Context-Free Languages

Regular Languages

- For the past several weeks, we have been looking at Regular Languages:
  - Means for defining: Regular Expression
  - Machine for accepting: Finite Automaton

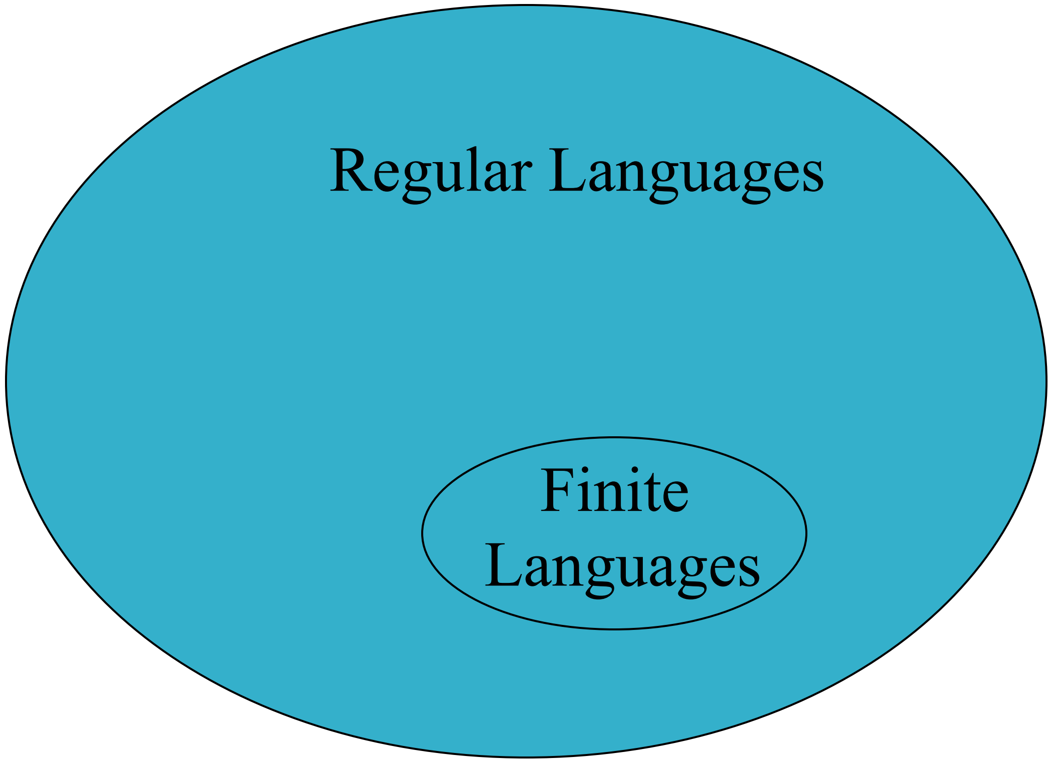
Language Space

- Venn diagram of languages

Is there
something
out here?

YES!

Non-regular
languages



Context-Free Languages

- Context-Free Languages (CFL)
  - The next class of languages that we will discuss
  - Means for defining: Context-Free Grammar
  - Machine for accepting: Pushdown Automata

  - We will eventually show that CFLs are a superset of the class of regular languages
    - Every regular language is also a CFL
    - There are other languages in the class CFL as well

Grammars

- Wikipedia says:
  - Languages can be described as a system of <u>symbols</u> and the <u>grammars (rules)</u> by which the symbols are manipulated
  - <u>Grammar</u> is the set of <u>rules</u> governing the use of language.

Grammars

- Grammars defined for CS Theory:

1. Terminals = Set of symbols that form the strings of the language being defined (alphabet)

2. Variables = Set of symbols representing categories

3. Start Symbol = variable that represents the "base category" that defines the language

4. Substitution rules = set of rules that recursively define the language (also called productions)

First Example

- Grammar G:
  - $A \rightarrow 0A1$
  - $A \rightarrow B$
  - $B \rightarrow \#$

What are the variables?
What are the terminals?

