Context Free Languages I

Grammars

Before We Start

- Any questions?

Plan for today

- Context Free Languages
  - Next class of languages in our quest!

Languages

- Recall.
  - What is a language?
  - What is a class of languages?

Regular Languages

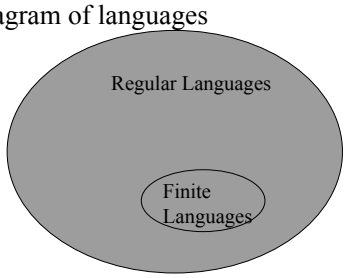
- For the past several weeks, we have been looking at Regular Languages:
  - Means of describing: Regular Expression
  - Machine for accepting: Finite Automata

Last class we discovered

- Venn-diagram of languages

Is there something out here?

YES!



Context Free Languages

- Context Free Languages(CFL) is the next class of languages outside of Regular Languages:
  - Means for defining: Context Free Grammar
  - Machine for accepting: Pushdown Automata

Plan for today

- Define and describe context free grammars
  - NOTE: Midterm will not cover CFLs.
- Questions?

Introduction

- Grammars
  - Think back to your days of learning English
  - Rules for constructing a valid sentence
    - Sentence = noun phrase + verb phrase
    - Noun phrase =
      - Name (Joe)
      - Article + noun (the car)
    - Verb Phrase =
      - Verb (runs)
      - Verb + prepositional phrase

Introduction

- Grammars
  - Rules for constructing a simple sentence
    - Sentence = noun phrase + verb phrase
    - Noun phrase =
      - Name (Joe)
      - Article + noun (the car)
    - Verb Phrase =
      - Verb (runs)
      - Verb + prepositional phrase
    - Prepositional Phrase =
      - Preposition + noun phrase (from the car)

Introduction

- Look at the sentence. Is this grammatically correct?
  - Joe runs from the car.
  - Sentence = noun phrase + verb phrase
  - = noun + verb phrase
  - = Name + verb phrase
  - = Joe + verb phrase
  - = Joe + verb + prepositional phrase
  - = Joe + verb + preposition + noun phrase
  - = Joe + verb + from + noun phrase
  - = Joe + verb + from + article +noun

Introduction

- Look at the sentence. Is this grammatically correct?
  - Joe runs from the car.
  - Sentence = Joe + verb + from + article +noun
  - = Joe + verb + from + the + car
  - = Joe + ran + from + the + car
- Valid sentence!

Introduction

- Observations
  - Our definition of a valid simple sentence includes:
    - A set of grammar categories (sentence, noun phrase, verb phrase, noun, verb, etc)
    - Rules for defining each category
    - Set of "final strings" (Joe, ran, from, the, car)
    - Initial category of what we wish to create (sentence)

Back to CS Theory

- CFLs are defined using Context Free Grammars(CFG)
  - Grammars, like their spoken language counterparts, provide a means to recursively "deriving" the strings in a language.

Back to CS Theory

- Recall our friend: Palindromes
  - A palindrome is a string that is the same read left to right or right to left
  - First half of a palindrome is a "mirror image" of the second half
  - Examples:
    - a, b, aba, abba, babbab.

Back to CS Theory

- Recursive definition for palindromes (pal) over Σ
  1. $\Lambda \in$ 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

Back to CS Theory

- A CFG for palindromes over {a,b}
  - Base cases
    1. $P \rightarrow \Lambda$
    2. $P \rightarrow a$
    3. $P \rightarrow b$
  - Recursion
    4. $P \rightarrow aPa$
    5. $P \rightarrow bPb$

Back to CS Theory

- Building the palindrome abba using grammar
  - $P \Rightarrow aPa$ (Rule 4)
  - $P \Rightarrow abPba$ (Rule 5)
  - $P \Rightarrow ab\Lambda ba$ (Rule 1)
  - $P \Rightarrow abba$

Back to CS Theory

- What we did:
  - At each step, we replaced the symbol P on the right with one of the definitions of P from our rule set.
  - Relating to our sentence example
    - Categories were replaced by Symbols
    - Final strings were replaced by Symbols
    - Start category was replaced by a symbol
    - We still have our set of rules.

Context Free Grammars

- Let's redefine grammars for CS Theory use:
  1. Terminals = Set of symbols that form the strings of the language being defined
  2. Variables = Set of symbols representing categories
  3. Start Symbol = variable that represents the "base category" that defines our language
  4. Production rules = set of rules that recursively define the language

Context Free Grammars

- Production Rules
  - Of the form $A \rightarrow B$
    - A is a variable
    - B is a string, combining terminals and variables
    - To apply a rule, replace an occurance of A with the string B.

Context Free Grammars

- Let's formalize this a bit:
  - A context free grammar (CFG) is a 4-tuple: (V, Σ, S, P) where
    - V is a set of variables
    - Σ is a set of terminals
    - V and Σ are disjoint (I.e. $V \cap \Sigma = \varnothing$)
    - $S \in V$, is your start symbol

Context Free Grammars

- Let's formalize this a bit:
  - Production rules
    - Of the form $A \rightarrow \beta$ where
      - $A \in V$
      - $\beta \in (V \cup \Sigma)^*$ string with symbols from V and Σ
    - We say that γ can be derived from α in one step:
      - $A \rightarrow \beta$ is a rule
      - $\alpha = \alpha_1 A \alpha_2$
      - $\gamma = \alpha_1 \beta \alpha_2$
      - $\alpha \Rightarrow \gamma$

Context Free Grammars

- Let's formalize this a bit:
  - Production rules
    - We say that the grammar is context-free since this substitution can take place regardless of where A is.
    - We write $\alpha \Rightarrow^* \gamma$ if γ can be derived from α in zero or more steps.

