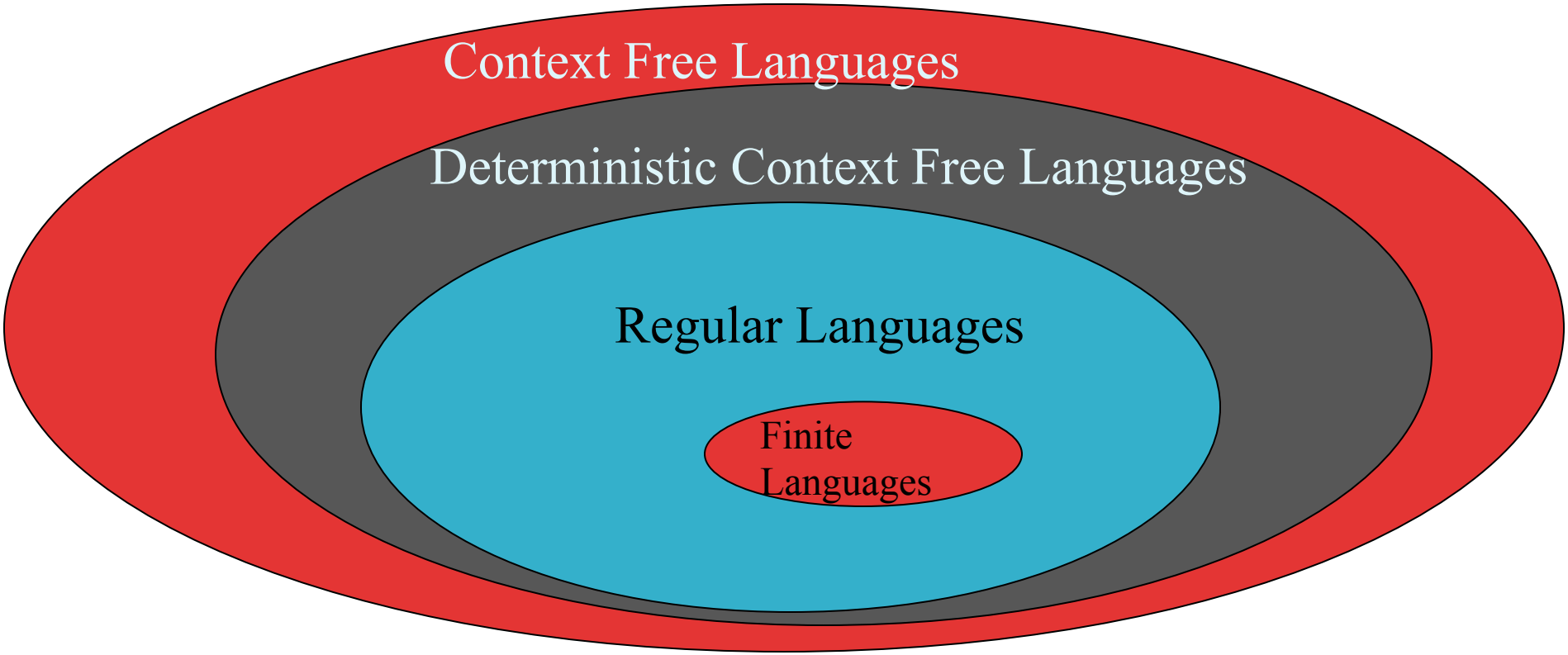
Turing Machines

Currently Our Picture Looks Like



We're going to start to look at languages out here

The Turing Machine

- We investigate the next (and final) classes of languages by first considering the machine
  - Turing Machine
    - Developed by Alan Turing in 1936
    - More than just recognizing languages
    - Foundation for modern theory of computation

The Turing Machine

- Motivating idea
  - Build a theoretical “human computer”
  - Likened to a human with a paper and pencil that can solve problems in an algorithmic way
  - Can do everything that a real computer can do
  - The theoretical machine provides a means to determine:
    - If an algorithm or procedure exists for a given problem
    - What that algorithm or procedure looks like
    - How long it would take to run this algorithm or procedure