- Start Variable: usually the variable on the left-hand side of the top rule (A)
- Substitution Rules: have a variable on the left, an arrow, and then a combination (string) of variables and other symbols called **terminals** on the right – note that this string can be empty
- Terminals: these are the alphabet we're used to
- Variables: usually capital letters, a variable to the left of an arrow has an equivalent expression to the right of the arrow

- So – what language is described here? $L = \{0^k\#1^k \mid k \geq 0\}$

Grammars

- Let's formalize this a bit:
  - A grammar is a 4-tuple: (V, Σ, R, S) where
    - V is a set of variables
    - Σ is a set of terminals
    - R is a set of productions
    - V and Σ are disjoint (i.e. $V \cap \Sigma = \emptyset$)
    - $S \in V$, is the start symbol

    - * NOTE – often we'll define a grammar (like in our first example) just by listing the rules, as this completely defines the variables (on the left), terminals (all the other symbols on the right), start symbol (left side of first rule).

First Example Formalized

- Grammar G:
  - $A \rightarrow 0A1$
  - $A \rightarrow B$
  - $B \rightarrow \#$

- $G = (V, \Sigma, R, S)$
  - $V = \{A,B\}$
  - $\Sigma = \{0,1,\#\}$
  - R = rules above
  - $S = A$

- G could also be written as: $A \rightarrow 0A1 \mid \#$
  - | is used to indicate ‘or’
  - B is superfluous in this example; can just be directly substituted with #

Context-Free Grammars

- Substitution Rules / Productions
  - Of the form $A \rightarrow \beta$
    - A is a <u>variable</u>
    - β is a string, combining terminals and variables (can be empty)
    - To apply a rule, replace an occurrence of A with the string β.
  - We say that the grammar is context-free since this substitution can take place regardless of where A is (as part of a longer string).
    - Another explanation of context-free is that only a single variable is allowed on the left-hand side of any production rule.

Context-Free Grammars

- Substitution Rules / Productions:
  - We say that γ can be derived from α in one step if:
    - $A \rightarrow \beta$ is a rule
    - $\alpha = \alpha_1 A \alpha_2$
    - $\gamma = \alpha_1 \beta \alpha_2$
  - We write $\alpha \Rightarrow \gamma$
  - We write $\alpha \Rightarrow^* \gamma$ if γ can be derived from α in zero or more steps.

  - NOTE that we use $\rightarrow$ to refer to a rule, and we use $\Rightarrow$ to refer to a derivation involving application of the rule on a particular string

Context-Free Grammars

- The language generated by a grammar:
  - Let $G = (V, \Sigma, R, S)$

  - $L(G)$ is the language generated by G
    - $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w\}$
    - Any string (just terminals) obtained from a finite application of the rules of the grammar

- A language A is a Context-Free Language (CFL) if and only if there is a CFG G, such that:
    - $A = L(G)$

Example 2 – Palindromes

- Recursive definition for palindromes (pal) over Σ:

1. $\epsilon \in$ pal
2. For any $a \in \Sigma$, $a \in$ pal
3. For any $x \in$ pal and $a \in \Sigma$, $axa \in$ pal
4. No string is in pal unless it can be obtained by rules 1–3

Example 2 – Palindromes

- A CFG for palindromes over {a,b}
  - Define the rules R:
  - Base cases:

$P \rightarrow \epsilon$	(rule 1)
$P \rightarrow a$	(rule 2)
$P \rightarrow b$	(rule 3)

  - Recursion

$P \rightarrow aPa$	(rule 4)
$P \rightarrow bPb$	(rule 5)

Palindrome Example: abba

- Building the palindrome abba using grammar
- Recall that any string that can be formed by a finite application of the rules is a member of the language of the grammar

  - $P \Rightarrow aPa$ (rule 4)
  - $aPa \Rightarrow abPba$ (rule 5)
  - $abPba \Rightarrow ab\ \epsilon\ ba$ (rule 1)

  - $P \Rightarrow^{*} abba$

Examples 3-6

- Find a CFG to describe:
  - $L = \{a,b\}^*$
- Find a CFG to describe:
  - $L = \varnothing$
- Find a CFG to describe:
  - $L = \{w \in \{a,b\}^* \mid w$ has at least 2 b's$\}$
- Find a CFG to describe:
  - $L = \{w \in \{a,b\}^* \mid |w|$ is even$\}$

These are all regular languages. We'll show later that every regular language can be described with a context-free grammar

