Theory of Computation
Chapter 3

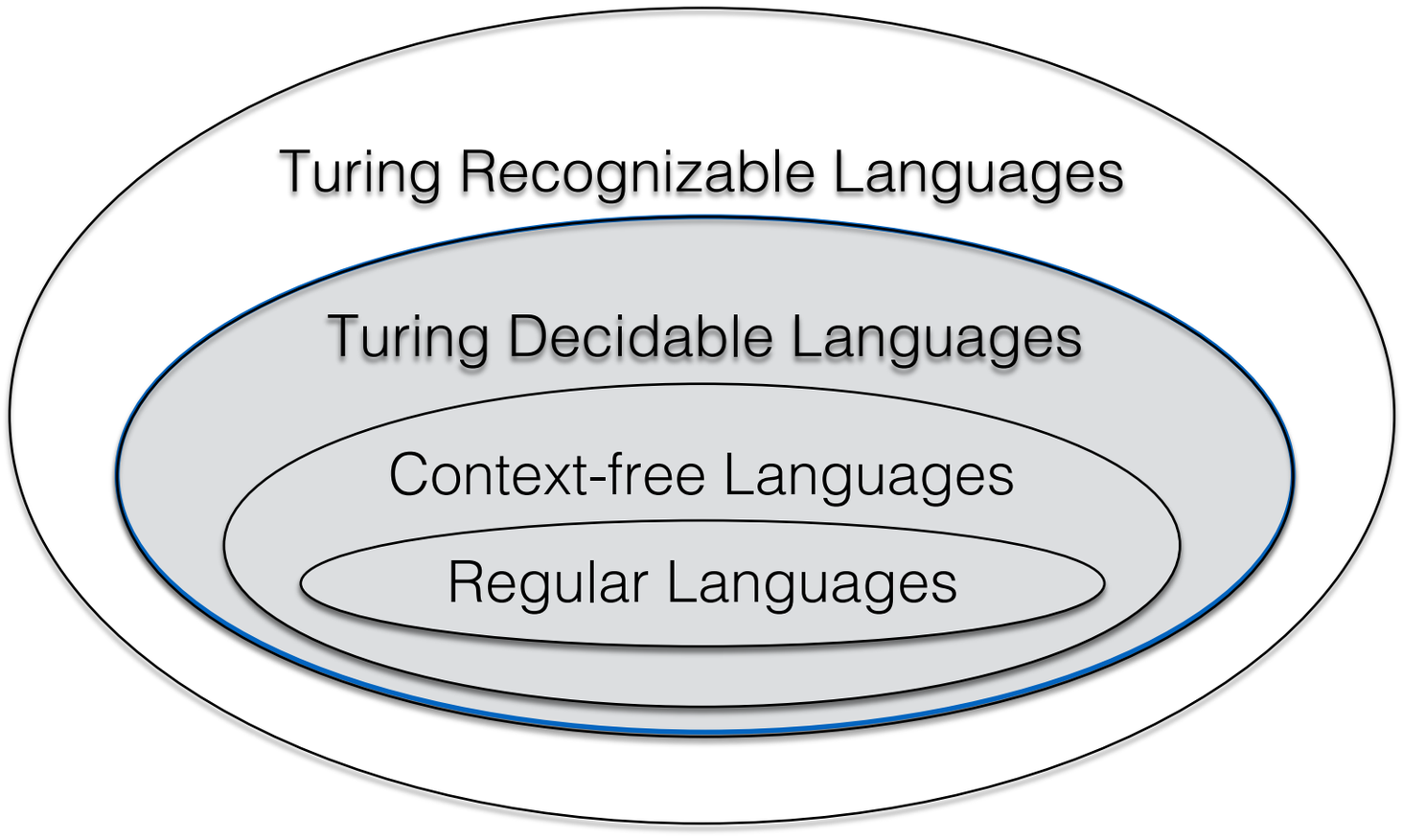
Turing Machines

School of Engineering | Computer Science

Alan Turing
1912-1954

- Proposed “Turing machines” – key theoretical underpinning for theory of computation (1936)
- Key in British decryption of German communication during WWII; co-designed “bombe” machine to automatically decrypt German messages
- Wrote “Computing machinery and intelligence” (1950), proposing the “Turing test” to define machine intelligence

