San José State University
Department of Computer Science

Ahmad Yazdankhah
ahmad.yazdankhah@sjsu.edu
www.cs.sjsu.edu/~yazdankhah

Grammars

(Part 3)

Lecture 22

Day 26/31

CS 154

Formal Languages and Computability

Spring 2018

Agenda of Day 26

- Solution and Feedback of Quiz 8
- Summary of Lecture 21
- Lecture 22: Teaching ...
  - Grammars (Part 3)

Solution and Feedback of Quiz 8 (Out of 30)

Metrics	Section 1	Section 2	Section 3
Average	25	??	24
High Score	29	22	28
Low Score	19	10	16

Summary of Lecture 21: We learned ...

Grammars

- Formal definition of grammar:

$$G = (V, T, S, P)$$

- Two grammars are equivalent iff ...
  - ... both generate the same language.
- Every grammar produces a language.
- Does every language have a grammar?
  - We don't know yet!

Types of Grammars

- A grammar G is linear if ...
  - ... the right hand side of every production rule has at most one variable.

- Right-linear grammar is ...
  - ... a linear grammar whose production rules are of the form:

  $A \rightarrow xB \mid x$ where $A, B \in V$ and $x \in T^*$

- Left-linear grammar is ...
  - ... a linear grammar whose production rules are of the form:

  $A \rightarrow Bx \mid x$ where $A, B \in V$ and $x \in T^*$

- A grammar is said to be regular if ...
  - ... it is either right-linear or left-linear.

Any Question

Summary of Lecture 21: We learned ...

Theorems

- Regular grammar produces regular language.
- Regular languages have regular grammars.

Context-Free Grammars (CFG)

- A context-free grammar is ...
  - ... a grammar whose production rules are of the form:

  $A \rightarrow v$ where $A \in V$ and $v \in (V \cup T)^*$

Any Question

Types of Grammars (cont'd)

Context-Free Grammars (CFG)

Example 21

- Is the following grammar context-free?

  $S \rightarrow S\,S \mid a\,S\,b \mid b\,S\,a \mid \lambda$

- What language does it produce?

- What would happen if:

  a = (

  b =)

Context-Free Languages (CFL)

