CS 302.1 - Automata Theory

Lecture 09

Shantanav Chakraborty

Center for Quantum Science and Technology (CQST)

Center for Security, Theory and Algorithms (CSTAR)

IIIT Hyderabad

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Quick Recap

Pumping Lemma for CFL: If L is Context Free, then there exists $p > 0$ (pumping length), such that, for any $w \in L$ of length $|w| \geq p$, w can be split into five parts, i.e. $w = uvxyz$
satisfying the following conditions:

- $|vy| \geq 1$
- $|vxy| \leq p$
- $uv^ixy^iz \in L, \ \forall i \geq 0$

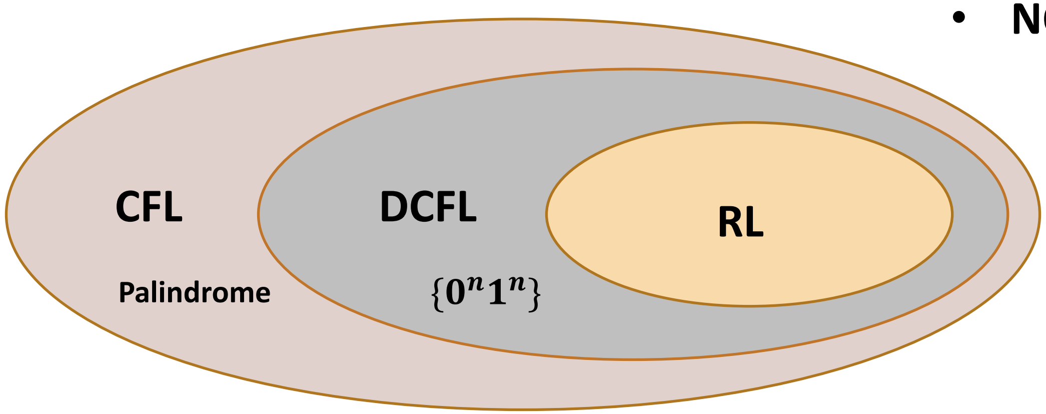
Closure properties of CFLs

CFLs are closed under

- **Union**
- **Star**
- **Concatenation**

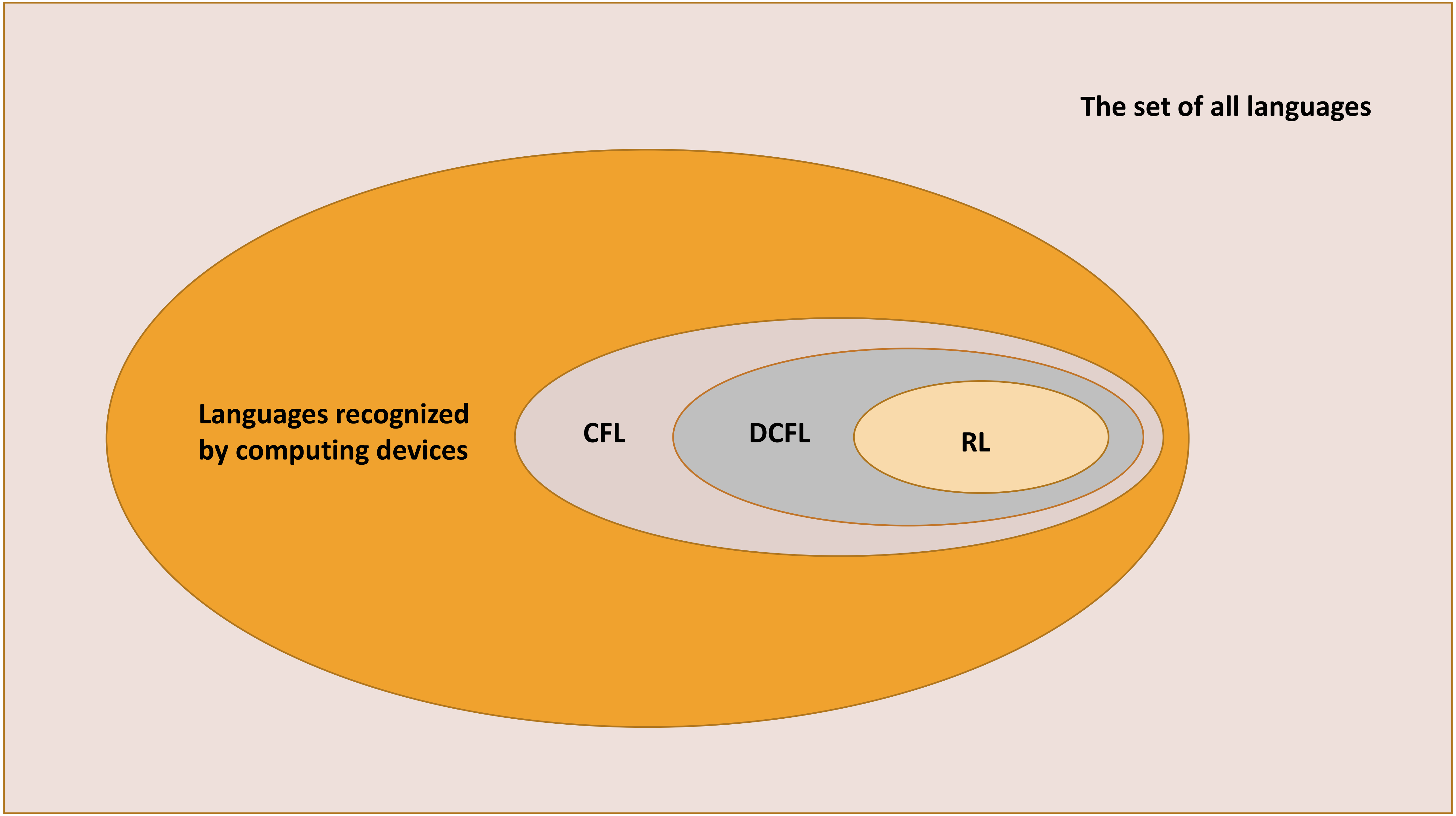
CFLs are NOT closed under

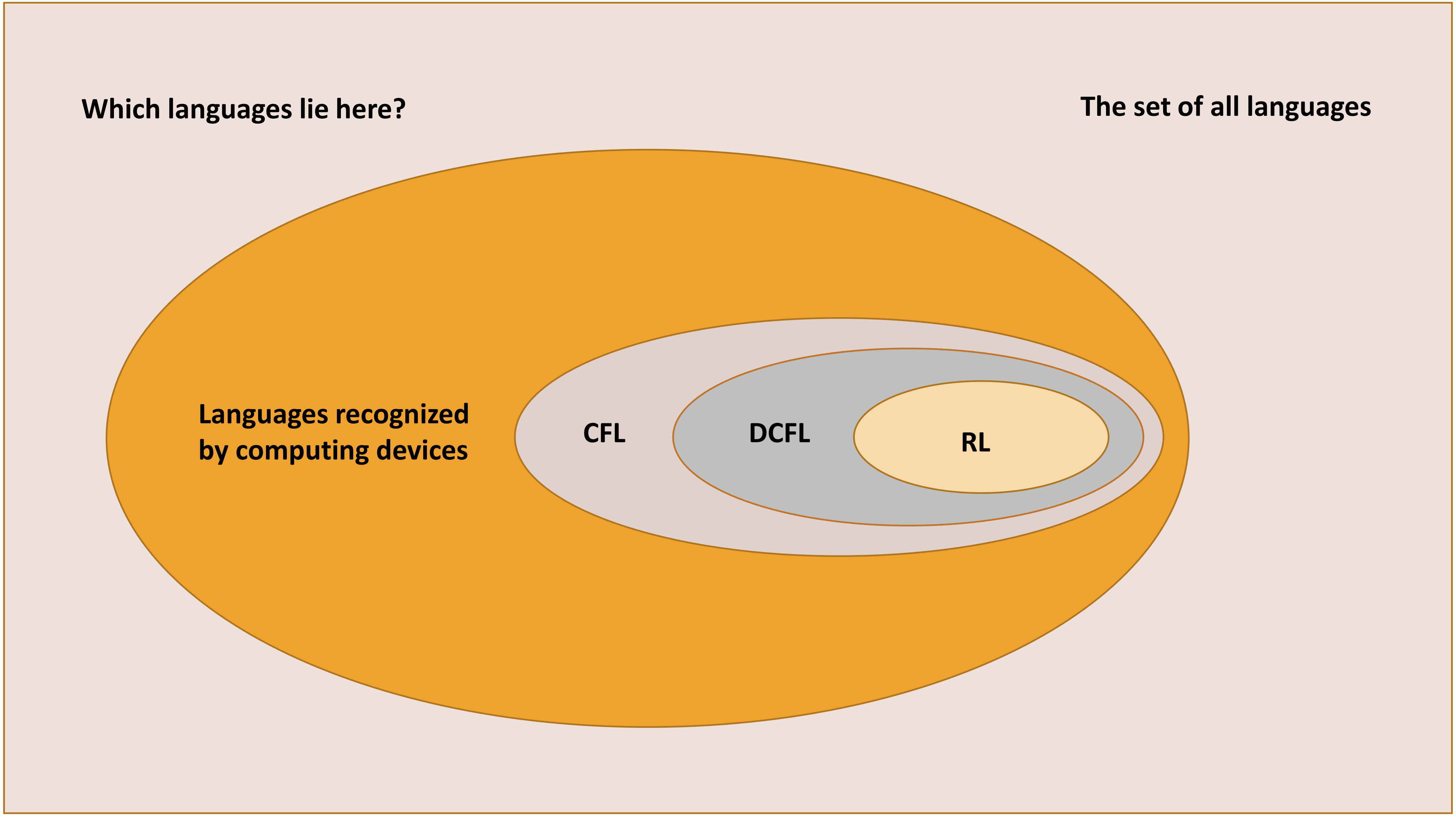
- **Complementation**
- **Intersection**



For DCFLs

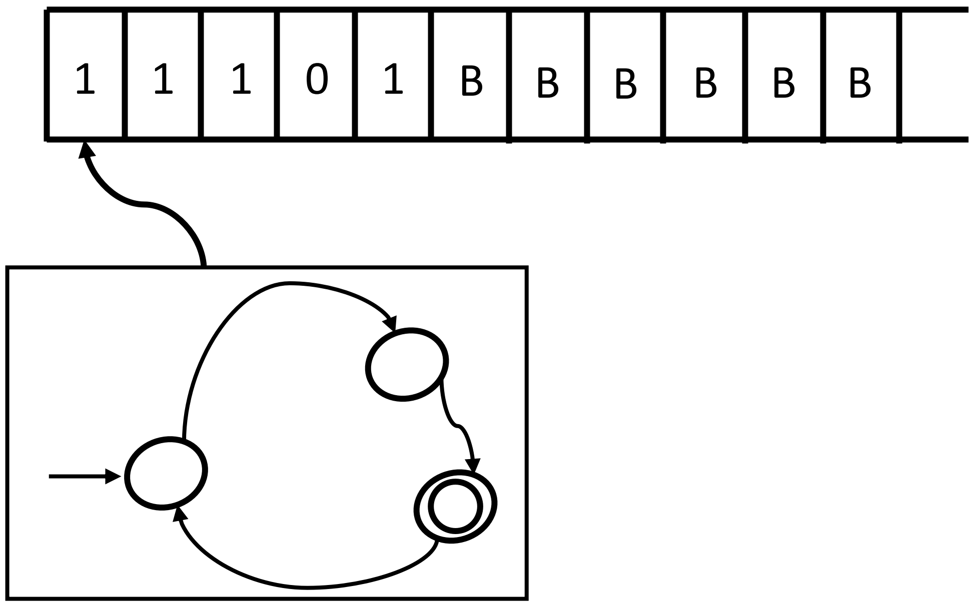
- **NOT closed Union**
- **Closed under complementation**
- **NOT closed under intersection**


The set of all languages
Languages recognized
by computing devices
CFL
DCFL
RL


Which languages lie here?
The set of all languages
Languages recognized by computing devices
CFL
DCFL
RL

Turing Machines

- A Turing machine is a FSM that has access to a infinite tape as its memory.
- The infinite tape contains in it, the input string followed by Blanks (indicated by B)
- The Turing machine can both read from the tape and write in it – one cell at a time, using a Read/Write head.
- The Read/Write head can move to the Left or to the Right – again one cell at a time.

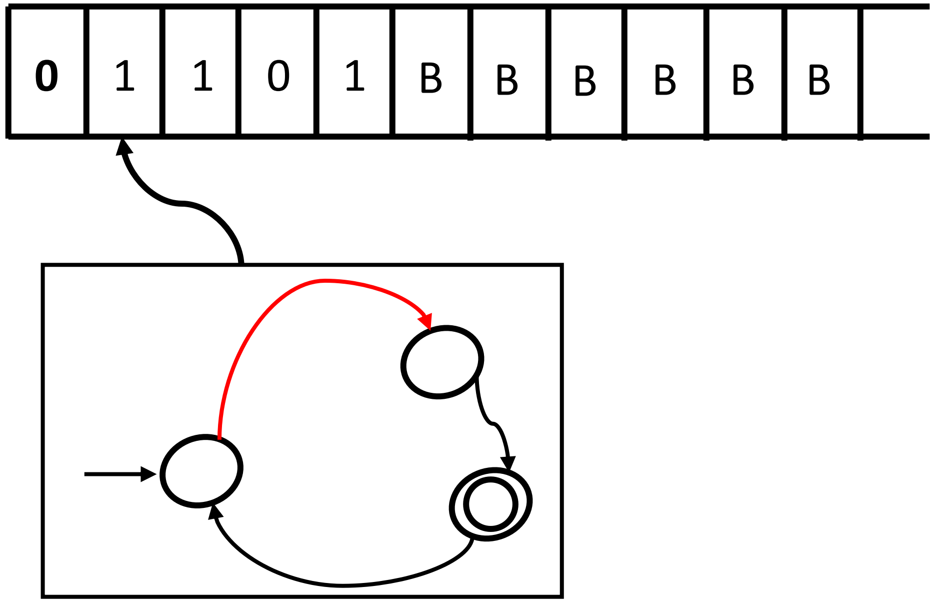


At each step, the TM:

- Replaces the symbol of the current cell in the tape with a new one
- Transitions from one state to another (or remains where it was)
- The Read/Write head moves either to the Left or to the Right

Turing Machines

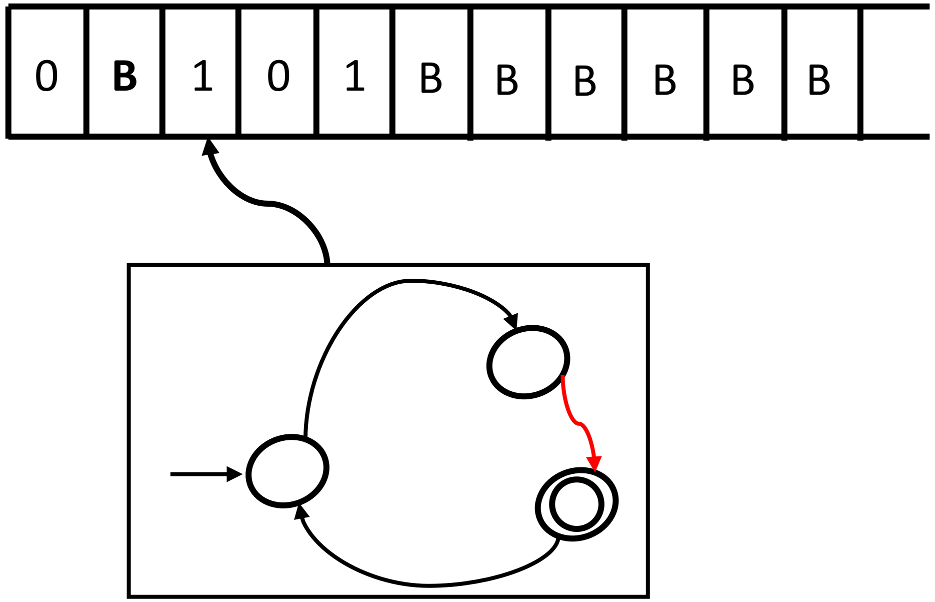
- A Turing machine is an FSM that has access to a infinite tape as its memory.
- The infinite tape contains in it, the input string followed by Blanks (indicated by B)
- The Turing machine can both read from the tape and write in it – one cell at a time, using a Read/Write head.
- The Read/Write head can move to the Left or to the Right – again one cell at a time.



At each step, the TM:
- Replaces the symbol of the current cell in the tape with a new one
- Transitions from one state to another (or remains where it was)
- The Read/Write head moves either to the Left or to the Right

Turing Machines

- A Turing machine is an FSM that has access to a infinite tape as its memory.
- The infinite tape contains in it, the input string followed by Blanks (indicated by B)
- The Turing machine can both read from the tape and write in it – one cell at a time, using a Read/Write head.
- The Read/Write head can move to the Left or to the Right – again one cell at a time.

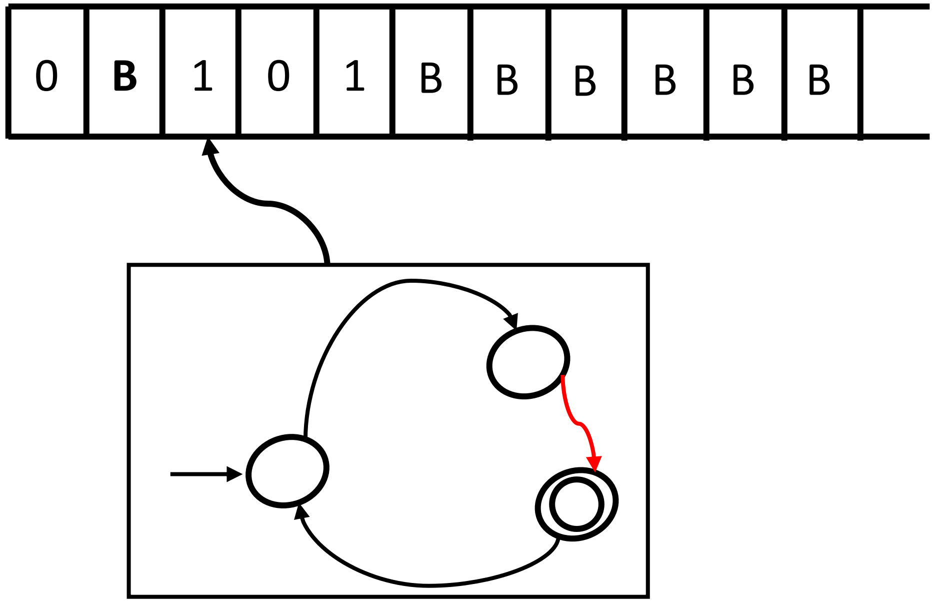


At each step, the TM:
- Replaces the symbol of the current cell in the tape with a new one
- Transitions from one state to another (or remains where it was)
- The Read/Write head moves either to the Left or to the Right

- Whenever the computation reaches an accept/reject state – the TM accepts or rejects the input string (halts)

Turing Machines

- A Turing machine is an FSM that has access to a infinite tape as its memory.
- The infinite tape contains in it, the input string followed by Blanks (indicated by B)
- The Turing machine can both read from the tape and write in it – one cell at a time, using a Read/Write head.
- The Read/Write head can move to the Left or to the Right – again one cell at a time.



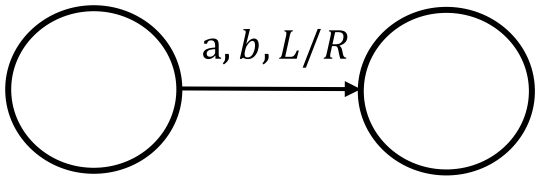
At each step, the TM:

- Replaces the symbol of the current cell in the tape with a new one
- Transitions from one state to another (or remains where it was)
- The Read/Write head moves either to the Left or to the Right

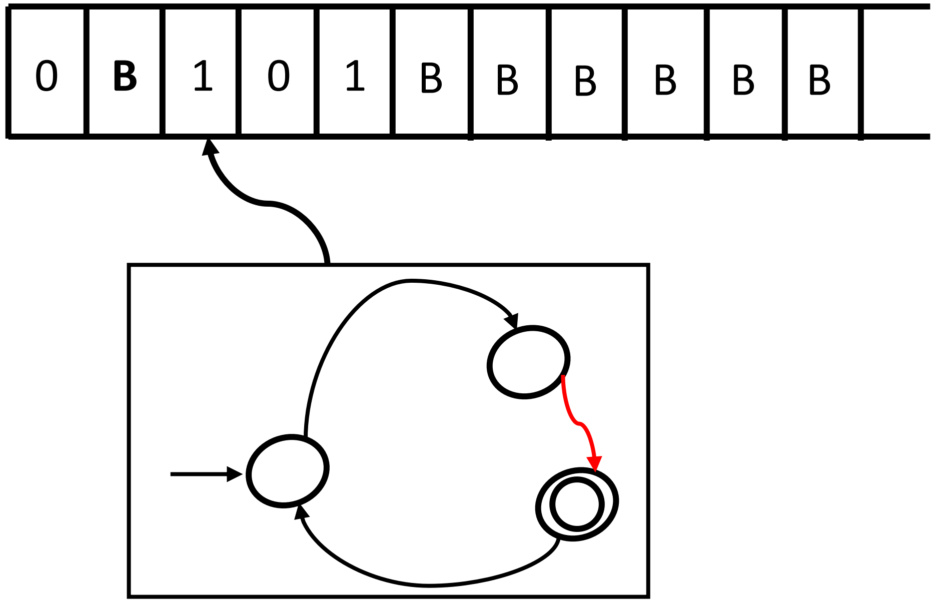
- Whenever the computation reaches an accept/reject state – the TM accepts or rejects the input string (halts)

- In a way these "added features" give TMs their power. (eg: ability to write on the tape)
- Notice: acceptance/rejection of a run is not tied to the input.
- Auxiliary computation can be performed – as much as needed, even when the input string has been scanned

Turing Machines



Transition $a, b, L/R$: Read a from the tape, replace with b and move L/R



TM may never halt – it may loop forever

Accept state

Reject state

TM halts and **accepts/rejects** on reaching these states

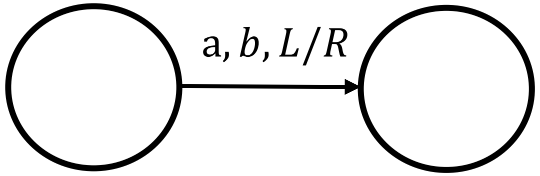
At each step, the TM:

- Replaces the symbol of the current cell in the tape with a new one
- Transitions from one state to another (or remains where it was)
- The Read/Write head moves either to the Left or to the Right

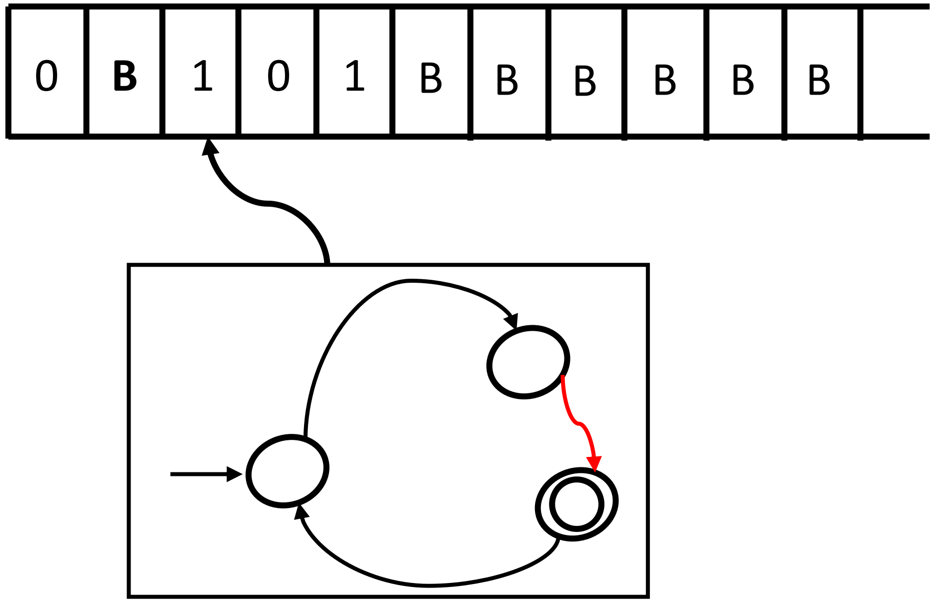
- Whenever the computation reaches an accept/reject state – the TM accepts or rejects the input string (halts)

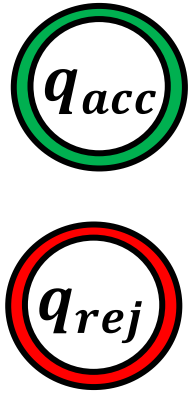
- In a way these "added features" give TMs their power. (eg: ability to write on the tape)
- Notice: acceptance/rejection of a run is not tied to the input.
- Auxiliary computation can be performed – as much as needed, even when the input string has been scanned

Turing Machines



Transition $a, b, L/R$: Read a from the tape, replace with b and move L/R





TM halts and **accepts/rejects** on reaching these states

So, given a TM M and an input ω,

$M(\omega)$ **accepts** if $\omega \in L(M)$
$M(\omega)$ **rejects** if $\omega \notin L(M)$
$M(\omega)$ **runs infinitely** if $\omega \notin L(M)$

At each step, the TM:

- Replaces the symbol of the current cell in the tape with a new one
- Transitions from one state to another (or remains where it was)
- The Read/Write head moves either to the Left or to the Right

- Whenever the computation reaches an accept/reject state – the TM accepts or rejects the input string (halts)

- In a way these "added features" give TMs their power. (eg: ability to write on the tape)
- Notice: acceptance/rejection of a run is not tied to the input.
- Auxiliary computation can be performed – as much as needed, even when the input string has been scanned

Turing Machines

Turing machines are named after **Alan Turing**. In 1936, he gave a negative answer to Hilbert's *Entscheidungsproblem* (Decision problem) – *Are all decision problems decidable?*

Turing claimed that anything "humanly computable" can be simulated by his machines

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

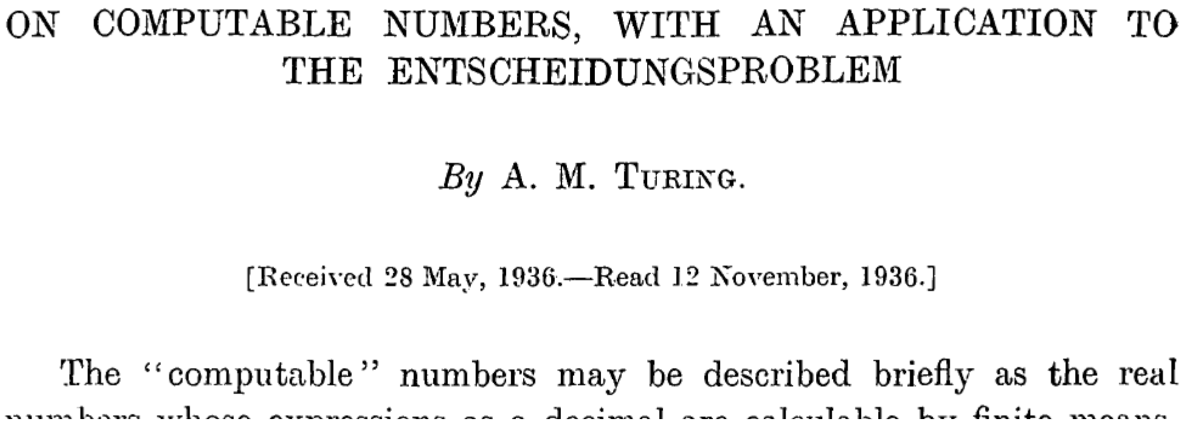
By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means

Turing Machines

Turing machines are named after **Alan Turing**. In 1936, he gave a negative answer to Hilbert's *Entscheidungsproblem* (Decision problem) – *Are all decision problems decidable?*


ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real

Turing claimed that anything "humanly computable" can be simulated by his machines

- Turing assumed that the human brain to be a finite state machine with a finite number of states
- Consider such a human being working on a problem with a notebook, pencil and an eraser.
- The pages of the notebook are laid out on the tape – each cell consists of one page, with a finite amount of information.
- Whatever the human being does with the notebook, can be simulated on the TM: reading, writing, erasing (writing a blank), moving left or right to a new page etc.

Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.

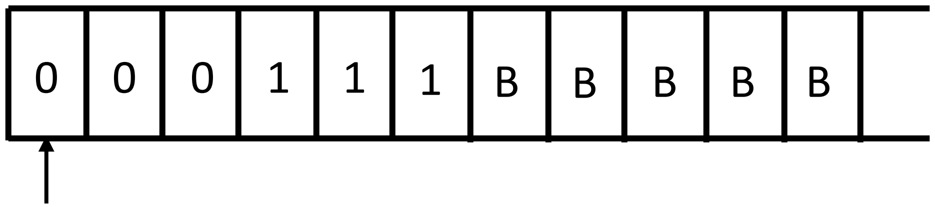
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



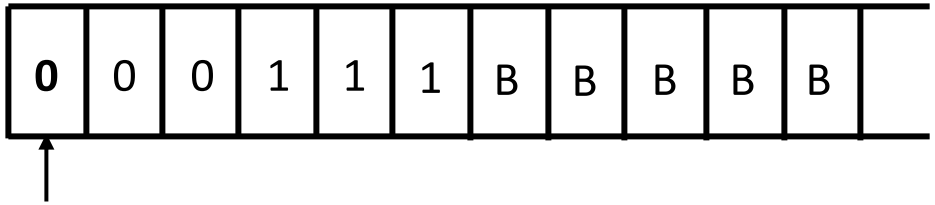
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



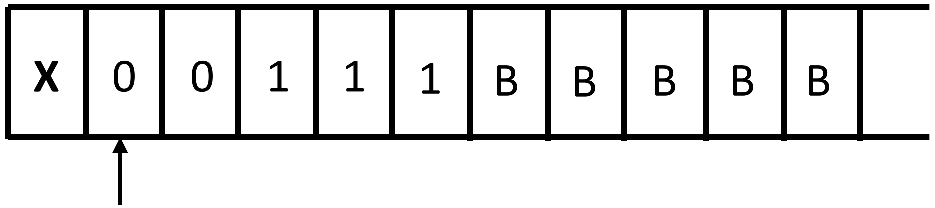
Turing Machines

Example: Let $L = \{0^n1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



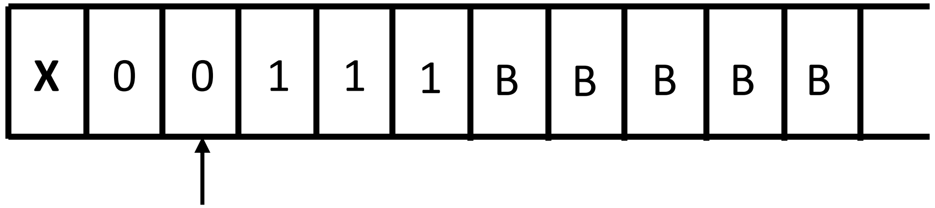
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.



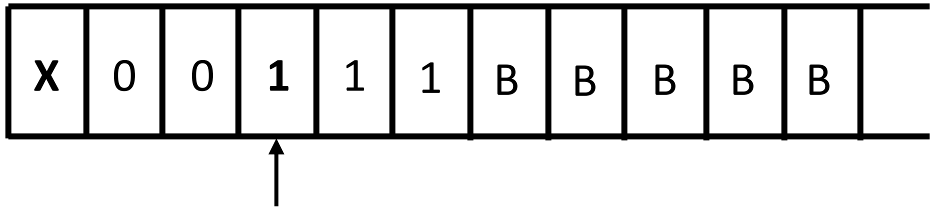
Turing Machines

Example: Let $L = \{0^n1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



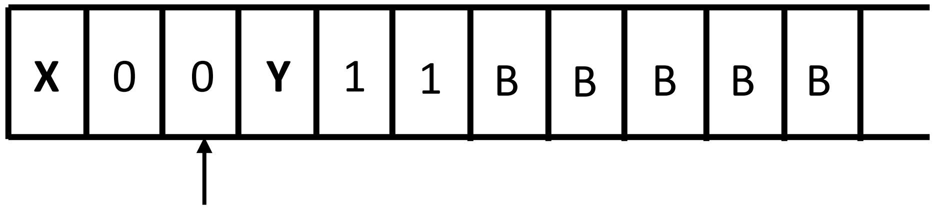
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.



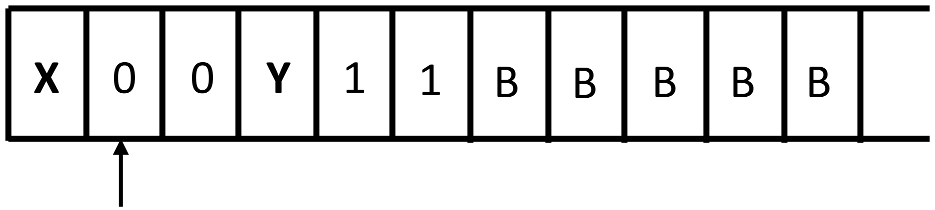
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



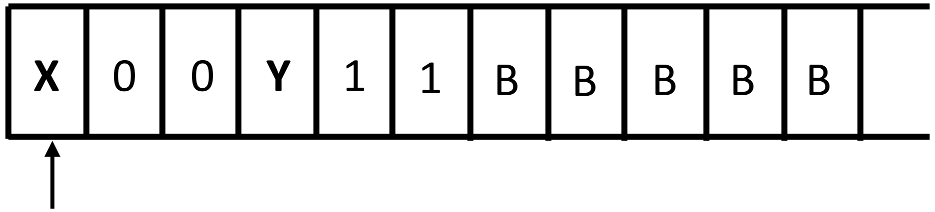
Turing Machines

Example: Let $L = \{0^n1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



While moving left, when an X is encountered, the head should move right until the next 0 to be marked is encountered $\Rightarrow$ **We need rules like (X, X, R)**

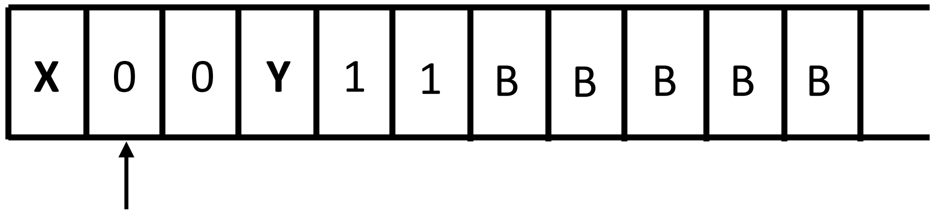
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



While moving left, when an X is encountered, the head should move right until the next 0 to be marked is encountered $\Rightarrow$ **We need rules like (X, X, R)**

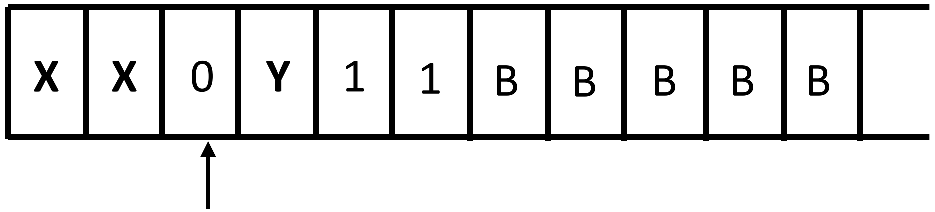
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.



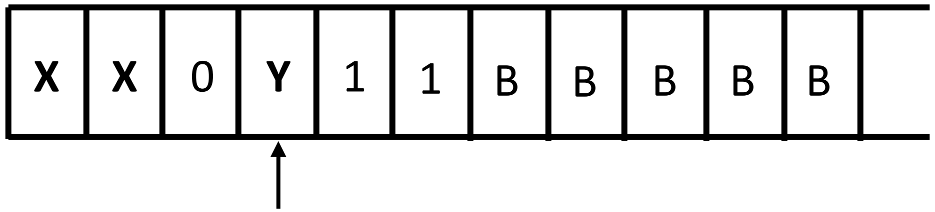
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.



While moving right, when a Y is encountered, the head should move right as that’s where the next 1 to be marked is $\Rightarrow$ **We need rules like $(\boldsymbol{Y}, \boldsymbol{Y}, \boldsymbol{R})$**

Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.

X	**X**	0	**Y**	**Y**	1	B	B	B	B	B	

While moving left, when a Y is encountered, the head should keep moving left as those $1'$s have been marked already $\Rightarrow$ **We need rules like** $(\boldsymbol{Y}, \boldsymbol{Y}, \boldsymbol{L})$

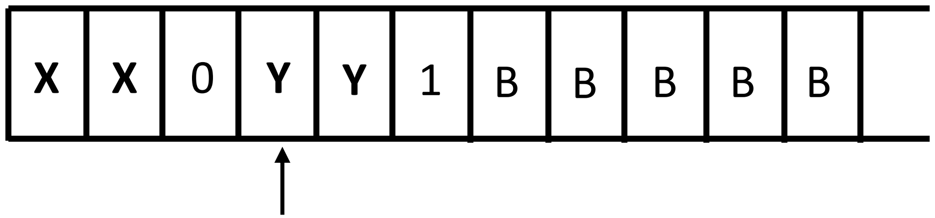
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



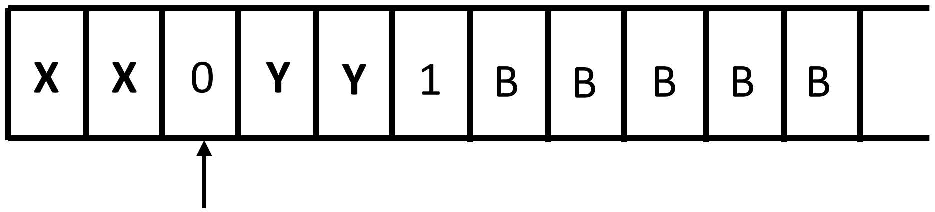
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



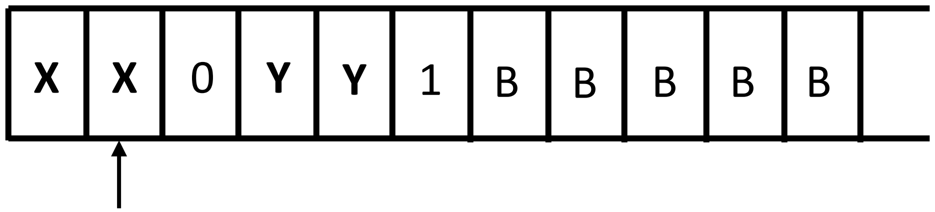
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



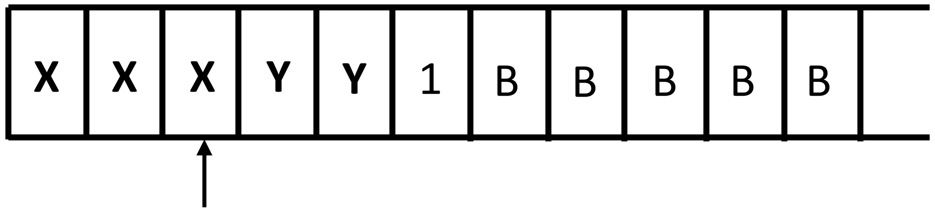
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



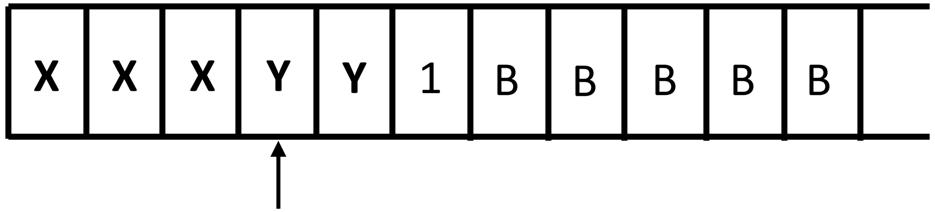
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



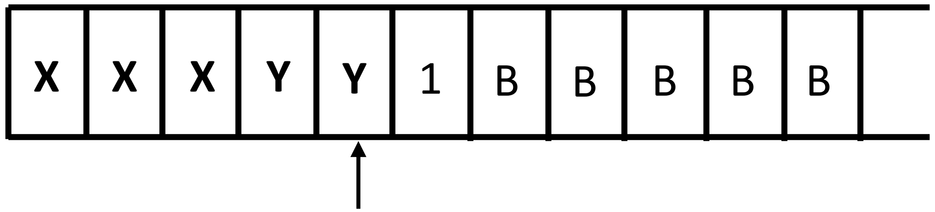
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



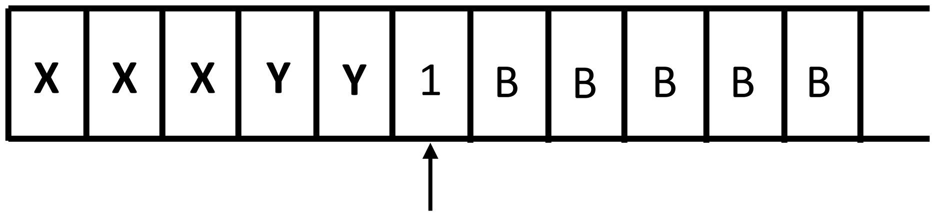
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



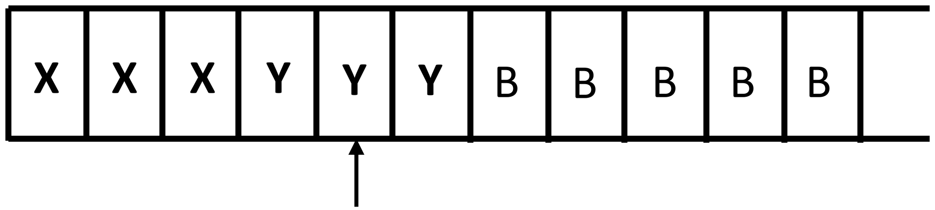
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.



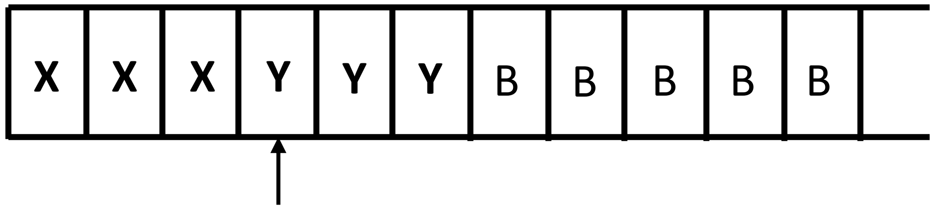
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



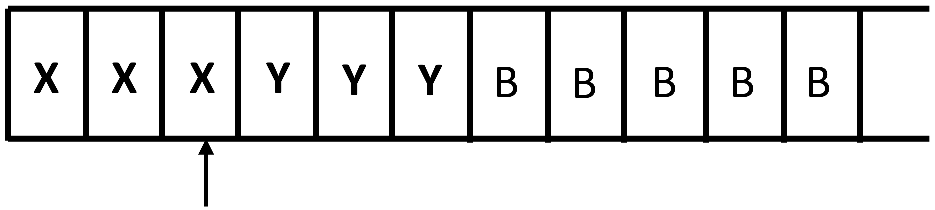
Turing Machines

Example: Let $L = \{\mathbf{0}^n\mathbf{1}^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



At this stage the head should move right to look for the next 0 to mark, but finds Y

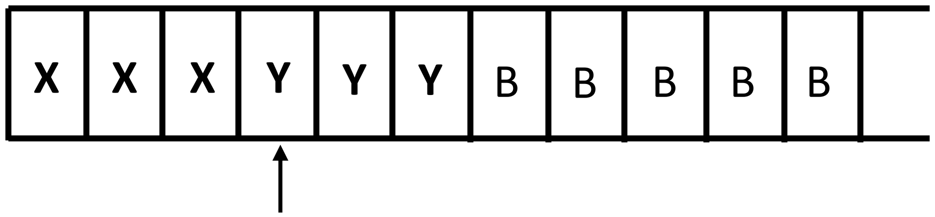
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.



At this stage the head should move right and looks for the next 0 to mark, but finds Y

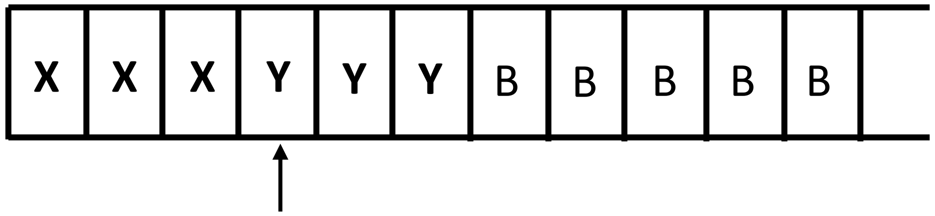
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



The head keeps moving right until it finds a B

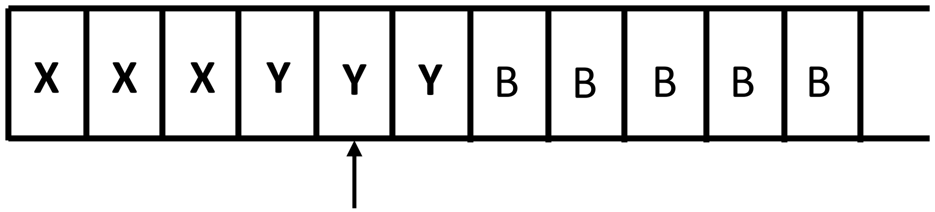
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



The head keeps moving right until it finds a B

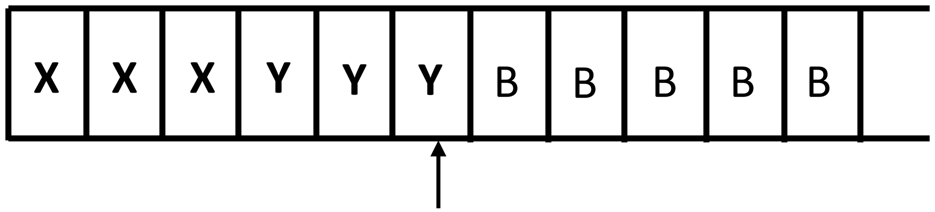
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



The head keeps moving right until it finds a B

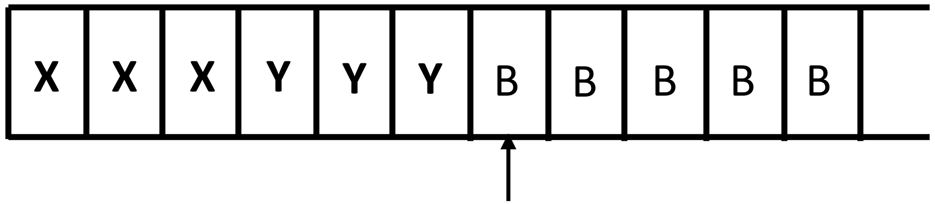
Turing Machines

Example: Let $L = \{0^n 1^n | n \geq 1\}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0's or $1's$ left in the tape.



The head keeps moving right until it finds a B

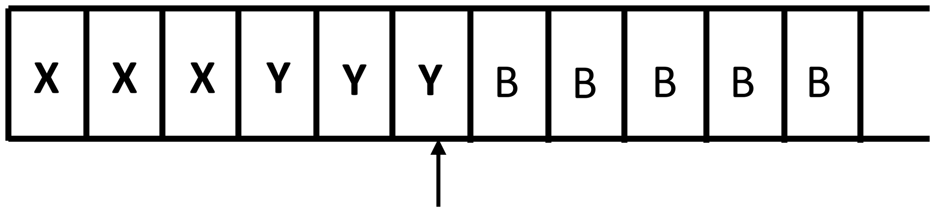
Turing Machines

Example: Let $\boldsymbol{L = \{0^n 1^n | n \geq 1\}}$

We will try to develop the basic idea in designing the Turing Machine for this language. Note that $L = CFL$.

