CS 3313
Foundations of Computing:

Modifications to the Turing Machine Model

http://gw-cs3313-2021.github.io

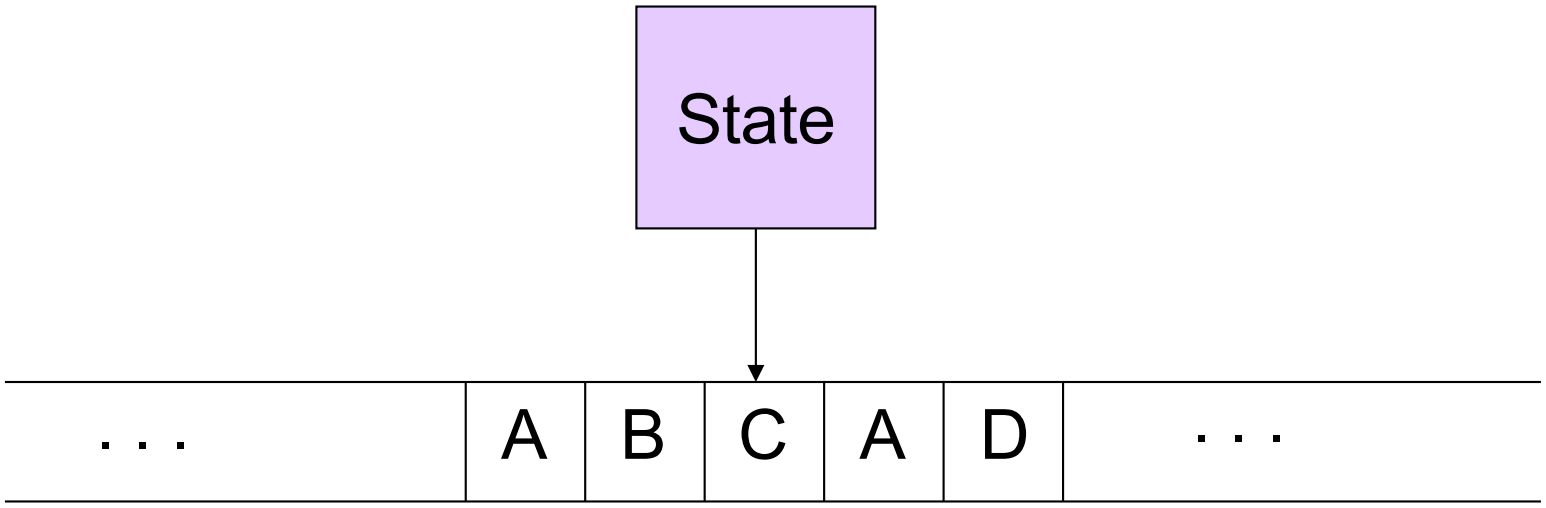
Outline..

- Turing Machine model
  - TM as an automaton
- Computing functions on a Turing machine
  - Turing machine as a “computer”
  - First step is encoding the integer arguments to the TM
- TM “programming” techniques
  - Storage in the state (you’ve seen this) – ‘caching’ a value
  - Checking symbols, Shifting over (skipping) tape symbols
  - Subroutines…
- Modifications to the basic TM model
  - Multiple tracks on the tape
  - Semi-infinite tape
  - Multi-Tape TMs
  - Non-deterministic TMs

Turing Machine

Action: based on the (i) state and)ii) the tape symbol under the read/write head:

- (1) change state, (2) write a symbol back to the tape and (3) move the head (left or right) one location/square on the tape.



