CS 3313
Foundations of Computing:

Modifications to the Turing Machine Model

http://gw-cs3313.github.io

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 Γ- Σ
     - All tape except for the input is blank initially.
  7. A set of *final states* $F \subseteq Q$

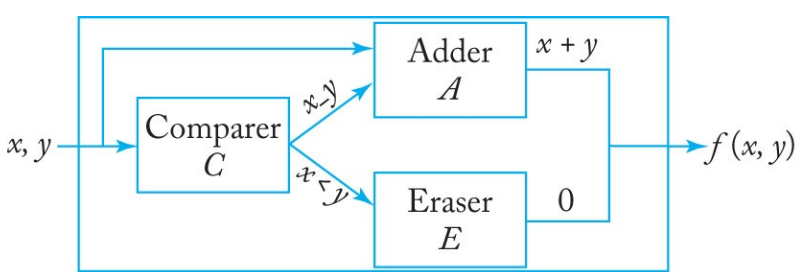
Turing Machine Model…where are we

- Turing Machine model
  - TM as an automaton
  - Computing functions on a Turing machine – using unary encoding
    - $q_0x \vdash^* q_fy \qquad y = f(x)$
- Input w to Turing machine M:
  - w is accepted iff M halts in a final state
  - w is rejected iff M halts in a non-final state
  - M may never halt on input w (ex: infinite loop) -- not the same as "reject"
- TM "programming" techniques
  - Storage in the state (you've seen this) ✓
  - Checking/marking symbols ✓
  - Shifting over (skipping) tape symbols
  - Subroutines ✓

3

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)



4

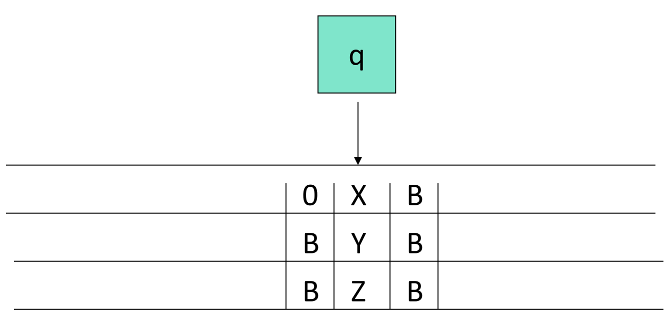
Next…Modifications to standard Turing machine

- Add new features/capabilities to standard turing machine….does this increase power/capabilities ?
  - Tape with multiple tracks
  - Tape with “stay” option
  - Semi-infinite tape
  - Multiple tapes
  - Multidimensional tapes
  - Non-deterministic Turing machines
- Turns out they are all equivalent to the standard TM
- Proofs: simulation of these models on the standard TM

5

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 ?

6

6

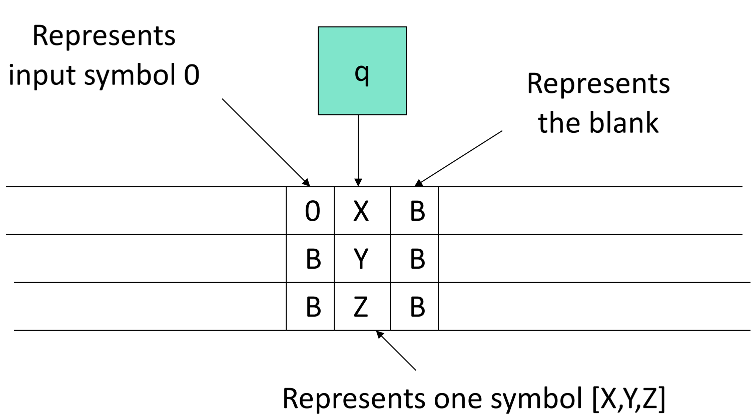
Multi track TM

Example: Tape alphabet = decimal digits {0,1,..,9}
What is the representation using base 2 (binary)?

7

Multiple Track TM = Single Track TM

Simply view the tape alphabet as a k-tuple!!
We are changing the "*data structure*" (data rep.)