Idea: An accepting run of a TM for L could look something like this:

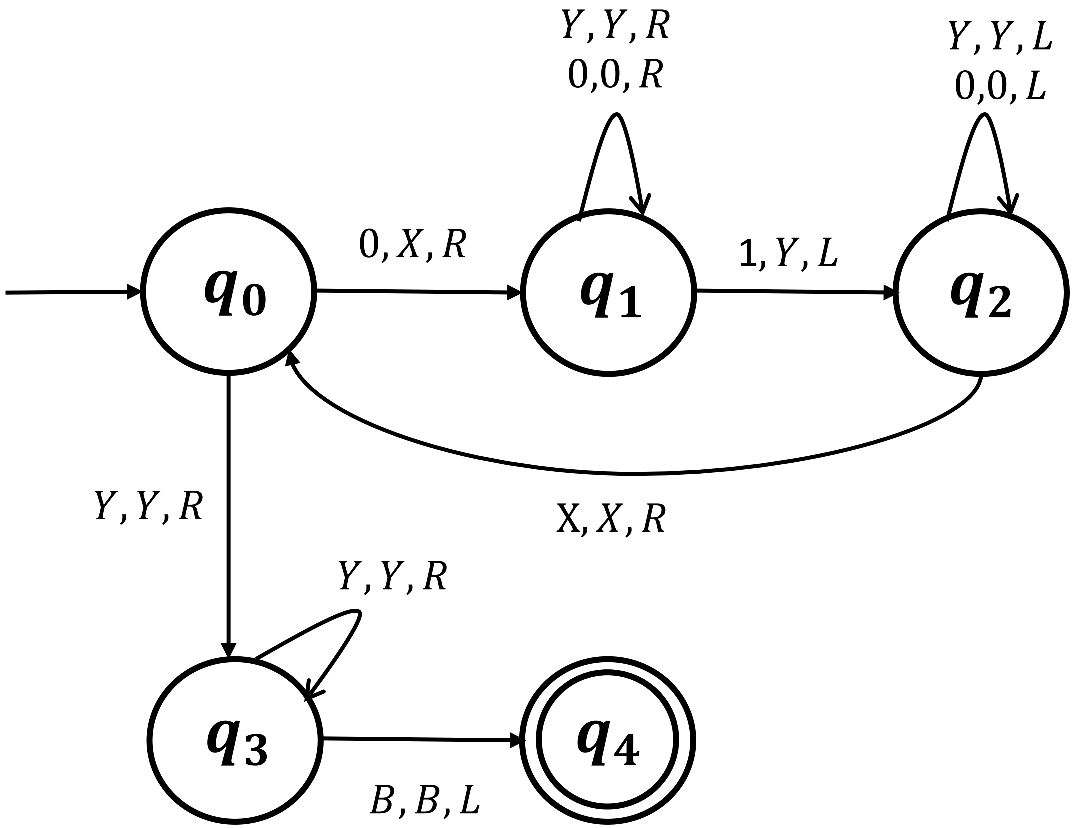
- Mark the first 0 (by replacing it with some special symbol say X)
- Continue to move to the right until the first 1 is encountered.
- Mark the first 1 (by replacing it with some special symbol say Y)
- Continue to move left until you encounter the second 0 (its to the right of the X you had marked before)
- Continue to move right until you encounter a second 1 (its to the right of the Y you had marked before)
- Go on repeating this, until there all the $0'$s and $1'$s have been marked (with X and Y).
- Accept if there are no 0’s or $1's$ left in the tape.



This is when the TM decides to accept the input string.

Turing Machines

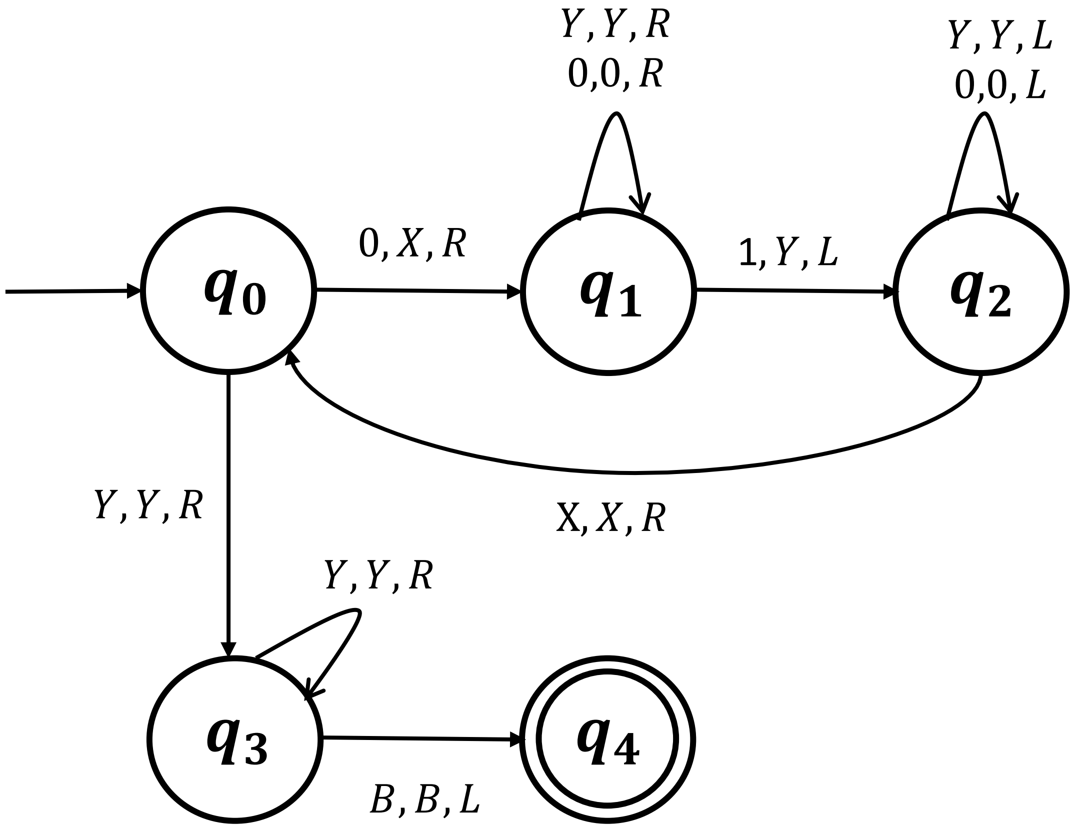
Example: Let $L = \{0^n 1^n | n \geq 1\}$

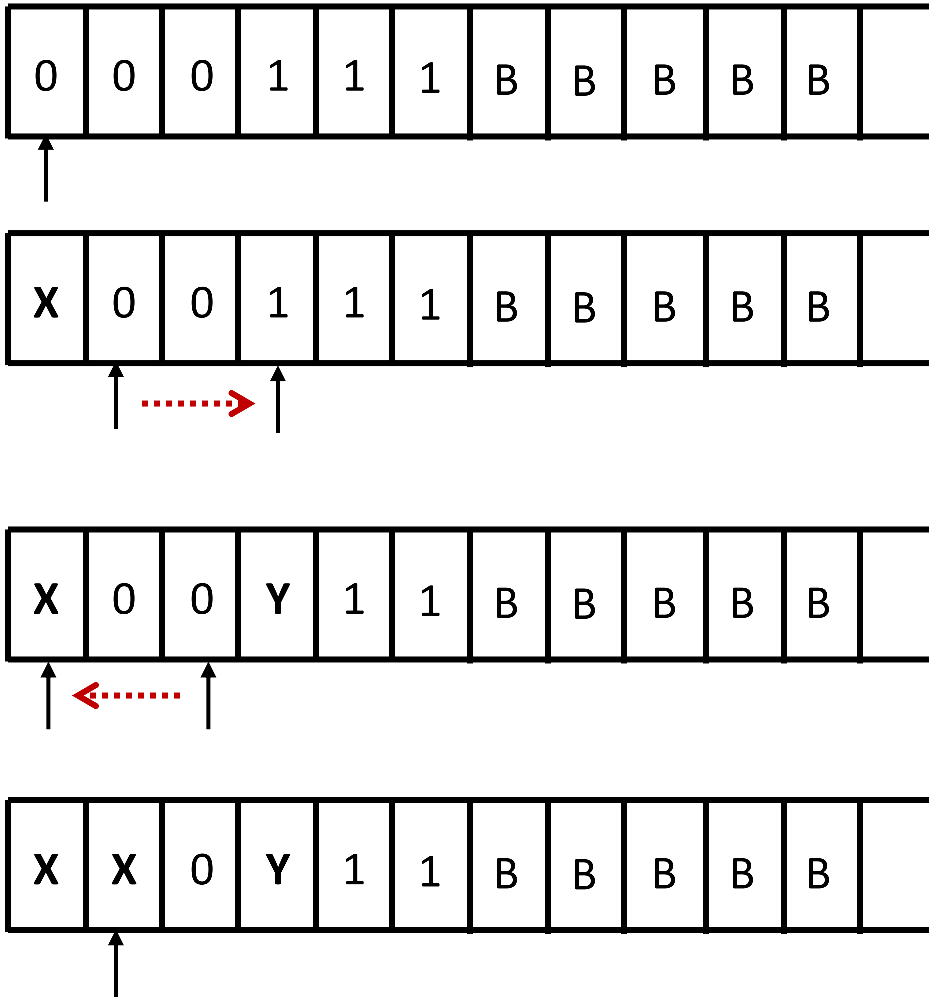


All missing transitions lead to the reject state and the input is rejected when this state is reached.

Turing Machines

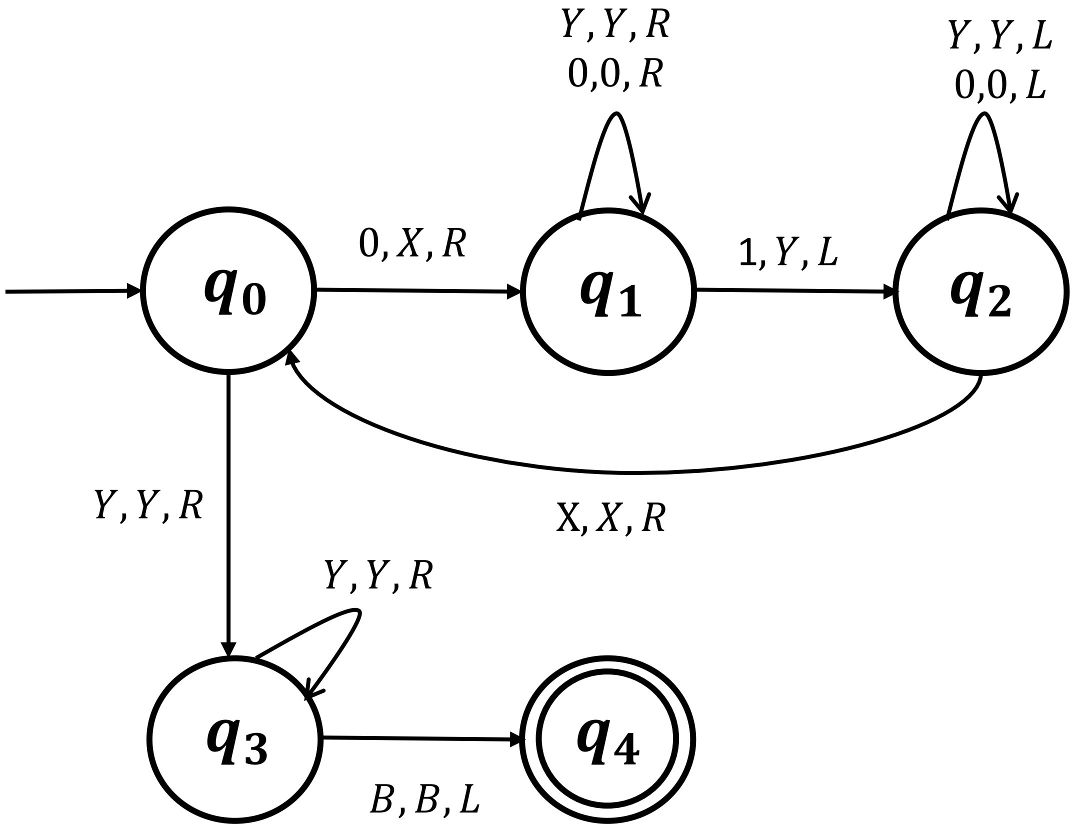
Example: Let $L = \{0^n 1^n | n \geq 1\}$

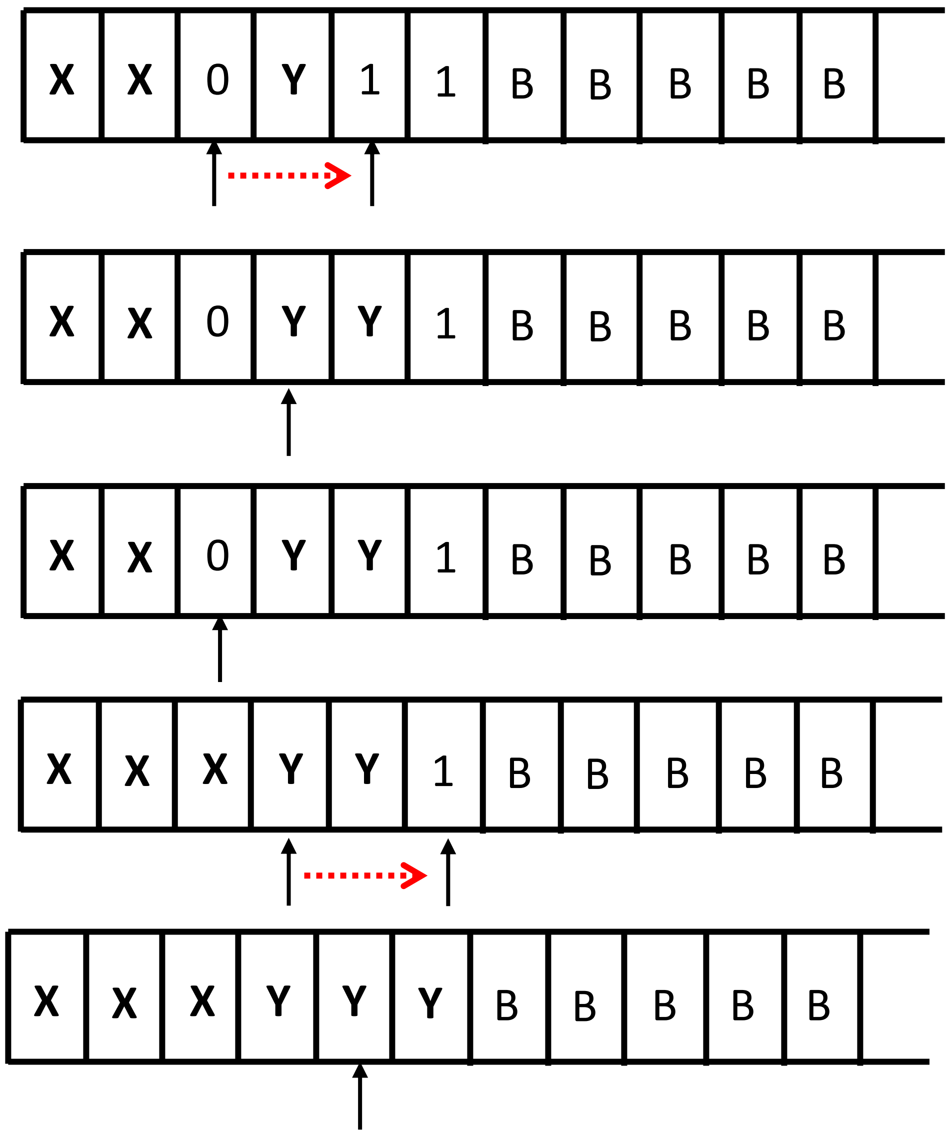




Turing Machines

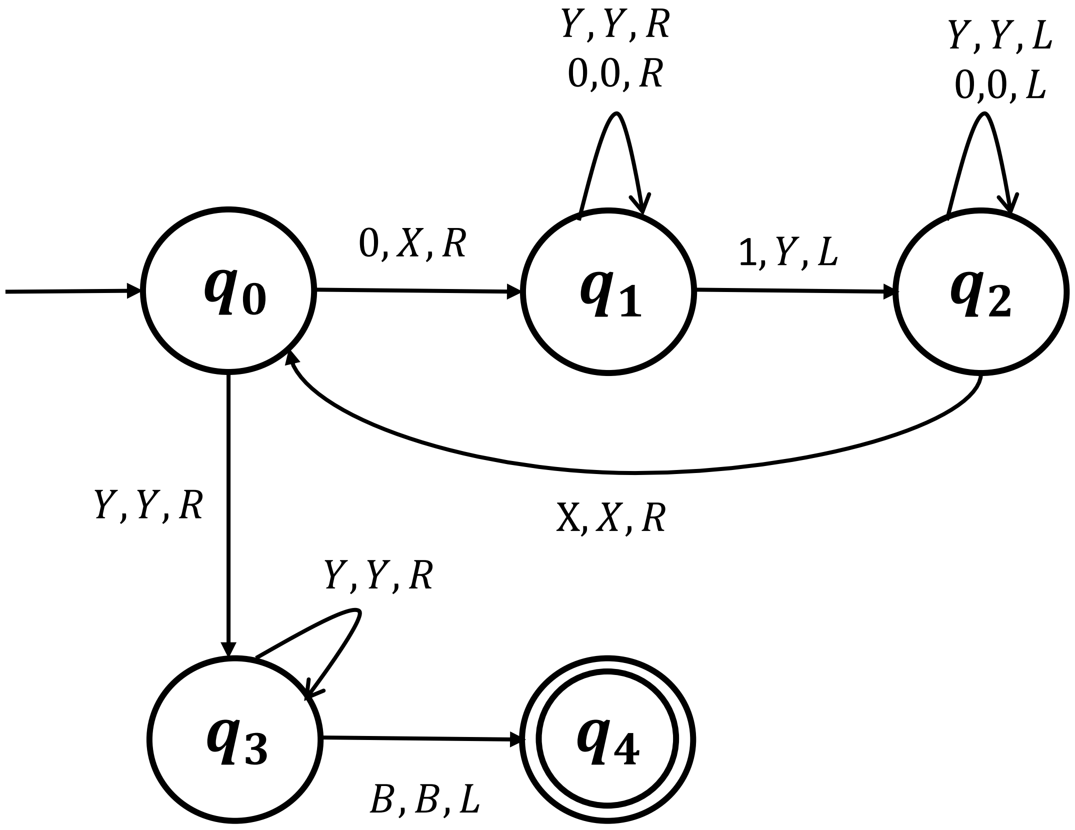
Example: Let $L = \{0^n 1^n | n \geq 1\}$

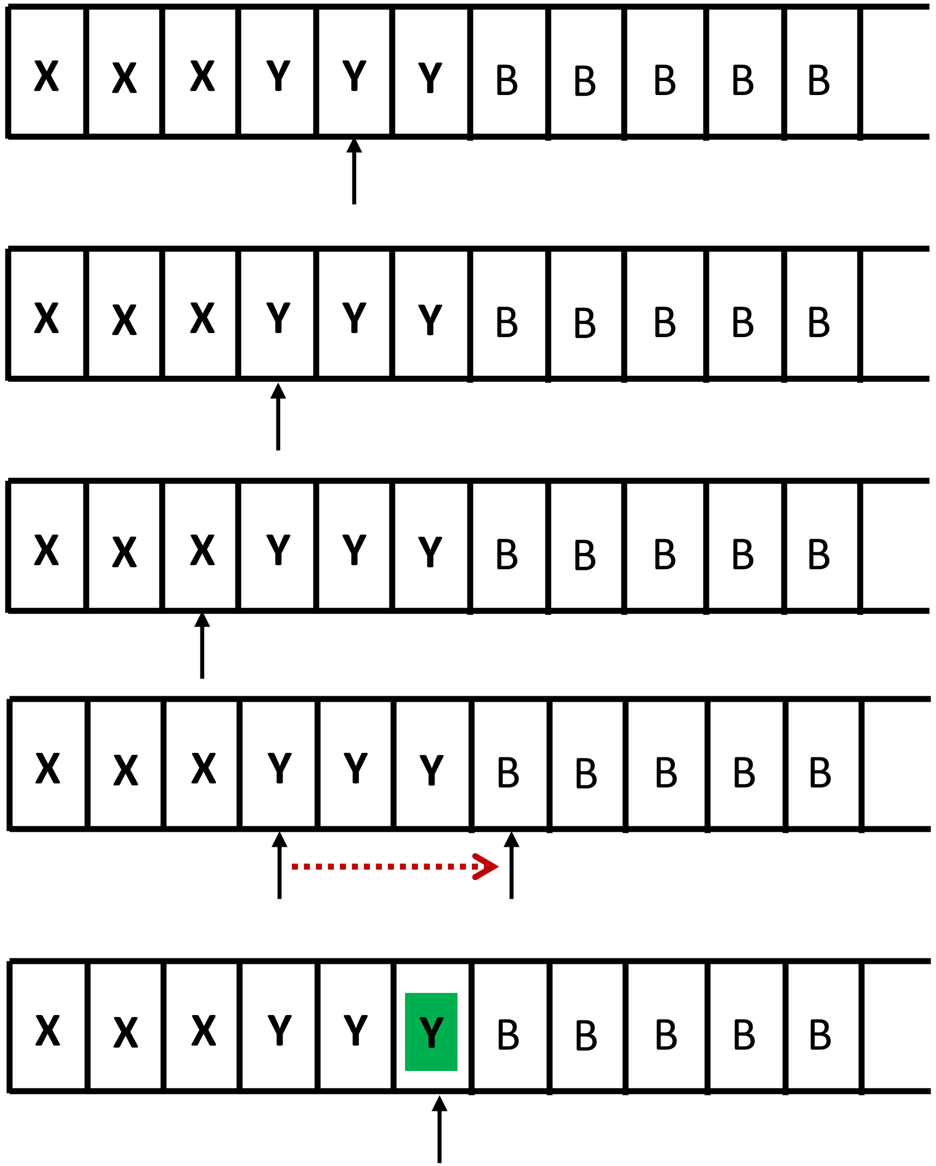




Turing Machines

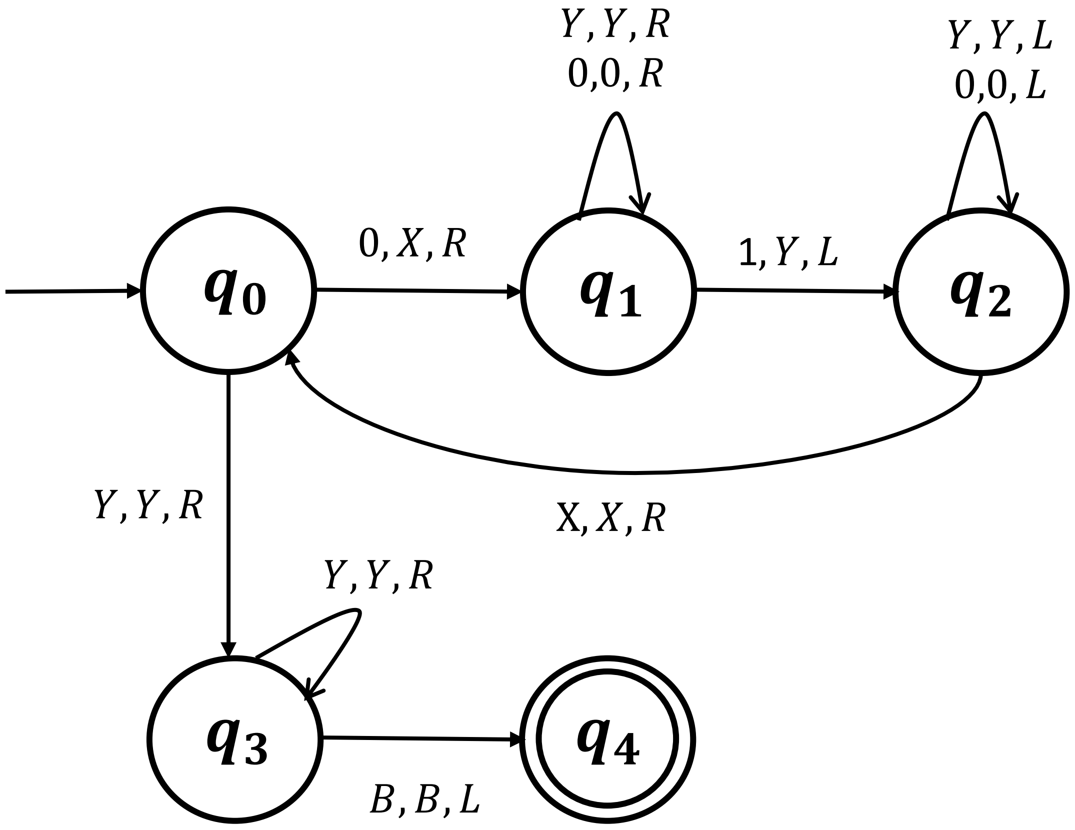
Example: Let $L = \{0^n 1^n | n \geq 1\}$