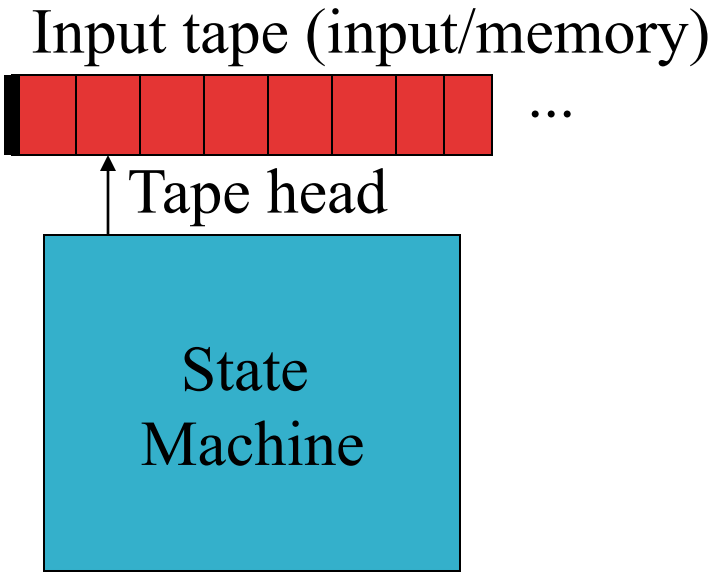
The Church-Turing Thesis (1936)

- "Any algorithmic procedure that can be carried out by a human or group of humans can be carried out by some Turing Machine"
  - Equating algorithm with running on a TM

Turing Machines

- A Turing Machine consists of:
  - A finite state machine
  - An input tape (infinite to the right)
  - A movable read/write tape head
- A move of a Turing Machine
  - Read the character on the tape at the current position of the tape head
  - Change the character on the tape at the current position of the tape head
  - Move the tape head
  - Change the state of the machine based on current state and character read

Turing Machines



- Tape that holds symbol string
- Movable tape head that reads and writes symbols
- Machine that changes state based on current state and what symbol is read

Turing Machines

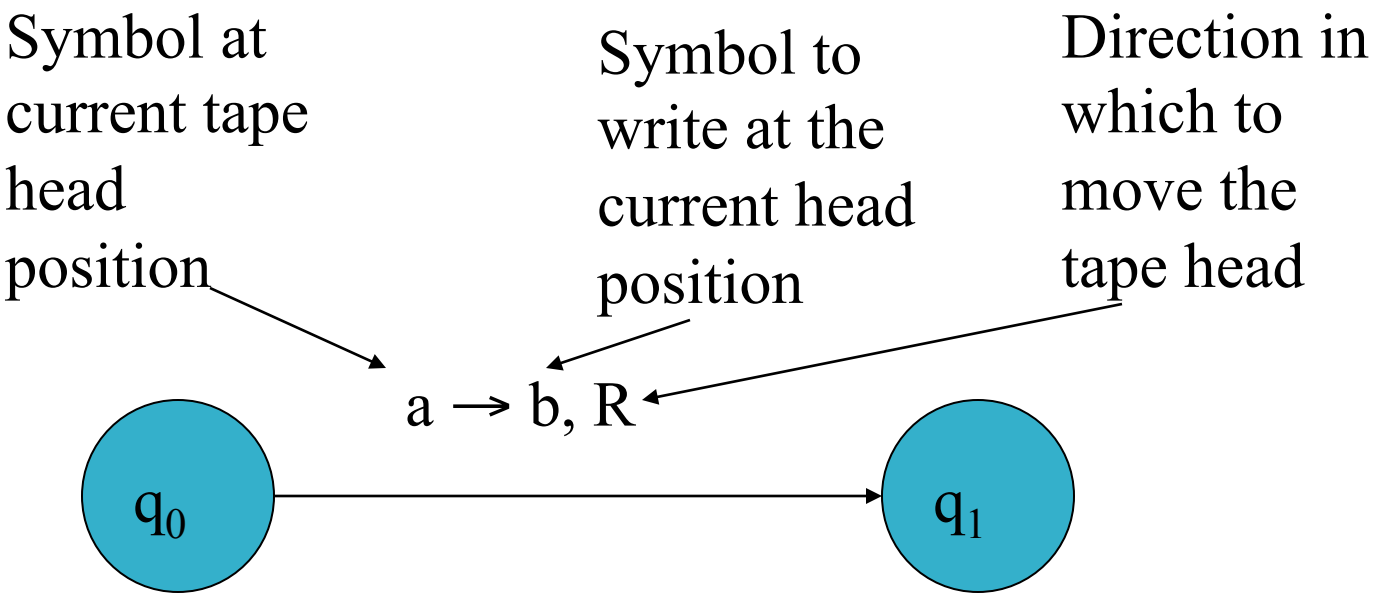
- Let's formalize this:
  - A Turing Machine M is a 7-tuple:
  - $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where
    - Q = a finite set of states
    - Σ = input alphabet (symbols initially on the tape, not including a special blank symbol $\sqcup$)
    - Γ = tape alphabet (symbols that can be written onto the tape. Includes symbols from Σ. Can include $\sqcup$)
    - $q_0 \in Q$ = start state
    - $q_{accept} \in Q$ = accept state
    - $q_{reject} \in Q$ = reject state ($q_{reject} \neq q_{accept}$)
    - δ = transition function