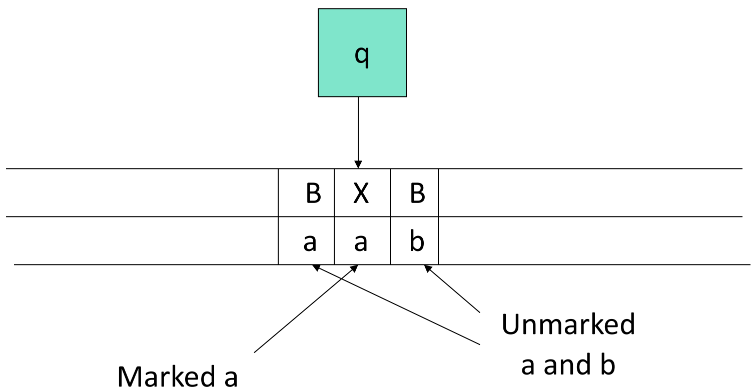


8

8

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 !



9

9

Multi-track TM for $L = \{ww\}$

10

Multi-track TM for $L = \{ww\}$

11

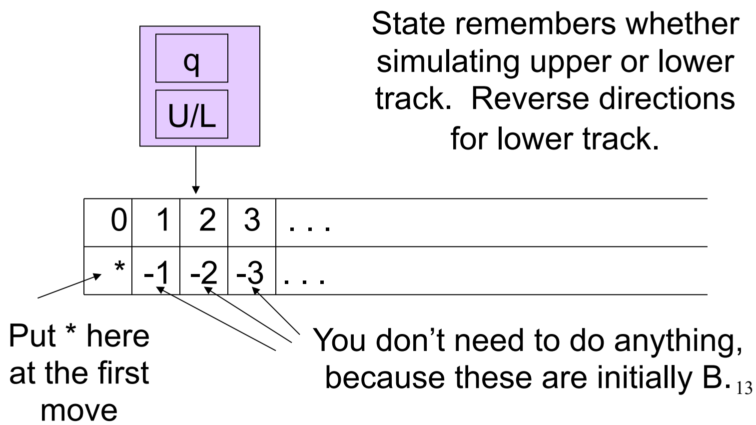
Semi-infinite Tape

- 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, …

12

12

Simulating Infinite Tape by Semi-infinite Tape



13

Turing Machines with Stay option

- *transition function* $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R, S\}$
- Does this add any power ?

14

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.

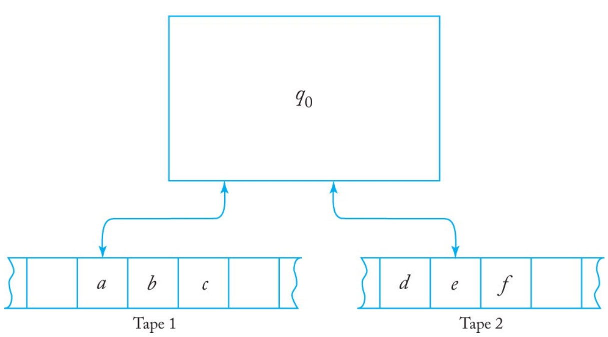
15

15

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:

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



16

Why bother with Multi-Tape TMs ?

- Makes your "algorithm" more "efficient" to design (& implement – number of moves of the TM).
- Ex 1: $L = \{ ww \}$
  - First find "middle" of the string…
  - Next match (check if equal) corresponding locations in left half and right half
  - Recall algorithm (from lab)
    - First sweep left to right, marking the positions until we find midpoint
    - Next sweep left to right (and back to left unmatched symbol) matching the symbols
    - Time complexity = $O(n^2)$ for length n input