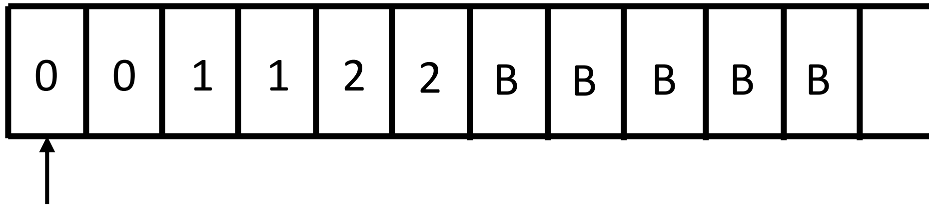


Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$

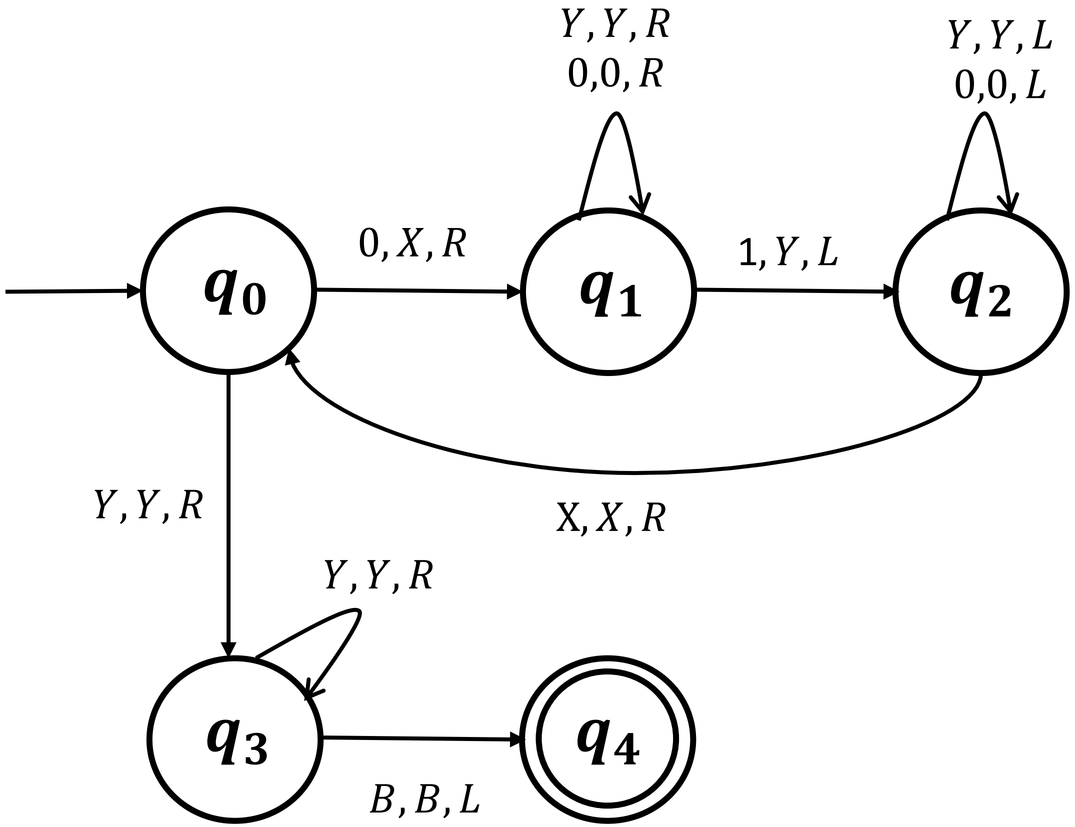


- We will start off with the TM for $\{0^n1^n\}$ and construct the TM for $\{0^n1^n2^n\}$
- **Very similar to the TM for $\{0^n1^n\}$**, except now the FSM would count the number of $2'$s as well. So it marks the $2's$ with another symbol (say Z)

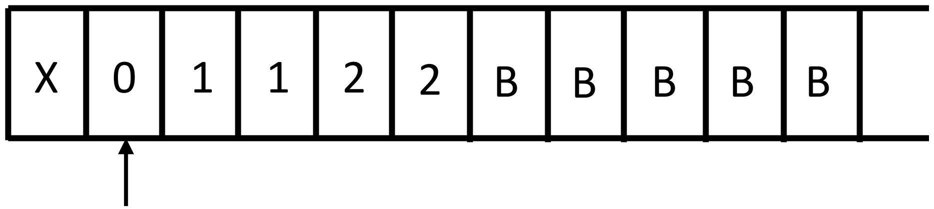


Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

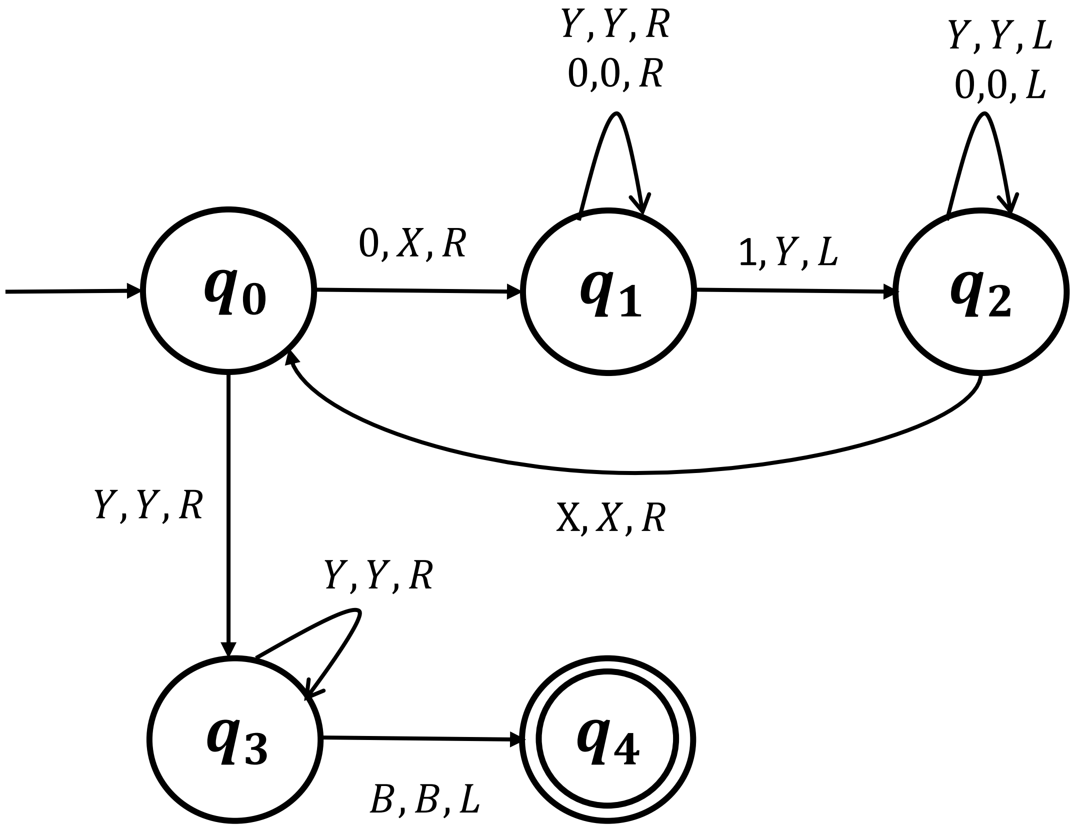


- We will start off with the TM for $\{0^n 1^n\}$ and construct the TM for $\{0^n 1^n 2^n\}$
- **Very similar to the TM for $\{0^n 1^n\}$**, except now the FSM would count the number of $2'$s as well. So it marks the $2's$ with another symbol (say Z)



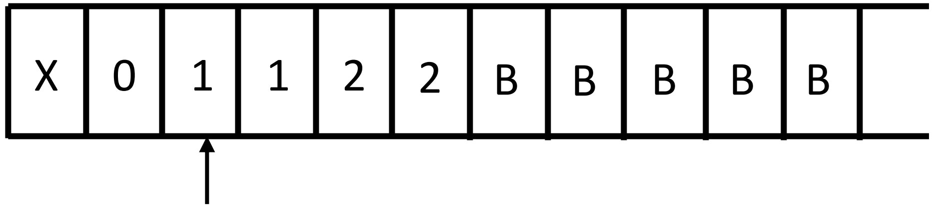
Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$



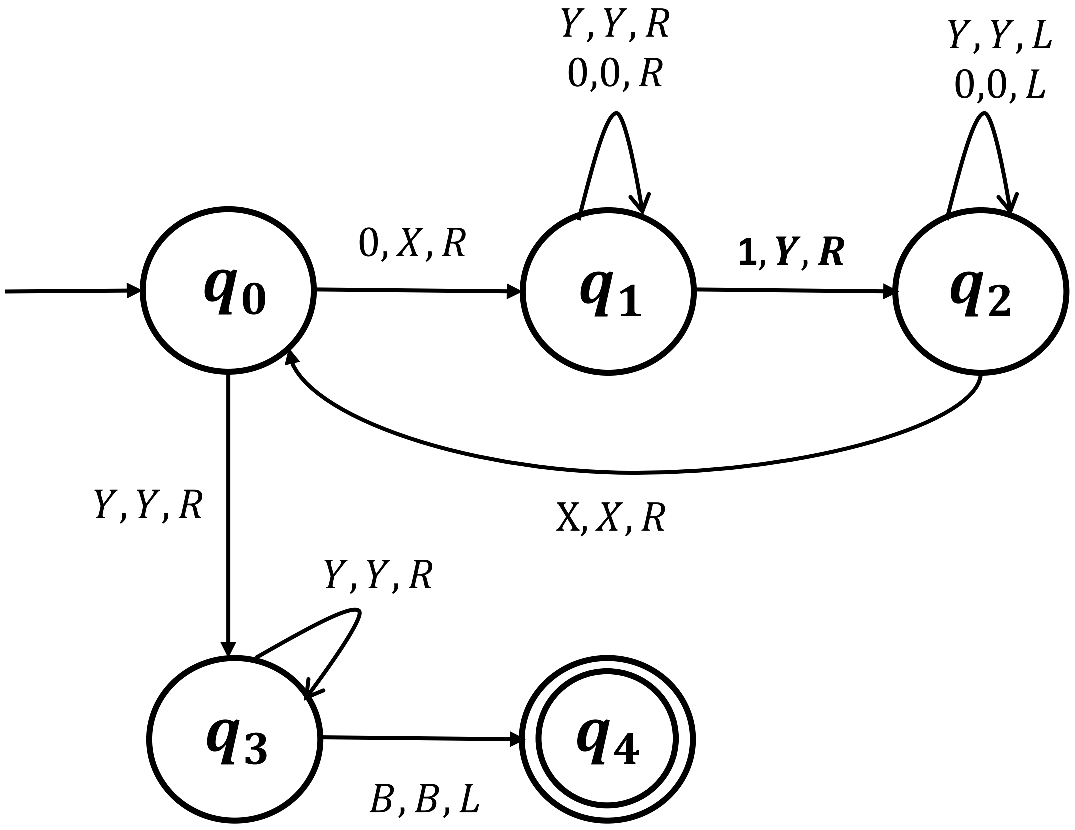
- We will start off with the TM for $\{0^n 1^n\}$ and construct the TM for $\{0^n 1^n 2^n\}$

- **Very similar to the TM for $\{0^n 1^n\}$**, except now the FSM would count the number of $2's$ as well. So it marks the $2's$ with another symbol (say Z)

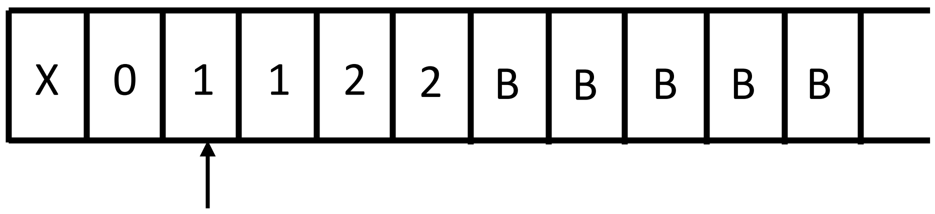


Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

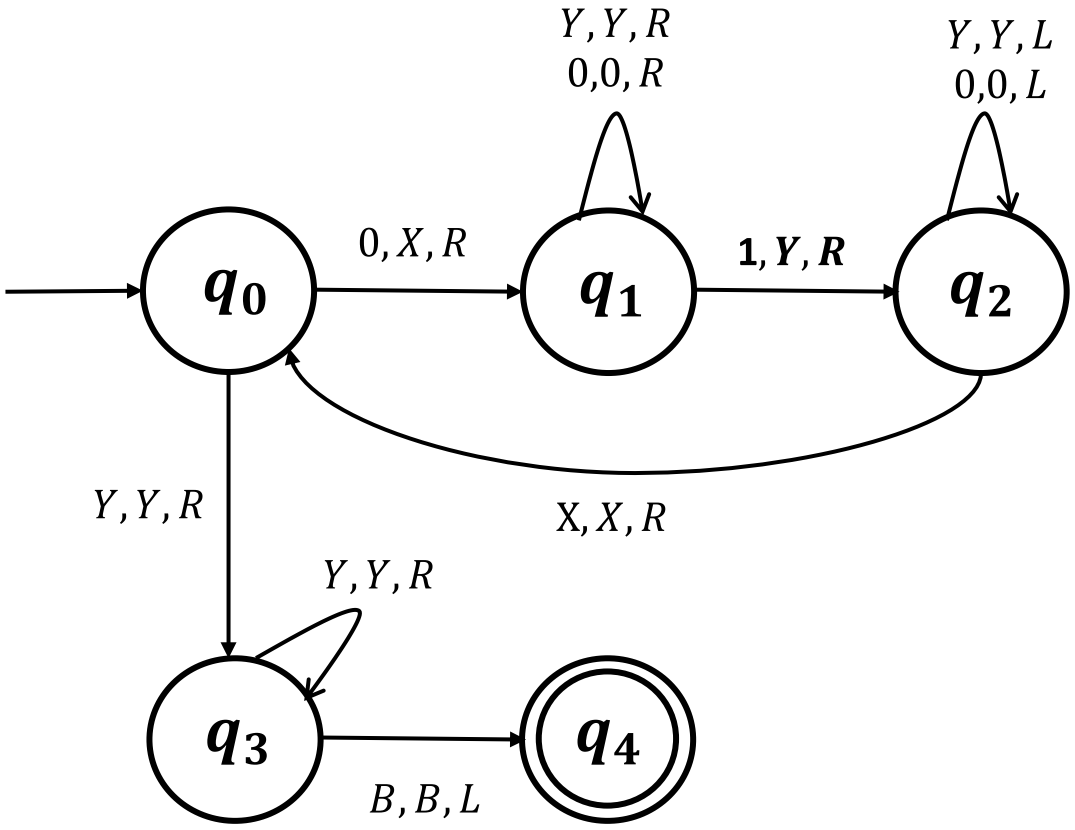


- Continue to go right to mark the next 2 with a Z.

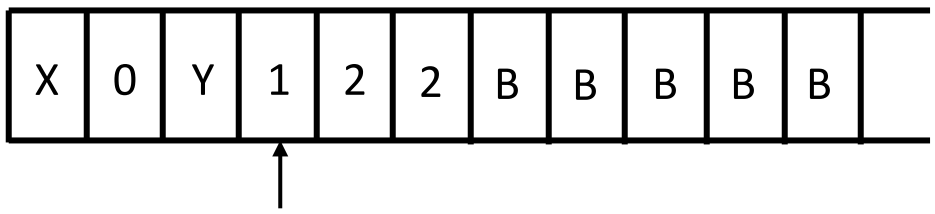


Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$

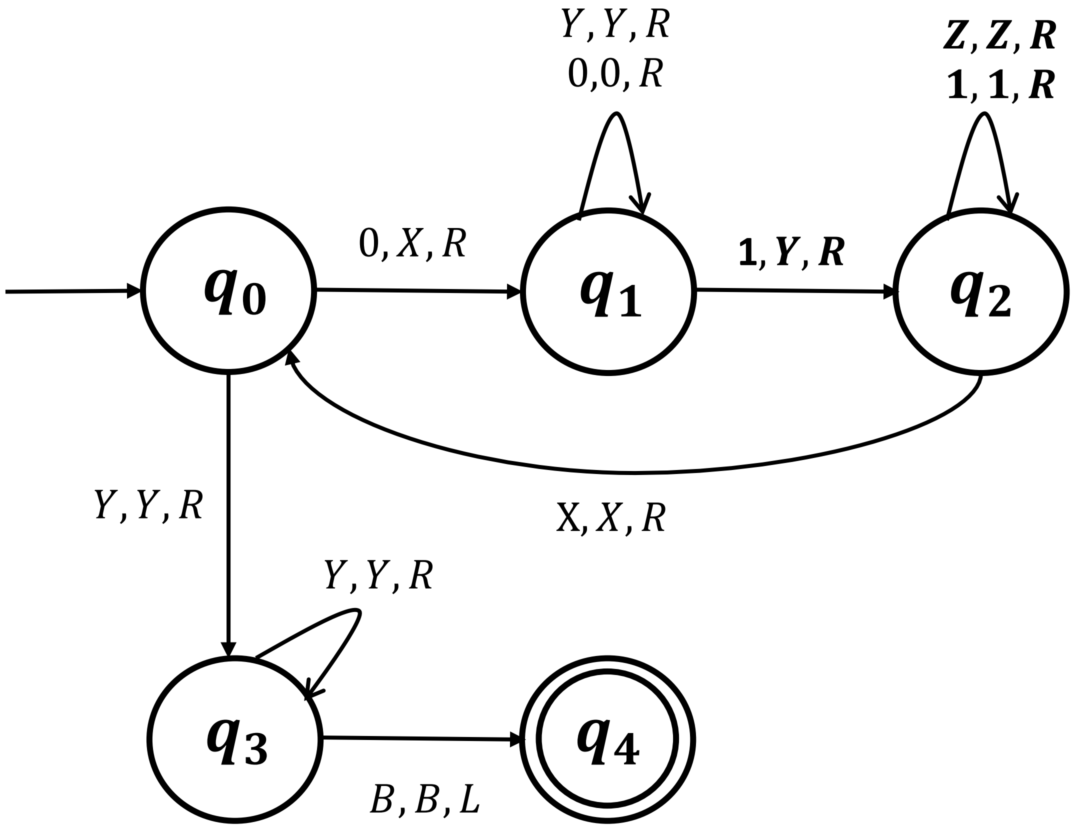


- Continue to go right to mark the next 2 with a Z.

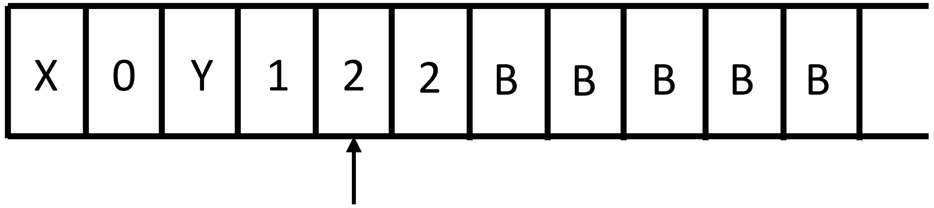


Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

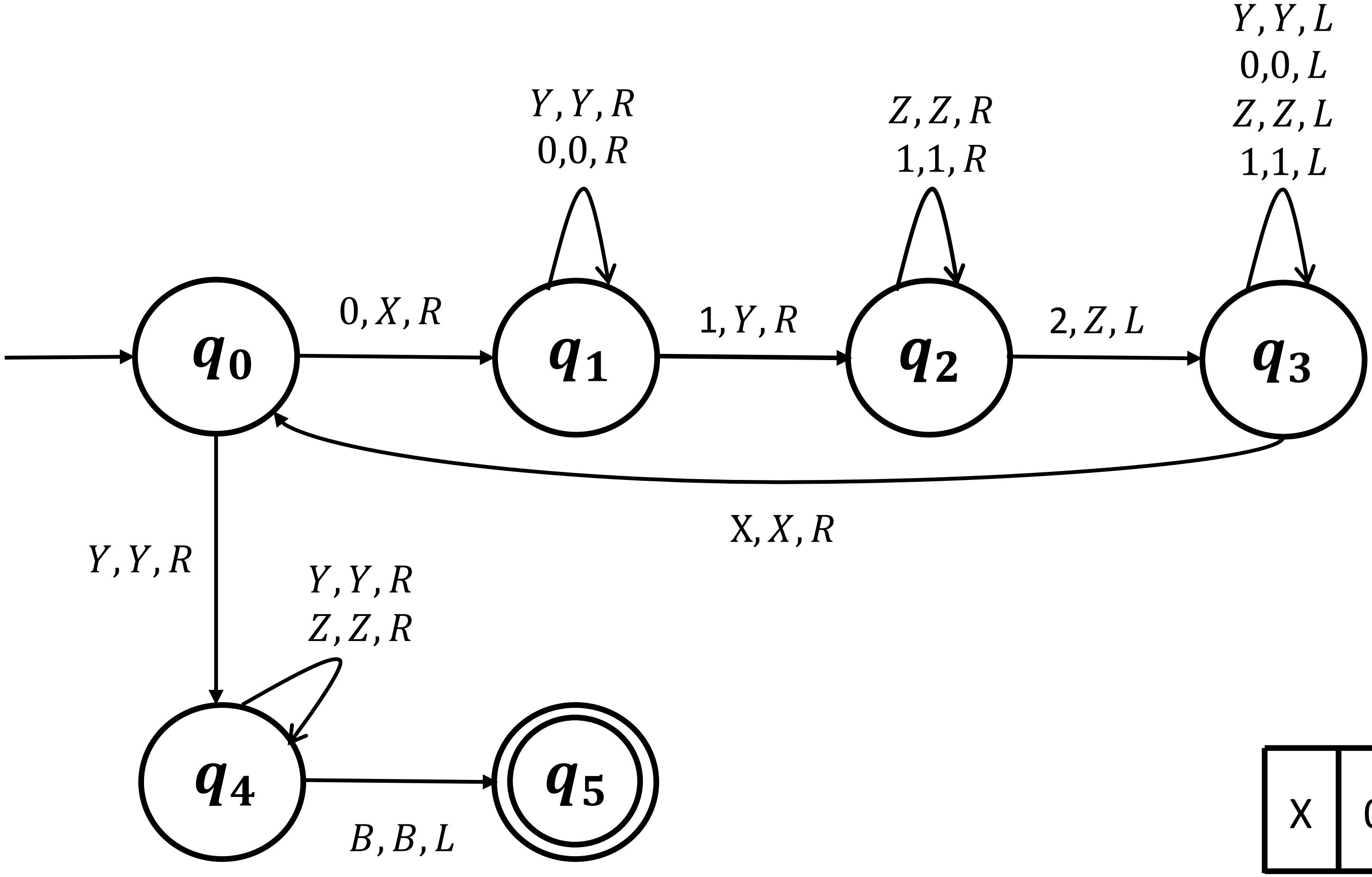


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.

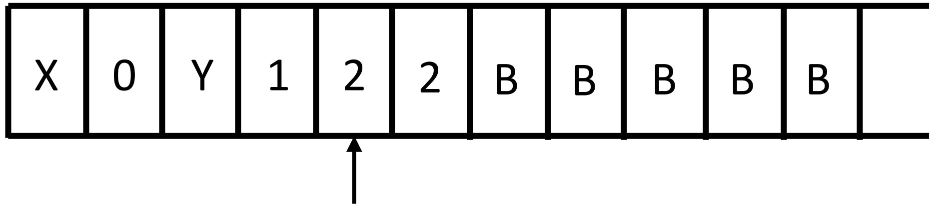


Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$

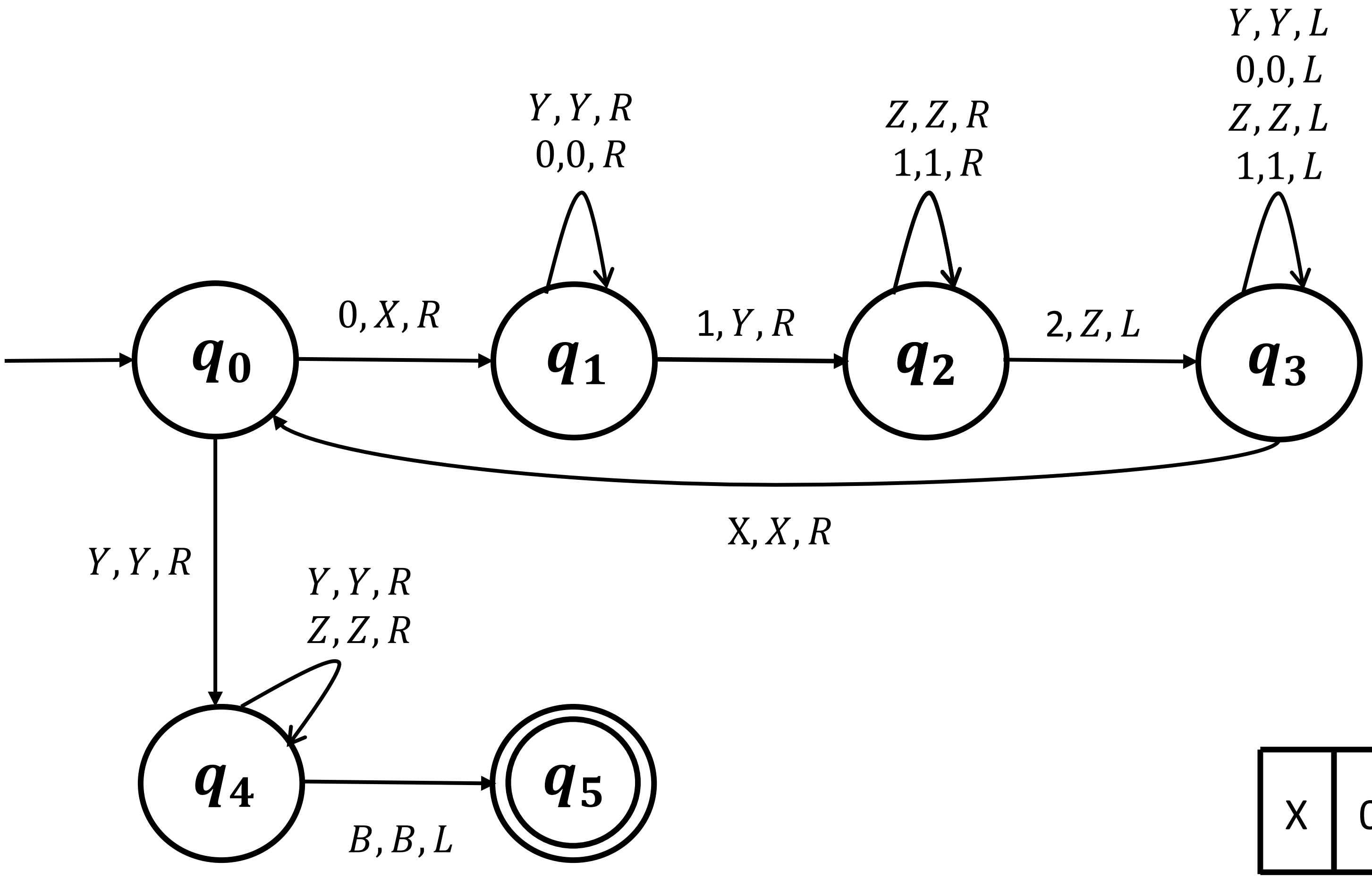


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.