Infinite tape with squares containing tape symbols chosen from a finite alphabet

Turing-Machine Formalism

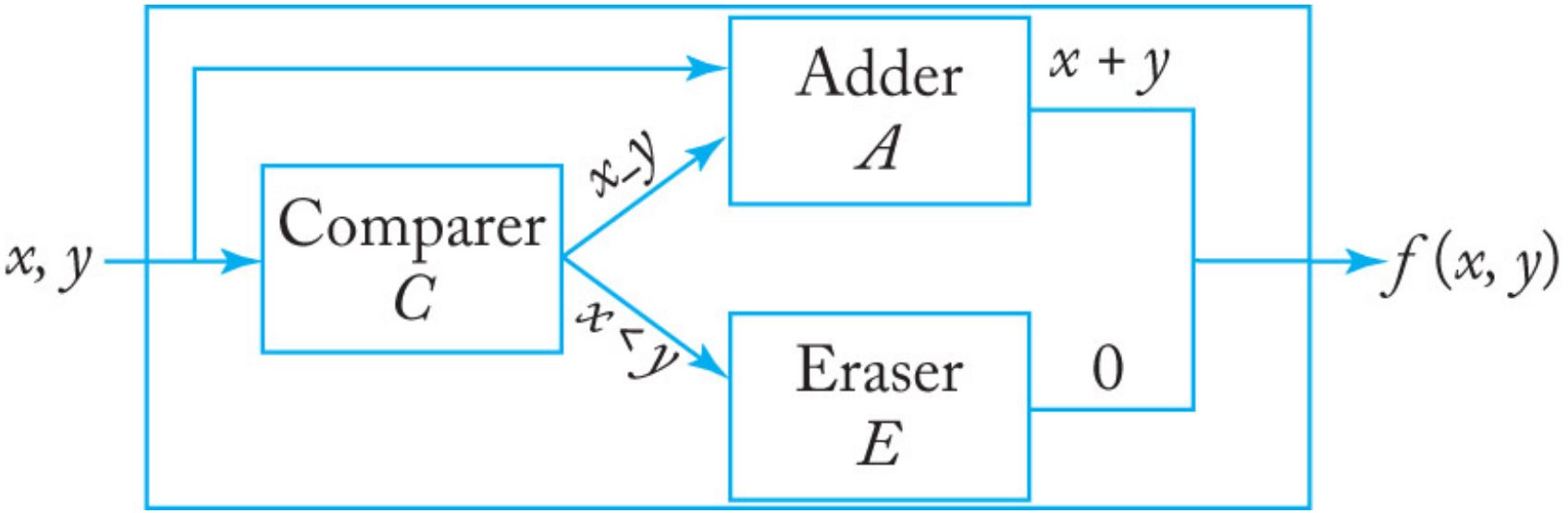
- A TM is described by:
  1. A finite set of *states* Q.
  2. An *input alphabet* Σ.
  3. A *tape alphabet* Γ (contains Σ).
  4. A *transition function* $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$
  5. A *start state* q_0 (in Q).
  6. A *blank symbol* B (or □) in $\Gamma - \Sigma$
     - All tape except for the input is blank initially.
  7. A set of *final states* $F \subseteq Q$

The Transition Function

- Takes two arguments:
  1. A state, in Q.
  2. A tape symbol in Γ.
- $\delta(q, Z)$ is either undefined or a triple of the form (p, Y, D).
  - p is a state.
  - Y is the new tape symbol.
  - D is a *direction*, L or R – move the tape head to the Left or Right
  - Convention: If undefined then TM halts
    - If it halts in a final state then it accepts
    - If it halts in a non-final state then it rejects

Taking stock: Combining Turing Machines

- By combining Turing Machines that perform simple tasks, complex algorithms can be implemented
- Example: assume the existence of
  - a machine to compare two numbers (comparer)
  - Machine to add two numbers (adder)
  - machine to erase the input (eraser)
- TM to compute function f(x, y) = x + y (if x ≥ y), 0 (if x < y)

