CS 3313
Foundations of Computing:

Turing Machine Examples

http://gw-cs3313.github.io

Turing Machine

- 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

Example: $L_3 = \{ww \mid w \in \{a, b\}^*\}$

- Not CFL, cannot be generated by CFG nor recognized by PDA.
  - Match symbols having same distances from the "mid-point".
  - Or mid-points, if we are dealing with, say, $\{a^i b^i c^j d^j\}$; etc.
  - red first, then blue, etc.

TOS

… $x_1 x_2 x_3$ … x_j ["mid-point"] x_{j+1} … $x_{n-2} x_{n-1} x_n$ …

- $L_0 = \{ww^R \mid w \in \{a, b\}^*\}$
  - ✓ CFG and PDA
  - ✓ TM [Lecture]: bounce back and forth using "same" approach
    - ➢ However, purple first, then green, etc., red last

Example from Lecture: L = { wcw }

- For TM, we can match symbols through other ways.
  - Recall $\{wcw \mid w \in \{a,b\}^*\}$

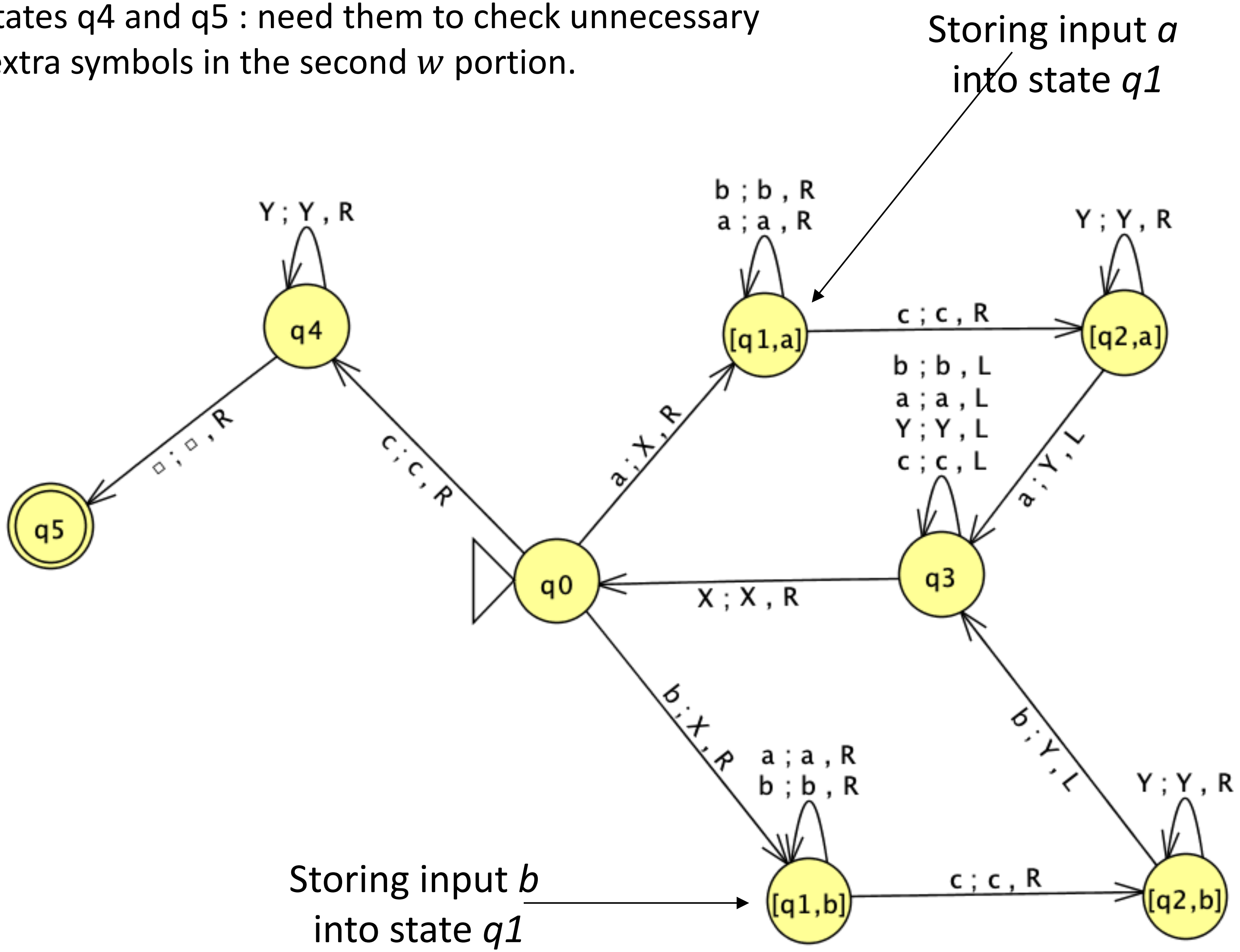
…BB $\cancel{x_1}\,\cancel{x_2}\,\cancel{x_3}$ ... $\cancel{x_j}$ c $\cancel{x_{j+1}}\,\cancel{x_{j+2}}\,\cancel{x_{j+3}}$ ... $\cancel{x_{2j}}$ BB…