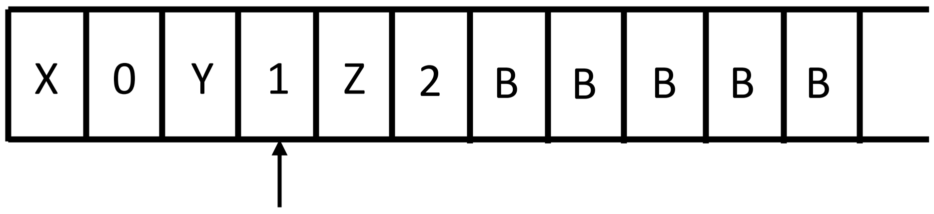


Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$

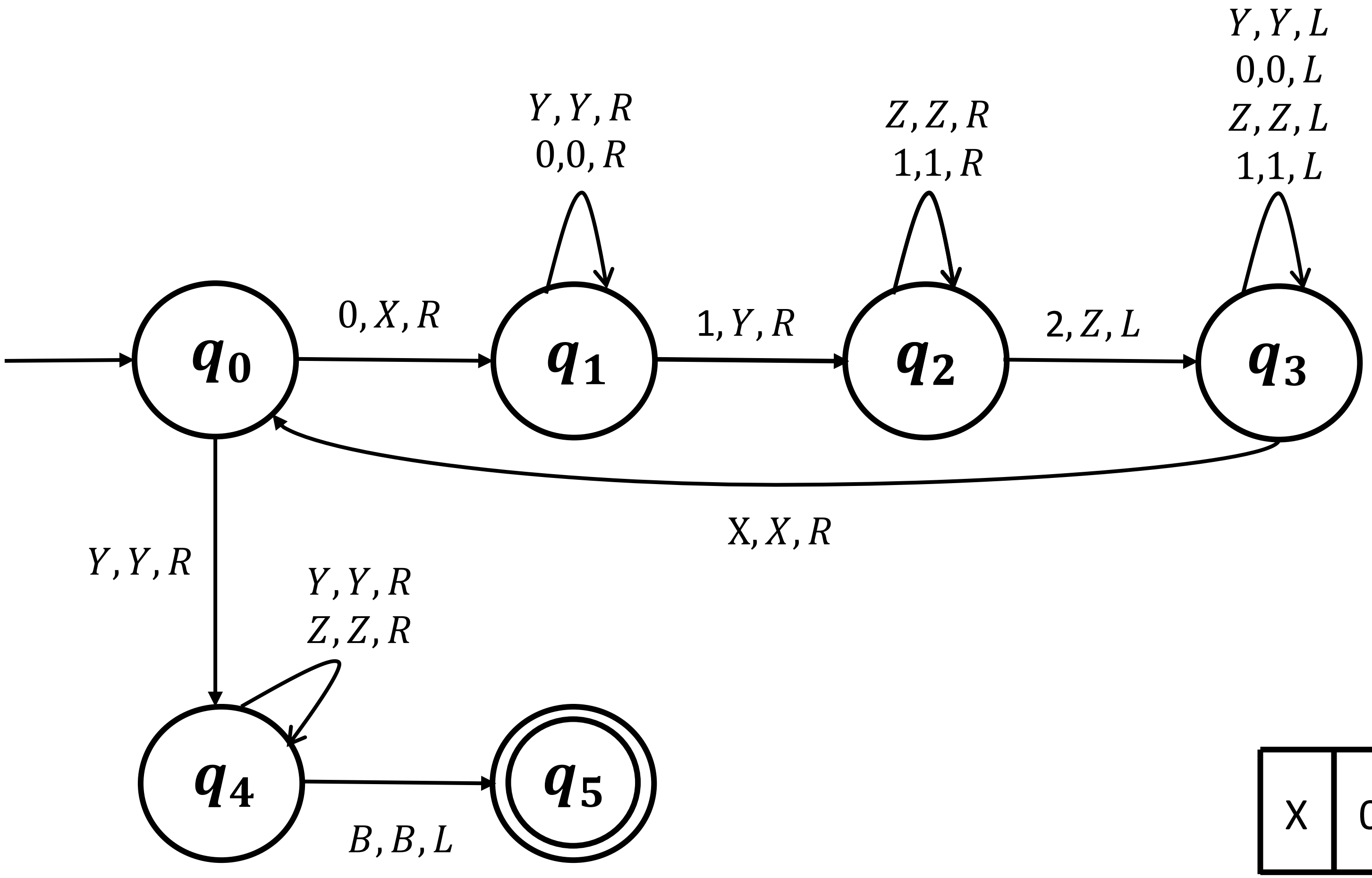


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.

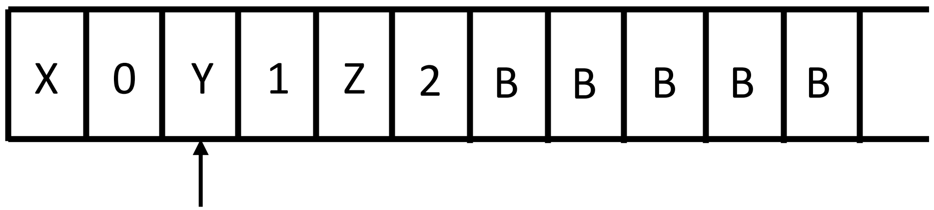


Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

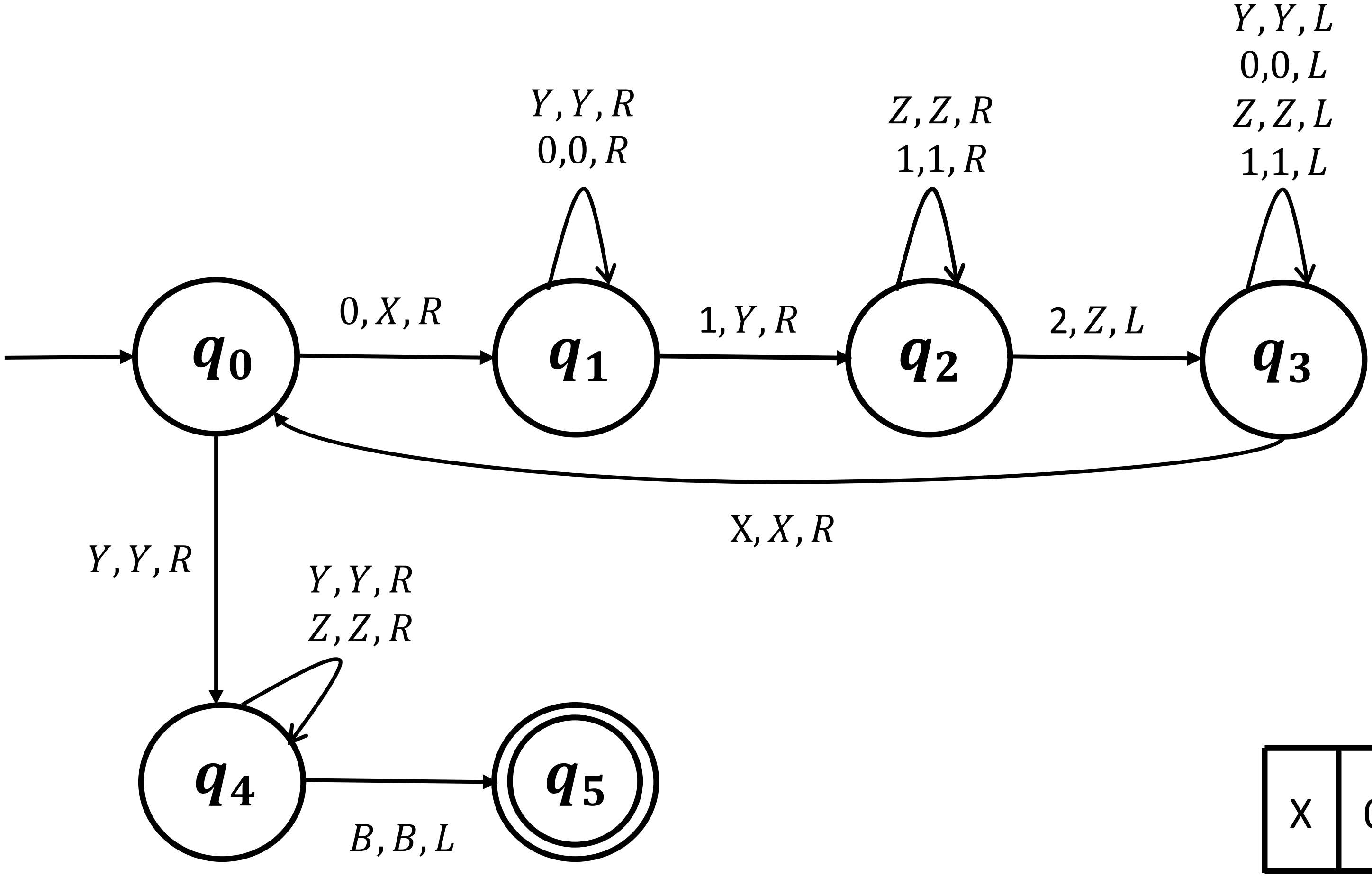


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.

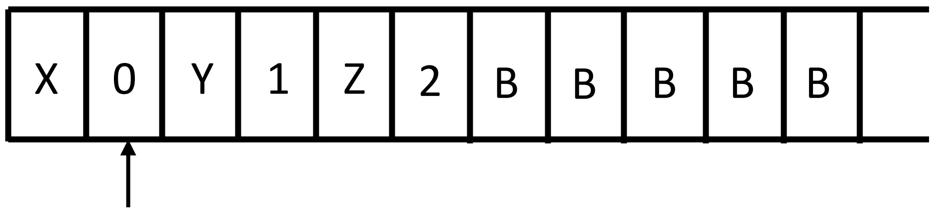


Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

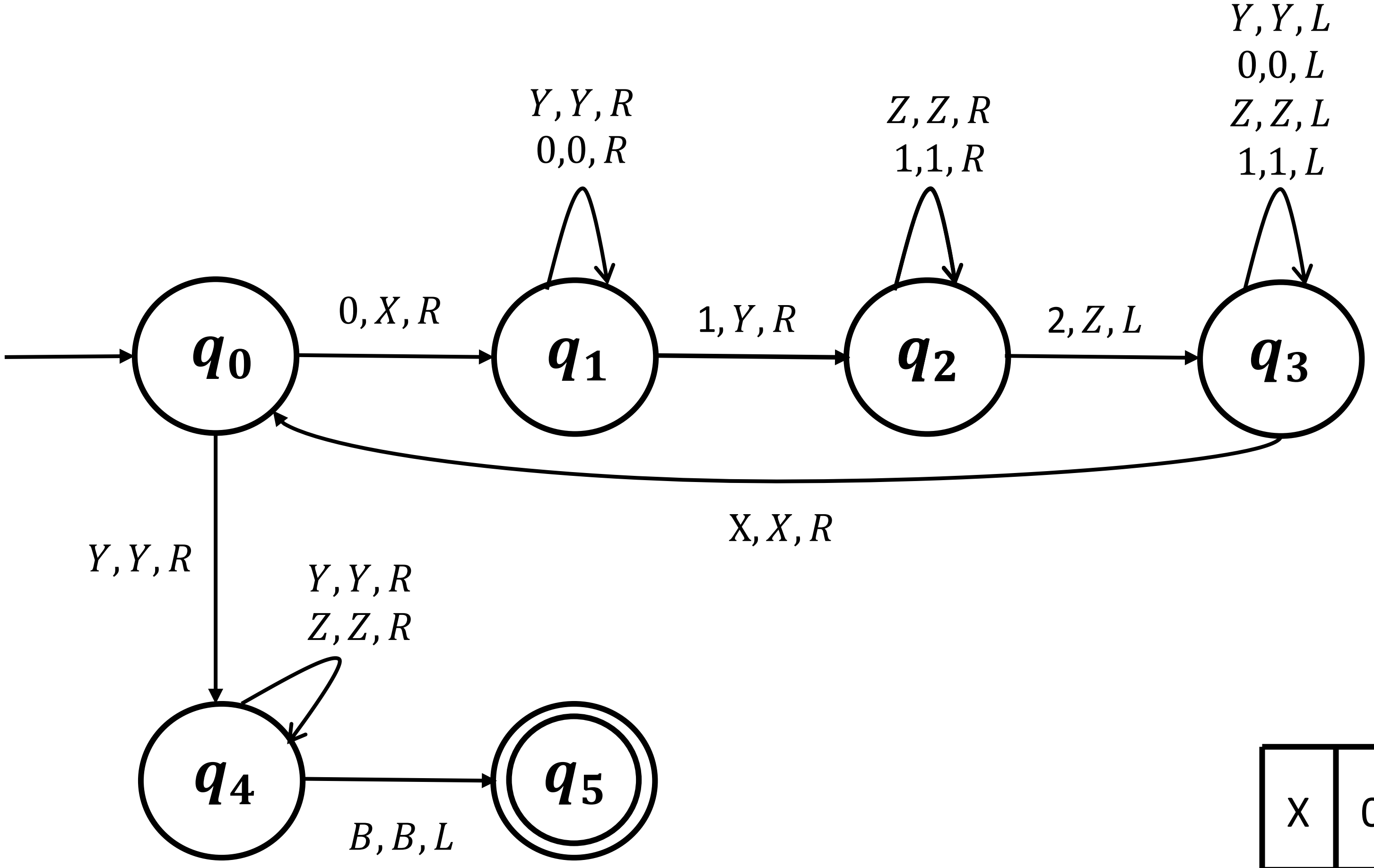


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.

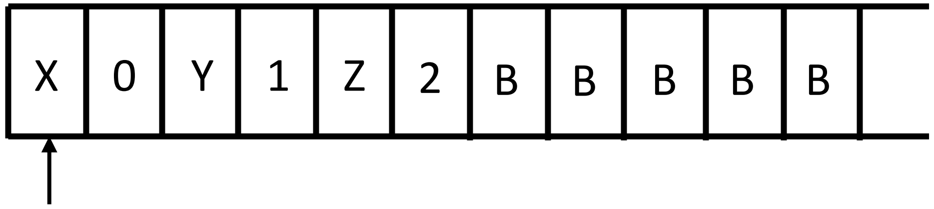


Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

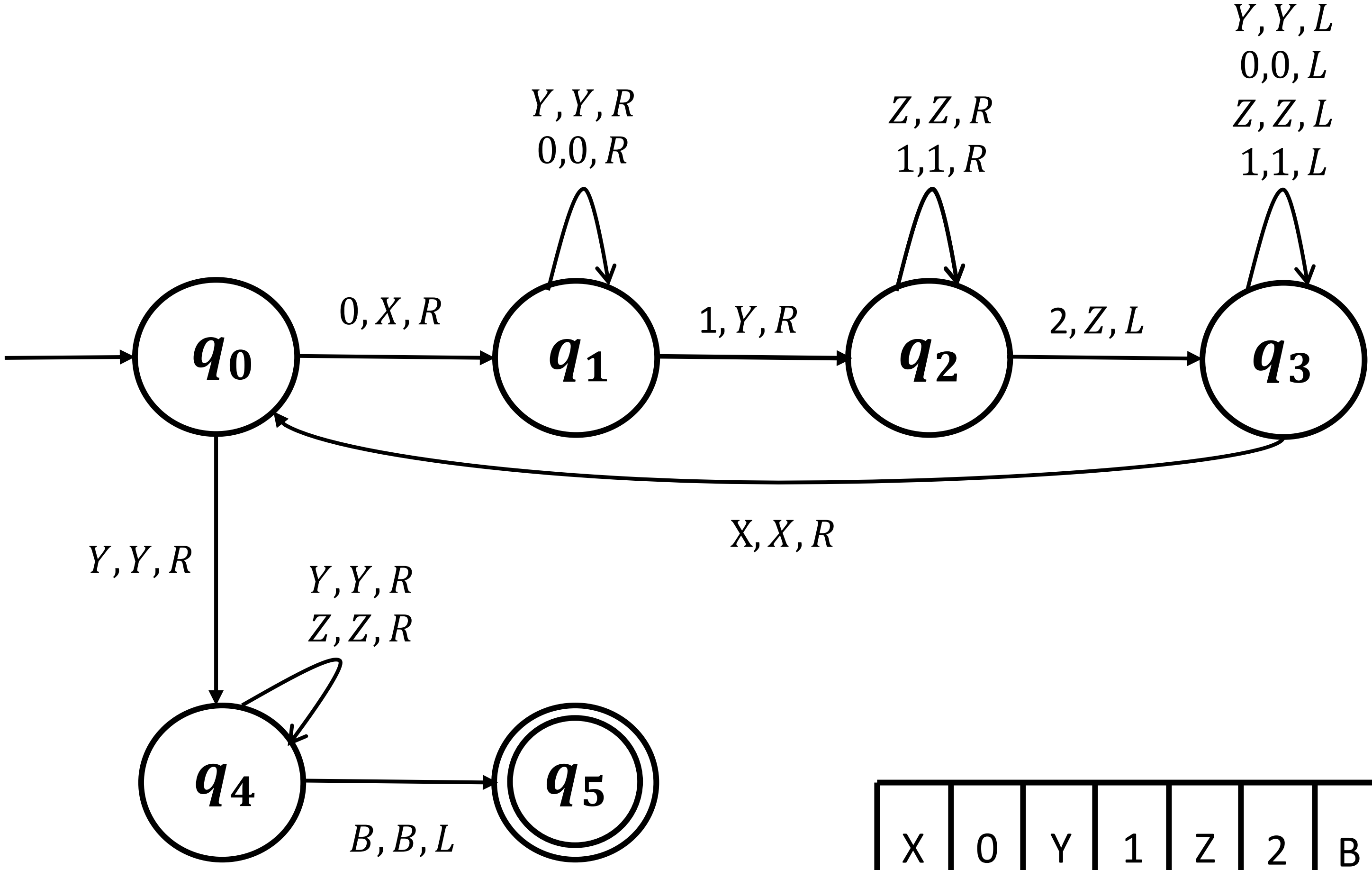


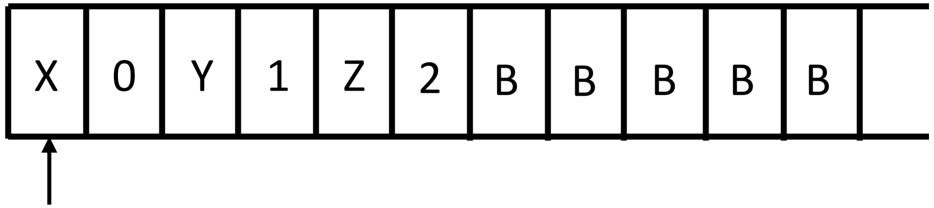
- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.



Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$





- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

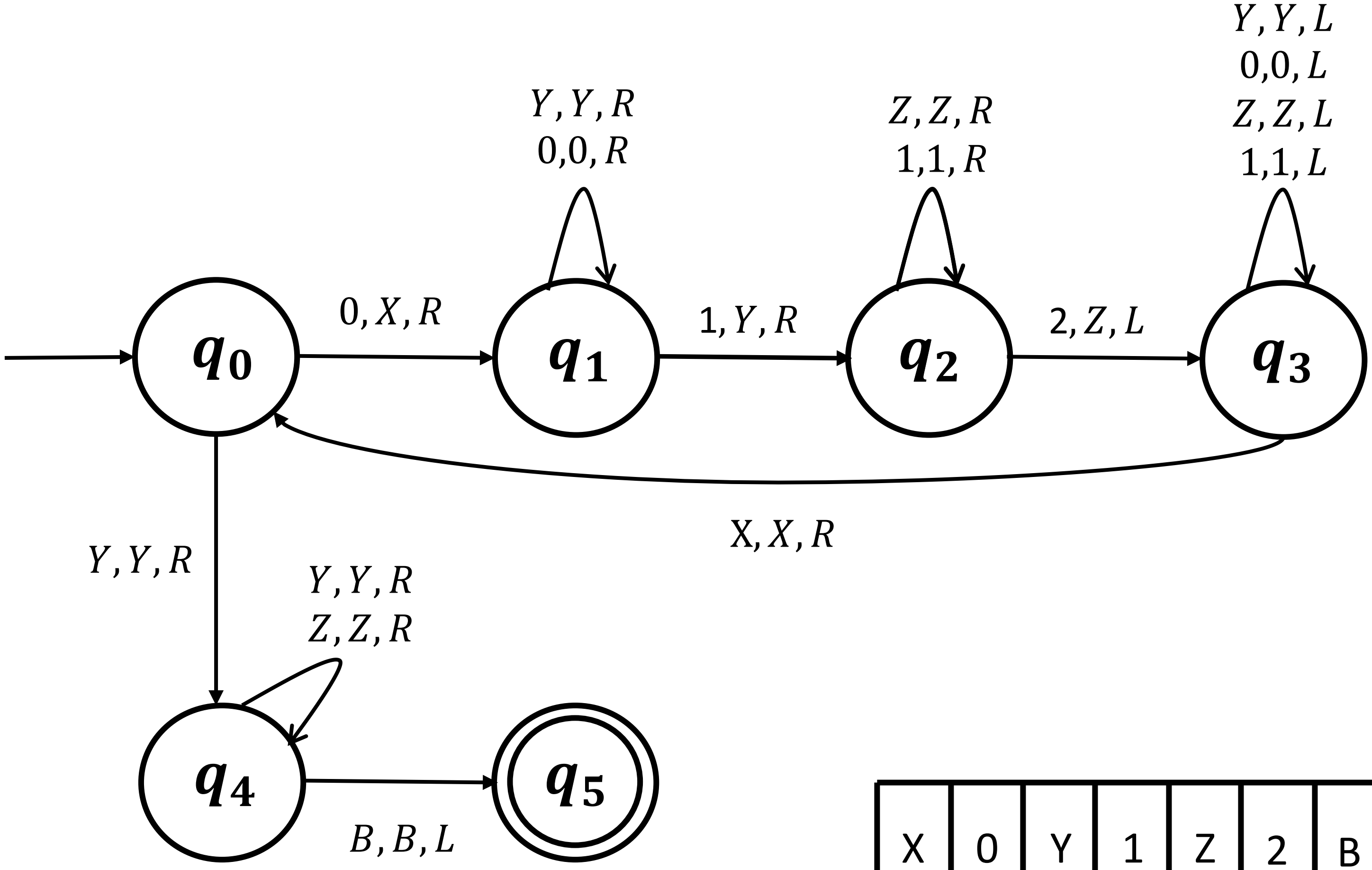
or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.