All Languages



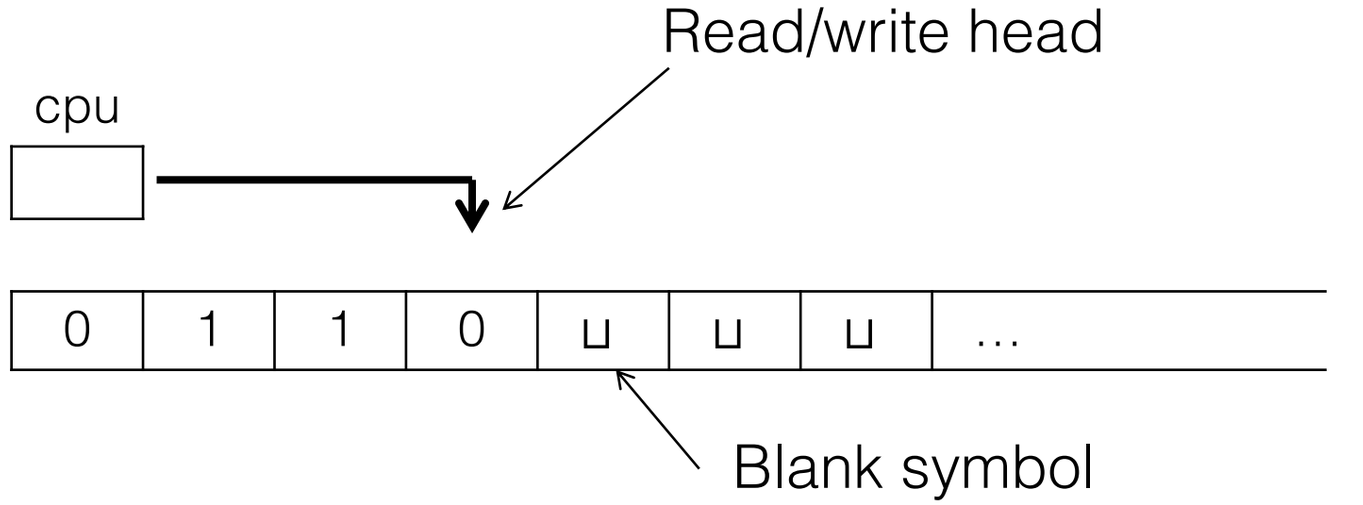
Turing Machines are <u>robust</u>: minor changes to the definition do not change the power of the TM.

Background on Turing Machines

- The Turing machine was designed by Alan Turing in 1936
- It is not a real machine, but a model of a general purpose computer
- You can think of it as a finite automata with unbounded and unrestricted memory
- Forms the current theoretical limit of computation

Turing Machines

- Parts of a Turing Machine (TM):
  - Memory
  - Read/write head
  - CPU/brain
- Rules:
  1. The head can move left or right
  2. The head can read from and write to the tape
  3. The tape is infinite



Turing Machine

- Formal Definition:
  - A TM is a 7-tuple (Q, Σ, Γ, δ, q_0, q_{accept}, q_{reject}) where:
    1. Q is a finite set of states
    2. Σ is the finite set of input alphabet symbols ($\sqcup \notin \Sigma$)
    3. Γ is the finite set of tape alphabet symbols (($\sqcup \in \Gamma$, $\Sigma \subset \Gamma$)
    4. Transition δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
    5. $q_0 \in Q$ is the start state
    6. $q_{accept} \in Q$ is the accept state
    7. $q_{reject} \in Q$ is the reject state

L = move left
R = move right

Conceptual Ex of a TM

- Use Turing Machines to decide membership in a language
  - B = {w#w | w ∈ {0, 1}*}.
- On input x ∈ {0, 1}*
  1. Scan the tape, if no # found, reject.
  2. Read and cross of the first non-crossed off symbol on the tape before # then read the first non-crossed off symbol on the right side of #. If they are the same, cross it off, scan left, and repeat this step.
  3. If there is no symbol on the right side to cross off, reject.
  4. If these symbols are not the same, reject.
  5. If all symbols to the left of the # symbol are crossed off, scan right, if any non-crossed off symbols remain to the right of the # symbol, reject.
  6. If all symbols are crossed off, accept.

