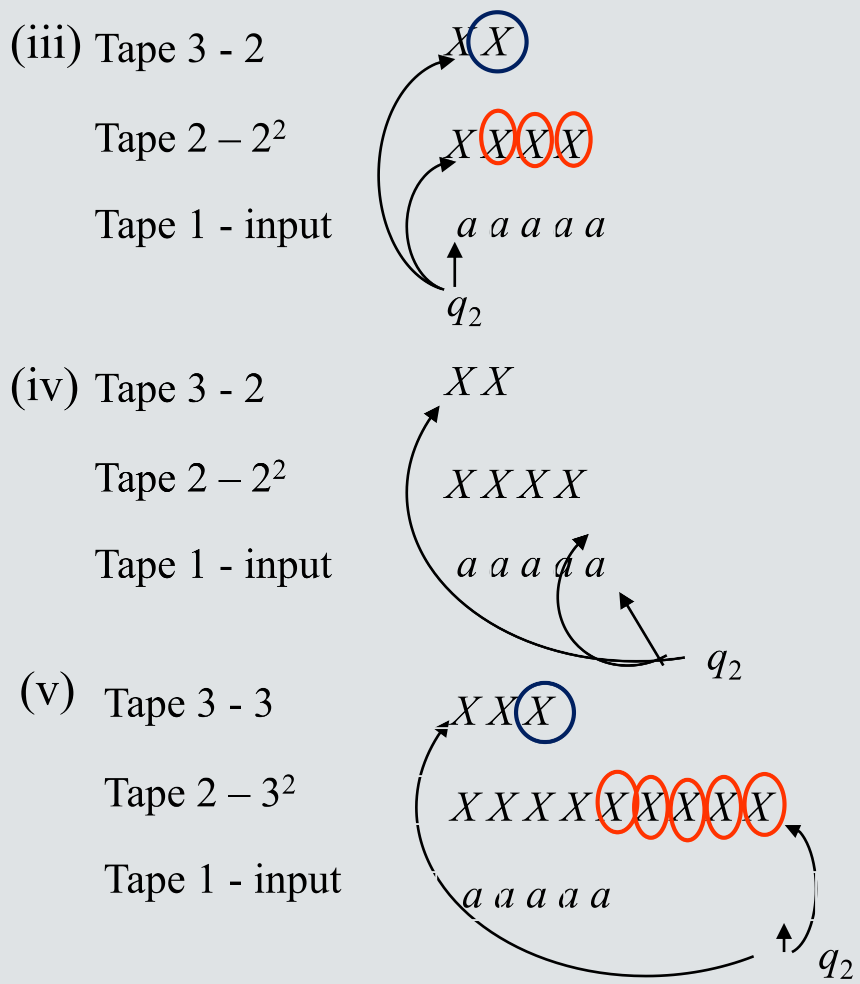


Step 1: Since the input is not a null string, initialize tapes 2 and 3 with an *X*, and all the tape head move to *Position* 1

Step 2: Move the heads of tapes 1 and 2 to the right, since they have scanned a *nonblank* square

Accept: if both read a *blank*
Reject: if tape head 1 reads a *blank* and tape head 2 reads an *X*

