CS 4410

Automata, Computability, and Formal Language

Dr. Xuejun Liang

Spring 2021

Chapter 11: A Hierarchy of Formal Languages and Automata

1. Recursive and Recursively Enumerable Languages
    - Languages That Are Not Recursively Enumerable
    - A Language That Is Not Recursively Enumerable
    - A Language That Is Recursively Enumerable But Not Recursive
2. Unrestricted Grammars
3. Context-Sensitive Grammars and Languages
    - Context-Sensitive Languages and Linear Bounded Automata
    - Relation Between Recursive and Context-Sensitive Languages
4. The Chomsky Hierarchy

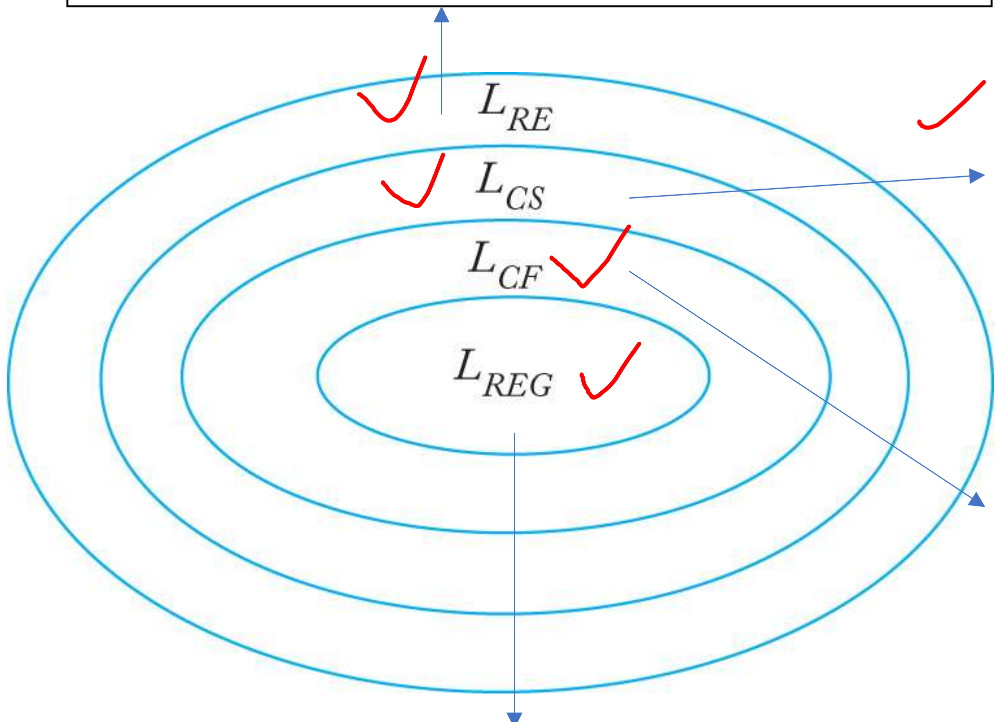
Learning Objectives

At the conclusion of the chapter, the student will be able to:

- Explain the difference between recursive and recursively enumerable languages
- Describe the type of productions in an unrestricted grammar
- Identify the types of languages generated by unrestricted grammars
- Describe the type of productions in a context sensitive grammar
- Give a sequence of derivations to generate a string using the productions in a context sensitive grammar
- Identify the types of languages generated by context-sensitive grammars
- Construct a context-sensitive grammar to generate a particular language
- Describe the structure and components of the Chomsky hierarchy

A Hierarchy of Formal Languages and Automata

Recursively enumerable language
Unrestricted Grammars, Turing Machine



Context-Sensitive Languages
Context-Sensitive grammars
Linear Bounded Automata

Context-free language
Context-free grammar, NDPA

Regular language, Regular grammar,
Regular expression, NFA, DFA

Recursive and Recursively Enumerable Languages

- A language L is *recursively enumerable* if there exists a Turing machine that accepts it (as we have previously stated, rejected strings cause the machine to either not halt or halt in a nonfinal state)
- A language L is *recursive* if there exists a Turing machine that accepts it and is guaranteed to halt on every valid input string
- In other words, a language is recursive if and only if there exists a membership algorithm for it