Conceptual Example of a TM

- Use Turing Machines to decide membership in a language B = {w#w | w ∈ {0, 1}*}
- Example:
  - 0101#01011 (Step 1 - check for #)
  - 0101 # 01011,
    x101 # x1011,
    xx01 # xx011,
    xxx1 # xxx11,
    xxxx # xxxx1 (Step 2 – zigzag across the tape and cross the symbols off)
  - xxxx # xxxx1 (Step 5 – when all the symbols to the left of the # are crossed off, if any non-blank symbols to the right of the # remain reject) - Reject

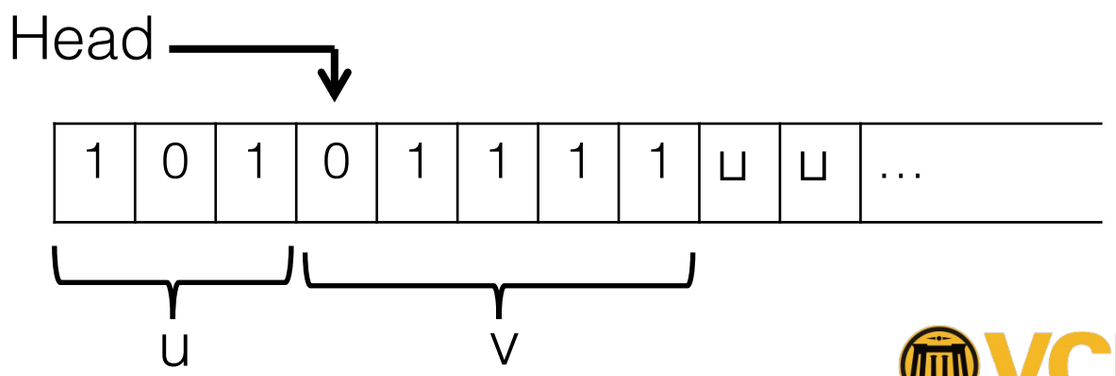
TM Computation

1. Initially the left-most n squares of the tape holds the input $w = w_1w_2\ldots w_n \in \Sigma^*$, the rest of the tape is blank (␣ symbols)
2. Once M starts, it follows the transitions in δ
3. If M tries to move its head off the left-end of the tape, the head will just stay where it is (over cell1)
4. Computation continues until M enters either q_{accept} or q_{reject}, at which point it stops (halts). Otherwise, the machine goes on forever.

Configuration of a TM

- The configuration of a Turing Machine consists of:
  1. Head location
  2. Tape contents
  3. Current state of the machine
- Represented as: uqv where u and v are in the tape alphabet and are tape contents ($u, v \in \Gamma^*$) and currently the "head" is located on the first symbol of v
- Example $\underbrace{101}_{u} q_{state} \underbrace{01111}_{v}$

current state on the machine = q_{state}



TM Configuration

- A configuration C_1 yields C_2 if M can legally go from C_1 to C_2 in a single step
- A step means the machine:
  1. Reads the current cell content
  2. Writes to the current cell
  3. Moves the head left or right by one step
- The start configuration is q_0w (q_0 is the start state, $w \in \Sigma^*$ and w is the input string on the tape)
- Accepting/rejecting states: q_{accept}, q_{reject} (halting configurations where computation stops)

TM Acceptance

- Formal Definition of Acceptance:
  - A Turing Machine accepts input w if a sequence of configurations $C_1 \ldots C_k$ exist where:
    1. C_1 is the starting configuration
    2. Each C_i yields C_{i+1}
    3. C_k is an accepting configuration