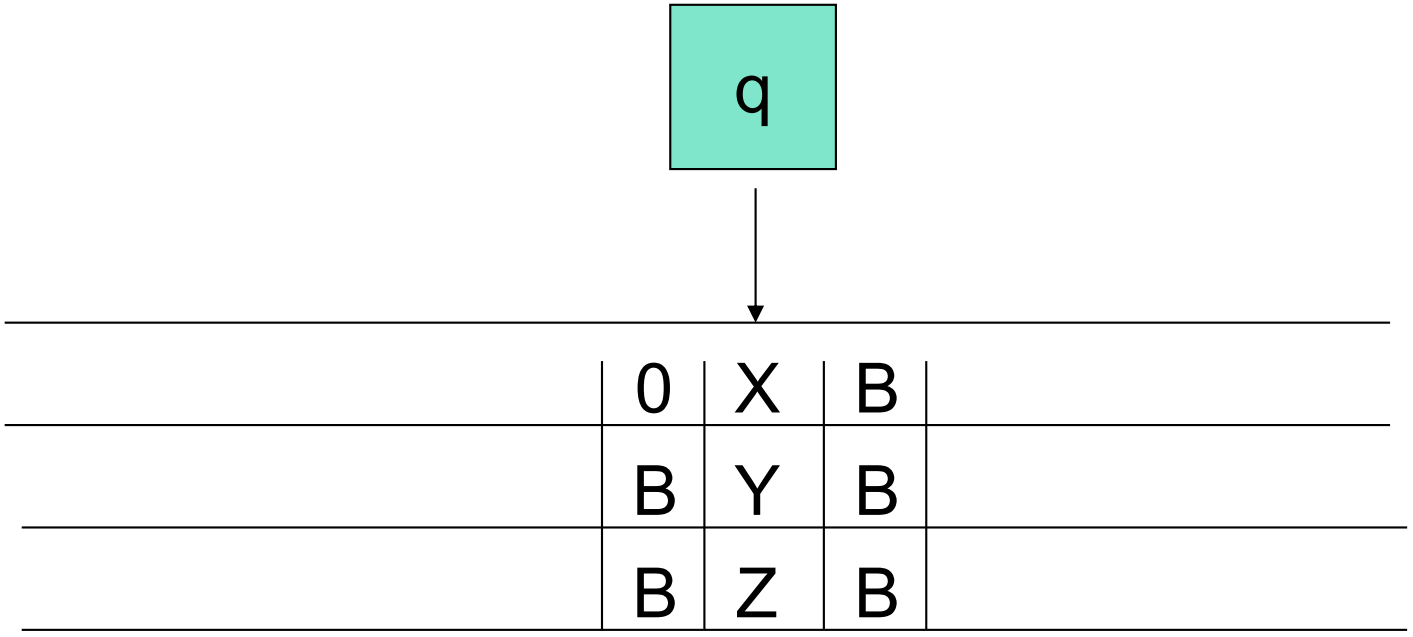


What happens if we “add” more “power” to the TM ?

- Change tape to multiple tracks
- Limit tape to be infinite at one end but with a left end to the tape.
  - Not as interesting to delve into details
- Multi-tape TMs
- Multi-dimensional Tapes
- Non-determinism

Multiple Track Turing Machine

- Tape consists of k tracks: each tape cell has k tracks
- Tape head reads from all k tracks in one step,
- Moves tape head to Left or Right
- Define transition function as $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R\}$

