Theory of Automata

Context Free Languages and Grammars

Dr. Sabina Akhtar

Revision

- Regular Languages
  - Finite Automata
  - Regular Expressions
  - Example:
    - $0^n \mid n>=0$
- Non-Regular Languages

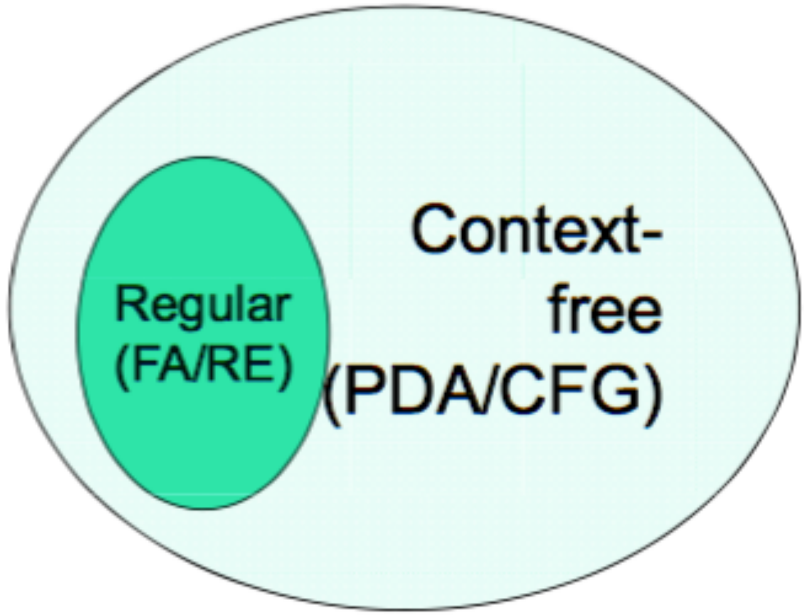
Not all languages are regular

- So what happens to the languages which are not regular?
- E.g., balanced paranthesis problem
- (5*(7+9))

- Can we still come up with a language recognizer?
  - i.e., something that will accept (or reject) strings that belong (or do not belong) to the language?

Context-Free Languages

- A language class larger than the class of regular languages
- Supports natural, recursive notation called "context- free grammar"
- Applications:
  - Parse trees, compilers
  - XML



Informal Comments

- A *context-free grammar* is a notation for describing languages.
- It is more powerful than finite automata or RE's, but still cannot define all possible languages.
- Useful for nested structures, e.g., parentheses in programming languages.

Informal Comments – (2)

- Basic idea is to use “variables” to stand for sets of strings (i.e., languages).
- These variables are defined recursively, in terms of one another.
- Recursive rules (“productions”) involve only concatenation.
- Alternative rules for a variable allow union.

Definition: CFG

- A context-free grammar G=(V,T,P,S), where:
  - V: set of variables or non-terminals
  - T: set of terminals (= alphabet U {ε})
  - P: set of *productions*, each of which is of the form
    V ==> α_1 | α_2 | …
    - Where each α_i is an arbitrary string of variables and terminals
  - S ==> start variable

Examples

- $L_1 = \{ 0^n \mid n \geq 0\}$
- $L_2 = \{w \mid w$ is of the form $0^n 1^n$, for all $n \geq 1\}$

- CFG for L_1
- $S \rightarrow \varepsilon$
- $S \rightarrow 0S$

- W=000

Example: CFG for $\{ 0^n1^n \mid n \geq 1\}$

- Basis: 01 is in the language.
- Induction: if w is in the language, then so is 0w1.

- Productions:
  - S -> 01
  - S -> 0S1

Structure of a production

head	derivation	body
A	=======>	$\alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_k$

The above is same as:

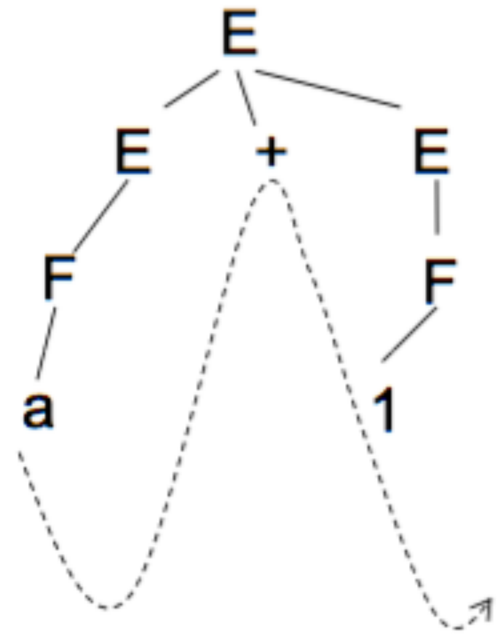
1. $A ==> \alpha_1$
2. $A ==> \alpha_2$
3. $A ==> \alpha_3$

...