Turing Machines

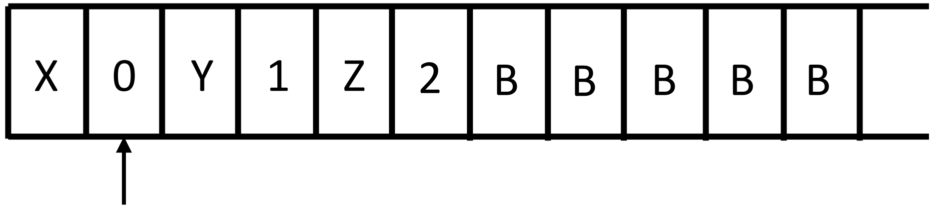
Example: Let $L = \{0^n1^n2^n | n \geq 1\}$



- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

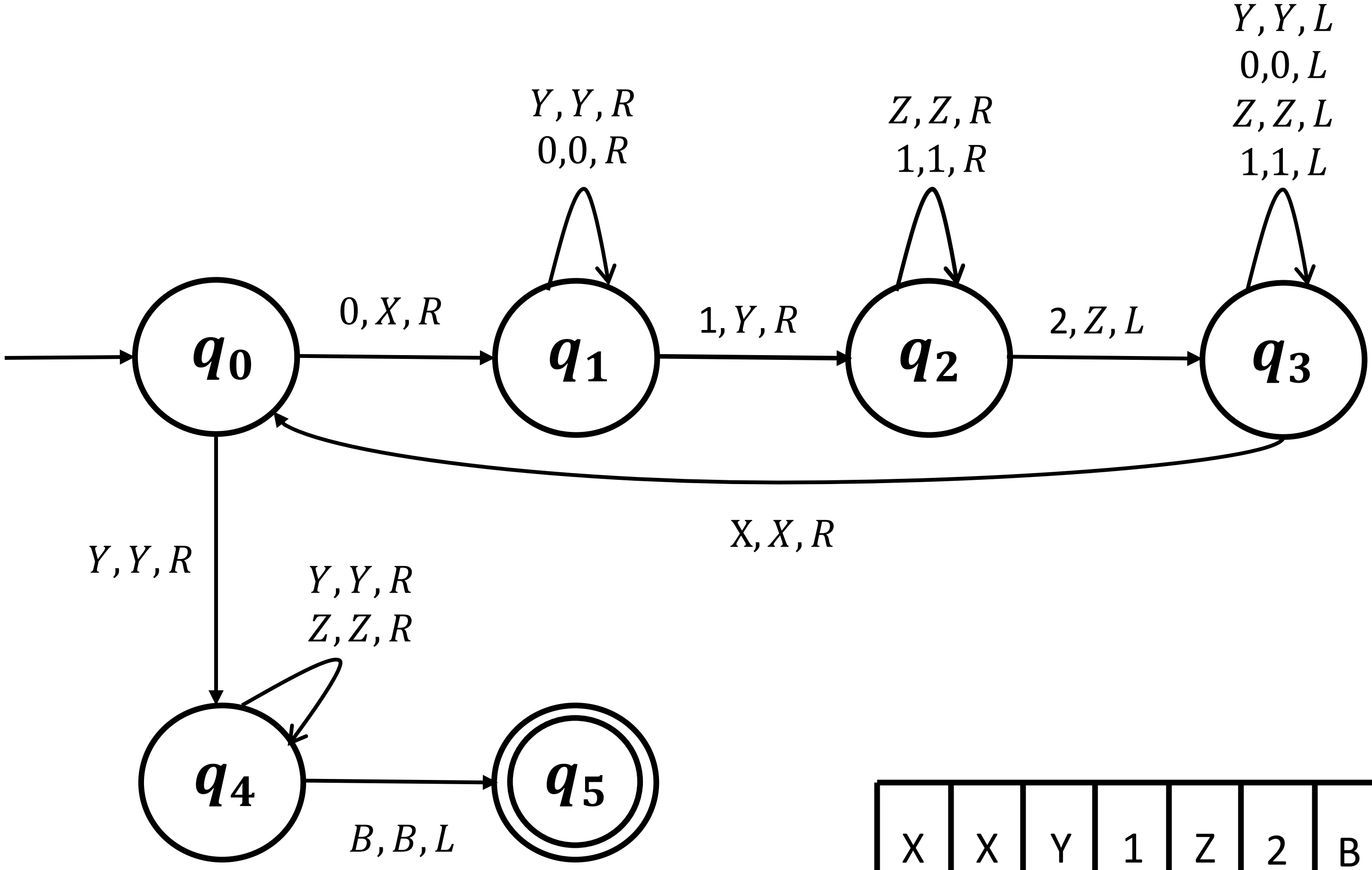
  or
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

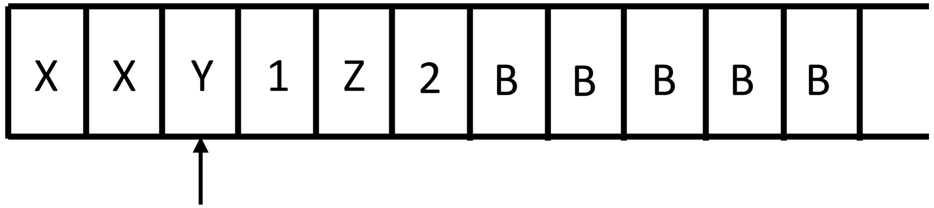
- Move left and accept the input string.



Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$





- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

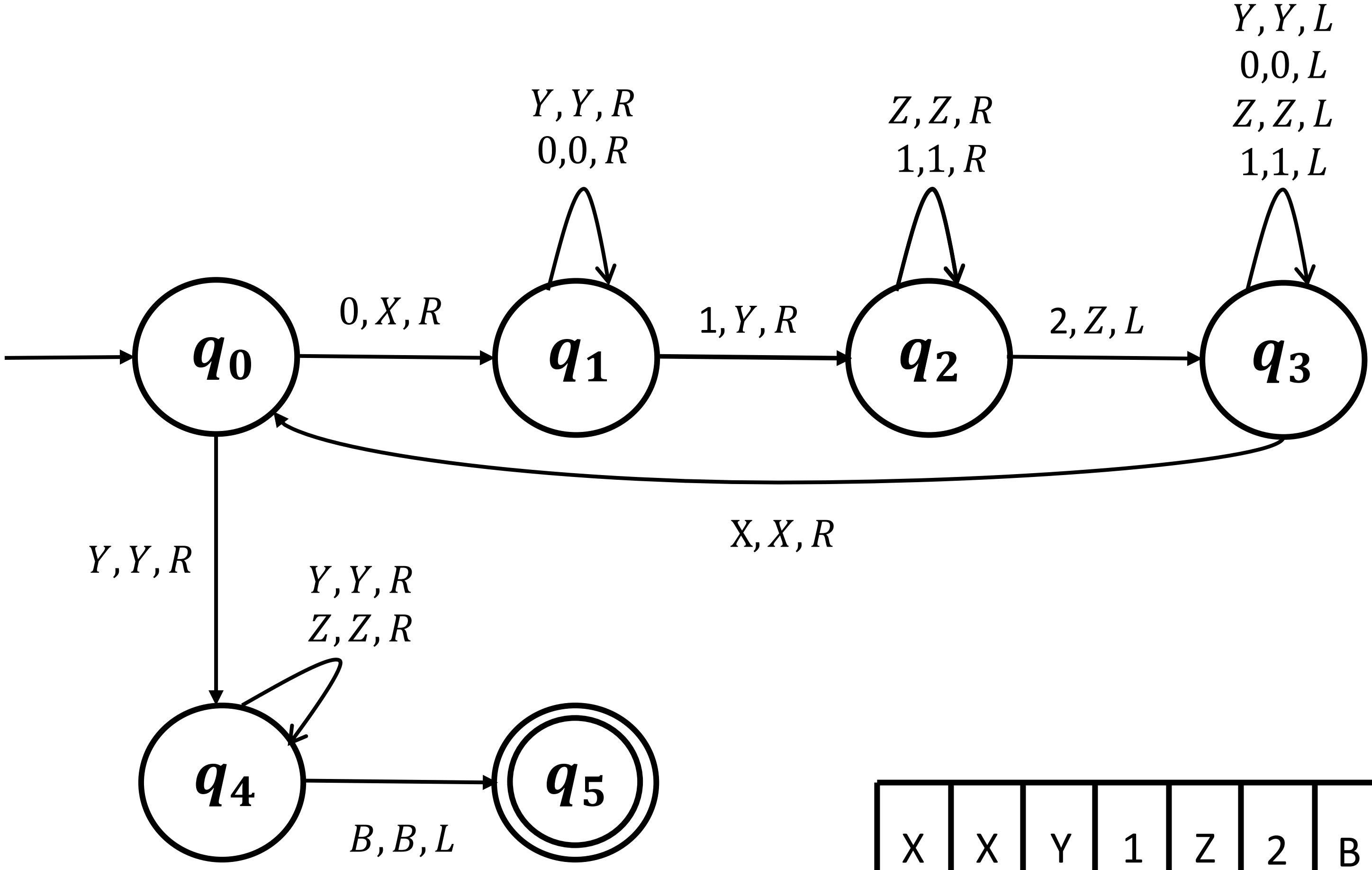
or

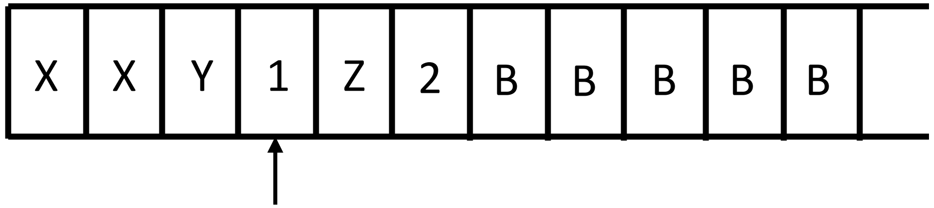
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.

Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$





- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

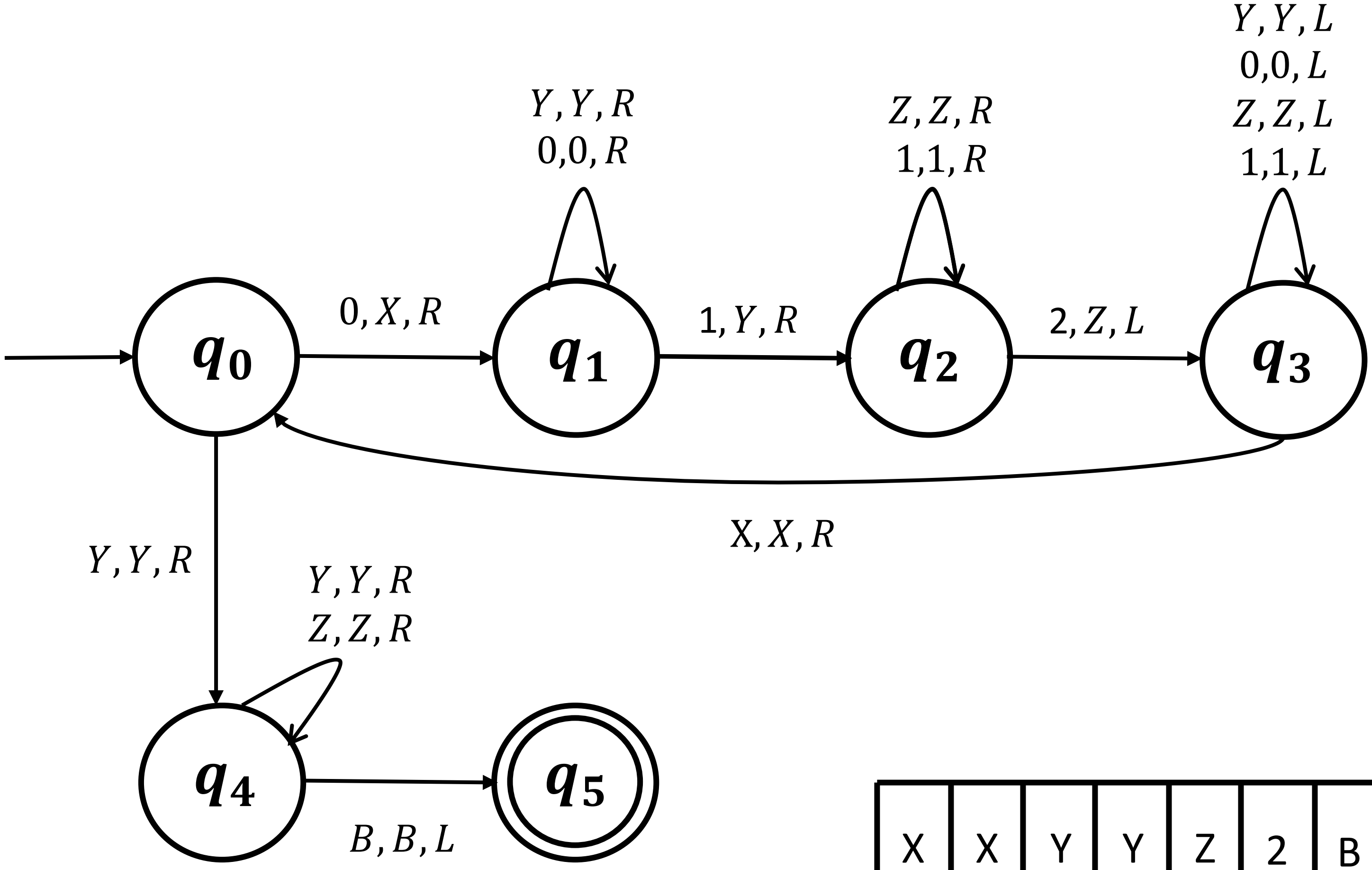
or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.

Turing Machines

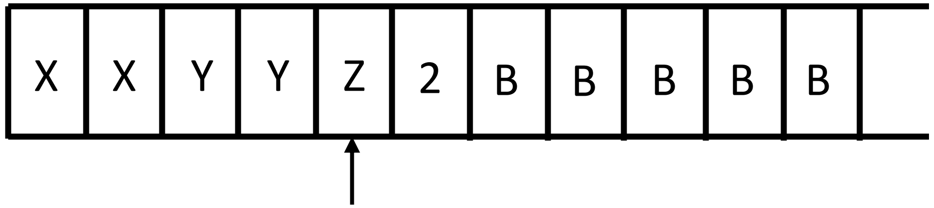
Example: Let $L = \{0^n1^n2^n | n \geq 1\}$



- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

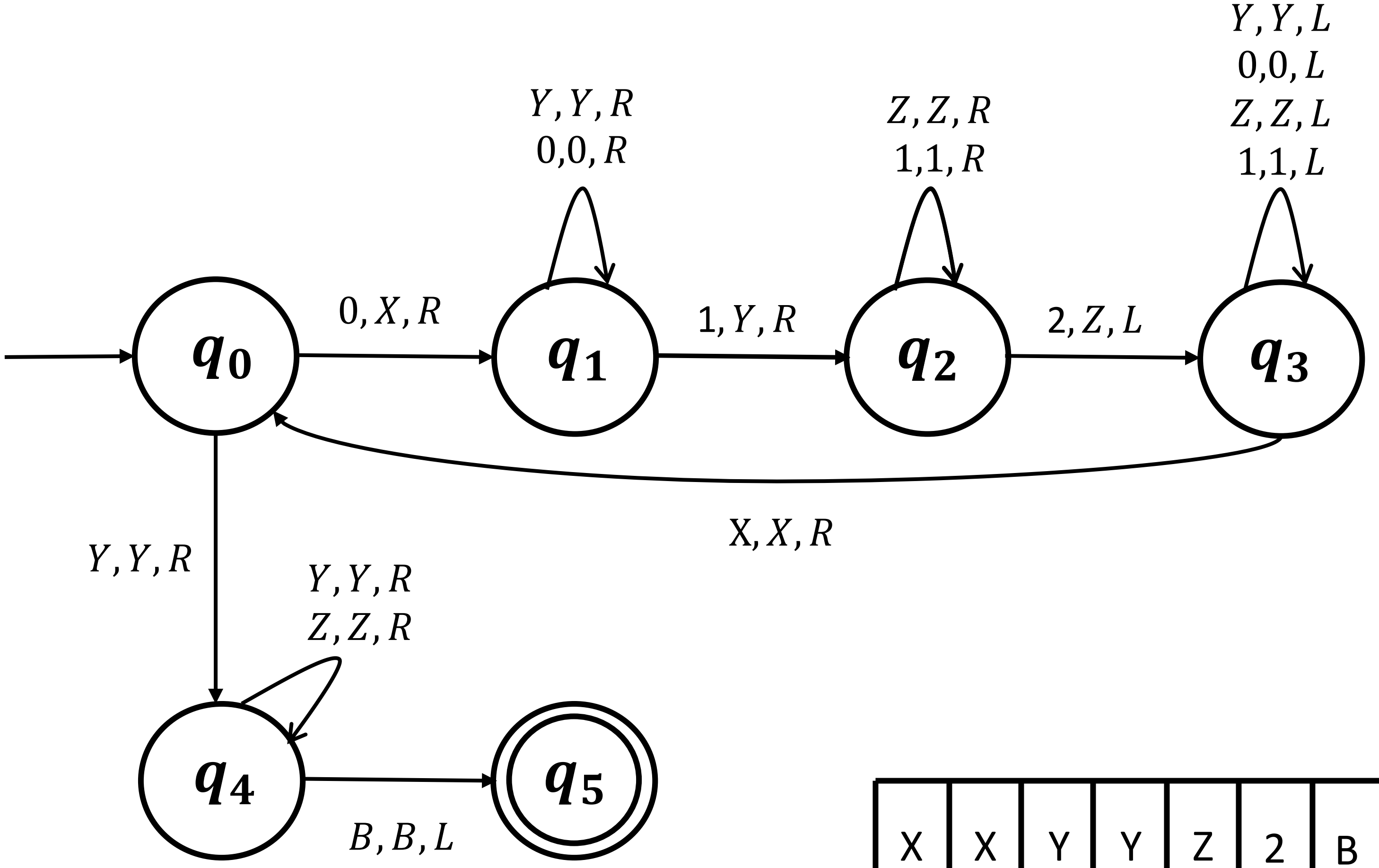
or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.
- Move left and accept the input string.



Turing Machines

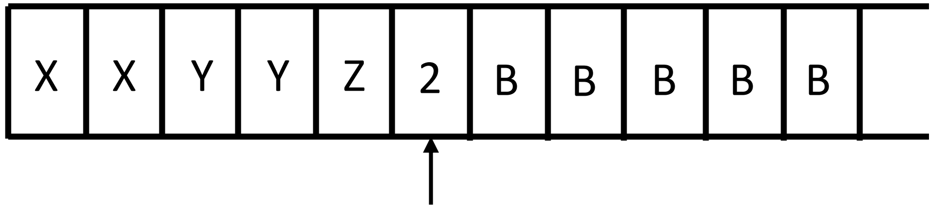
Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$



- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

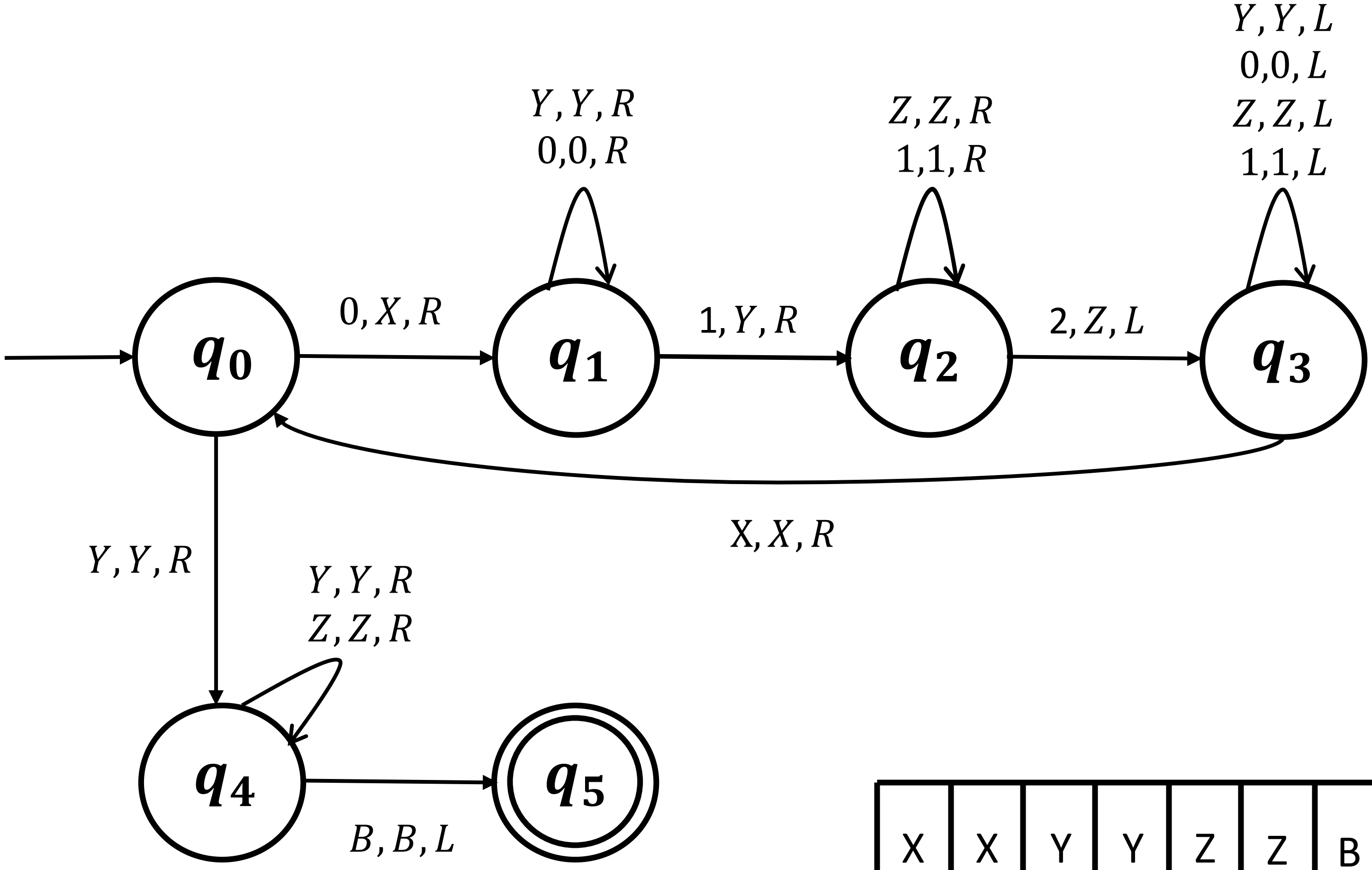
or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.
- Move left and accept the input string.



Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$

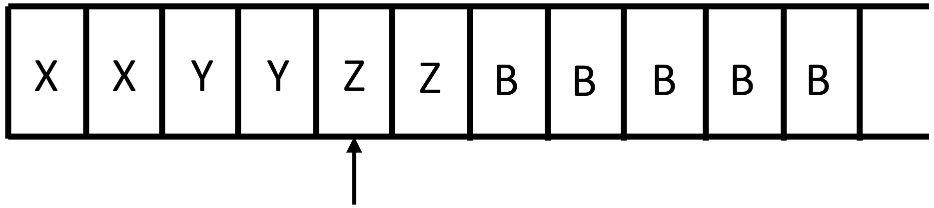


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

or

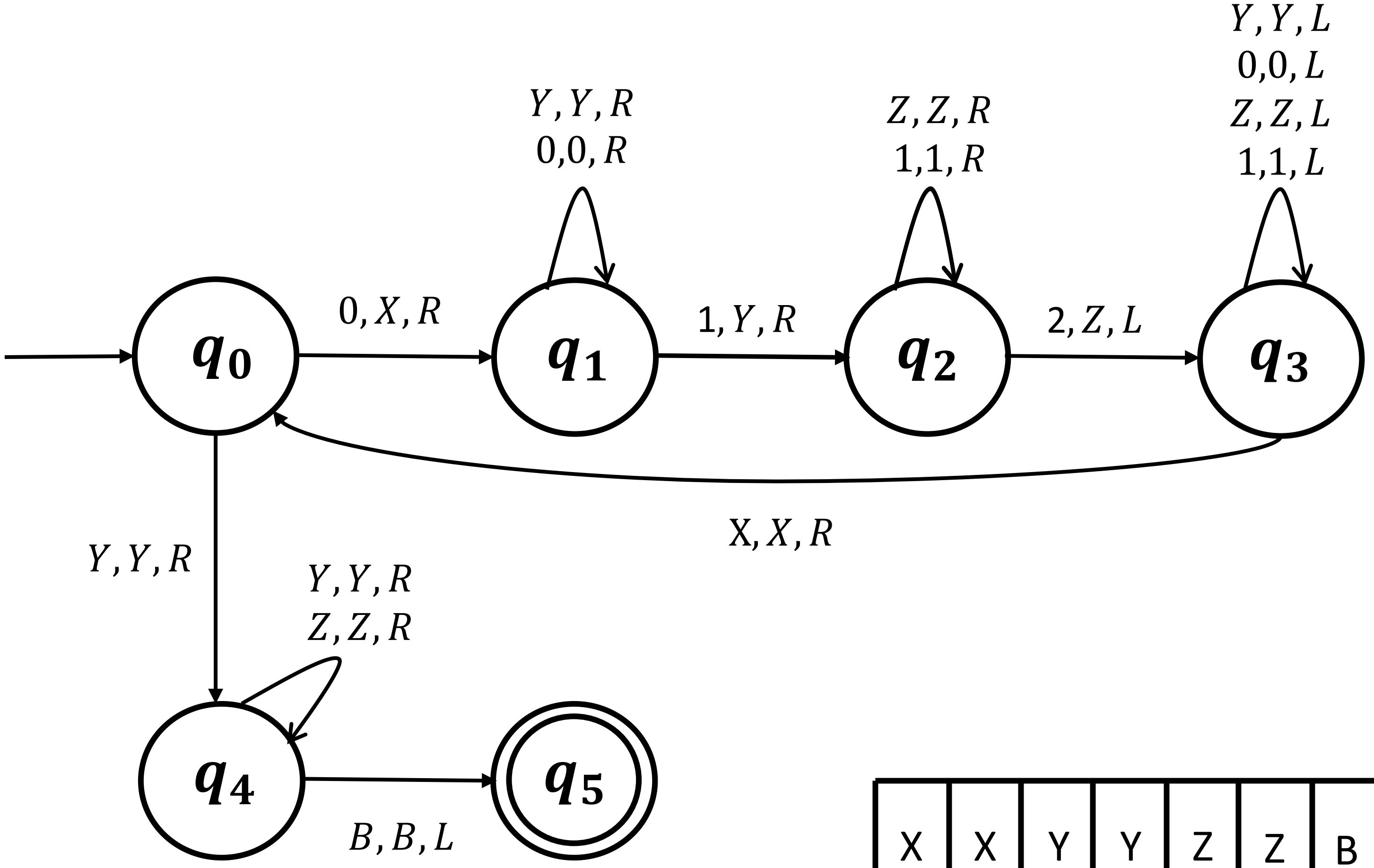
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.



Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

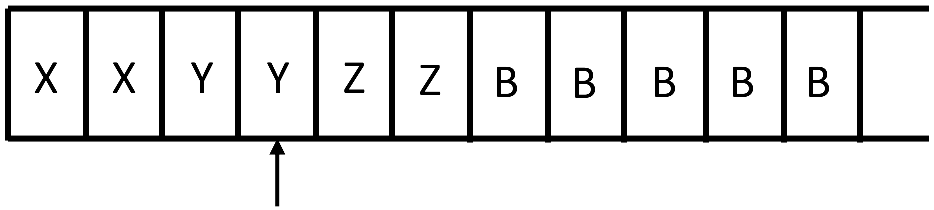


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

or

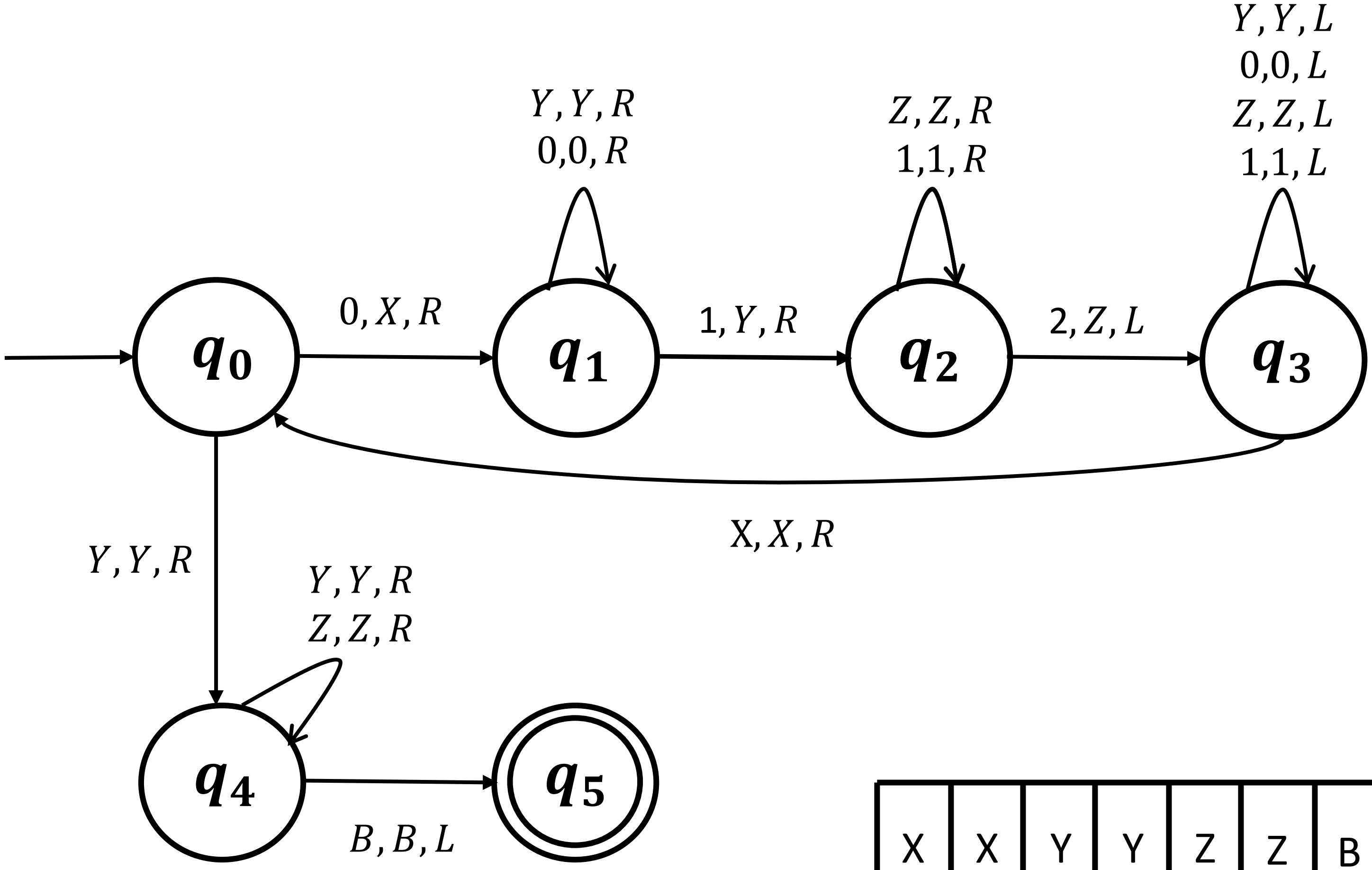
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

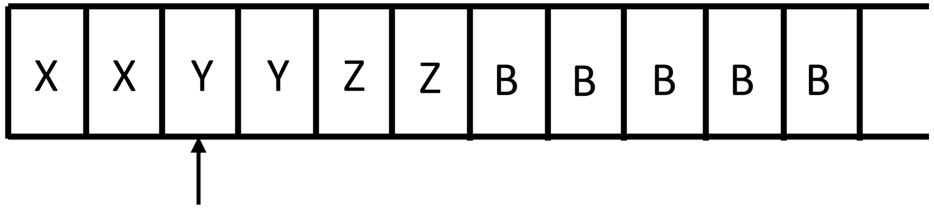
- Move left and accept the input string.



Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$





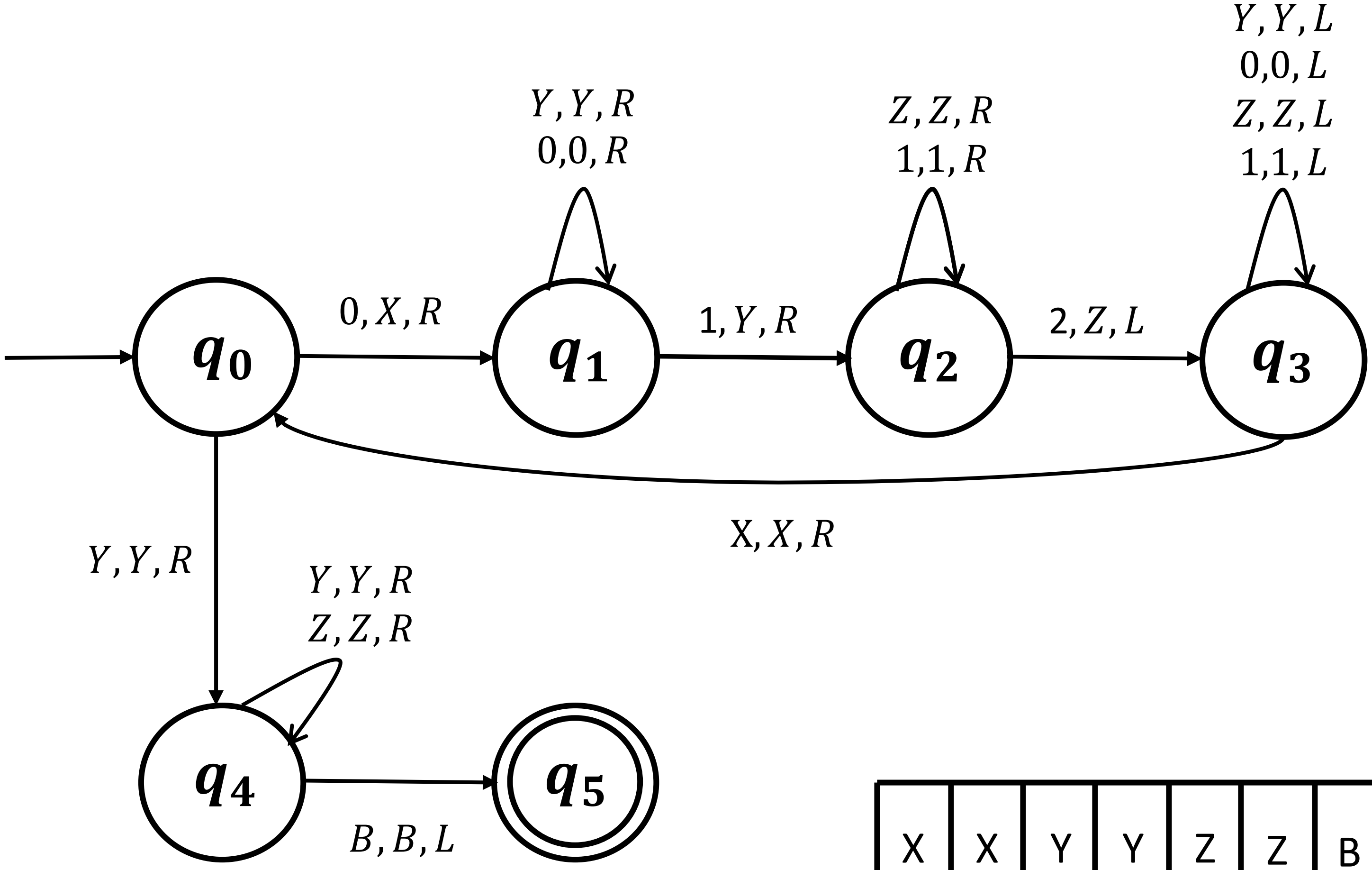
- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

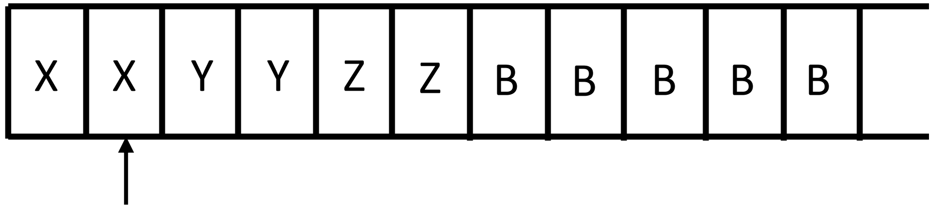
or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.
- Move left and accept the input string.

Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$





- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

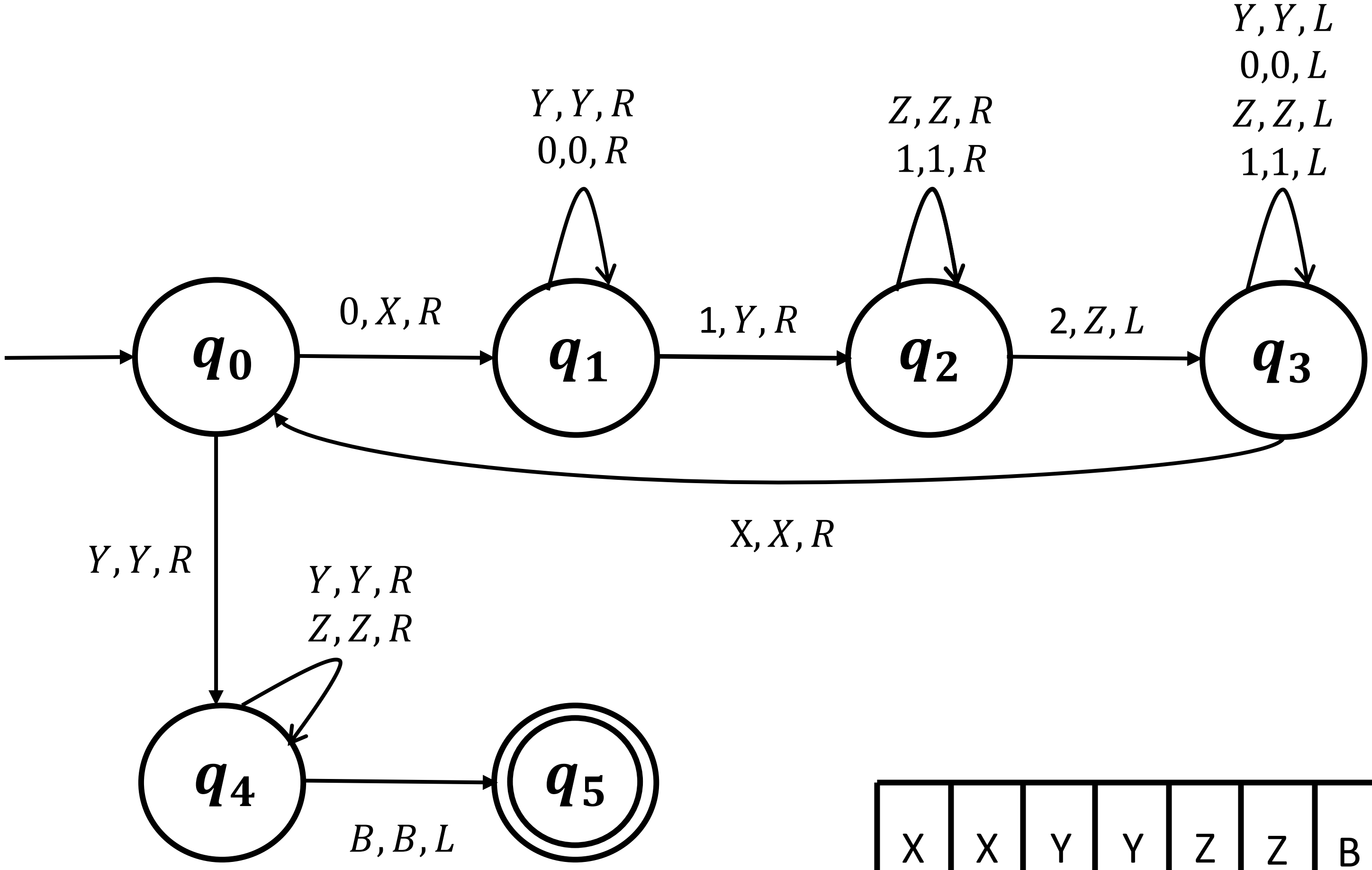
or

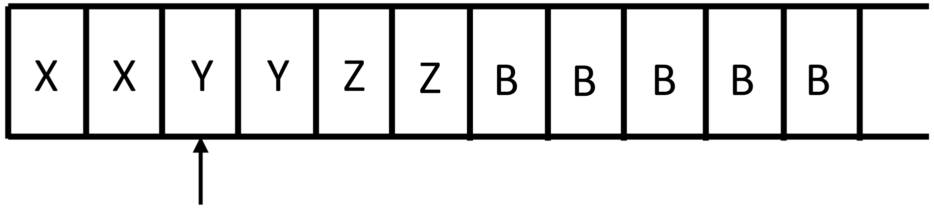
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.

Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$





- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

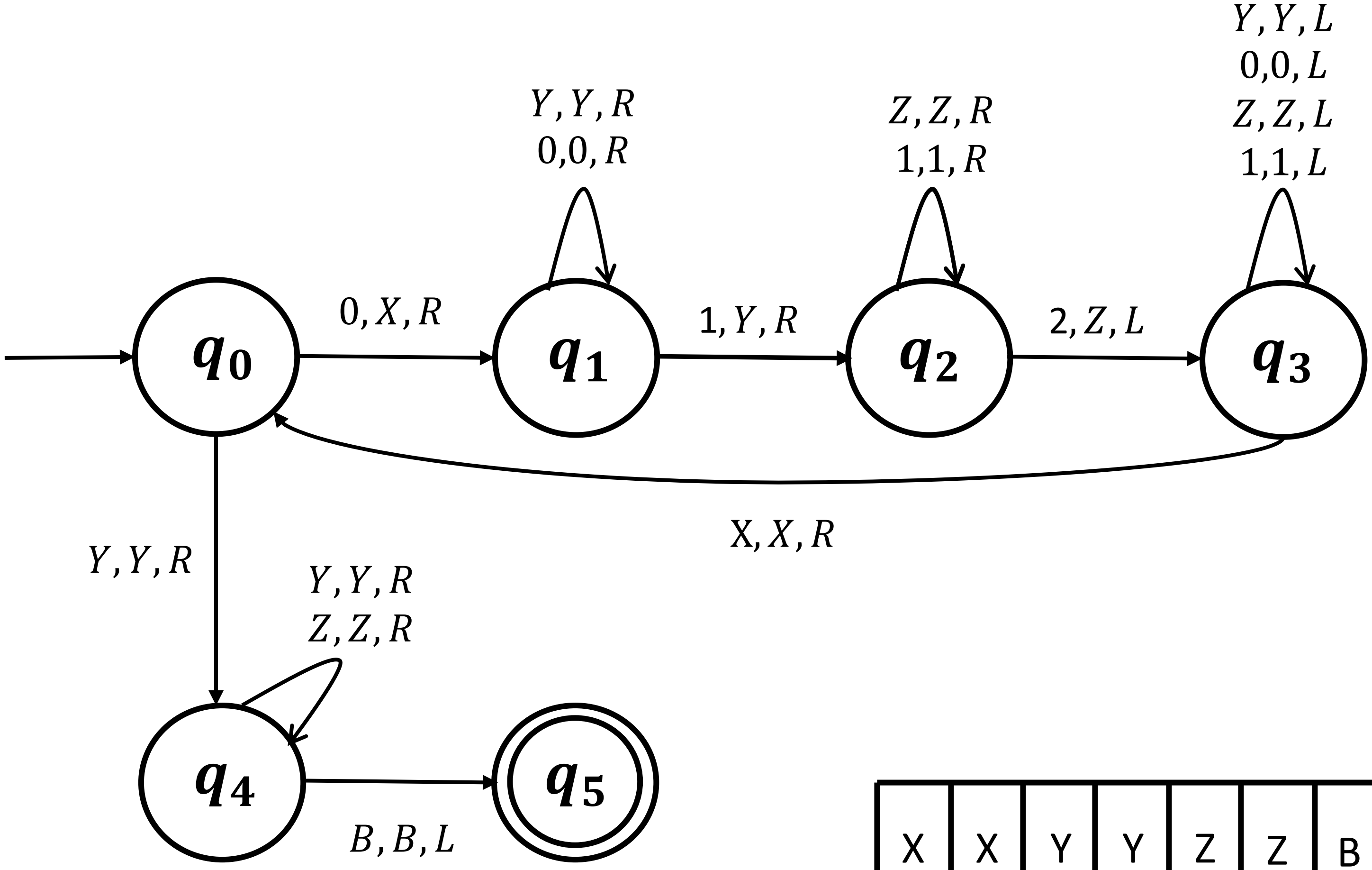
or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.

Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$

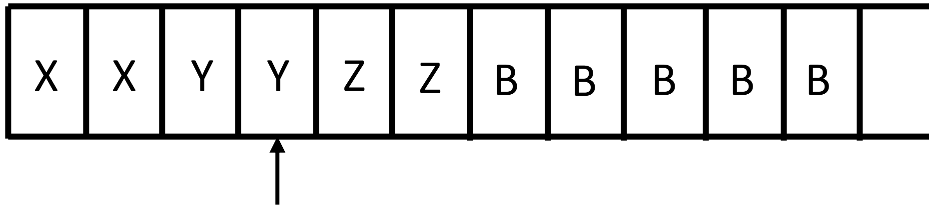


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

or

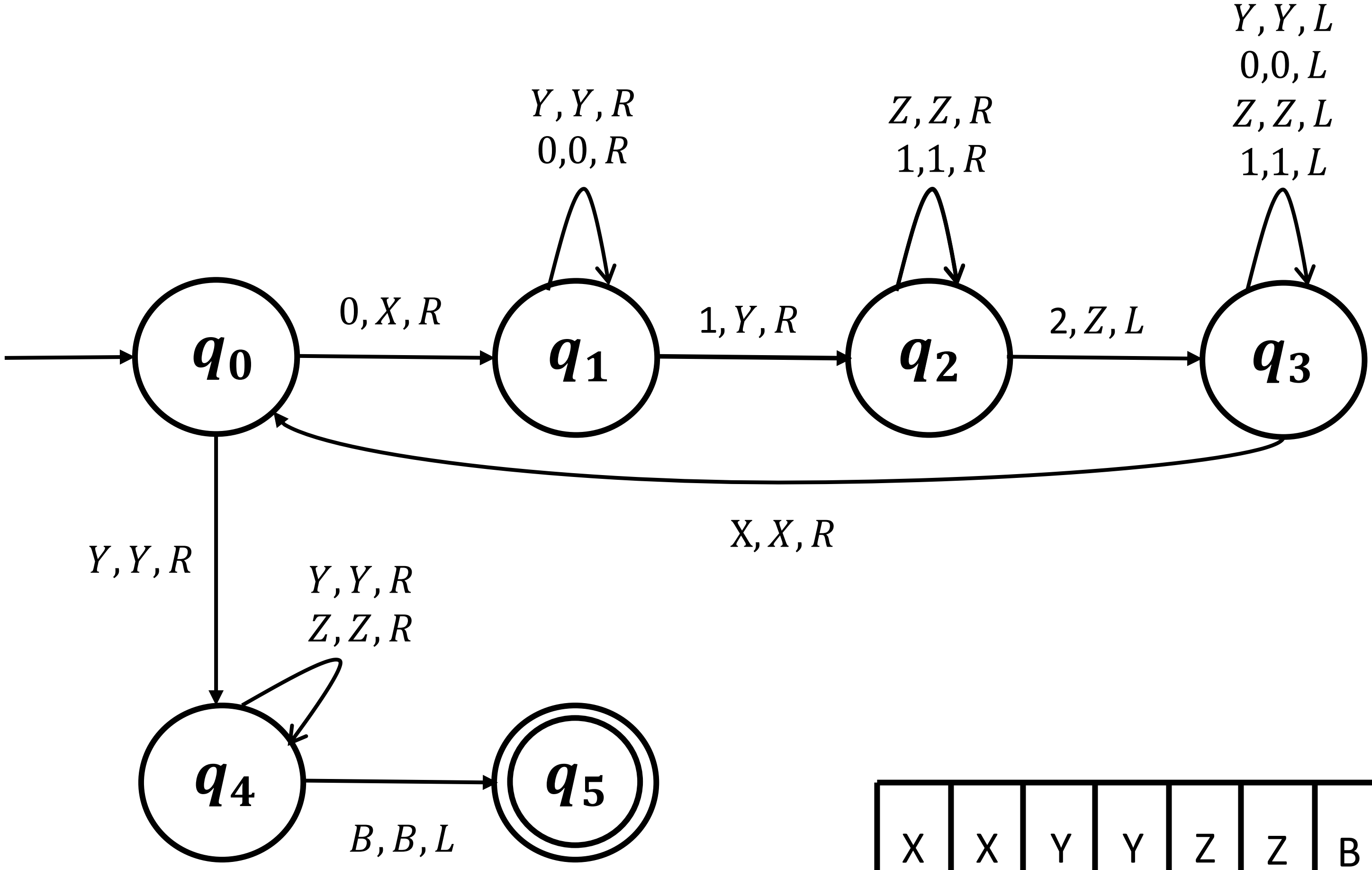
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.



