CS 3313
Foundations of Computing:

Computation Models and Turing Machine Model

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

Next..

- Turing Machine model
  - TM as an automaton
- Changing the basic TM model…..
  - Multiple tracks, multiple tapes, two-way tape/storage
  - Non-deterministic TM
  - Equivalence of Deterministic and Non-deterministic TMs
    - Simulation procedure
  - Simulation of RAM (Random Access Machine) on a TM
- Solvable and Unsolvable problems…
- Time and space complexity
- Other models of computation: λ-calculus (functional programming)

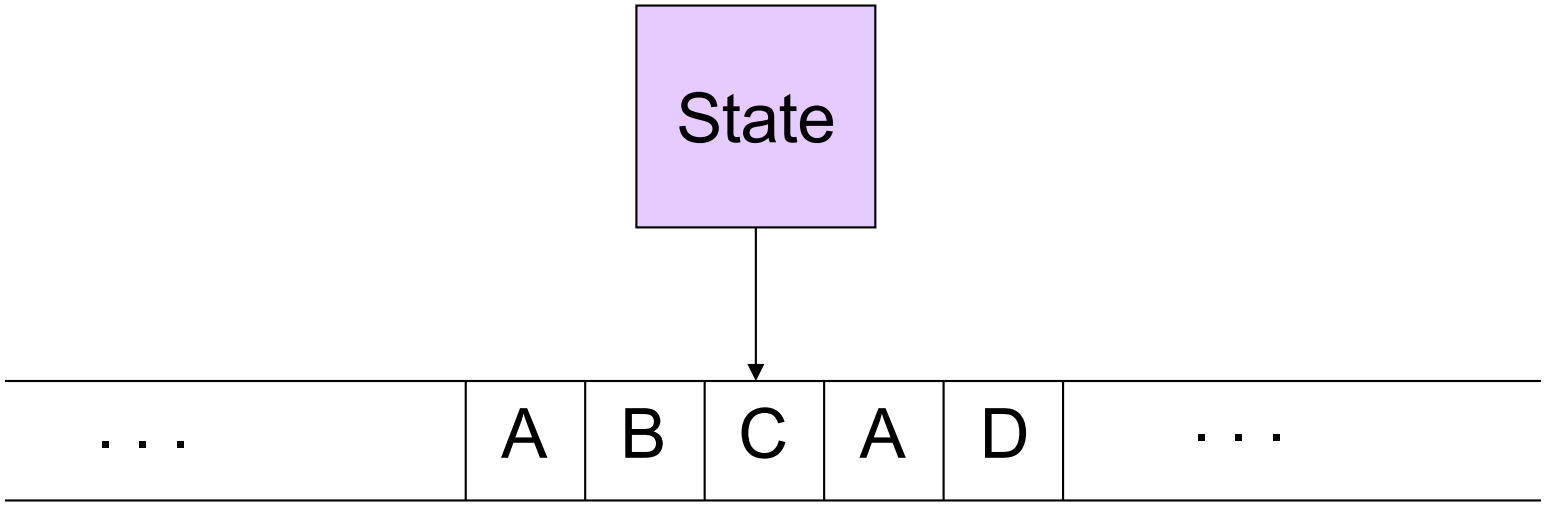
Moving on from PDAs….Turing Machines

- Finite State Automata (DFA/FSM): finite number of states
  - States store summary of past input/events
  - No external storage …so cannot have a counter to store variable
- Pushdown Automata (PDA): Add a “external” stack storage to a NFA
  - Single stack – first in-last out
  - What is stored in stack comes out in reverse when it is popped
- Extend PDA…..Two stacks, two-way input tape, etc…..OR
- Generalize the storage form to random access
  - Can store into any location and read from any location
  - Instead of a “box” as storage, we move to a line of bookshelves
- Turing Machine: NFA + external storage on a tape