K. $A ==> \alpha_k$

Parse Tree

Draw parse tree for
a1*(1+b0)



Parse tree for *a + 1*

Start Variable: E

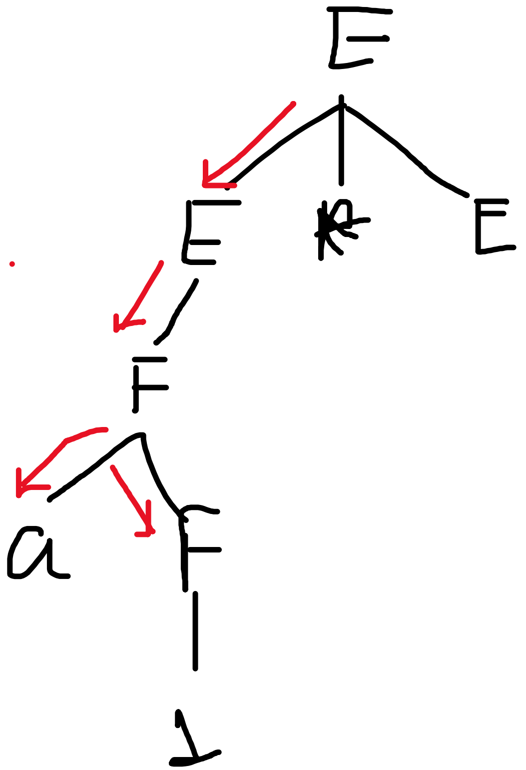
V = ?

T = ?

G:
E => E+E | E*E | (E) | F
F => aF | bF | 0F | 1F | 0 | 1 | a | b

Parse Tree

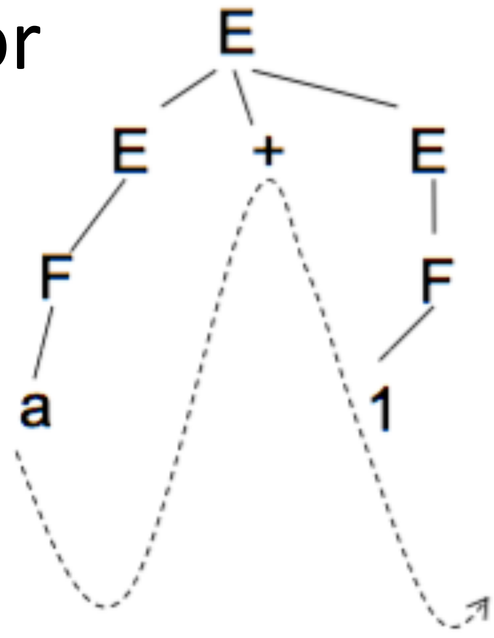
Draw parse tree for a1*(1+b0)



Start Variable: E

V = ?

T = ?



Parse tree for *a + 1*

G:

E => E+E | E*E | (E) | F

F => aF | bF | 0F | 1F | 0 | 1 | a | b

An Example

- A palindrome is a word that reads identical from both ends
  - E.g., madam, redivider, malayalam, 010010010
- Let L = { w | w is a binary palindrome}
- Is L regular?

An Example

- A palindrome is a word that reads identical from both ends
  - E.g., madam, redivider, malayalam, 010010010
- Let L = { w | w is a binary palindrome}
- Is L regular?
  - No.
  - <u>Proof:</u>
    - Let $w=0^N10^N$ (assuming N to be the p/l constant)
    - By Pumping lemma, w can be rewritten as xyz, such that xy^kz is also L (for any $k\geq 0$)
    - But $|xy|\leq N$ and $y\neq\varepsilon$
    - ==> $y=0^+$
    - ==> xy^kz *will NOT* be in L for k=0
    - ==> Contradiction

But the language of palindromes…

<u>is a CFL</u>, because it supports recursive substitution (in the form of a CFG)

- This is because we can construct a "*grammar*" like this:

Productions

1. $A \Longrightarrow \varepsilon$
2. $A \Longrightarrow 0$
3. $A \Longrightarrow 1$
4. $A \Longrightarrow 0A0$
5. $A \Longrightarrow 1A1$

Terminal

Variable or non-terminal

Same as:
$A \Rightarrow 0A0 \mid 1A1 \mid 0 \mid 1 \mid \varepsilon$

How does this grammar work?

Class Activity

- Draw the parse tree for 0110 using
- G:
  A -> 0A0|1A1| 0|1|Ɛ

How does the CFG for palindromes work?

An input string belongs to the language (i.e., accepted) iff it can be generated by the CFG

G:
A => 0A0 | 1A1 | 0 | 1 | ε

- Example: w=01110
- G can generate w as follows:

1. A => 0A0
2. => 01A10
3. => 01110

Generating a string from a grammar:

1. Pick and choose a sequence of productions that would allow us to generate the string.
2. At every step, substitute one variable with one of its productions.