Definition

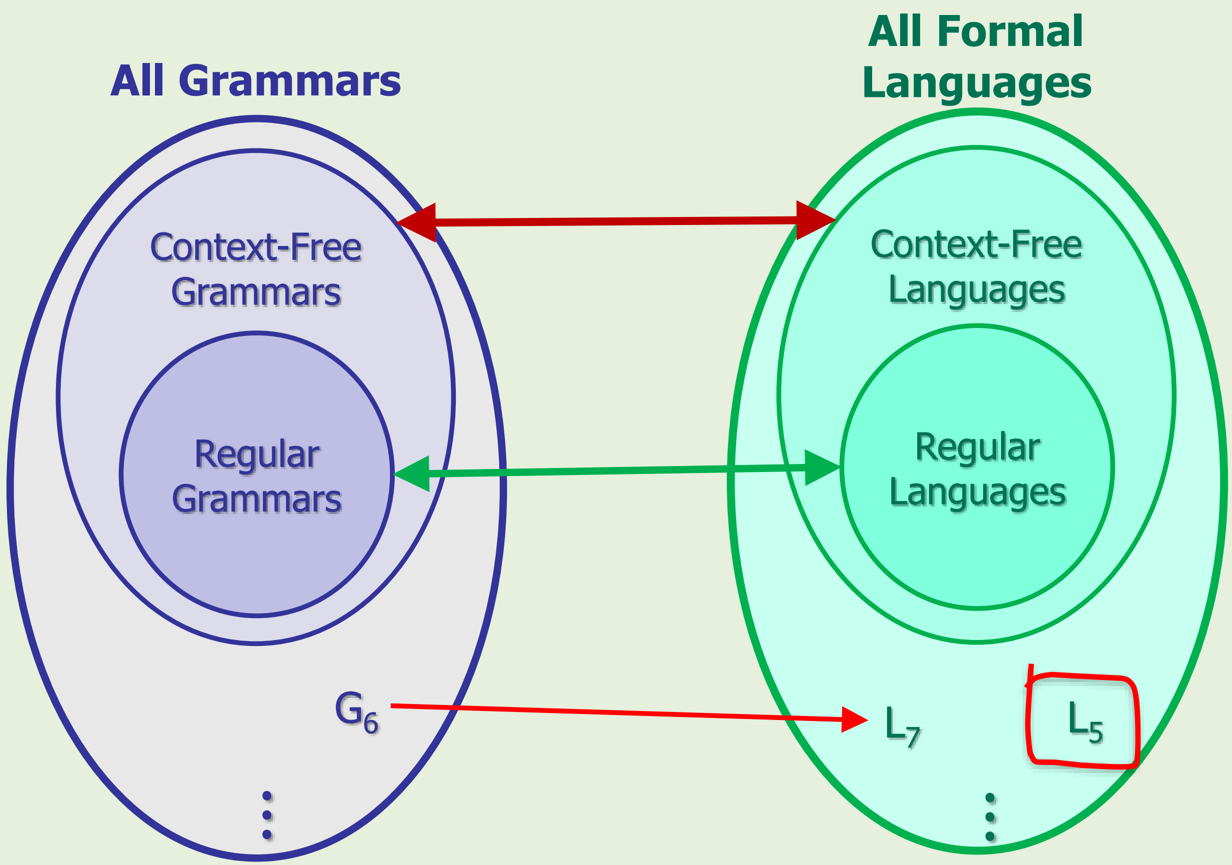
- A language L_1 is said to be context-free iff there is a context-free grammar G such that $L_1 = L(G)$.
  - In other words, CFGs generates CFLs.

- Therefore, all of the following languages are context-free:
- $L = \{a^n b^n : n \geq 0\}$
- $L = \{ww^R : w \in \Sigma^* \}$
- $L = \{w : n_a(w) = n_b(w), w \in \{a, b\}^* \}$

- Note that a regular grammar is a CFG but NOT vice-versa!

Grammars and Languages Association

Revisited



Simple Grammars (S-Grammars)

Definition

- A context-free grammar G is said to be simple grammar (aka s-grammar) if the following two conditions are satisfied:

Condition #1

All production rules are of the form:

$A \to av$ Where $A \in V$, $a \in T$, $v \in V^*$

Means: One terminal as prefix and any number of variables as suffix.

Condition #2

Any pair (A, a) occurs only once in all production rules.

Simple Grammar (S-Grammar)

Example 22

- Is the following grammar s-grammar?
  $S \rightarrow aS \mid bSS \mid c$

Solution

Simple Grammar (S-Grammar)

Example 23

- Is the following grammar s-grammar?
  $S \rightarrow bSS \mid aS \mid c \mid aSS$

Solution

Derivations Techniques

Derivations Techniques

- Consider a production rule that has two or more variables.

  $S \rightarrow a\ A\ B$

  $A \rightarrow \ldots$

  $B \rightarrow \ldots$

- To derive a string, we should substitute A and B with some other production rules.
- But in what order?
  - We can substitute them randomly.
  - Or we can pick a specific order. (e.g. left var first or right var first ...)

Derivations Techniques

Example 24

- Derive string "aab" from the following context-free grammar:

  $S \rightarrow AB$
  $A \rightarrow aaA \mid \lambda$
  $B \rightarrow Bb \mid \lambda$

- **Approach 1**: Substitute the leftmost variables first

  $$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

- **Approach 2**: Substitute the rightmost variables first

  $$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{5}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

- Both derivations yielded the same results.

Leftmost / Rightmost Derivations

Definition

- A derivation is said to be leftmost if in each step the leftmost variable in the sentential form is substituted.