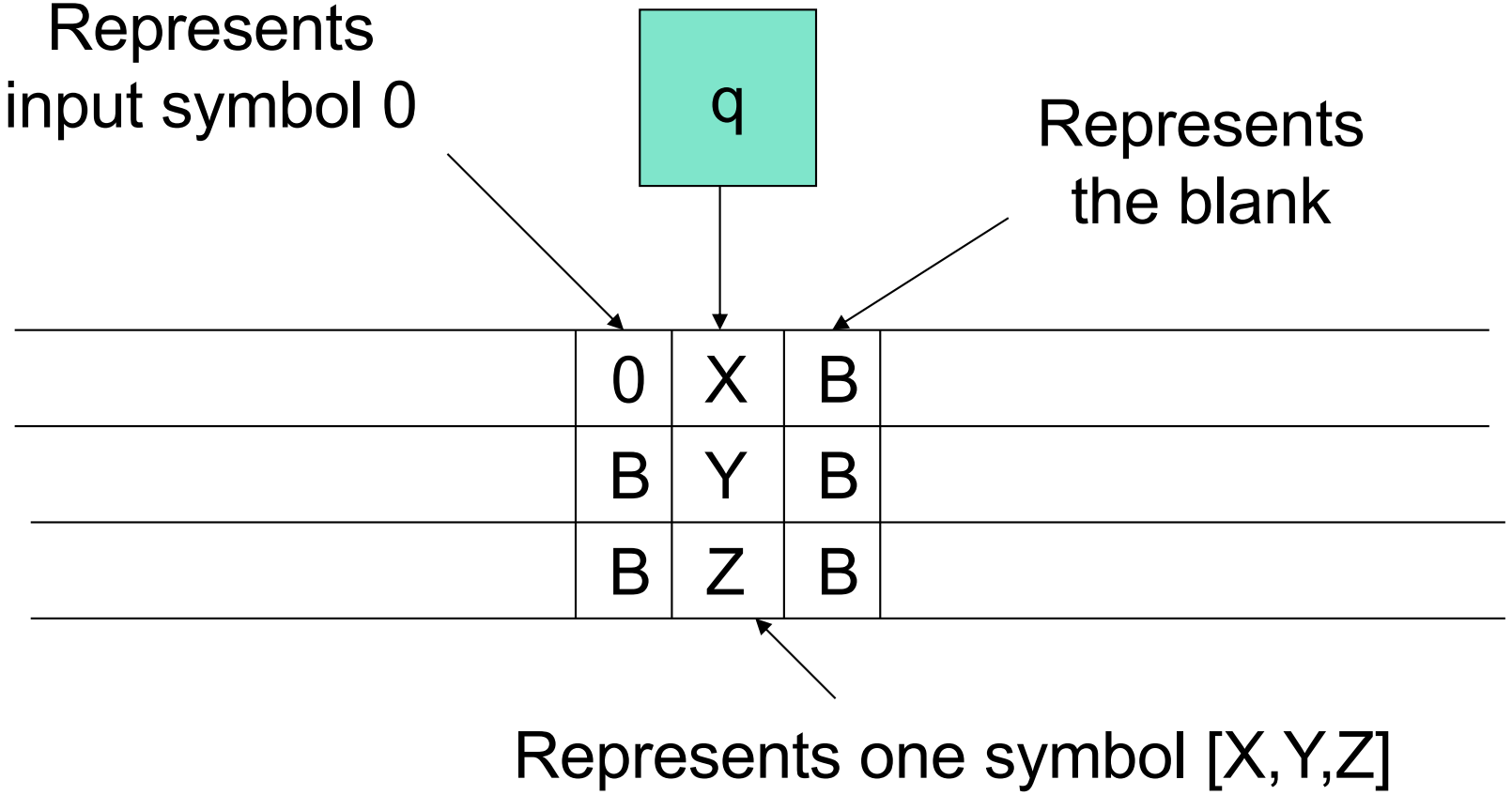


Is this really different from the “standard” 1-track TM ?

Multiple Track TM = Single Track TM