Definition CFG

- A context-free grammar G=(V,T,P,S), where:
  - V: set of variables or non-terminals
  - T: set of terminals (= alphabet U {ε})
  - P: set of *productions*, each of which is of the form
    $V \Longrightarrow \alpha_1 \mid \alpha_2 \mid \ldots$
    - Where each α_i is an arbitrary string of variables and terminals
  - S ==> start variable

CFG for the language of binary palindromes:
G=({A},{0,1},P,A)
P: A ==> 0 A 0 | 1 A 1 | 0 | 1 | ε

CFG conventions

- A, B, C,… are variables.
- a, b, c,… are terminals.
- …, X, Y, Z are either terminals or variables.
- …, w, x, y, z are strings of terminals only.
- α, β, γ,… are strings of terminals and/or variables.

Example: Formal CFG

- Here is a formal CFG for $\{ 0^n 1^n \mid n \geq 1\}$.
- Terminals = {0, 1}.
- Variables = {S}.
- Start symbol = S.
- Productions =

  S -> 01

  S -> 0S1

Class Activity: Example #2

- Provide formal CFG for
  - A grammar for $L = \{0^m 1^n \mid m \geq n\}$

Class Activity: Example #2

- Provide formal CFG for
  - A grammar for $L = \{0^m 1^n \mid m \geq n\}$
  - CFG?

G:
$S \Rightarrow 0S1 \mid A$
$A \Rightarrow 0A \mid \varepsilon$

How would you interpret the string "00000111" using this grammar?

Class Activity: Example #3

- Provide formal CFG for
  - Language of balanced paranthesis
    e.g., ()((((()))))((())))....

Class Activity: Example #3

- Provide formal CFG for
  - Language of balanced paranthesis
    e.g., ()(((())))((()))....
  - CFG?

<u>G:</u>
S => (S) | SS | ε

How would you “interpret” the string “(((()))()())” using this grammar?

Exercise

[A]**2.3** Answer each part for the following context-free grammar G.

$$
\begin{aligned}
R &\to XRX \mid S \\
S &\to \mathtt{a}T\mathtt{b} \mid \mathtt{b}T\mathtt{a} \\
T &\to XTX \mid X \mid \varepsilon \\
X &\to \mathtt{a} \mid \mathtt{b}
\end{aligned}
$$

a. What are the variables of G?

b. What are the terminals of G?

c. Which is the start variable of G?

d. Give three strings in $L(G)$.

e. Give three strings *not* in $L(G)$.

f. True or False: $T \Rightarrow \mathtt{aba}$.

g. True or False: $T \overset{*}{\Rightarrow} \mathtt{aba}$.

h. True or False: $T \Rightarrow T$.

i. True or False: $T \overset{*}{\Rightarrow} T$.

j. True or False: $XXX \overset{*}{\Rightarrow} \mathtt{aba}$.

k. True or False: $X \overset{*}{\Rightarrow} \mathtt{aba}$.

l. True or False: $T \overset{*}{\Rightarrow} XX$.

m. True or False: $T \overset{*}{\Rightarrow} XXX$.

n. True or False: $S \overset{*}{\Rightarrow} \varepsilon$.

o. Give a description in English of $L(G)$.

Class Activity

- Provide CFG for accepting simple expressions like
  - a+b, b*b, …
  - 1+2,….

Example #4

A program containing **if-then(-else)** statements

if *Condition* **then** *Statement* **else** *Statement*

(Or)

if *Condition* **then** *Statement*

CFG?

More examples

- Parenthesis matching in code
- Syntax checking

- In scenarios where there is a general need for:
  - Matching a symbol with another symbol, or
  - Matching a count of one symbol with that of another symbol, or
  - Recursively substituting one symbol with a string of other symbols

Applications of CFLs & CFGs

- Compilers use parsers for syntactic checking
- Parsers can be expressed as CFGs
  1. Balancing paranthesis:
     - B ==> BB | (B) | *Statement*
     - *Statement* ==>
  2. If-then-else:
     - S ==> SS | *if Condition then Statement else Statement* | *if Condition then Statement* | *Statement*
     - *Condition* ==>
     - *Statement* ==>
  3. C paranthesis matching { … }
  4. Pascal *begin-end* matching
  5. YACC (Yet Another Compiler-Compiler)

Class Activity

- Design grammar that accepts addition and subtraction operations on all the numbers using plain and extended BNF format.
- Example:
  - 1 + 2,
  - 2 – 3 + 5,….

Exercise

2.4 Give context-free grammars that generate the following languages. In all parts, the alphabet Σ is $\{0,1\}$.

A**a.** $\{w \mid w \text{ contains at least three } 1\text{s}\}$

b. $\{w \mid w \text{ starts and ends with the same symbol}\}$

c. $\{w \mid \text{the length of } w \text{ is odd}\}$

A**d.** $\{w \mid \text{the length of } w \text{ is odd and its middle symbol is a } 0\}$

e. $\{w \mid w = w^{\mathcal{R}}, \text{ that is, } w \text{ is a palindrome}\}$

f. The empty set

References

- Book Chapter
- Lectures from Stanford University
  - http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE%20NOTES
- Lectures from Washington State University
  - http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/