Definition

- A derivation is said to be rightmost if in each step the rightmost variable in the sentential form is substituted.

- The default method would be "leftmost" if we don't mention specifically.

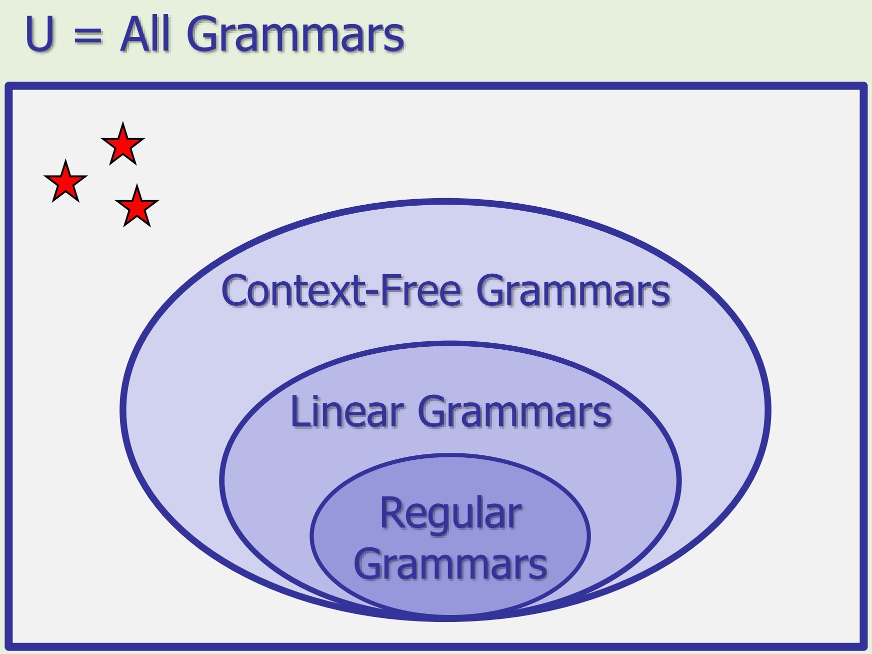
Homework

- Derive string "abbbb" from the following grammar:
  1. $S \rightarrow aAB$
  2. $A \rightarrow bBb$
  3. $B \rightarrow A \mid \lambda$

- Leftmost derivation:

- Rightmost derivation:

Grammars Hierarchy



- Note that there are still some grammars that are not CFG.
- They are called: "context-sensitive", and "recursively enumerable".
- We won't cover those in this course but as an FYI, I'd like to mention them in the next slide.

Chomsky's Hierarchy

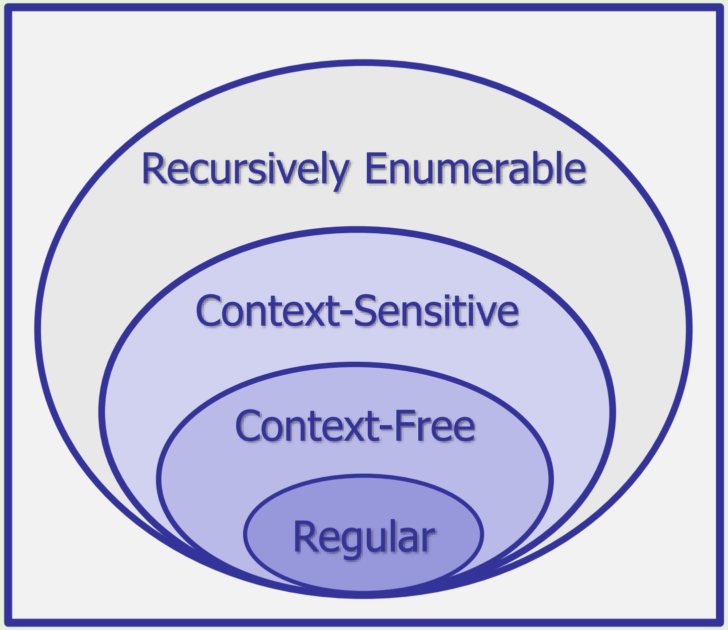
FYI

- Avram Noam Chomsky, the American linguist, philosopher, and historian (1928 - ?), has categorized formal languages that is called "Chomsky's Hierarchy".