Simply view the tape alphabet as a k-tuple!!

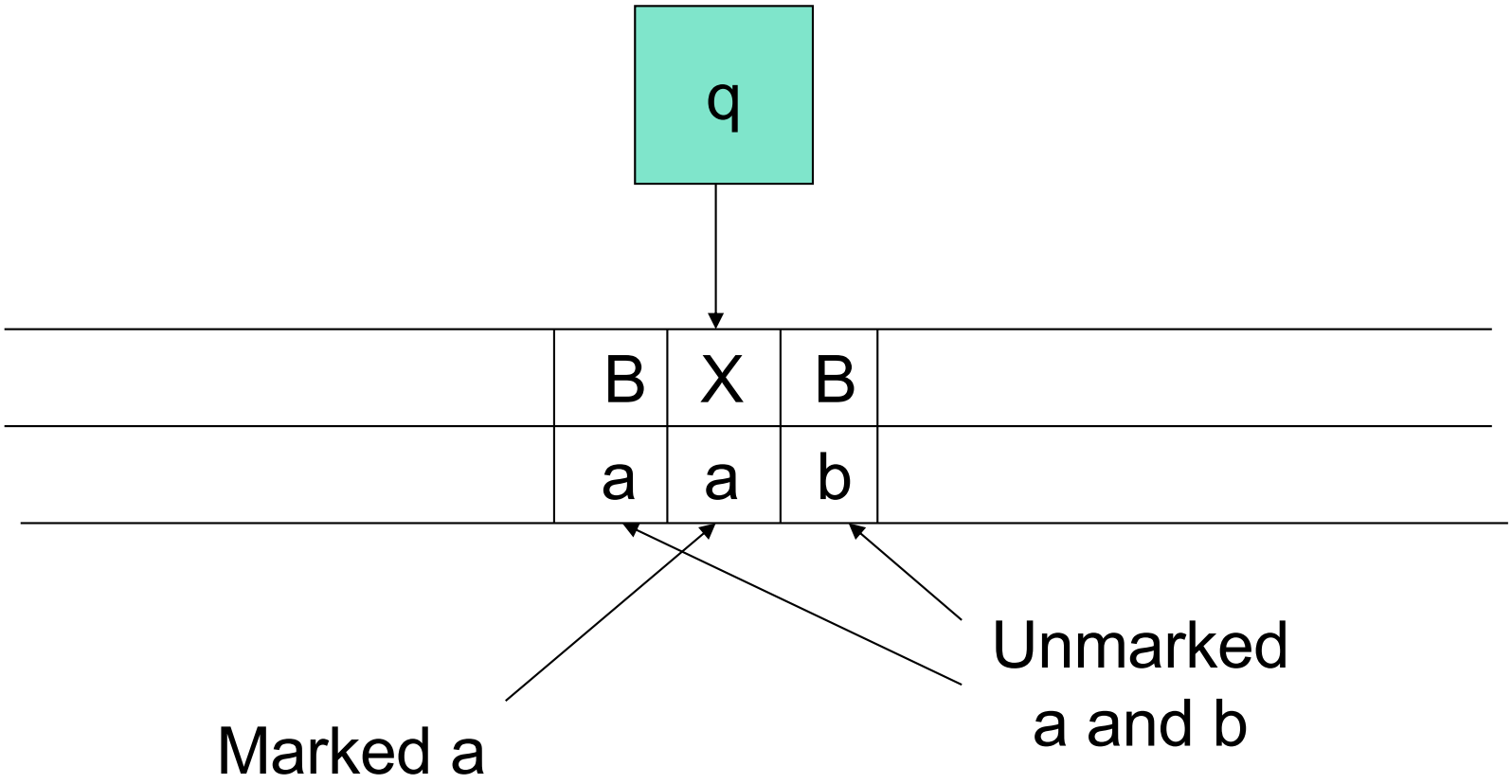
We are changing the "*data structure*"