Sweep

Accept

Example : L = { wcw | w in {a,b}* }

States q4 and q5 : need them to check unnecessary extra symbols in the second *w* portion.



Example 1: $L_3 = \{ww \mid w \in \{a, b\}^*\}$

- Quite similar, but we do NOT know where the mid-point is.
- Any thought?

Non-determinism? Will talk about NTM later.

Let’s try to find the “mid-point”, deterministically.

But, rather, we differentiate the first and second w’s.
Then, we can apply similar approach.

- **Take-away**: Use a sequence of sub-TMs to “divide & conquer” the problem.

Example 1: $L_3 = \{ww \mid w \in \{a, b\}^*\}$

- **Key observation**: if we know the midpoint then we can leverage the solution we used for *wcw*
- Decompose the problem: view it as two problems:
  1. Identify midpoint
  2. Check if first/left half of input = right half of input
- Use a sequence of sub-TMs to “divide & conquer” the problem.
  1. Design solution/TM transitions to identify midpoint
  2. Design solution/TM to check if left half = right half
  3. Call (1) and after it ends, go to (2)

Example 1: $L_3 = \{ww \mid w \in \{a, b\}^*\}$

- Differentiate the two portions

$$\ldots BB\ \cancel{x_1}\,\cancel{x_2}\,\cancel{x_3}\ \ldots\ \cancel{x_j}\ c\ \cancel{x_{j+1}}\,\cancel{x_{j+2}}\,\cancel{x_{j+3}}\ \ldots\ \cancel{x_{2j}}\ BB\ldots$$

$$\ldots BB\ \cancel{x_1}\,\cancel{x_2}\,\cancel{x_3}\ \ldots\ \cancel{x_j}\ \lambda\ \cancel{x_{j+1}}\,\cancel{x_{j+2}}\,\cancel{x_{j+3}}\ \ldots\ \cancel{x_{2j}}\ BB\ldots$$

Read λ ?

Not doable: again, no way to know the mid-point.
How can we achieve the same setup alternatively?

Example 1: $L_3 = \{ww \mid w \in \{a, b\}^*\}$

- Differentiate the two portions

$$\ldots \text{BB } \cancel{x_1}\, \cancel{x_2}\, \cancel{x_3} \ldots \cancel{x_j}\ \lambda\ x_{\cancel{j+1}}\, x_{\cancel{j+2}}\, x_{\cancel{j+3}} \ldots \cancel{x_{2j}} \text{ BB} \ldots$$

Now identified the "mid-point"!

Can make the second portion different from the first.

However, since we want to perform matching in the next step; let's also differentiate the symbols at this step. So, we can "recover", say, the first potion later.

Example 1: $L = \{ww \mid w \in \{a, b\}^*\}$

- Differentiate the two portions and recover the first portion

YX XY

…BB $\overset{X}{x_1}\,\overset{Y}{x_2}\,\overset{X}{x_3} \dots \overset{X}{x_j}\ \lambda\ \overset{X}{x_{j+1}}\,\overset{X}{x_{j+2}}\,\overset{Y}{x_{j+3}} \dots \overset{X}{x_{2j}}$ BB…

So, if we have ababaaababaa then write X for *a* and write Y for *b*

We will end up with XYXYXXXYXYXX

Then, by recovering, we get ababaaXYXYXX

Now, we can try to match a with the first X we encounter, and b with the first Y we encounter.