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

Example 1: Turing Machine

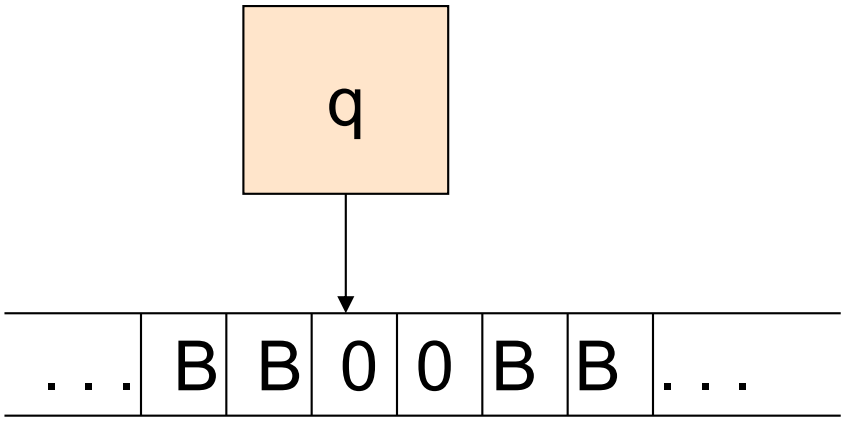
- This TM scans its input right, looking for a 1
- If it finds one, it changes to 0, goes to final state and halts
- If it reaches a blank, it changes it to a 1 and moves left
- States = {q (start), f (final)}.
- Input symbols = {0, 1}.
- Tape symbols = {0, 1, B}.
- $\delta(q, 0) = (q, 0, R)$.
- $\delta(q, 1) = (f, 0, R)$.
- $\delta(q, B) = (q, 1, L)$.

Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

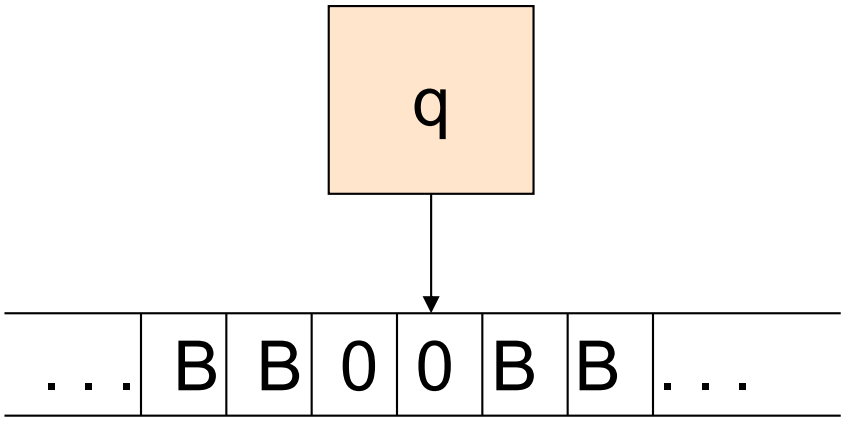


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

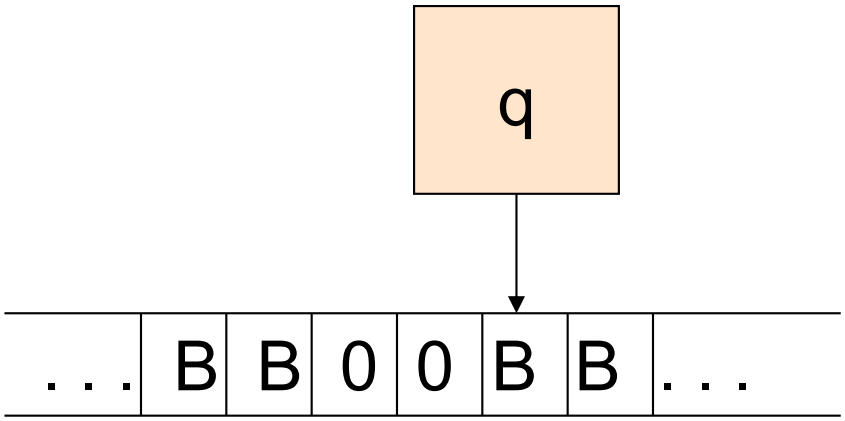


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

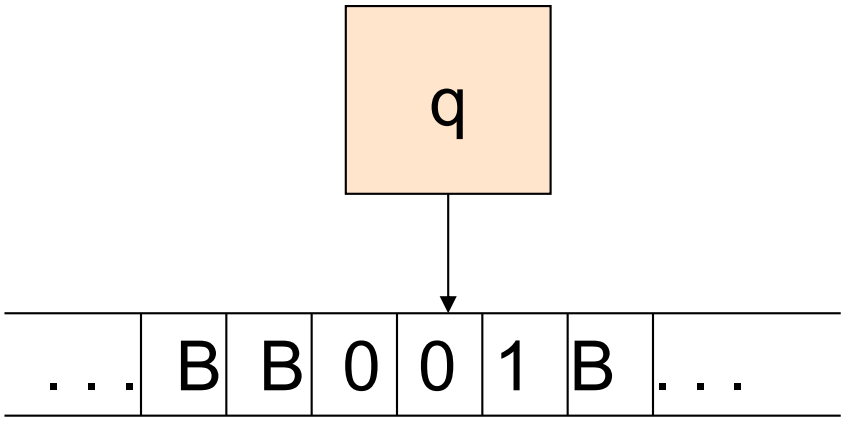


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

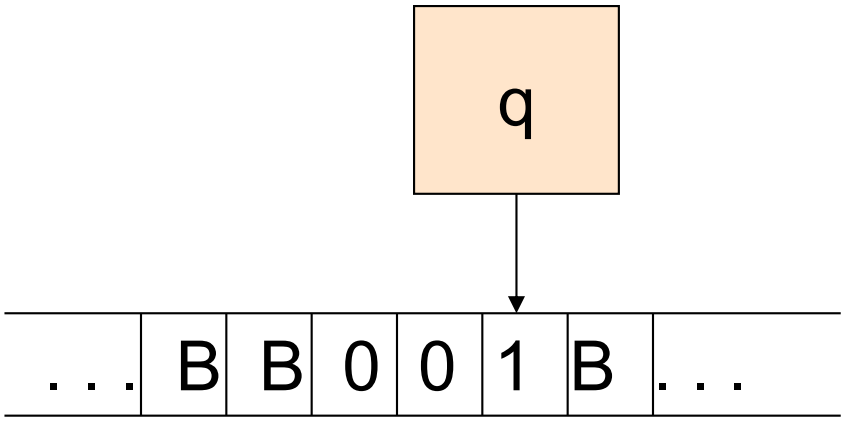


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$

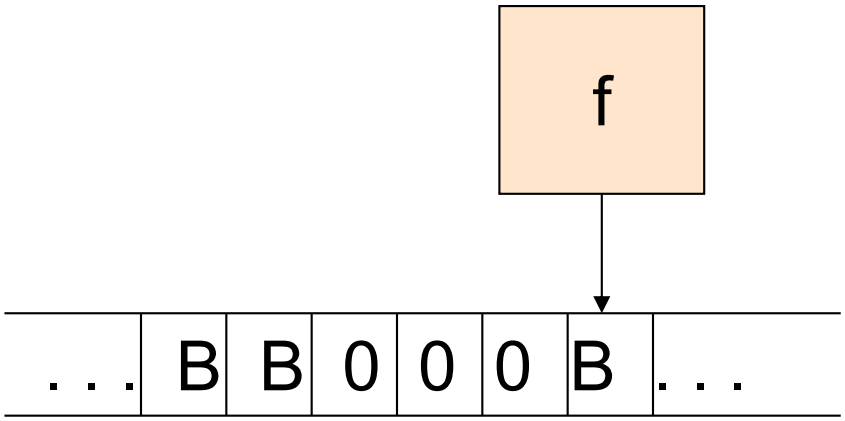


Simulation of TM

$\delta(q, 0) = (q, 0, R)$

$\delta(q, 1) = (f, 0, R)$

$\delta(q, B) = (q, 1, L)$



No move is possible.
The TM halts and accepts.

Example 2: Turing Machine

- States = {q (start), f (final)}. Input symbols = {0, 1}.
- Tape symbols = {0, 1, B}.
- $\delta(q, 0) = (q, 0, R)$.
- $\delta(q, 1) = (q, 0, R)$.
- $\delta(q, B) = (q, B, L)$.

- What happens in this TM for input w = 010 ?