TM Acceptance

- Formal Definition of Acceptance:
  - $L(M)$ is the language accepted / recognized by M
  - $L(M) = \{x \in \Sigma^* \mid M \text{ halts on } x \text{ and accepts } x\}$
    - Ex: M "recognizes" L
      - If we feed $x \in L$ into M, M halts on x and accepts
      - What if we feed x $\notin$ L into M?
        - M can either halt and reject or loop forever

Turing-Recognizable

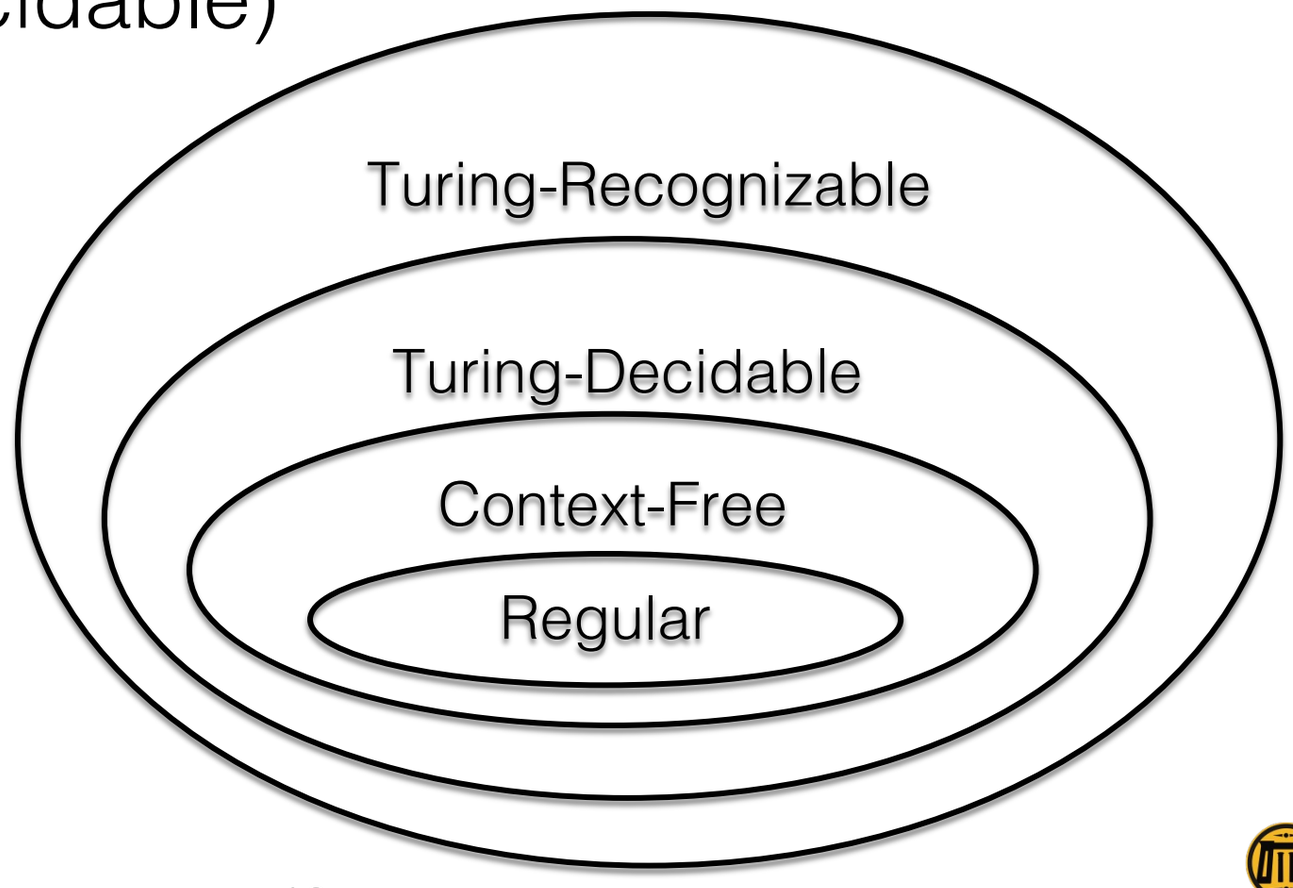
- Turing-Recognizable or Recursively Enumerable:
  - A language is Turing-recognizable if some Turing Machine "recognizes" it
  - Recognizes means:
    1. If input $x \in L$, the Turing Machine halts and accepts
    2. If input $x \notin L$, the Turing Machine either:
       a. Halts and rejects
       b. Loops forever

