Turing Machines

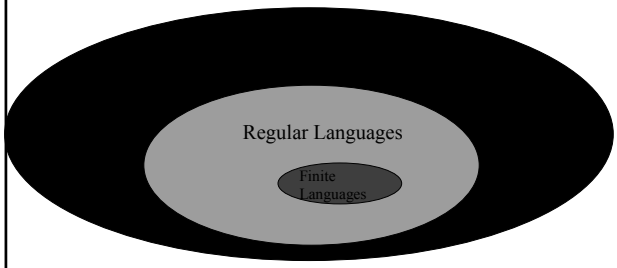
Before We Start

- Any questions?

Languages

- The $64,000 Question
  - What is a language?
  - What is a class of languages?

Now our picture looks like



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

The Turing Machine

- We investigate the next 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

Theory Hall of Fame

- Alan Turing
  - 1912 – 1954
  - b. London, England.
  - PhD – Princeton (1938)
  - Research
    - Cambridge and Manchester U.
    - National Physical Lab, UK
  - Creator of the Turing Test

More about Turing

- "Breaking the Code"
  - Movie about the personal life of Alan Turing
    - Death was by cyanide poisoning (some say suicide)
  - Turing worked as a code breaker for the Allies during WWII.
  - Turing eventually tried to build his machine and apply it to mathematics, code breaking, and games (chess).
    - Was beat to the punch by vonNeumann

The Turing Machine

- Some history
  - Created in response to Kurt Godel's 1931 proof that formal mathematics was incomplete
    - There exists logical statements that cannot be proven by using formal deduction from a set of rules
      - Good Reading: "Godel, Escher, Bach" by **Hofstadter**
    - Turing set out to define a process by which it can be decided whether a given mathematical can be proven or not.

Theory Hall of Fame

- Kurt Godel
  - 1906 -- 1978
  - b. **Brünn, Austria-Hungary**
  - PhD – University of Vienna (1929)
  - Research
    - Princeton University
  - Godel's Incompleteness Theorem

The Turing Machine

- Motivating idea
  - Build a theoretical a "human computer"
  - Likened to a human with a paper and pencil that can solve problems in an algorithmic way
  - 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 would it 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 Machine is still a valid computational model for most modern computers.

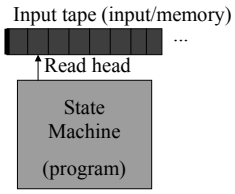
Theory Hall of Fame

- Alonso Church
  - 1903 -- 1995
  - b. **Washington D.C.**
  - PhD – Princeton (1927)
  - Mathematics Prof (1927 – 1967)
  - Advisor to both Turing and Kleene

Turing Machine

- A Machine consists of:
  - A state machine
  - An input tape
  - A movable r/w tape head
- A move of a Turning 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 Machine



- Tape that holds character string
- Movable tape head that reads and writes character
- Machine that changes state based on what is read in

Turing Machine

- In the original Turing Machine
  - The read tape was infinite in both directions
  - The text describes a semi-infinite TM where
    - The tape is bounded on the left and infinite on the right
    - It can be shown that the semi-infinite TM is equivalent to the basic TM.
  - As not to confuse, I'll follow the conventions in the book…(as much as that bothers me!)

Turing Machines

- About the input tape
  - Bounded to the left, infinite to the right
  - All cells on the tape are originally filled with a special "blank" character Δ
  - Tape head is read/write
  - Tape head can not move to the left of the start of the tape
    - If it tries, the machine "crashes"

Turing Machines

- About the machine states
  - A Turing Machine does not have a set of accepting states
  - Instead, each TM has a special halting state, h.
    - Once in the halting state, the machine halts.
  - Unlike PDAs, the basic TM is deterministic!

Turing Machines

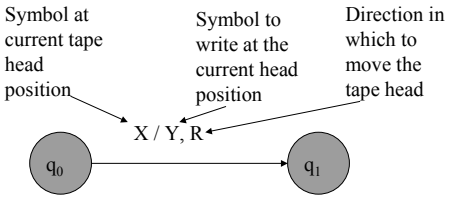
- Let's formalize this
  - A Turing Machine M is a 5-tuple:
  - $M = (Q, \Sigma, \Gamma, q_0, \delta)$ where
    - Q = a finite set of states (assumed not to contain the halting state h)
    - Σ = input alphabet (strings to be used as input)
    - Γ = tape alphabet (chars that can be written onto the tape. Includes symbols from Σ)
    - Both Σ and Γ are assumed not to contain the "blank" symbol
    - δ = transition function

Turing Machines

- Transition function:
  - δ: Q x ($\Gamma \cup \{ \Delta \}$) $\rightarrow$ (Q $\cup$ {h}) x ($\Gamma \cup \{ \Delta \}$) x {R, L, S}
  - Input:
    - Current state
    - Tape symbol read at current position of tape head
  - Output:
    - State in which to move the machine (can be the halting state)
    - Tape symbol to write at the current position of the tape head (can be the "blank" symbol)
    - Direction in which to move the tape head (R = right, L = left, S= stationary)