Context Free Grammars

- The language generated by a CFG
  - Let G = (V, Σ, S, P)
  - The language generated by G, L(G)
    - $L(G) = \{ x \in \Sigma^* \mid S \Rightarrow^* x\}$
  - A language L is a Context Free Language (CFL) iff there is a CFG G, such that
    - L = L(G)

Example

- Find a CFG to describe:
  - $L = \{x \in \{0,1\}^* \mid n_0(x) = n_1(x)\}$
  - Basic idea (define recursively)
    - Λ 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.
    - The concatenation of any two strings in the language will also be in the language

Example

- Find a CFG to describe:
  - $L = \{x \in \{0,1\}^* \mid n_0(x) = n_1(x)\}$
    - $S \rightarrow \Lambda$ (1)
    - $S \rightarrow 0S1$ (2)
    - $S \rightarrow 1S0$ (3)
    - $S \rightarrow SS$ (4)

Example

- Let's derive a string from L
  - 00110110
  - $S \Rightarrow SS$ rule 4
    $\Rightarrow 0S1\ 0S1$ rule 2
    $\Rightarrow 00S11\ \ 00S11$ rule 2
    $\Rightarrow 00\Lambda11\ \ 00\ \Lambda11$ rule 1
    $= 00110011$

Another example

- 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.

Another example

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

Another example

- Let's derive a string from L: aabbbcc
  - $S \Rightarrow aSc$ rule 2
    $S \Rightarrow aaScc$ rule 2
    $S \Rightarrow aaBcc$ rule 1
    $S \Rightarrow aabBcc$ rule 3
    $S \Rightarrow aabbBcc$ rule 3
    $S \Rightarrow aabbbBcc$ rule 3
    $S \Rightarrow aabbb\Lambda cc$ rule 4
    $= aabbb\Lambda cc$

One more example

- Defining the grammar for algebraic expressions:
  - Let a = a numeric constant
  - Set of binary operators = {+, -, *, /}
  - Expressions can be parenthesized

One more example

- Defining the grammar for algebraic expressions:
  - G = (V, Σ, S, P)
  - V = {S}
  - Σ = { a, -, +, *, /, (,) }
  - S = S
  - P = see next slide

One more example

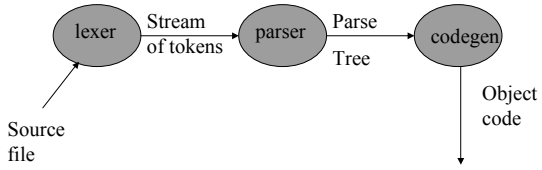
- Defining the grammar for algebraic expressions – Production rules
  - $S \rightarrow S + S$ (1)
    $S \rightarrow S - S$ (2)
    $S \rightarrow S * S$ (3)
    $S \rightarrow S / S$ (4)
    $S \rightarrow (S)$ (5)
    $S \rightarrow a$ (6)

One more example

- Show derivation for a + (a * a) / a
  - $S \Rightarrow S + S$ rule 1
    $S \Rightarrow a + S$ rule 6
    $S \Rightarrow a + S / S$ rule 4
    $S \Rightarrow a + (S) / S$ rule 5
    $S \Rightarrow a + (S * S) / S$ rule 3
    $S \Rightarrow a + (a * S) / S$ rule 6
    $S \Rightarrow a + (a * a) / S$ rule 6
    $S \Rightarrow a + (a * a) / a$ rule 6

Practical uses for grammars

- How a compiler works



A real practical example

- Grammars for programming languages
  - <stmt> → … | <for-stmt> | <if-stmt> | …
  - <stmt> → { <stmt> <stmt> } | Λ
  - <if-stmt> → **if (** <expr> **) then** <stmt>
  - <for-stmt> → **for (** <expr>**;** <expr>**;** <expr>**)** <stmt>

A real practical example

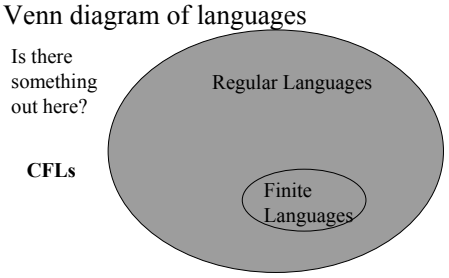
- Grammars for programming languages
  - Keywords and punctuation are terminals
  - Program constructs are variables
  - Production rules define the syntax of the language
  - This is really the second step in building a compiler!

Context Free Grammars

- Questions?

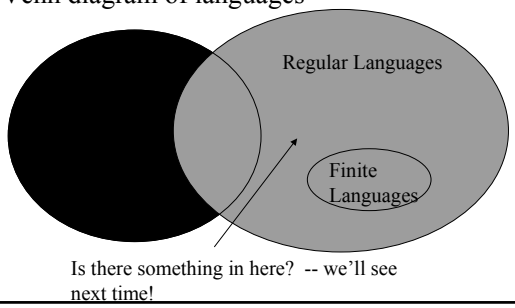
CFLs and Regular Languages

- Venn diagram of languages



CFLs and Regular Languages

- Venn diagram of languages



Summary

- Next class of languages Context Free Languages.
  - Described by Context Free Grammars
    - G = (V, Σ, S, P)
- Questions?