Instantaneous Descriptions (ID) of a Turing Machine

- Initially, a TM has a tape consisting of a string of input symbols surrounded by an infinity of blanks in both directions.
- The TM is in the start state, and the head is at the leftmost input symbol.
- An ID is a string $\alpha q \beta$, where $\alpha\beta$ includes the tape between the leftmost and rightmost nonblanks.
  - The state q is immediately to the left of the tape symbol scanned
  - If q is at the right end, it is scanning B.
    - If q is scanning a B at the left end, then consecutive B's at and to the right of q are part of α.

TM ID's – (2)

- As for PDA's we may use symbols ⊢ and ⊢* to represent "becomes in one move" and "becomes in zero or more moves," respectively, on ID's.
- Example: The moves of the previous TM are

  q00 ⊢ 0q0 ⊢00q ⊢ 0q01 ⊢ 00q1 ⊢ 000f

Formal Definition of Moves: Instantaneous Description (ID)

- At any instant in time, the TM is in a state q, its tape head is reading some symbol Z, the string α is to the left of the tape head, and the string β is to the right of the tape head:

  this ID is denoted as $\alpha q Z \beta$

- If $\delta(q, Z) = (p, Y, R)$, then $\alpha q Z \beta \vdash \alpha Y p \beta$
  - If Z is the blank B, then also $\alpha q \vdash \alpha Y p$
- If $\delta(q, Z) = (p, Y, L)$, then for any X, $\alpha X q Z \beta \vdash \alpha p X Y \beta$
  - In addition, $qZ\beta \vdash pBY\beta$

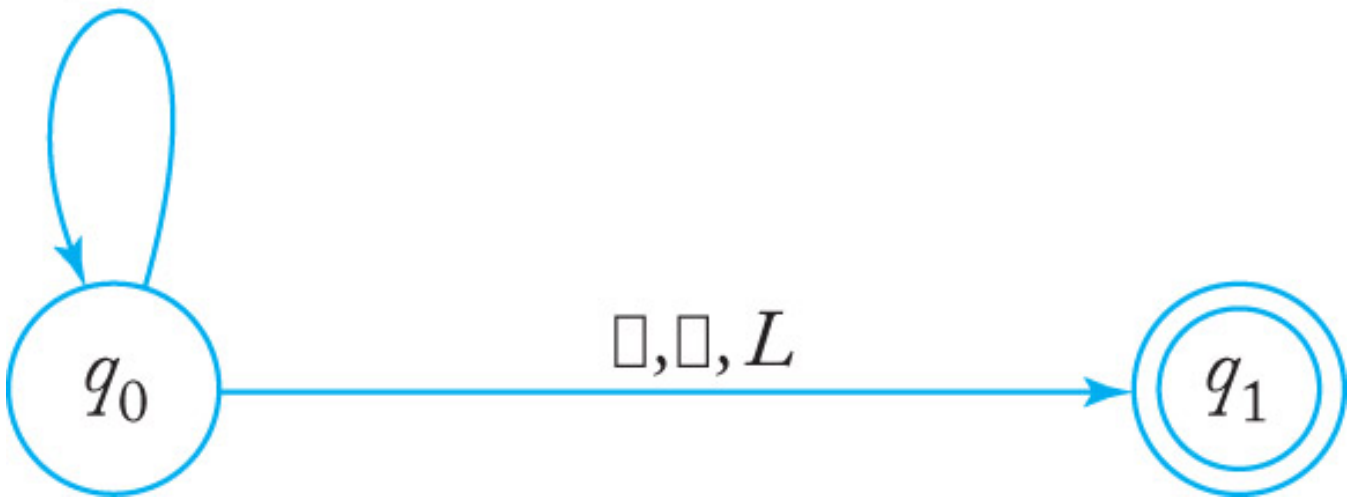
Languages of a TM

- A TM defines a language by final state, as usual.
- $L(M) = \{w \mid q_0 w \vdash^* I$, where I is an ID with a final state$\}$.
  - TM halts in this configuration
  - Alternate definition: accepts as long as state is a final state

Transition Graphs for Turing Machines

- In JFLAP, you will see a transition graph for a TM
- In a Turing machine transition graph, each edge is labeled with three items:
    1. current tape symbol,
    2. new tape symbol, and
    3. direction of the head move

$$\delta(q_0,a)=(q_0,b,R)$$
$$\delta(q_0,b)=(q_0,b,R)$$
$$\delta(q_0,B)=(q_1,B,L)$$



Recursively Enumerable Languages

- Consider the class of languages L, where if w is in L then there is a TM that halts on input w
  - Does not say what happens when w is not in the language
    - The TM may never halt
- This class of languages is called the *recursively enumerable languages*.
  - Why? The term actually predates the Turing machine and refers to another notion of computation of functions.