- He categorized formal languages into 4 types as:

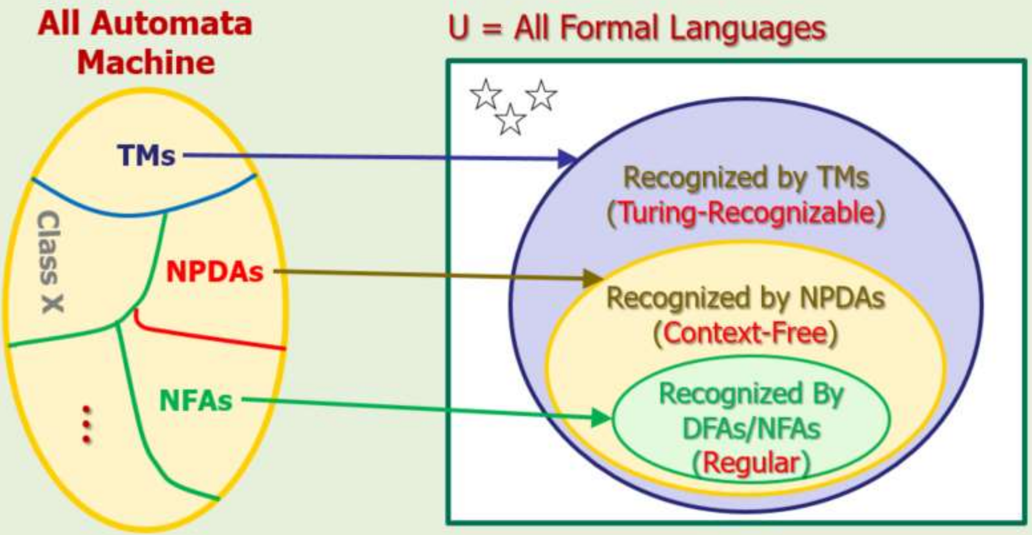
- Type 0: Recursively-enumerable
- Type 1: Context-sensitive
- Type 2: Context-free
- Type 3: Regular

U = All Grammars



Machines and Languages Association

Revisited



Notes

- Turing-recognizable is also known as recursively enumerable.
- We mentioned context-free languages before and now we defined them.
- We learned before that $\{ww : w \in \Sigma^*\}$ cannot be recognized by NPDAs.
- This language is one example of non-CFL.

Parsing

Introduction

- Parsing is a very important topic in computer science.
- There are many theorems, efficient algorithms, and a lot of researches about it.

- In this lecture, we give you only a big picture about it.
- So, consider this as a very short introduction about parsing.

- For more information, you need to take "Compiler Course".

Motivation

- Assume you have the following statement in your Java program:

```
if (x > 5) {
  y = y * 2 + 1;
}
```

- How does Java compiler know that this is a valid statement?
  - valid = well-formed

- To answer this question, let's remove all whitespaces:

```
if(x>5){y=y*2+1;}
```

- This is just a string like other strings that we have seen so far.
- So, this string is well-formed if we can derive it from a grammar.

A Simplified Grammar for If-Statement

Example 25

Construct a grammar to produce if-statements like:

```
if(Condition){Statement}
```

Simplified Requirements

1. Condition: only one condition containing '>' or '<' or '=' symbols
   - e.g. "x<5", "5<x", "x<y", "y>2", "x=3", …
2. Statements: only one Java assignment-statement
   - e.g. "**x=y*2+3;**", "**y=1+x;**", …
3. Identifiers: only x or y.
4. Operators: * , +