Turing-Decidable

- Turing-Decidable or Decidable Recursive:
  - A language L is Turing-decidable if some Turing Machine "decides" it
  - Decides means:
    1. If input $x \in L$, the Turing Machine halts and accepts
    2. If input $x \notin L$, the Turing Machine halts and rejects (there is no looping forever)

Turing Recognizable vs. Turing Decidable

- Notes:
  - If L is Turing-decidable ⇒ L is Turing-recognizable
  - If L is Turing-recognizable ⇏ L is Turing-decidable (may not be decidable)



Example of Decidable Language

- Example of a Turing-Decidable Language:
  - L = $\{0^{2^n} \mid n \geq 0\}$ (assume that $\Sigma = \{0\}$)
  - Design TM M so that it accepts L on inputs w ∈ Σ^*

1. If the tape is empty, reject (first symbol is ⊔)
2. The machine sweeps left to right on the tape:
   a. If it sees a single 0, then it accepts.
   b. If it sees k 0's for $k > 1$ and k is odd, reject.
3. The machine sweeps left and right across the tape and crosses off every other zero, then returns to Step 2.

Example of Decidable Language

- Example of a Turing-Decidable Language:
  - L = $\{0^{2^n} \mid n \geq 0\}$ (assume that $\Sigma = \{0\}$)
  - Design TM M so that it accepts L on inputs $w \in \Sigma^*$
    - What does this look like:
      - $0^{2^3} = 0^8 = 00000000$ (> 1, and not odd, so go to Step 3)
        x0x0x0x0 (cross off every other 0, return to Step 2)
        0000 (> 1, and not odd, so go to Step 3)
        x0x0 (cross off every other 0, return to Step 2)

        00 (> 1, and not odd, so go to Step 3)
        x0 (cross off every other 0, return to Step 2)
        0 (single 0, so accept)