Recursive Languages

- An *algorithm* is a TM, accepting by final state, that is *guaranteed to halt* whether or not it accepts.
- If L = L(M) for some TM M that is an algorithm, we say L is a *recursive language*.
  - Why? Again, don't ask; it is a term with a history.

Turing Machine Design: Examples

- A TM function can be captured by describing its behavior by an “algorithm”
  - First describe how TM works
  - Then capture how states can be encoded to capture specific actions/steps
  - Finally, formally specify the transition function

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

- We know this can be accepted by a PDA…so use this example to set things up.
- Input on the tape: string x followed by Blanks (B)
  - Tape head is at the leftmost symbol of x

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

- Algorithm:
  1. Read symbol (a or b), mark it as checked (with an X) and "store" it in the state
     - If the symbol read is a Blank then accept.
  2. Move right until you hit a B, then move one position left.
  3. Check if symbol on tape = symbol stored in state, mark with B, move left, go to 4
     1. Else halt and reject
  4. Go to the left end of the input – skip over all symbols until you hit an X, move one tape cell to the right, and goto 1.

TM Trick…Storage in state

- "store" it in the state…what/how ?
- Recall – a state can store/summarize finite amount of information…..
  - We have read an "a"
  - We have read a "b"
  - ….etc.
- *Think of the state as having two components* $[q, X]$
  - State q, and symbol X

Example 1: L = { w w^R | w in {a,b}* }

- Algorithm:

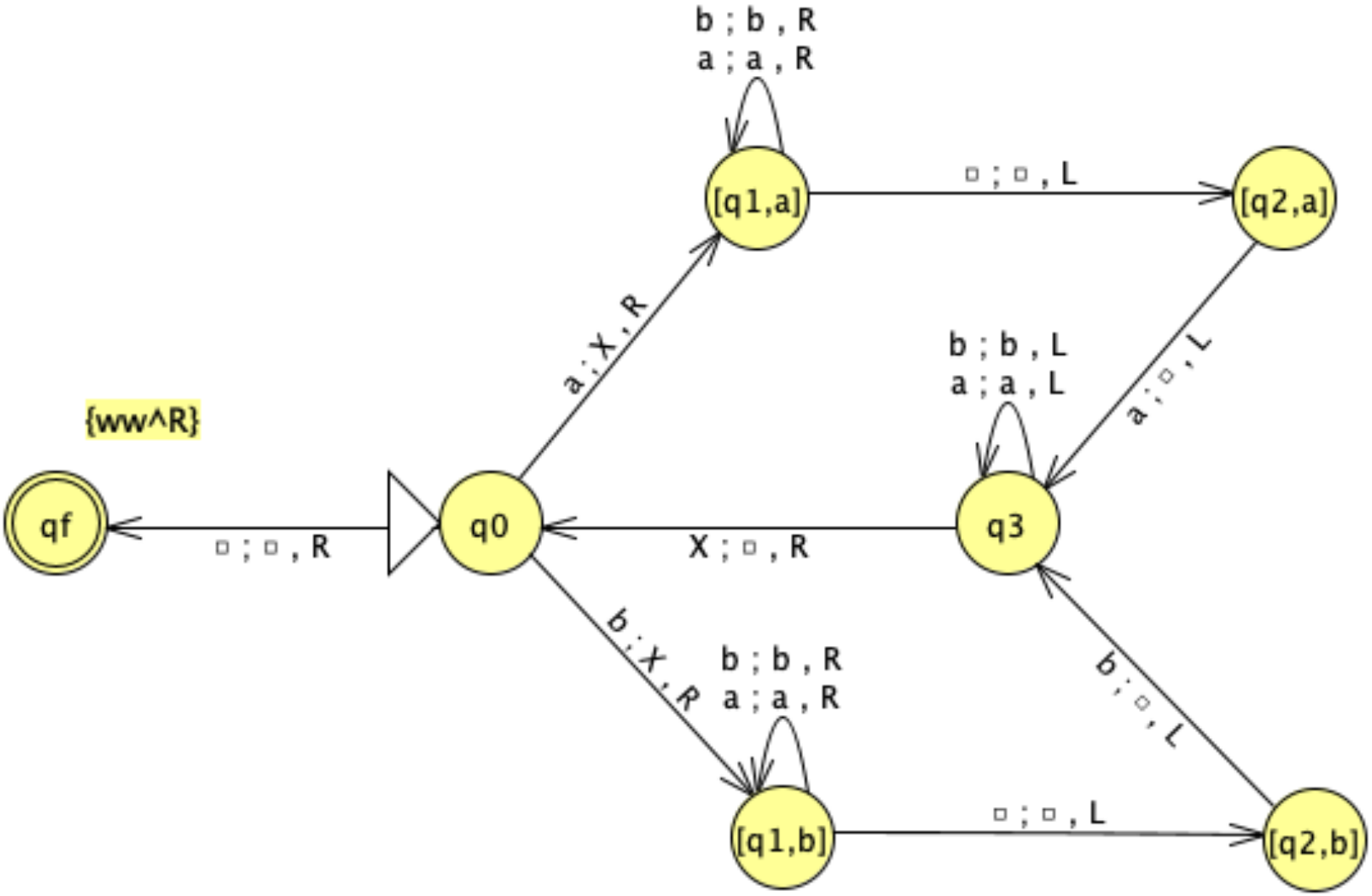
1. State q_0 :
   - Read symbol a, write X to tape, move right, goto state $[q_1,a]$
   - Read symbol b, write X to tape, move right, goto state $[q_1,b]$
   - Read B, write B, move right goto final state q_f
2. State q_1: Move right until you hit a B, then move one position left and goto state q_2…but keep the symbol stored in the state
   - From $[q_1,a]$ goto $[q_2,a]$
   - From $[q_1,b]$ goto $[q_2,b]$
3. State q_2: Check if symbol on tape = symbol stored in state, mark with B, move left, go to 4 else halt and reject
   - If state = $[q_2,a]$ then check if input is a then goto q_3 write B to tape
   - If state = $[q_2,b]$ then check if input is b then goto q_3 write B to tape
4. State q_3: Go to the left end of the input – skip over all symbols until you hit an X, move one tape cell to the right, and goto 1 (state q_0)

Example 1: L = { w w^R | w in {a,b}* }

- Transition Function: start state is q_0

1. state q_0
   - $\delta(q_0, a) = ([q_1, a], X, R)$ /* read symbol, store in state */
   - $\delta(q_0, b) = ([q_1, b], X, R)$ /* write X , move right */
   - $\delta(q_0, B) = (q_f, B, R)$ /* change state to q1 */
2. State q_1:
   - $\delta([q_1, b], a/b) = ([q_1, b], a/b, R)$ /* skip over all symbols until */
   - $\delta([q_1, b], B) = ([q_2, b], B, L)$ /* you read B */
   - $\delta([q_1, a], B) = ([q_2, a], B, L)$
3. State q_2:
   - $\delta([q_2, a], a) = (q_3, B, L)$ /* check if symbol on tape equal to */
   - $\delta([q_2, b], b) = (q_3, B, L)$ /* symbol stored in state, mark with B */
     /* else reject and halt
4. State q_3:
   - $\delta(q_3, a) = (q_3, a, L)$ /* go to left, skip over all symbols until X */
   - $\delta(q_3, b) = (q_3, b, L)$ /* you are going to the leftmost input that */
   - $\delta(q_3, X) = (q_0, X, R)$ /* not been checked yet */

Transition graph for $\{ww^R\}$



Exercise 1: L = { wcw | w in {a.b}* }

- Recall this was not a CFL…
- Design a TM to accept this language.