Why bother with Multi-track Tapes?

- Useful “programming tricks”…
  - Add to our bag of TM programming techniques!
- Can use one track to check off symbols !

Marking

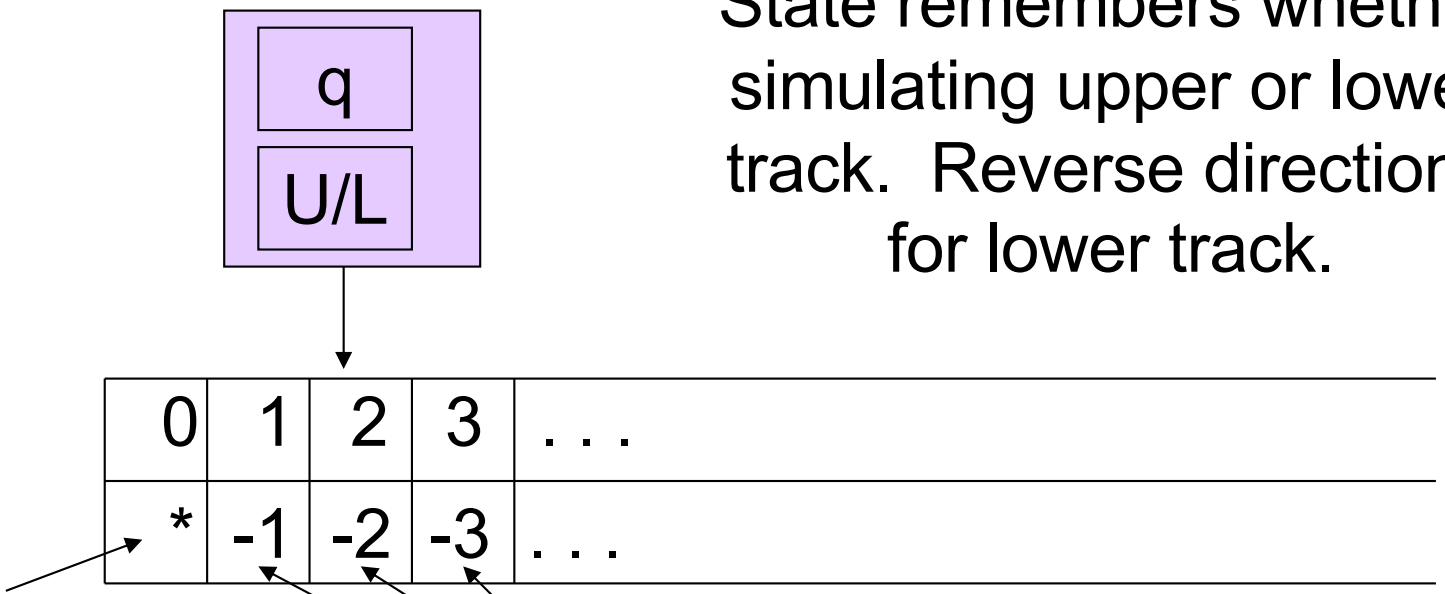


Multi-track TM for L = {ww}

Semi-infinite Tape

- We can assume the TM never moves left from the initial position of the head….this "restricted TM" can simulate a two-way infinite TM!
- Let this position be 0; positions to the right are 1, 2, … and positions to the left are –1, –2, …
- New TM has two tracks.
  - Top holds positions 0, 1, 2, …
  - Bottom holds a marker, positions –1, –2, …

Simulating Infinite Tape by Semi-infinite Tape



State remembers whether simulating upper or lower track. Reverse directions for lower track.

Put * here at the first move

You don't need to do anything, because these are initially B.

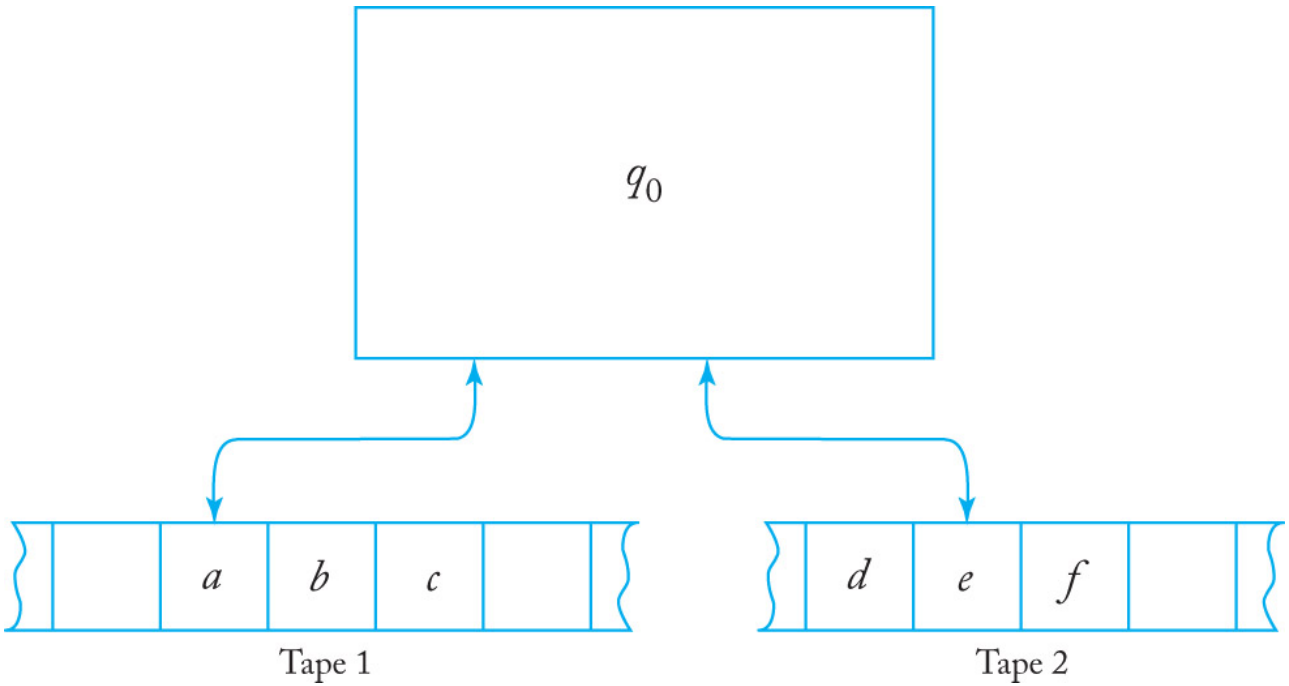
Multitape Turing Machines

- Allow a TM to have k tapes for any fixed k.
- Move of the TM depends on the state and the symbols under the head for each tape.
- In one move, the TM can change state, write symbols under each head, and move each head independently.