17

Multi-tape TM *Ex 1: L = { ww}*

- Using a 2 tape TM to accept $L = \{ ww\}$

1. Find "middle" (to recognize left half and right half)

2. Match symbols in left half and right half

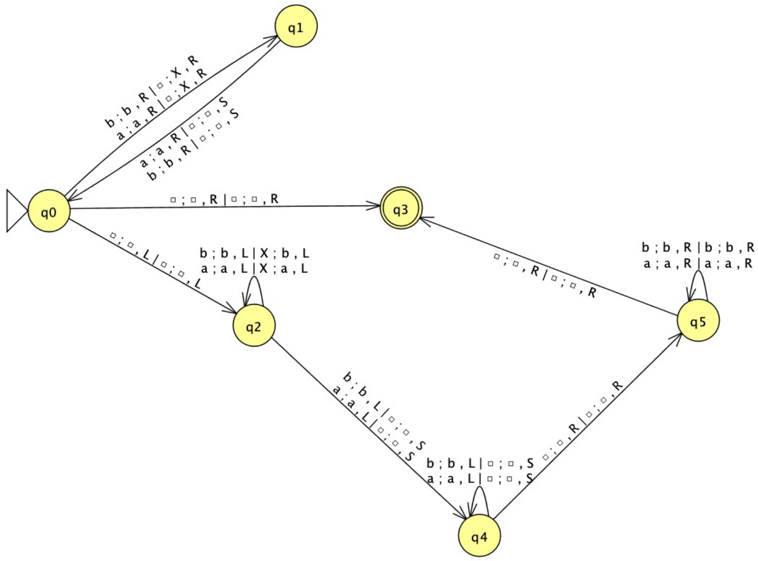
18

Multi-tape TM *Ex 1: L = { ww}*

- Algorithm:

19

Multi-tape TM *Ex 1: L = { ww}*



20

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

21

Multi-tape TM *Exercise:* $L = \{ w \mid w = w^R \}$

22

Is k-tape TM > 1-tape TM ?

- So can a k-tape TM do more than a 1-tape TM ?
- *Transition function for a k-tape TM*

$\delta(q_1, a_1. a_2...,a_k) = (q_2, b_1. b_2,...,b_k, D_1, D_2,...D_k)$

  - a_i , $b_i \in \Gamma$ $D_i \in \{L,R\}$

- To simulate a move of the k-TM, we need to read/write *k* symbols from tape and specify *k* tape head moves
- Question: How do we specify the ID (snapshot) of a k-tape TM ?

23

Moves in the k-tape TM

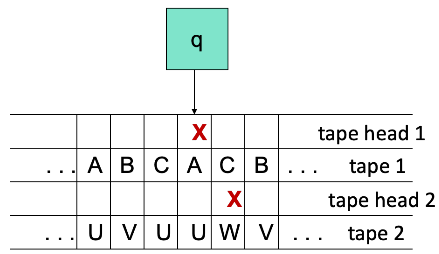
- k tapes, k tape heads
- One move: read from k tapes, write to k tapes, move each tape head L or R

24

24

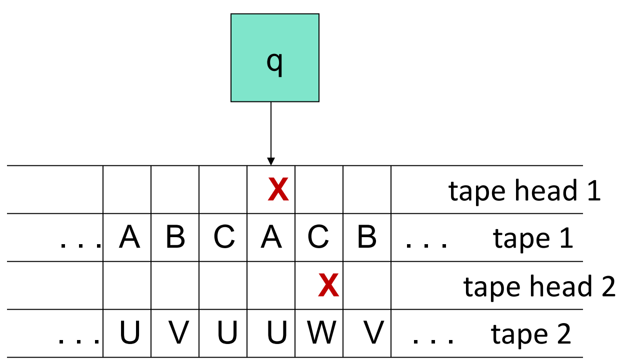
Simulating k Tapes by One: capturing the ID (snapshot)

- 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.