Multitape TM



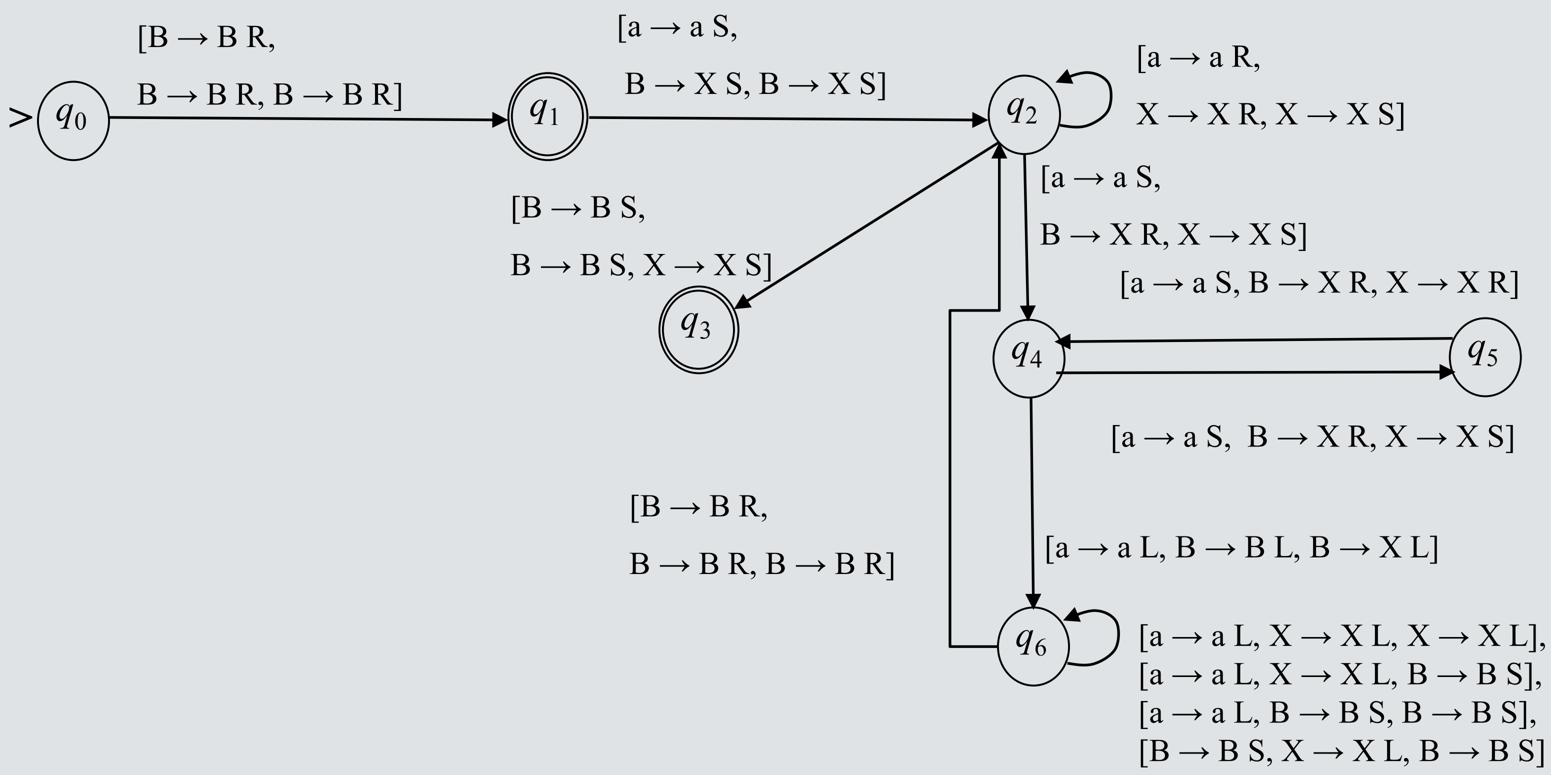
Step 3: *Reconfiguration* for comparison with the next perfect square by

- adding an *X* on tape 2 to yield k^2+1 *X*'s
- appending *two* copies of the string on tape 3 to the end of the string on tape 2 to yield $(k+1)^2$ *X*'s
- adding an *X* on tape 3 to yield $(k + 1)$ *X*'s on tape 3
- moving all the tape heads to *Position* 1