Turing Machines

- Transition function:
  - δ: Q x $\Gamma \rightarrow$ Q x Γ x {R, L}
  - Input:
    - Current state
    - Tape symbol read at current position of tape head
  - Output:
    - State in which to move the machine
    - Tape symbol to write at the current position of the tape head
    - Direction in which to move the tape head (R = right, L = left)

Turing Machines

- Transition Function



Note, Sipser uses shorthand to indicate case when tape is not changed: $a \rightarrow R$
Also, for multiple symbols with common behavior : $a,b,c \rightarrow R$

Turing Machines

- Configuration of a TM
  - Current configuration of a TM represented as:

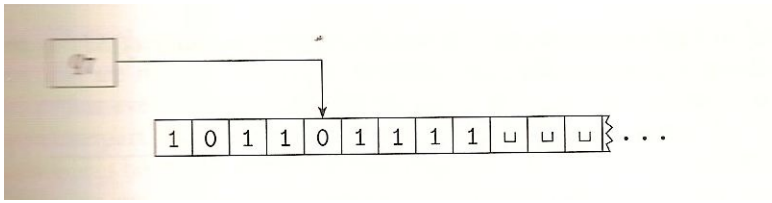
uqv

Contents of tape before tape head

Current state

Contents of tape at and after tape head (until all blanks)

$1011q_701111$



(Figure 3.4 in Sipser)

Turing Machines

- We say configuration C_1 yields configuration C_2 if the Turing machine can legally go from C_1 to C_2 in one move. We indicate:
  - uaq_ibv yields uq_jacv if
    - $\delta(q_i,b) = (q_j,c,L)$
  - uaq_ibv yields $uacq_jv$ if
    - $\delta(q_i,b) = (q_j,c,R)$
  - We can write $C_1 \rightarrow C_2$
  - We can write $C_1 \rightarrow^* C_k$ to indicate C_1 yields C_k in $k-1$ steps

Turing Machines

- Start configuration:
  - To run an input string w on a TM,
    - Start in the starting state
    - Place the string on the tape
    - Place the head at the start of this string:
    - Configuration: q_0w
    - (By definition, trying to move left past the left edge of the tape results in staying in the same position.)

Turing Machines

- Accepting or Rejecting a string
  - An *accepting configuration* is a configuration with state q_{accept}
  - A *rejecting configuration* is a configuration with state q_{reject}
  - The accepting and rejecting configurations are exactly the *halting configurations*. They do not yield further configurations.