Multitape Turing Machines

- a *multitape Turing machine* has several tapes, each with its own independent read-write head
- A sample transition rule for a two-tape machine must consider the current symbols on both tapes:

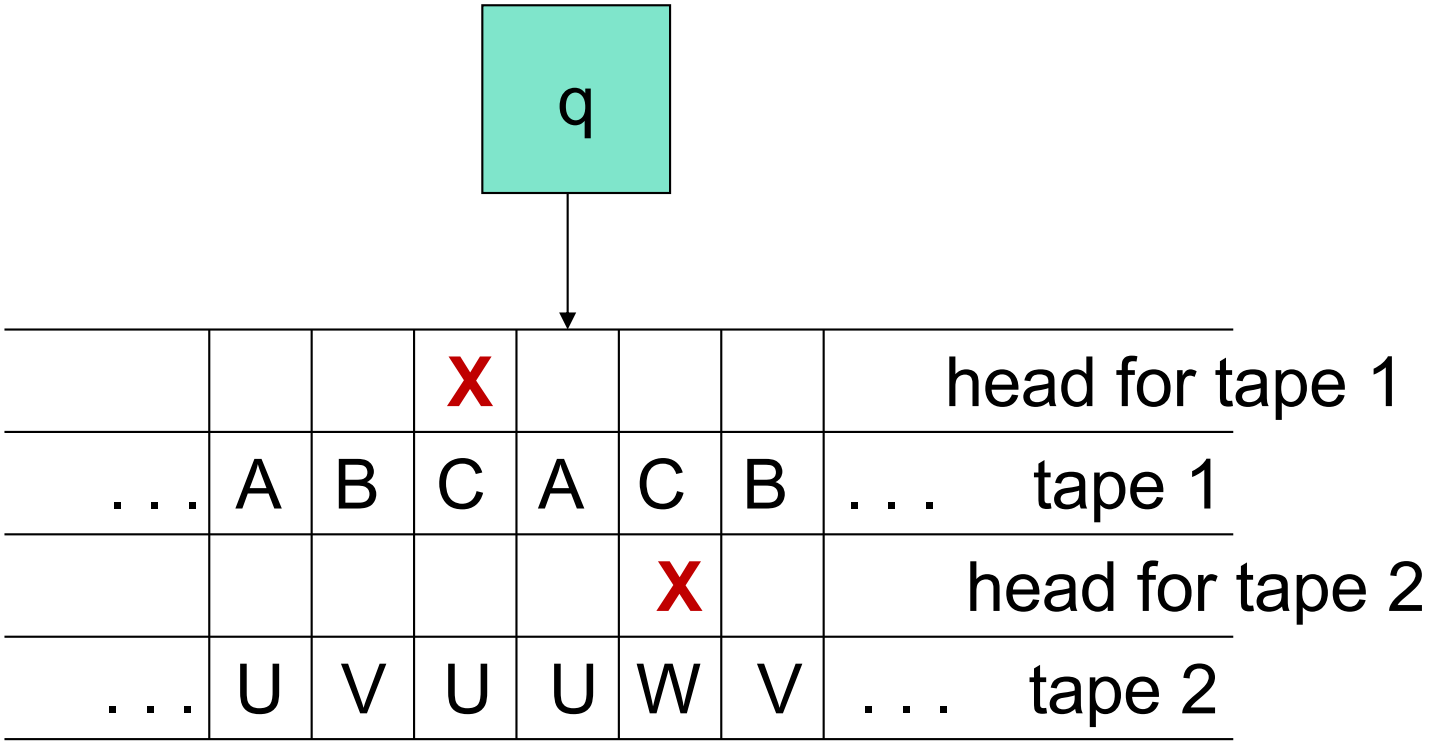
$$\delta(q_0, a, e) = (q_1, x, y, L, R)$$



Simulating k Tapes by One

- Use 2k tracks.
- Each tape of the k-tape machine is represented by a track.
- The head position for each track is represented by a mark on an additional track.

Picture of Multitape Simulation



Why bother with Multi-Tape TMs ?

- Makes your ”algorithm” more “efficient” to design (and implement – number of moves of the TM).
- Ex 1: L = { ww }
  - You’ve figured out how to find “middle” of the string…
  - Next step ???

Multi-tape TM Ex 2: L = { $a^n b^n c^n$}

Multi-tape TM Ex.3: L = { w | w = w^R}

Finally….Nondeterministic TM's

- Allow the TM to have a choice of move at each step.
  - Each choice is a state-symbol-direction triple, as for the deterministic TM.
- The TM accepts its input if any sequence of choices leads to an accepting state.