Step 4: Repeat Steps 2 through 3

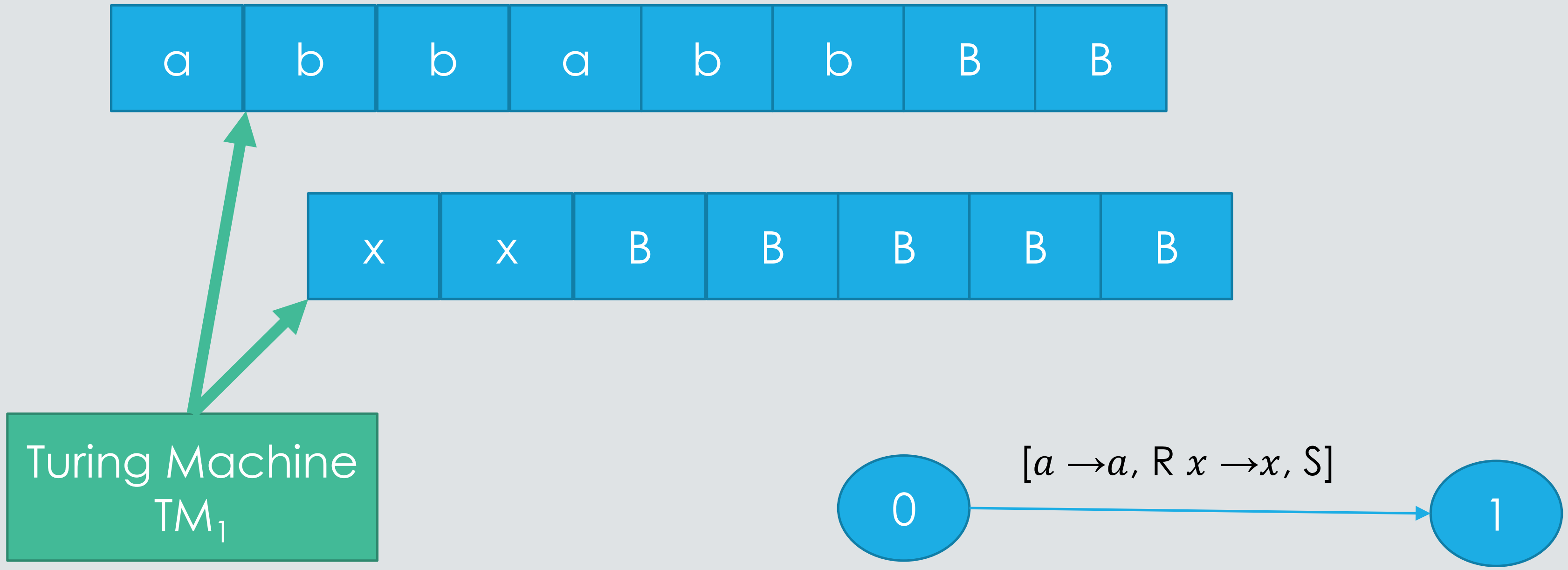
Another iteration of Step 2 halts and rejects the input