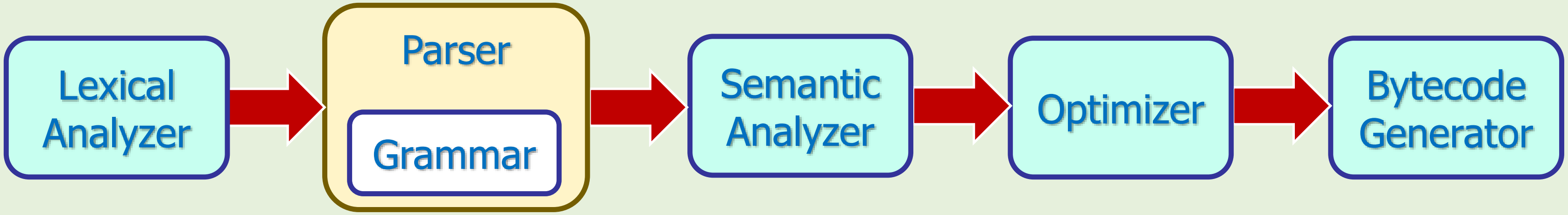
Note
The provided grammar is just for getting some idea.
It is neither efficient nor practical!

Solution

Java Compiler (From Compiler Course!)

FYI

1. **Lexical Analyzer** (aka Lexer or scanner): breaks the entire code up into words (tokens)
2. **Parser**: by using the grammar, generates the parse-tree, checks the syntax of the sentences
3. **Semantic Analyzer**: checks the sentences meaning
4. **Optimizer**: optimizes the sentences to be more efficient
5. **Code Generator**: produces the bytecode



Parse Trees

Parse Trees

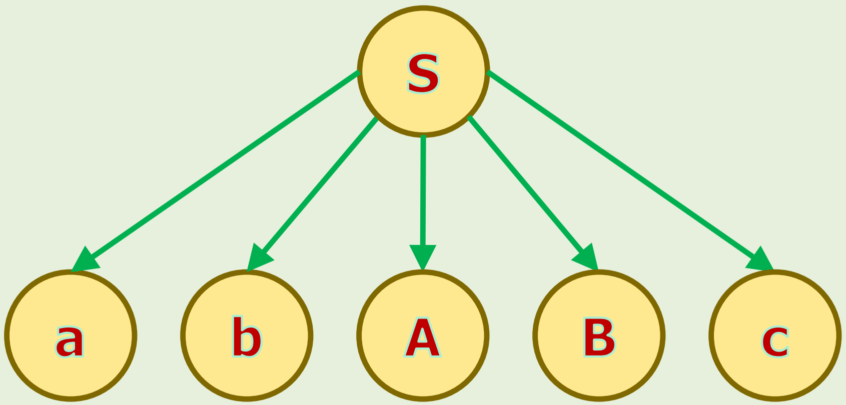
- Let's explain it through some examples.
- The first example shows how to construct a parse-tree for a production-rule.

Example 26

- Construct a parse-tree for the following production rule.

  $S \rightarrow abABc$

- Note that the order of children matters.



Parse Trees

Example 27

- Given the following grammar:
  1. $S \rightarrow AB$
  2. $A \rightarrow aaA \mid \lambda$
  3. $B \rightarrow Bb \mid \lambda$
- Construct a parse-tree for the string aab.

Homework

- Given the following grammar:
  1. $S \rightarrow aAB$
  2. $A \rightarrow bBb$
  3. $B \rightarrow A \mid \lambda$

- Construct a parse-tree for the following strings:
  a. $w = abbb$
  b. $w = abbbb$
  c. $w = abbbbb$

References

1. Linz, Peter, "An Introduction to Formal Languages and Automata, 5th ed.," Jones & Bartlett Learning, LLC, Canada, 2012
2. Michael Sipser, "Introduction to the Theory of Computation, 3rd ed.," CENGAGE Learning, United States, 2013
   ISBN-13: 978-1133187790
3. The ELLCC Embedded Compiler Collection, available at: http://ellcc.org/