25

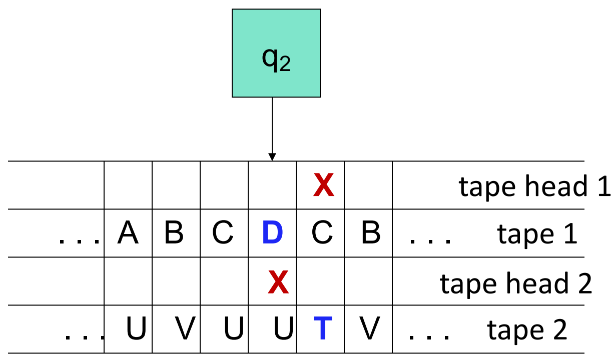
Picture of Multitape Simulation



26

26

Picture of Multitape Simulation

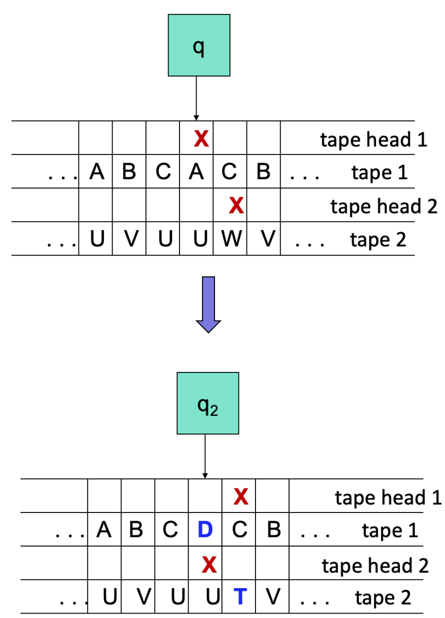


27

27

Simulating a move on 2k track 1-tape TM

$\delta(q, A, W) = (q_2, D, T, R, L)$



28

Simulating k Tapes by One tape TM

29

29

Simulating k Tapes by One tape TM

30

30

Simulating k Tapes by One

- 2k tracks to simulate the k tape TM
  - For each tape: 1 track for tape contents and 1 track to indicate location of tape head (indicated by an X)
- How many *X*'s to find = *k*
- Where to store symbol above X (read by a tape) = in the state!
- State in 1-tape *2k* track TM is $[q,\ a_1,a_2,..a_k]$
- To read all k symbols from the k tapes (on the k tracks)
  1. Sweep tape (Left to Right) past k "X" markers and store the k tape symbols in the state
  2. Sweep tape (Right to Left)
     1. Write symbol to the track
     2. Write X below to track below it and move R or L
  3. If TM halts in final state then accept, else go to 1

31

31

Summary of Results

- Theorem 1: A *k*-track single tape TM is equivalent to a one tape one track TM.
- Theorem 2: A two-way infinite TM is equivalent to a one way infinite tape TM.
- Theorem 3: A *k*-tape TM is equivalent to a one tape TM
  - Simulation used a 2k track TM, but then apply Theorem 1 to get equivalence to the basic TM
- Other results:
  - A Multi-dimensional TM Is equivalent to a one dimensional tape TM
  - A multi-tape head single tape TM is equivalent to a one head one tape TM
- None of these models can solve a problem that cannot be solved by the standard TM…..however, they add expressive power (analogy= programming languages)

32

Next….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.

33

33

Non-determinism

- From a single state, the machine can go to any of k states
  - Abstraction model: simultaneously go to all k, and replicate the machine
  - In reality: we cannot replicate the machine to arbitrarily large numbers
- In NFA and PDA: machine accepts the input w, if there is one sequence of choices that lead it to a final state.
  - Some of the sequence of choices may not lead to acceptance.
- NFA to DFA simulation: constructed power set of states
  - Won't work for TM (or PDA) since we also have to construct power set of the infinite storage
    - Power set of an infinite set is an uncountable set !!!

34

Why use non-determinism

- Powerful expressive model to describe a solution
  - If we are only interested in showing there is a solution
  - So first focus on what non-deterministic machines can solve and how to construct solution
- Can simplify the solution in some cases
  - Ex: "guess" the mid point of *ww* non-deterministically
  - In coding: imagine you spawn multiple threads, as many as you want, and then wait for one of the threads to complete.
  - Since multiple "transitions" may be applied at each step:

    the program (i.e., machine) may have multiple active simultaneous threads,

    any of which may accept the input string when the thread halts

35

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.
- Transition function: δ: $Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R\}}$
  - Set of choices
  - Each choice: goes to a state, writes to tape, moves L or R
  - $\delta(q_0, a) = \{(q_1, b, R), (q_2, c, L)\}$
- Theorem: For every non-deterministic Turing machine there is an equivalent deterministic turing machine that accepts the same language
  - i.e., a deterministic TM that simulates the non-deterministic TM

36

36

37