Multitape TM



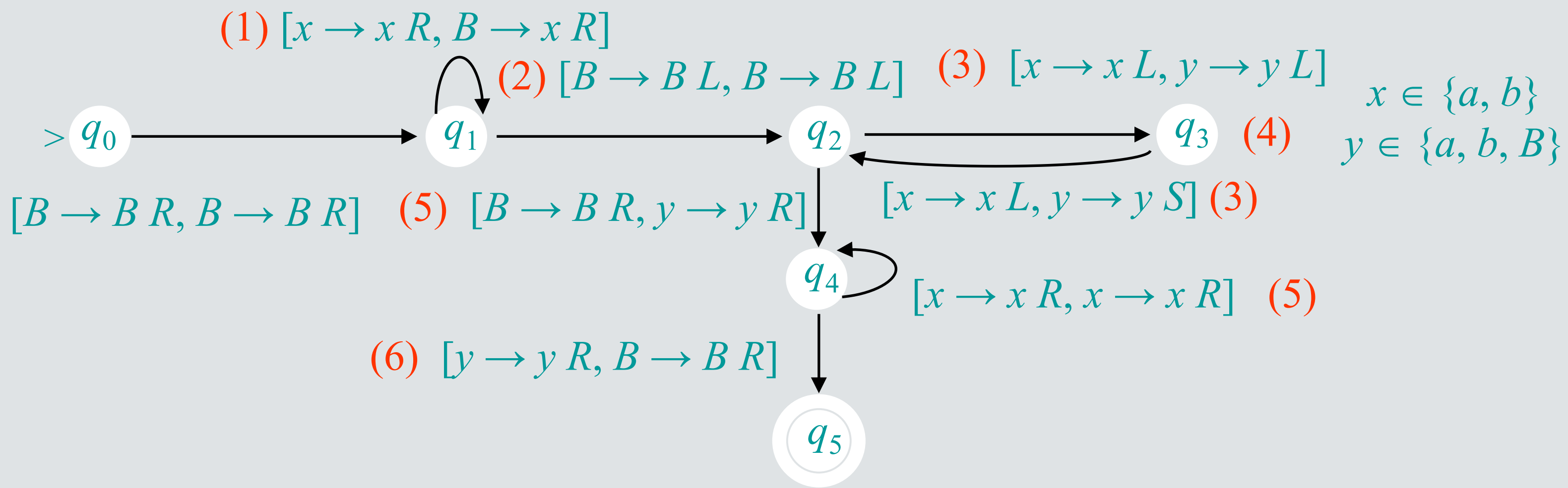
Variants of TM

- "Stay Put"
  - Besides moving left (L) and right (R), the TM may "stay put" (S) after reading/writing on a tape
  - Equivalent in power to ordinary TM



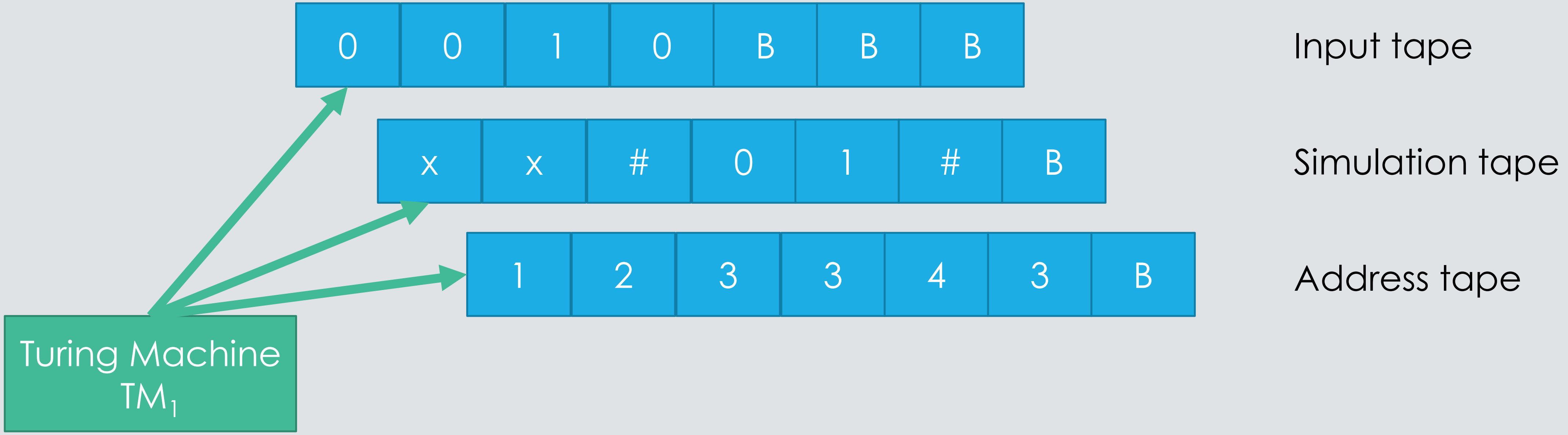
Another Multitape, "Stay-Put" Example

- A 2-tape TM that accepts $\{ uu \mid u \in \{ a, b \}^* \}$
  - Computation:
    1) Make a *copy* of the input *S* (on tape 1) to tape 2; tape heads: right of *S*.
    2) Move both tape heads one step to the left.
    3) Move the head of tape 1 *two* squares for each square move of tape 2.
    4) *Reject* the input *S* if the TM halts in q_3. (i.e. $|S|$ is *odd*.)
    5) *Compare* the 1st half with the 2nd half of *S* in q_4
    6) *Accept* *S* in q_5