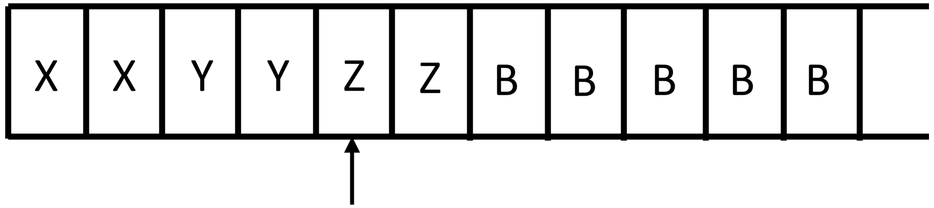
Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$



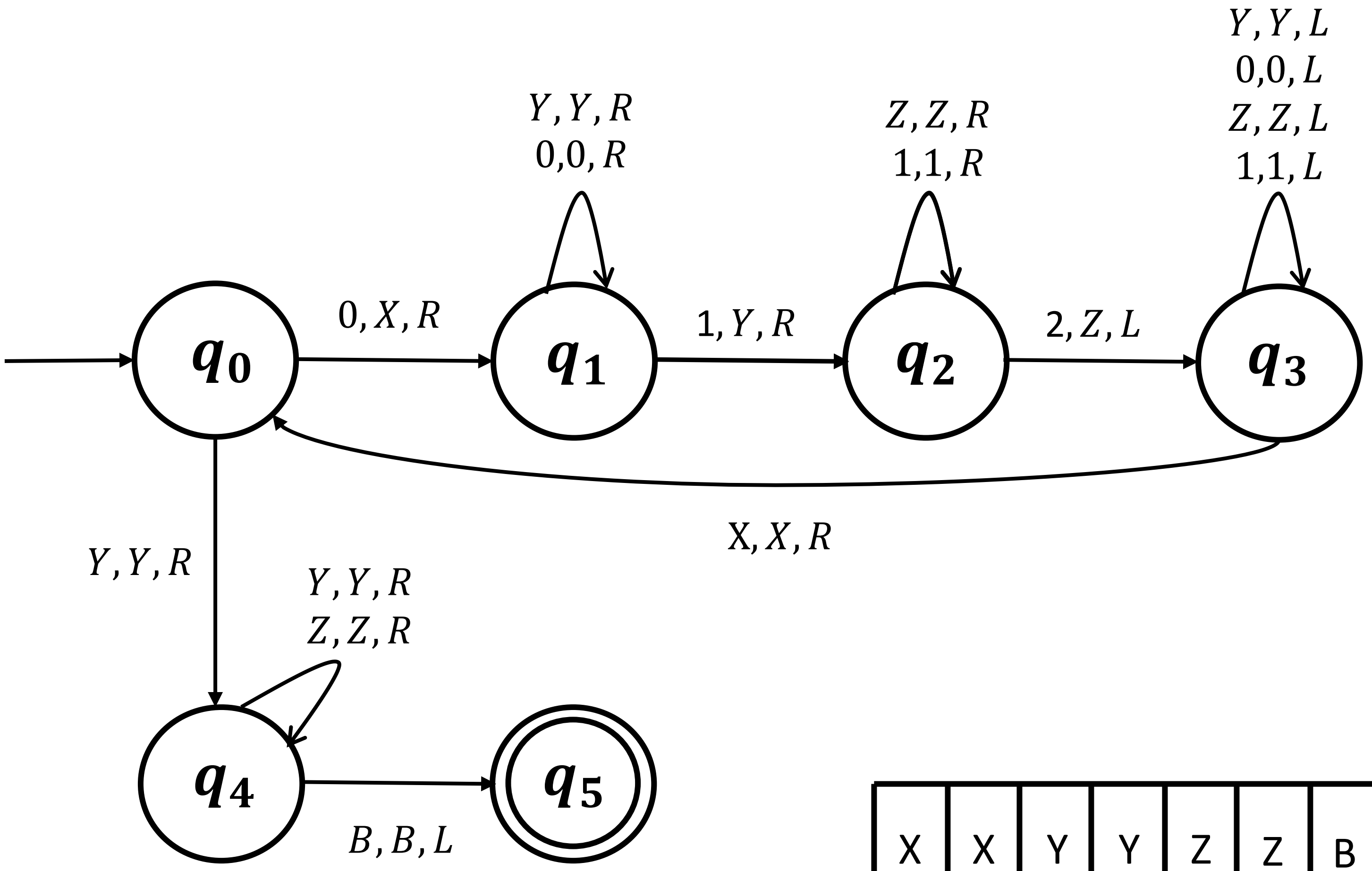
- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

  or
- Skip across the Y's and Z's to the right end of the tape until a blank is found.
- Move left and accept the input string.



Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

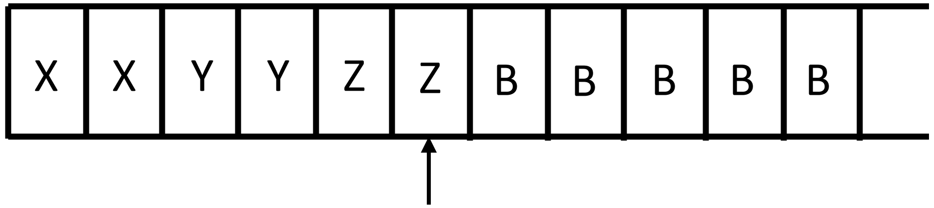


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

or

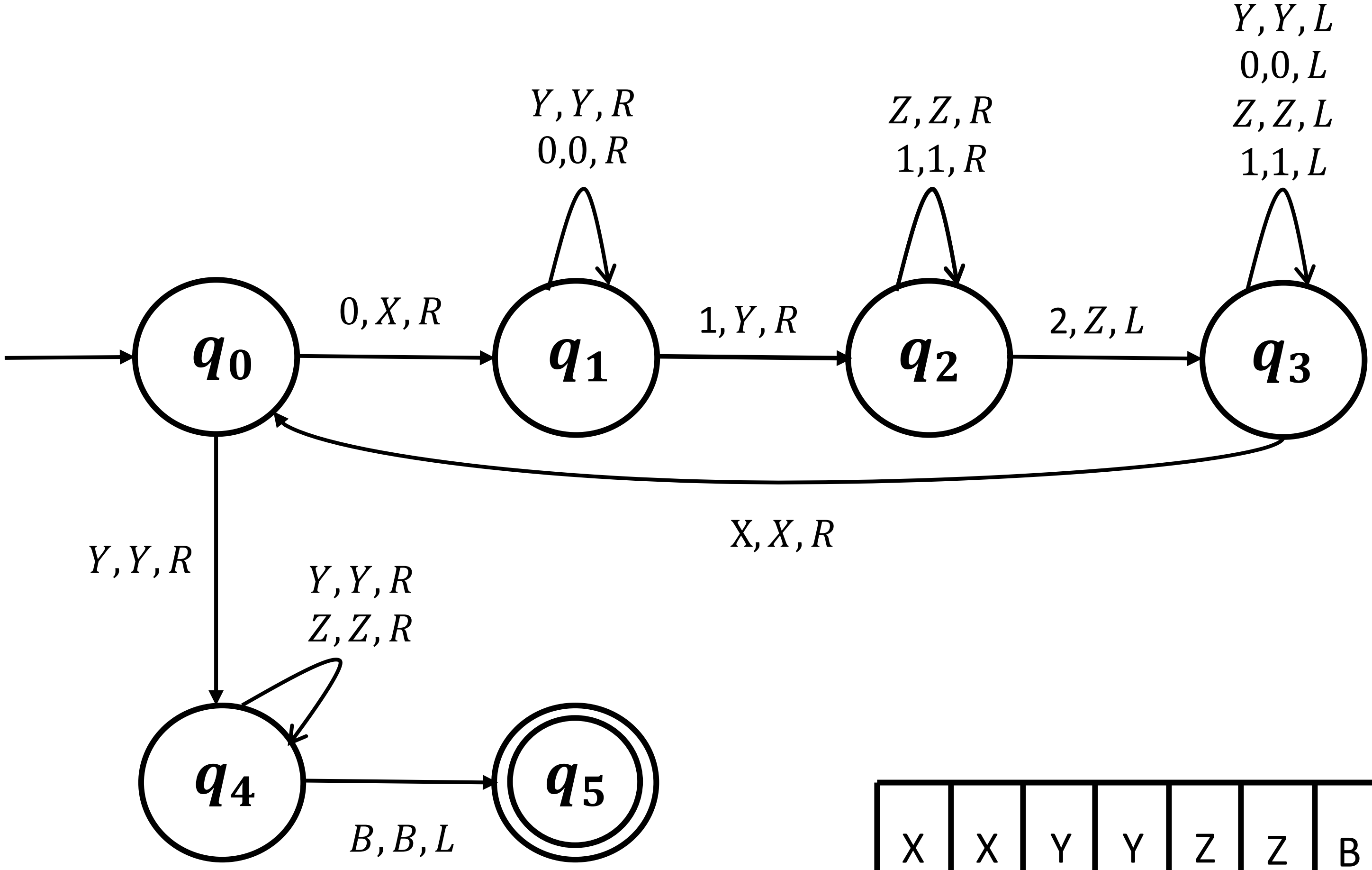
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.



Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$

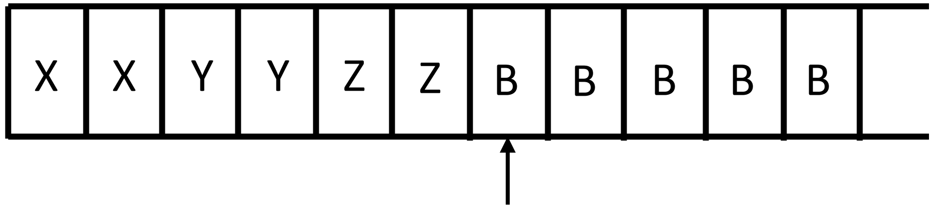


- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

or

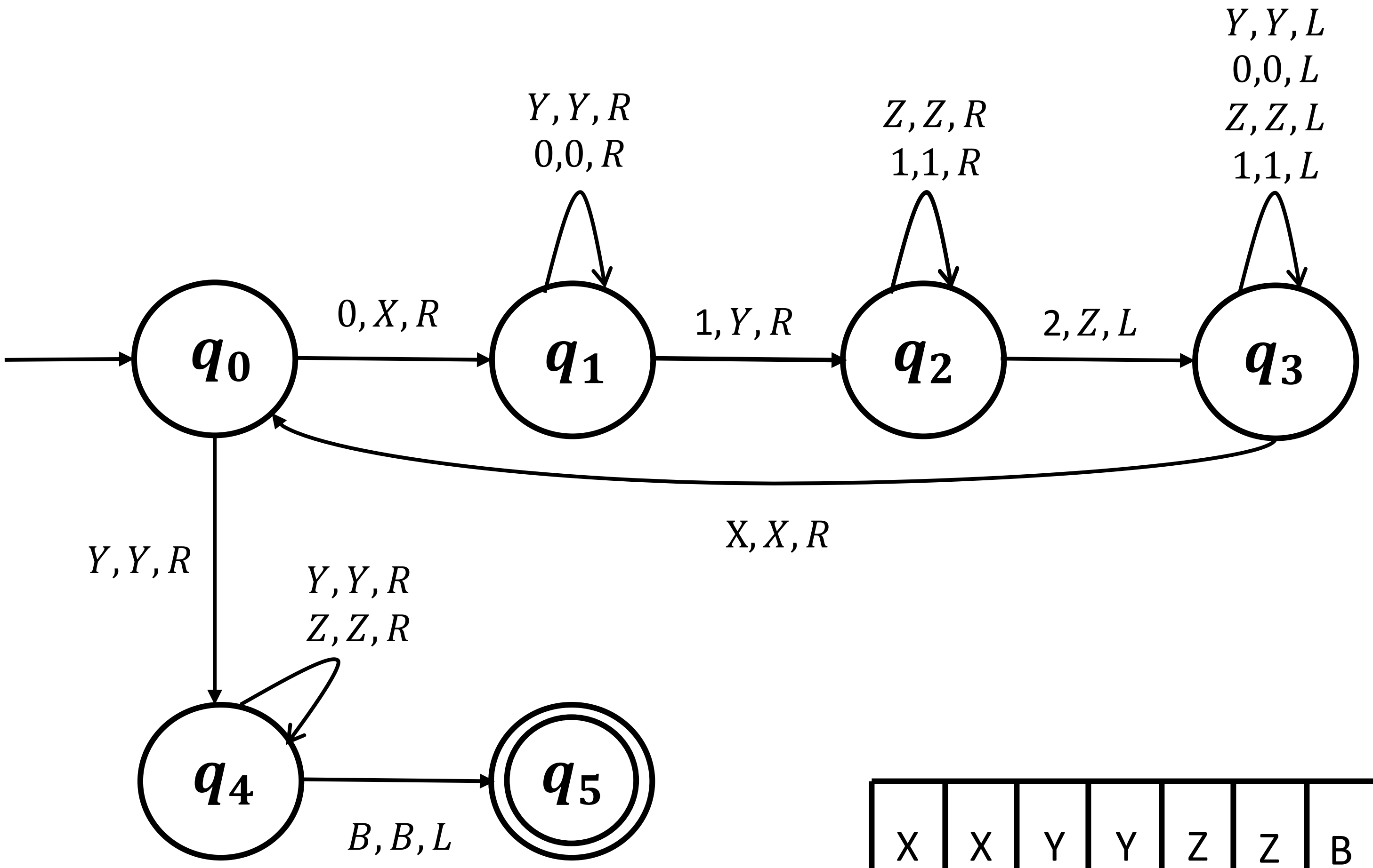
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.



Turing Machines

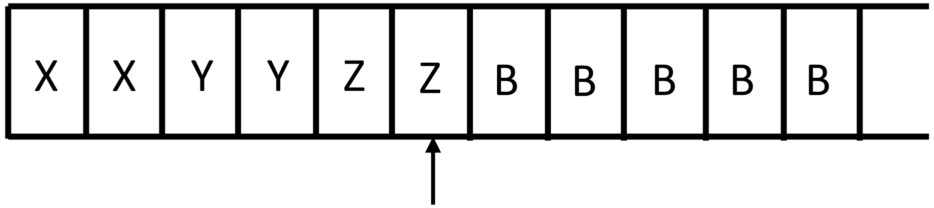
Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$



- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

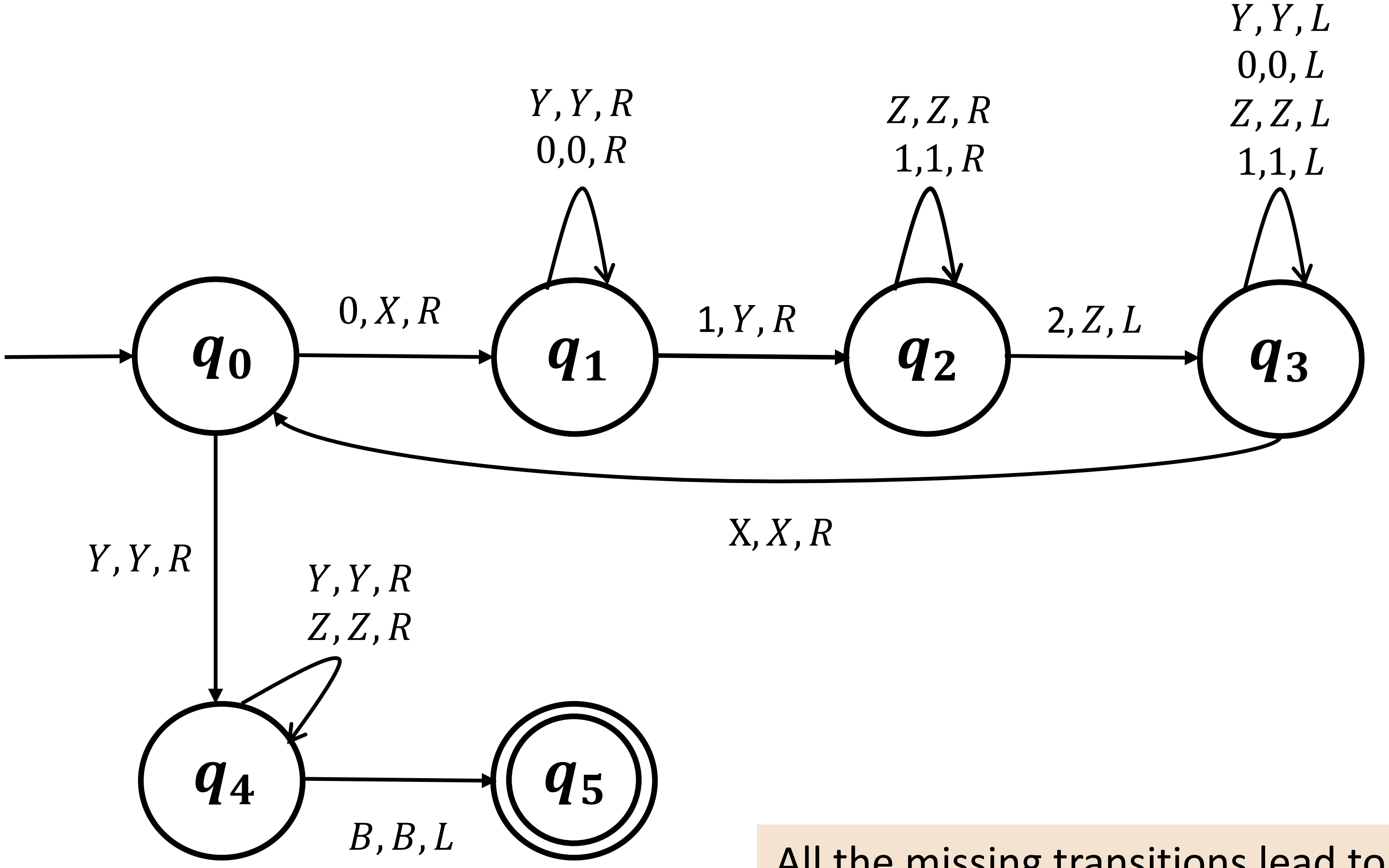
or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.
- Move left and accept the input string.



Turing Machines

Example: Let $L = \{0^n 1^n 2^n | n \geq 1\}$



- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

or

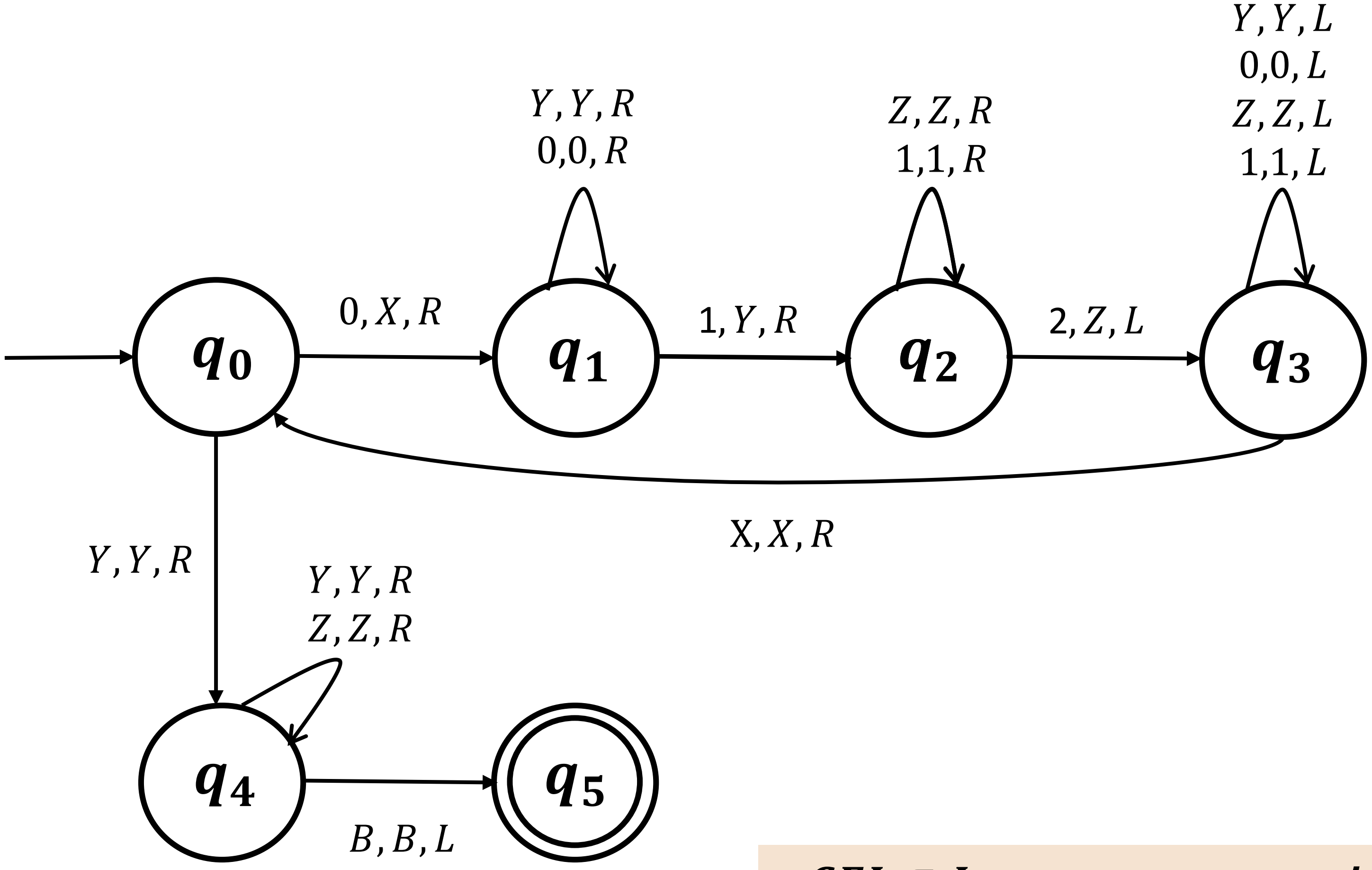
- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.

All the missing transitions lead to a reject state and so any input not of the form $\{0^n 1^n 2^n\}$ is rejected.

Turing Machines

Example: Let $L = \{0^n1^n2^n | n \geq 1\}$



- Continue to go right to mark the next 2 with a Z.
- Skip all the $1'$s and move right/ Also, move across (to the right) the Z's already marked.
- Mark a new 2 with a Z and start moving left.
- Keep moving left until an X is encountered.
- Either repeat this process if there are $0'$s, 1's or $2'$s left to mark

or

- Skip across the Y's and Z's to the right end of the tape until a blank is found.

- Move left and accept the input string.

$CFL \subseteq$ Language recognized by TM

Turing Machines

Formally, a Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets*** not containing the blank symbol B.
- Γ is the ***tape alphabet***, where $B \subseteq \Gamma$ and $\Sigma \subseteq \Gamma$.
- $\delta: Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ is the ***transition function***
- $q_0 \in Q$ is the ***start state***.
- $q_{accept} \in Q$ is the ***accepting state***.
- $q_{reject} \in Q - \{q_{accept}\}$ is the ***reject state.***

Configuration of a TM: Combination of the current state, the current tape contents and the current head location.

Formally, it is a triple: $(\boldsymbol{q}, \boldsymbol{a}, \boldsymbol{x})$, where $\boldsymbol{q} \in Q$, $\boldsymbol{a} \in \Gamma$, $\boldsymbol{x} \in Z_+$

At each step, the Turing machine configuration changes. We say C_1 **yields** C_2 if the TM changes from C_1 to C_2 in one step.

Turing Machines

Formally, a Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets*** not containing the blank symbol B**.**
- Γ is the ***tape alphabet***, where $B \subseteq \Gamma$ and $\Sigma \subseteq \Gamma$**.**
- $\delta: Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ is the ***transition function***
- $q_0 \in Q$ is the ***start state***.
- $q_{accept} \in Q$ is the ***accepting state***.
- $q_{reject} \in Q - \{q_{accept}\}$ is the ***reject state.***

A TM $\boldsymbol{M}$ **accepts** $\boldsymbol{w}$ if there exists a sequence of configurations C_1 to C_k, where

- C_1 is the start configuration M on w.
- Each C_i yields C_{i+1}.
- C_k is an accepting configuration

Language recognized a TM M:

$$\boldsymbol{L}(\boldsymbol{M}) = \{\boldsymbol{w} | \boldsymbol{M} \textbf{ accepts } \boldsymbol{w}\}$$

Configuration of a TM: Combination of the current state, the current tape contents and the current head location.

$X\ 0\ 0\ 1\ 1\ 1\ B\ B\ B\ B\ \ldots$

$\uparrow q_1$

At each step, the Turing machine configuration changes. We say C_1 **yields** C_2 if the TM changes from C_1 to C_2 in one step.

Start configuration:

$0\ 0\ 0\ 1\ 1\ 1\ B\ B\ B\ B\ \ldots$

$\uparrow q_0$

Accept configuration:

$X\ X\ X\ Y\ Y\ Y\ B\ B\ B\ B\ \ldots$

$\uparrow q_{accept}$

Reject configuration:

$X\ X\ X\ Y\ Y\ 0\ B\ B\ B\ B\ \ldots$

$\uparrow q_{reject}$

Turing Machines

Formally, a Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where

- Q is a finite set called the ***states.***
- Σ is the set of input ***alphabets*** not containing the blank symbol B**.**
- Γ is the ***tape alphabet**,* where $B \subseteq \Gamma$ and $\Sigma \subseteq \Gamma$**.**
- $\delta: Q \times \Gamma \mapsto Q \times \Gamma \times \{L, R\}$ is the ***transition function***
- $q_0 \in Q$ is the ***start state***.
- $q_{accept} \in Q$ is the ***accepting state***.
- $q_{reject} \in Q - \{q_{accept}\}$ is the ***reject state.***

A TM $\boldsymbol{M}$ **accepts** $\boldsymbol{w}$ if there exists a sequence of configurations C_1 to C_k, where

- C_1 is the start configuration M on w.
- Each C_i yields C_{i+1}.
- C_k is an accepting configuration

Language recognized a TM M:

$$\boldsymbol{L(M) = \{w | M \text{ accepts } w\}}$$

Configuration of a TM:

- Combination of the current state, the current tape contents and the current head location.
- At each step, the Turing machine configuration changes. We say C_1 **yields** C_2 if the TM changes from C_1 to C_2 in one step.

Next Lecture

Various TM model **variants**: Robustness of the standard TM model

Thank You!