Turing Machines

- Accepting a string
  - A Turing Machine, M, accepts input w if
    - $C_1 \rightarrow^* C_k$, where C_1 is the start configuration of M on input w and C_k is an accepting configuration
  - The collection of strings that M accepts is the language of M

Turing Machines

- Running a Turing Machine
  - The execution of a TM can result in 3 possible cases:
    - Accept
      - The machine halts in the accepting state
    - Non-accept
      - The machine halts in the reject state
      - The machine goes into an “infinite loop”

Turing Machines

- Any language that is recognized by some Turing Machine is said to be **Turing-recognizable**.
  - The Turing Machine may loop on some rejected strings
  - Sometimes termed **recursively enumerable** in other books.
- A language is **Turing-decidable** (or just decidable) if some Turing Machine decides it
  - It is recognized by a Turing Machine that halts on all input
    - No infinite loops
  - Sometimes termed a **recursive** language in other books.

Example – Sipser ex. 3.9

- $B = \{w\#w \mid w \in \{0,1\}^* \}$

- Basic idea for TM decider
  1. Read first unmarked symbol…mark it.
  2. "Remember if a 0 or 1"
  3. Move to first unmarked symbol after the #
  4. If doesn't match remembered symbol then fail.
  5. Otherwise mark.
  6. Rewind to start of string.
  7. Repeat until all 0s and 1s are marked.

Example

- Example 3.7 in book

$$A = \{0^{2^n} \mid n \geq 0\}$$

  - Language of all strings of 0's with length a power of 2.
  - Build a TM that will **decide** if a string w is in the language.

Example

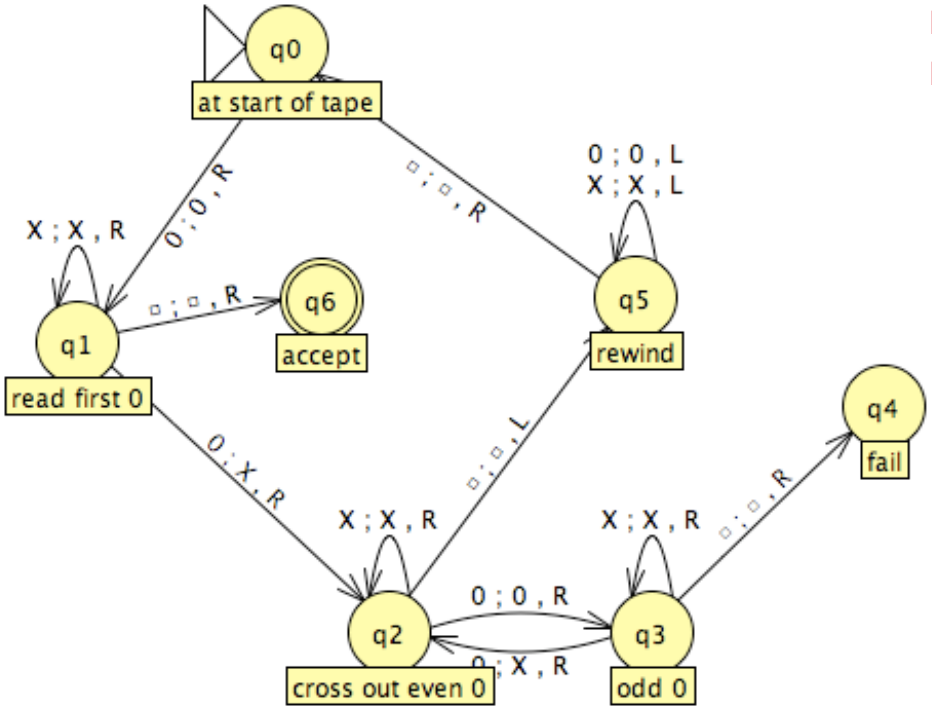
- Example 3.7 in book

$$A = \{0^{2^n} \mid n \geq 0\}$$

- Basic idea:
  1. Sweep L to R, crossing off every other 0
  2. If single 0, then accept
  3. If more than one 0 but odd # of 0's reject
  4. Return to left hand end of tape
  5. Goto 1.

Example

- Example 3.7 in text



Note slightly different notation here. Transition is indicated as X:Y, R indicating symbol X at tape head is replaced by Y and head moves to the right

Note also that this solution assumes that there exists a blank space to the left of the beginning of the input sequence. While this is ultimately equivalent to Sipser's definition, he has a slightly different solution in the book because he assumes that the input tape begins with the input sequence, and there is nothing to the left of it.