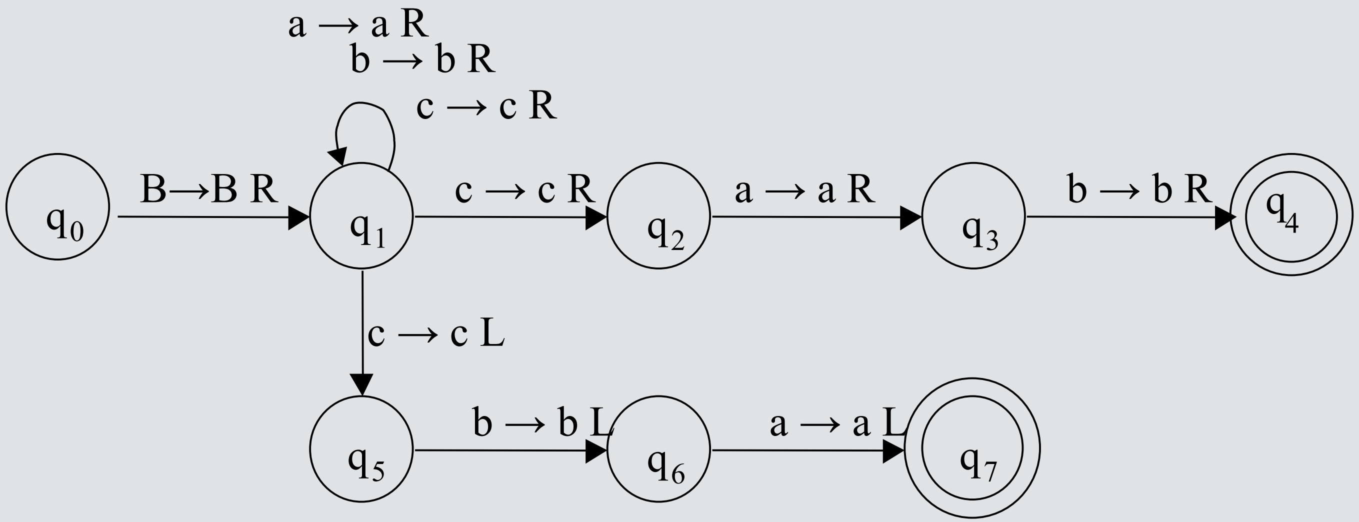
Variants of TM

- Nondeterministic, multitape TMs
  - A TM will explore several possibilities in determining if a string is accepted/rejected
  - The computation of a nondeterministic TM is a tree whose branches correspond to different possibilities for the machine. If some branch of the computation leads to the accept state, then the TM accepts its input