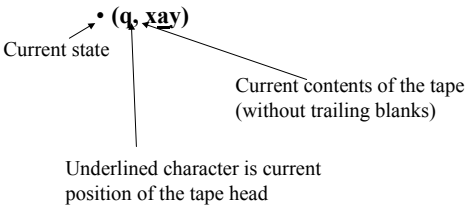
Turing Machines

- Transition Function



Turing Machine

- Configuration of a TM
  - Gives the current "configuration" of a TM



Turing Machine

- We indicate
  - (q, x<u>a</u>y) $\mapsto$ (p, u<u>b</u>v)
  - If you can go from one configuration to another on a single move, and…
    - (q, x<u>a</u>y) $\mapsto^*$ (p, u<u>b</u>v)
  - If you can go from one configuration to another on 0 or more moves.

Turing Machine

- Initial configuration:
  - To run an input string x on a TM,
    - Start in the stating state
    - place the string after the leftmost blank on the tape
    - place the head at this leftmost blank:
    - (q_0, <u>Δ</u>x)

Turing Machine

- Running a Turing Machine
  - The execution of a TM can result in 4 possible cases:
    - The machine "halts" (ends up in the halting state)
    - The machine has nowhere to go (at a state, reading a symbol where no transition is defined)
    - The machine "crashes" (tries to move the tape head to before the start of the tape)
    - The machine goes into an "infinite loop" (never halts)

Turing Machine

- Accepting a string
  - A string x is accepted by a TM, if
    - Starting in the initial configuration
    - With x on the input tape
    - The machine eventually ends up in the halting state.
  - I.e.
    - $(q_0, \underline{\Delta}x) \mapsto^* (h, x\underline{a}y)$
      - for $x,y \in (\Gamma \cup \{\Delta\})^*$, $a \in (\Gamma \cup \{\Delta\})$

Turing Machine

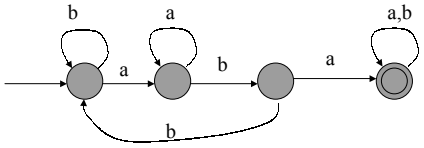
- Running a Turing Machine
  - The execution of a TM can result in 4 possible cases:
    - The machine "halts" (ACCEPT)
    - The machine has nowhere to go (REJECT)
    - The machine "crashes" (REJECT)
    - The machine goes into an "infinite loop" (REJECT but keeps us guessing!)

Turing Machine

- Language accepted by a TM
  - The language accepted by a TM is the set of all input strings x on which the machine halts.
- Questions?

TMs and Regular Languages

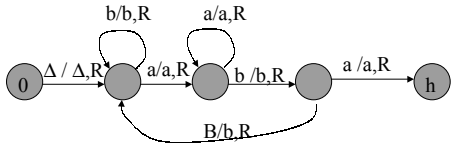
- Example
  - L = { x ∈ { a, b }* | x contains the substring aba }



TMs and Regular Languages

- Example
  - L = { x ∈ { a, b }* | x contains the substring aba }
  - Build a TM that mimics the FA that accepts this language

TMs and Regular Languages



TMs and Regular Languages

- Do you think that JFLAP can handle TMs?
  - You bet!
  - One difference, JFLAP simulates the "original" TM (with an infinite tape in both directions).

Theory Hall of Fame

- Susan Rodgers
  - PhD – Purdue (1985)
  - CS Prof
    - RPI (1989-1994)
    - Duke (1995 – present)
  - Creator and keeper of JFLAP

TMs and Regular Languages

- Example
  - Observations
    - Like FAs TM tape head will always move to the right
    - Like FAs, TM will not write new chars onto the tape
    - Can enter halt state even before the machine reads all the characters of x.

TMs and Context Free Language

- Example
  - Our old friend the palindrome:
  - $L = \{ x \in \{a, b\}^* \mid x = x^r \}$
  - We won't simulate a PDA since the PDA for pal is non-deterministic (we'll deal with non-deterministic TMs later).

TMs and Context Free Language

- Example
  - $L = \{ x \in \{a, b\}^* \mid x = x^r \}$
  - Basic idea:
    - Compare the first character with the last character.
    - If they match compare the second character with the second to last character
    - If they match, compare the 3rd character with the 3rd to last character
    - And so on…
    - For x in pal, eventually we will end up with 0 or 1 unmatched characters.

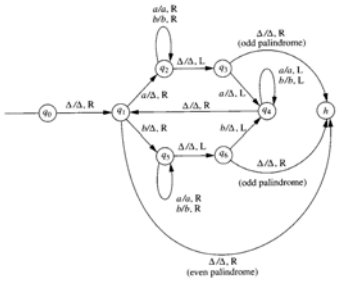
TMs and Context Free Language

- Example
  - $L = \{ x \in \{a, b\}^* \mid x = x^r \}$
  - How to compare characters?
    - Read a character and replace it with blanks.
    - Move across the tape to first blank character
    - Check the character to the left
      - If it's the character that you initially read in, replace it with a blank, move the tape head left until you reach the first blank character and so on.