Example 1: $L_3 = \{ww \mid w \in \{a, b\}^*\}$

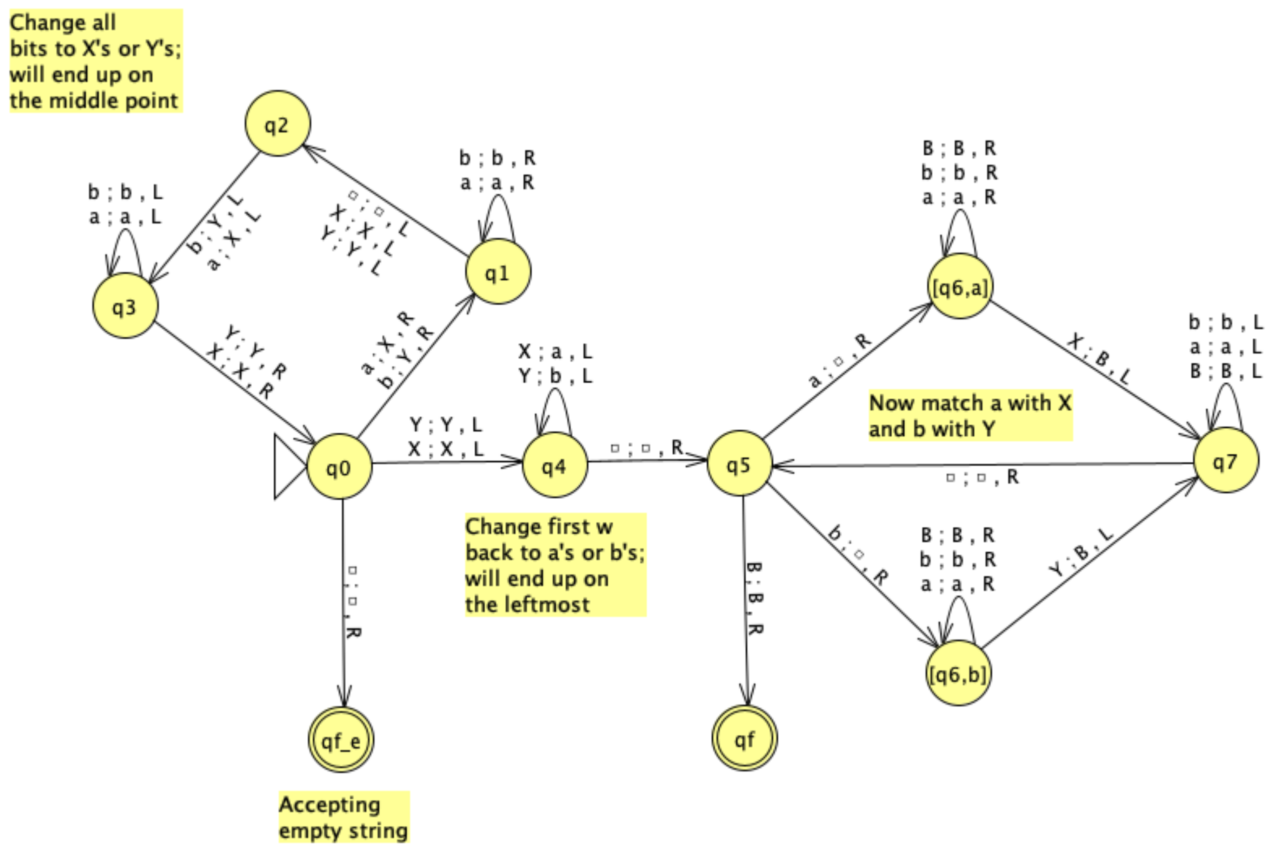
- Design the algorithm:
- Phase 1: Find midpoint
  - For every leftmost a/b write X/Y and move to rightmost unmarked a/b
    - Since should be the leftmost unmarked in the second *w*
  - Change a/b to X/Y and go left to find leftmost a/b (unmarked symbol)
  - If immediately left of rightmost a/b is a X/Y then we are at the midpoint
  - Change all X/Ys left of midpoint to a/b
- Phase 2: Check if left half of input is equal to right half
  - Read

Example 1: $L_3 = \{ww \mid w \in \{a, b\}^*\}$

- Design the algorithm: Exercise
- Phase 1: Find midpoint – write out the steps/states
- Phase 2: Check if left half of input is equal to right half – write the steps/states

Example 1: $L = \{ww \mid w \in \{a, b\}^*\}$

- Try to come up with the "Algorithm" and its corresponding state and transitions.
- Here's the transition diagram



Exercise 2: $L = \{ a^i b^j c^i d^j \mid i,j>0 \}$

- Describe TM to accept L