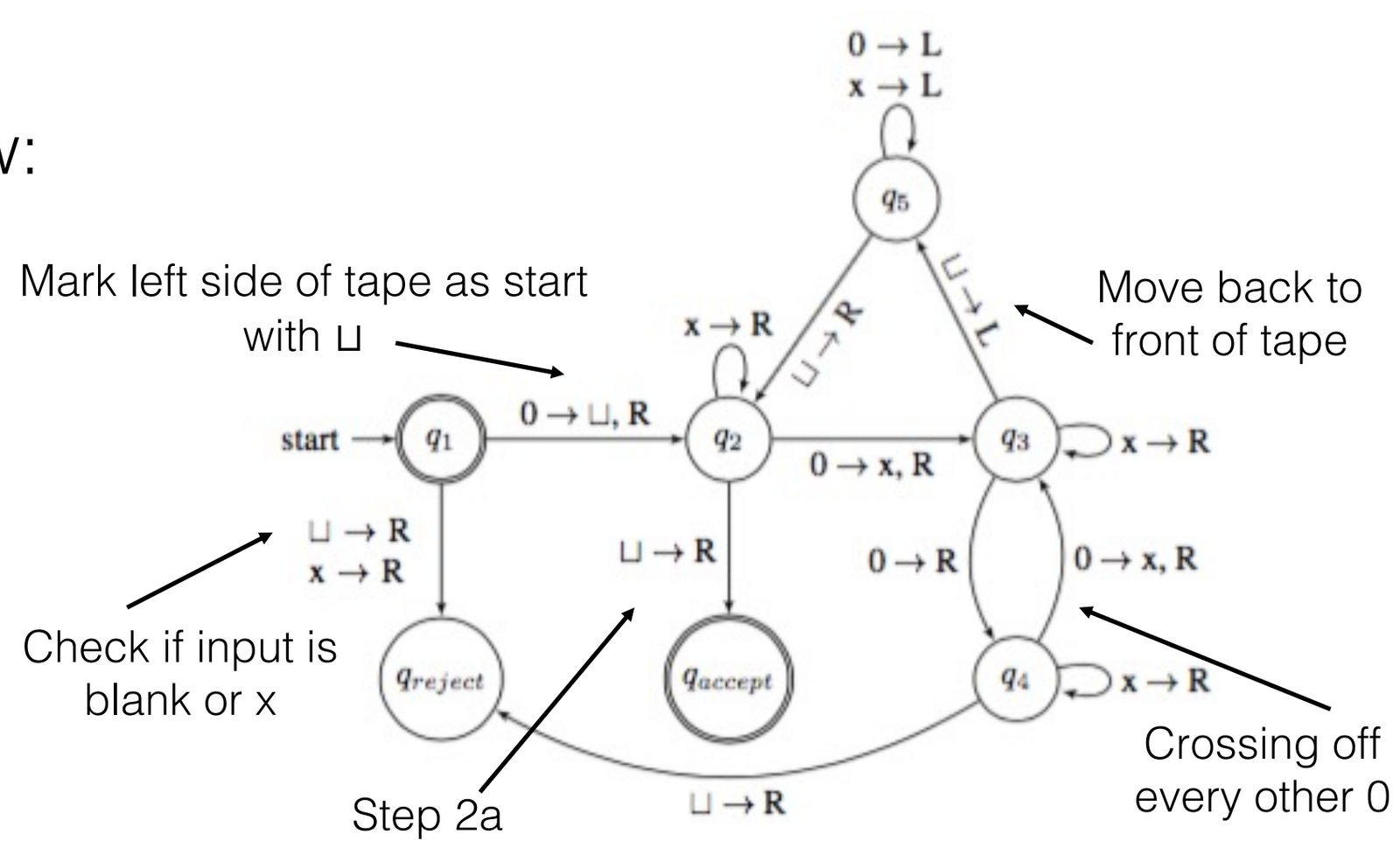
Example of Decidable Language

- Example of a Turing-Decidable Language:
  - L = $\{0^{2^n} \mid n \geq 0\}$ (assume that $\Sigma = \{0\}$)
  - Formal Description of TM M, where M accepts L on inputs w ∈ Σ^*
    - Q = $\{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$
    - $\Sigma = \{0\}$
    - $\Gamma = \{0, x, \sqcup\}$
    - $q_0 = q_1$
    - Accept/ reject states = q_{accept}, q_{reject}
    - δ Cont.

Example of Decidable Language

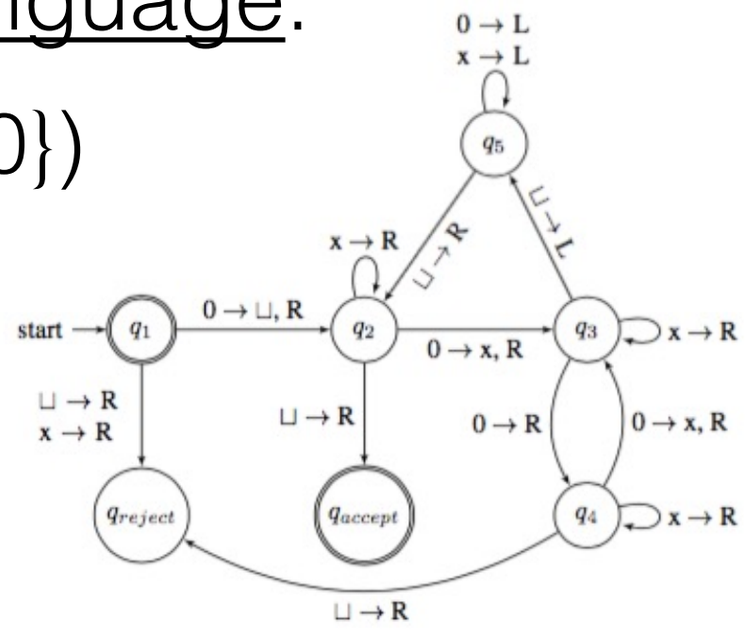
- Example of a Turing-Decidable Language:
  - L = $\{0^{2^n} \mid n \geq 0\}$ (assume that $\Sigma = \{0\}$)
  - Formal Description of M, where M accepts L on inputs $w \in \Sigma^*$
  - δ is as below:

Notation:
- a → b, L(read 'a' from tape, write 'b' to tape, move in L direction)
- a → R (read 'a' from tape, move in R direction)



Example of Decidable Language

- Example of a Turing-Decidable Language:
  - L = $\{0^{2^n} \mid n \geq 0\}$ (assume that $\Sigma = \{0\}$)
  - Ex: 00 (process to get to accept state – configuration form: $uq_{state}v$)



Configuration						Tape Alphabet				
q_1	0	0	⊔	⊔	→	0	0	⊔	⊔	…
⊔	q_2	0	⊔	⊔	→	⊔	0	⊔	⊔	…
⊔	x	q_3	⊔	⊔	→	⊔	x	⊔	⊔	…
⊔	q_5	x	⊔	⊔	→	⊔	x	⊔	⊔	…
q_5	⊔	x	⊔	⊔	→	⊔	x	⊔	⊔	…
⊔	q_2	x	⊔	⊔	→	⊔	x	⊔	⊔	…
⊔	x	q_2	⊔	⊔	→	⊔	x	⊔	⊔	…
⊔	x	⊔	q_{accept}		→	⊔	x	⊔	⊔	…

More on TM's

- <u>How can we find the left-end of a given tape?</u>
  - For each a ∈ Σ, we can use a new tape alphabet symbol containing a and $\dot{a}$, where the new symbol is the same as the old, but with a dot over it. We can mark the first symbol on the tape with $\dot{a}$.
  - We can also place a temporary symbol in the current cell such as a $ and then move left, if it reads the $ at the left-hand end of the tape, then it restores the original symbol that the $ overwrote.

TM Example

- Example 3.12: Decide L = {#x_1#x_2#x_3…#x_i | each x ∈ {0, 1}* and i is distinct (no two strings are the same)}
  - TM M = "on inputs w ∈ {0, 1}*
    1. If the first symbol is ⊔, accept.
    2. Place a mark on the top of the left-most tape symbol. If the first symbol is not #, reject.
    3. Scan right to the next # and place a mark on it. If no # symbol is encountered before ⊔, accept.
    4. Compare the two strings to the right of the marked #'s zigzagging back and forth. If they are equal, the machine rejects.
    5. Move the right-most of the two marks to the next # symbol to the right. If no # is found before ⊔, move the first mark to the next # to its right and the second mark to the next # after that. If no # is found for the second mark, accept. (All strings are compared.)
    6. Go to Step 4.

Try It

- Using the configuration from Slide 7, below, walk through this string to show how a Turing Machine would either accept or reject it. Show the steps you took.
  - Language B = {w#w | w ∈ {0, 1}*} String w = 0110#011
- On input x ∈ {0, 1}*
  1. If no # found, reject.
  2. Zigzag across the tape to the corresponding positions on either side of the # symbol to check if these positions contain the same symbol, if not reject.
  3. “Cross off” symbols as they are checked to keep track of progress
  4. When all symbols to the left of the # symbol are crossed off, if any non-blank symbols remain to the right of the # symbol, reject.

Try It

- Using the configuration on Slide 7, walk through this string to show how a Turing Machine would either accept or reject it. Show the steps you took.
- Language B = {w#w | w ∈ {0, 1}*}
- String w = 0110#011

- 0110#011
- x110#011
- x110#011
- x110#x11
- x110#x11
- xx10#x11
- xx10#x11
- xx10#xx1
- xx10#xx1
- xxx0#xx1
- xxx0#xxx␣
- xxx0#xxx␣
- xxxx#xxx␣
- xxxx#xxx␣
- Reject since ␣ does not equal 0