TMs and Context Free Language

- Example
  - L = { x ∈ {a, b}* | x = x^r }
  - Halting condition:
    - If when you moved left/right after finding the first blank, the character found is a blank, we have found a palindrome!

TMs and Context Free Language



TMs and Context Free Language

- Example
  - L = { x ∈ {a, b}* | x = x^r }
  - States:
    - q_1 – at the leftmost blank
    - q_2 – read an a, move right until you find a blank
    - q_3 – looking for an a, look left after finding rightmost blank
    - q_4 – matched first character read, move left till you find the leftmost blank
    - q_5 – read an b, move right until you find a blank
    - q_6 – looking for an b, look left after finding rightmost blank

TMs and Context Free Language

- Example
  - L = { x ∈ {a, b}* | x = x^r }
  - Let's go to the video tape

Let's try a non-context free language

- Example
  - L = { xx | x ∈ { a, b }* }
  - Basic idea
    - Find and mark the middle of the string
    - Compare characters starting from the start of the string with characters starting from the middle of the string.

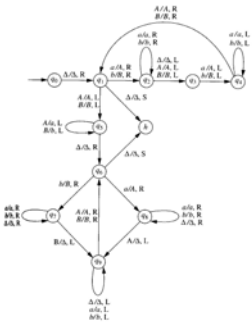
Let's try a non-context free language

- Example
  - L = { xx | x ∈ { a, b }* }
  - Finding the middle of the string
    - Convert first character to it's upper case equiv.
    - Move all the way to the right to the last lower case character and change it to upper case.
    - Move all the way back left to the first lower case character and change it to upper case
    - And so on.

Let's try a non-context free language

- Once you've found the middle,
  - Convert the 1st half of the string back to lower case.
  - Start from the left of the tape
    - Match upper case chars in 1st half with lower case chars in the 2nd.
    - Replace a matched upper case char with blanks
    - Repeat until the 1st half of the string is all blank.

Let's try a non-context free language



Find middle of string

1st half to lower case

Character matching

Let's try a non-context free language

- Let's go to the video tape

Turing Machines

- Questions?

Combining Turing Machines

- Suppose an algorithm has a number of tasks to perform
  - Each task has its own TM
    - Much like subroutines or functions
  - They can be combined into a single TM
    - T_1T_2 is the composite TM
    - $T_1 \rightarrow T_2$

Combining Turing Machines

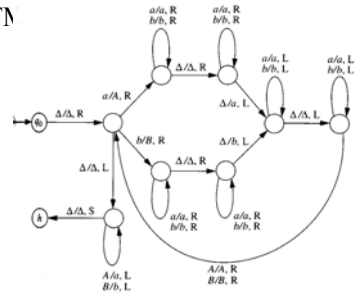
- Composite TMs
  - T_1T_2
    - Start at start state of T_1
    - For any move that causes T_1 to enter the halt state, have T_1 move to the start state of T_2.
    - So T_2 will get executed immediately after T_1 as long as T_1 will halt on a given input.
  - Allows one to define subroutines.

Combining Turing Machines

- Useful "subroutines"
  - Copy TM
    - Creates a copy of the input string to the right of the input separated by a blank
      - $(q_0, \underline{\Delta}x) \mapsto^* (h, \underline{\Delta}x\ \Delta x)$
    - Basic idea
      - Examines each character in turn and writes it after the 1st blank to the right of the input string
      - Keeps track of its progress by converting copied characters to upper case
      - When finished, coverts original string back to lower case.

Combining Turing Machines

- Copy TM

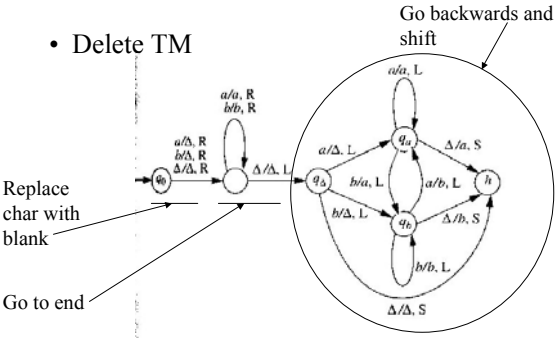


Combining Turing Machines

- Useful "subroutines"
  - Delete TM
    - Deletes a character from a string
    - $(q, y\underline{a}z) \mapsto^* (h, y\underline{z})$
    - Basic idea:
      - Replace the char to be deleted with a blank
      - Go to the rightmost non-blank character of the string
      - Move left towards the "blanked out character" shifting each character along the way to the left.

Combining Turing Machines

- Delete TM



Combining Turing Machines

- Useful "subroutines"
  - Delete TM
  - Something to note
    - The TM must be set up to be in the correct configuration before delete is "called"
    - I.e. the tape head must be at the character to be deleted.
  - Questions?

Summary

- Turing Machines
- Questions
- Stick around for homework help