Example 7

- Find a CFG to describe:
  - $L = \{w \in \{0,1\}^* \mid n_0(w) = n_1(w)\}$
  - Basic idea (define recursively)
    - ϵ is certainly in the language
    - For all strings in the language, if we add a 0 and 1 to the string, the result is in the language.
      - Just consider all the different ways to add a 0 and 1
      - $S \rightarrow \epsilon \mid 01S \mid 10S \mid 0S1 \mid 1S0 \mid S01 \mid S10$
      - Clearly every string in L(G) will have equal 1's and 0's, but have we captured all such strings?
        - What about 00111100 ?

Example 7

- Find a CFG to describe:
  - $L = \{w \in \{0,1\}^* \mid n_0(w) = n_1(w)\}$
  - Basic idea (define recursively)
    - ϵ is certainly in the language
    - For all strings in the language, if we add a 0 to one end, and a 1 to the other end of the string, the result is in the language.
    - The concatenation of any two strings in the language will also be in the language

Example 7

- Find a CFG to describe:
  - $L = \{w \in \{0,1\}^* \mid n_0(w) = n_1(w)\}$

    - $S \to \epsilon$ (rule 1)
    - $S \to 0S1$ (rule 2)
    - $S \to 1S0$ (rule 3)
    - $S \to SS$ (rule 4)
  - Why does this work?
    - One work to think of it is to consider what $w \in L$ looks like and break it back down
      - If it's ϵ then we're ok
      - Or it has a 0 on one end and a 1 on the other, in which case it was built up by rule 2 or 3 from a smaller string
      - Or it has the same symbol on each end, in which case there must be some dividing point in the middle of the string where each portion of the string has equal 0's and 1's
        - Why? Because to start the string you see a 0 and now your counter is at one extra 0. By the time you get to the last symbol your counter is at -1 extra 0's. It must have crossed equal somewhere.

Example 7

- Let's derive a string from L
  - 00111100
  - $S \Rightarrow SS$ rule 4

    $\Rightarrow 0S1\ S$ rule 2

    $\Rightarrow 0S1\ 1S0$ rule 3

    $\Rightarrow 00S11\ \ 1S0$ rule 2

    $\Rightarrow 00S11\ \ 11S00$ rule 3

    $\Rightarrow 00\ \epsilon\ 11\ \ 11\ \epsilon\ 00$ rule 1

    $= 00111100$

Example 7

- Let's derive a string from L
  - 00110011
  - $S \Rightarrow SS$ rule 4

    $\Rightarrow 0S1\ 0S1$ rule 2

    $\Rightarrow 00S11\ \ 00S11$ rule 2

    $\Rightarrow 00\ \epsilon\ 11\ \ 00\ \epsilon\ 11$ rule 1

    $= 00110011$

  or
  - $S \Rightarrow 0S1$ rule 2

    $\Rightarrow 00S11$ rule 2

    $\Rightarrow 001S011$ rule 3

    $\Rightarrow 0011S0011$ rule 3

    $\Rightarrow 0011\epsilon0011$ rule 1

    $= 00110011$

Example 8

- Find a CFG to describe:
  - $L = \{a^i b^j c^k \mid i = k\}$
    - Number of a's equals the number of c's with any number of b's between them
    - Use variable B to represent b^j
    - Every time you add a to the left of B you need to add c to the right.

Example 8

- Find a CFG to describe:
  - $L = \{a^i b^j c^k \mid i = k\}$
    - $S \rightarrow B$ (rule 1)
    - $S \rightarrow aSc$ (rule 2)
    - $B \rightarrow bB$ (rule 3)
    - $B \rightarrow \epsilon$ (rule 4)
  - Can also write as
    - $S \rightarrow B \mid aSc$
    - $B \rightarrow bB \mid \epsilon$

Example 8

- Let's derive a string from L: aabbbcc
  - S ⇒ aSc rule 2

S ⇒ aSc	rule 2
⇒ aaScc	rule 2
⇒ aaBcc	rule 1
⇒ aabBcc	rule 3
⇒ aabbBcc	rule 3
⇒ aabbbBcc	rule 3
⇒ aabbb ϵ cc	rule 4
= aabbbcc	