Nondeterministic TM

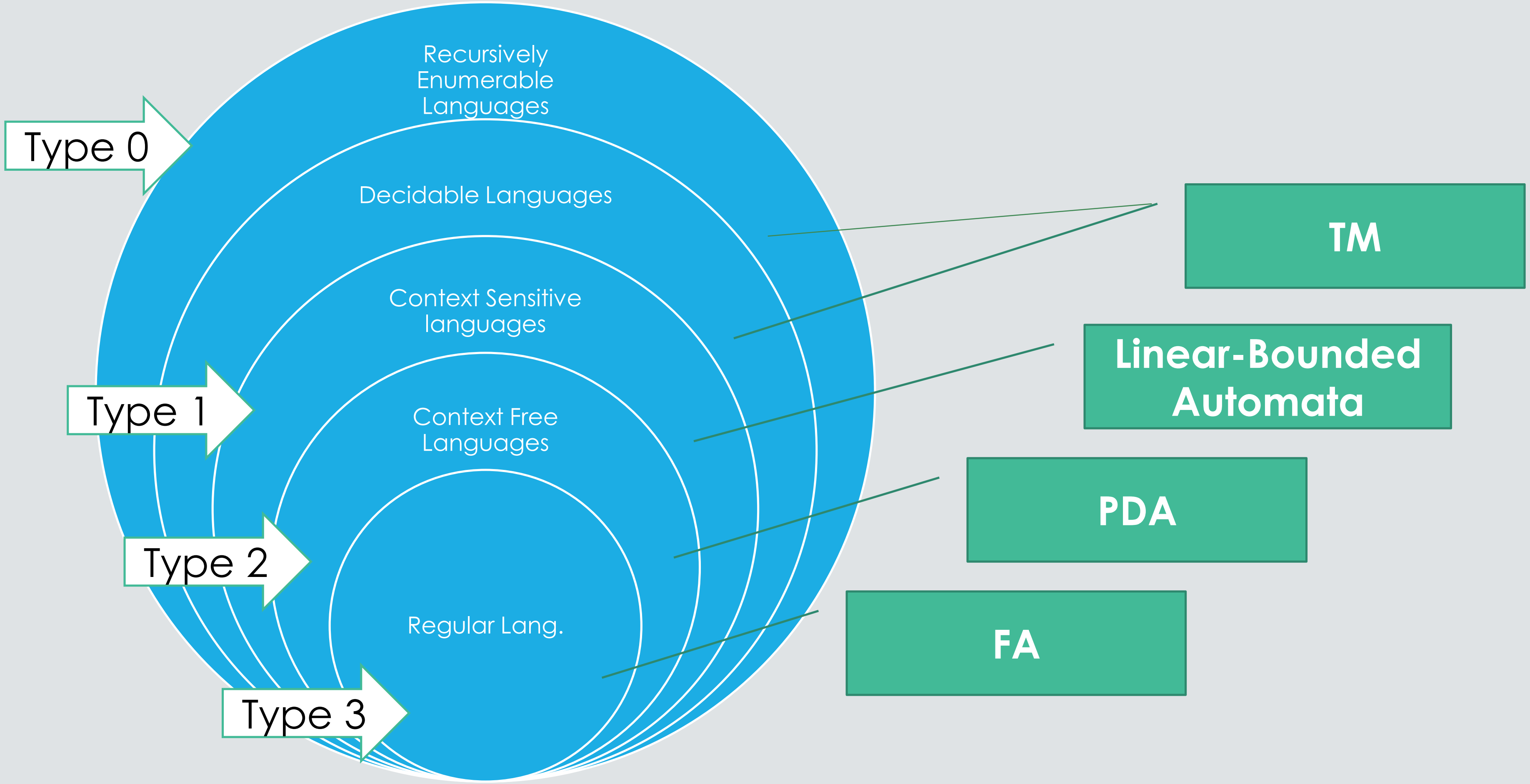
- NTM M accepts strings over {a, b, c}* containing a ‘c’, which is either *preceded* or *followed* by ‘*ab*’
  - Is “ccb” accepted by M?
  - Are “abc” / “cab” accepted by M?



q_0 B**abc**B
B$q_1$abcB
Ba$q_1$bcB
Babq_1cB
Ba$q_5$bcB
B$q_6$abcB
q_7 BabcB

q_0 B**cab**B
B$q_1$cabB
Bc$q_2$abB
Bcaq_3bB
Bcabq_4B

Chomsky Hierarchy



Remember

A language accepted by a TM is a **recursively enumerable** language (or TM-recognizable)

A language that is accepted by a TM that *halts* for all input strings is said to be **recursive** (or TM-decidable)

It is often accepted that any *algorithm* that can be carried out at all (by humans, a computer, or a computation model) can be carried out by a TM.
(**Church –Turing Thesis**)

Definition: An **algorithm** is a procedure that can be executed on a TM. (*If a problem cannot be solved by an algorithm, i.e., no TM can be designed for it, then a real computer cannot solve it.*)