Languages That Are Not Recursively Enumerable

- Theorem 11.2 states that, for any nonempty alphabet, there exist languages not recursively enumerable
- One proof involves a technique called diagonalization, which can be used to show that, in a sense, there are fewer Turing Machines than there are languages
- More explicitly, Theorem 11.3 describes the existence of a recursively enumerable language whose complement is not recursively enumerable
- Furthermore, Theorem 11.5 concludes that the family of recursive languages is a proper subset of the family of recursively enumerable languages

Theorem 11.1: Let S be an infinite countable set. Then its power set 2^S is not countable

Let S = {s1, s2, s3, ...}. Then any element of 2^S can be represented by a sequence of 0's and 1's. For examples:

1 2 3 4 5 6 7 8 9

the set {s2, s3, s6} = 0 1 1 0 0 1 0 0 0 ...

the set {s1, s3, s5} = 1 0 1 0 1 0 0 0 0 0 ...

Now, suppose that 2^S were countable and 2^S ={t1, t2, t3, …}

Pick t = 0011…

Then $t \notin 2^S$

A contradiction!

So, 2^S is not countable

Diagonalization

t_1	1	0	0	0	0	. . .
t_2	1	1	0	0	0	. . .
t_3	1	1	0	1	0	. . .
t_4	1	1	0	0	1	. . .
⋮					⋱	

Unrestricted Grammars

- An *unrestricted grammar* has essentially no restrictions on the form of its productions:
  - Any variables and terminals on the left side, in any order
  - Any variables and terminals on the right side, in any order
  - The only restriction is that λ is not allowed as the left side of a production
- A sample unrestricted grammar has productions

$$
\begin{aligned}
S &\rightarrow S_1B \\
S_1 &\rightarrow aS_1b \\
bB &\rightarrow bbbB \\
aS_1b &\rightarrow aa \\
B &\rightarrow \lambda
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow S_1B \Rightarrow aS_1bB \\
&\stackrel{*}{\Rightarrow} a^n S_1 b^n B \\
&\Rightarrow a^{n-1} aa b^{n-1} B \\
&\Rightarrow a^{n+1} b^{n-1} bbB
\end{aligned}
$$

Unrestricted Grammars and Recursively Enumerable Languages

- Theorem 11.6: Any language generated by an unrestricted grammar is recursively enumerable
- Theorem 11.7: For every recursively enumerable language L, there exists an unrestricted grammar G that generates L
- These two theorems establish the result that unrestricted grammars generate exactly the family of recursively enumerable languages, the largest family of languages that can be generated or recognized algorithmically

Context-Sensitive Grammars

- In a context-sensitive grammar, the only restriction is that, for any production, length of the right side is at least as large as the length of the left side
- Example 11.2 introduces a sample context-sensitive grammar with productions

$S \rightarrow abc \mid aAbc$

$Ab \rightarrow bA$

$Ac \rightarrow Bbcc$

$bB \rightarrow Bb$

$aB \rightarrow aa \mid aaA$

Derive the string aabbcc

$S \Rightarrow aAbc$

$\Rightarrow abAc$

$\Rightarrow abBbcc$

$\Rightarrow aBbbcc$

$\Rightarrow aabbcc$

Characteristics of Context-Sensitive Grammars

- An important characteristic of context-sensitive grammars is that they are **noncontracting**, in the sense that in any derivation, the length of successive sentential forms can never decrease
- These grammars are called context-sensitive because it is possible to specify that variables may only be replaced in certain contexts
- For instance, in the grammar of Example 11.2, variable A can only be replaced if it is followed by either b or c

$$Ab \rightarrow bA$$
$$Ac \rightarrow Bbcc$$

Context-Sensitive Languages

- A language L is context-sensitive if there is a context-sensitive grammar G, such that either L = L(G) or L = L(G) $\cup \{ \lambda \}$
- The empty string is included, because by definition, a context-sensitive grammar can never generate a language containing the empty string
- As a result, it can be concluded that the family of context-free languages is a subset of the family of context-sensitive languages
- The language $\{ a^n b^n c^n : n \geq 1 \}$ is context-sensitive, since it is generated by the grammar in Example 11.2

aabbcc / aaabbbccc

Context-Sensitive Languages and Linear Bounded Automata

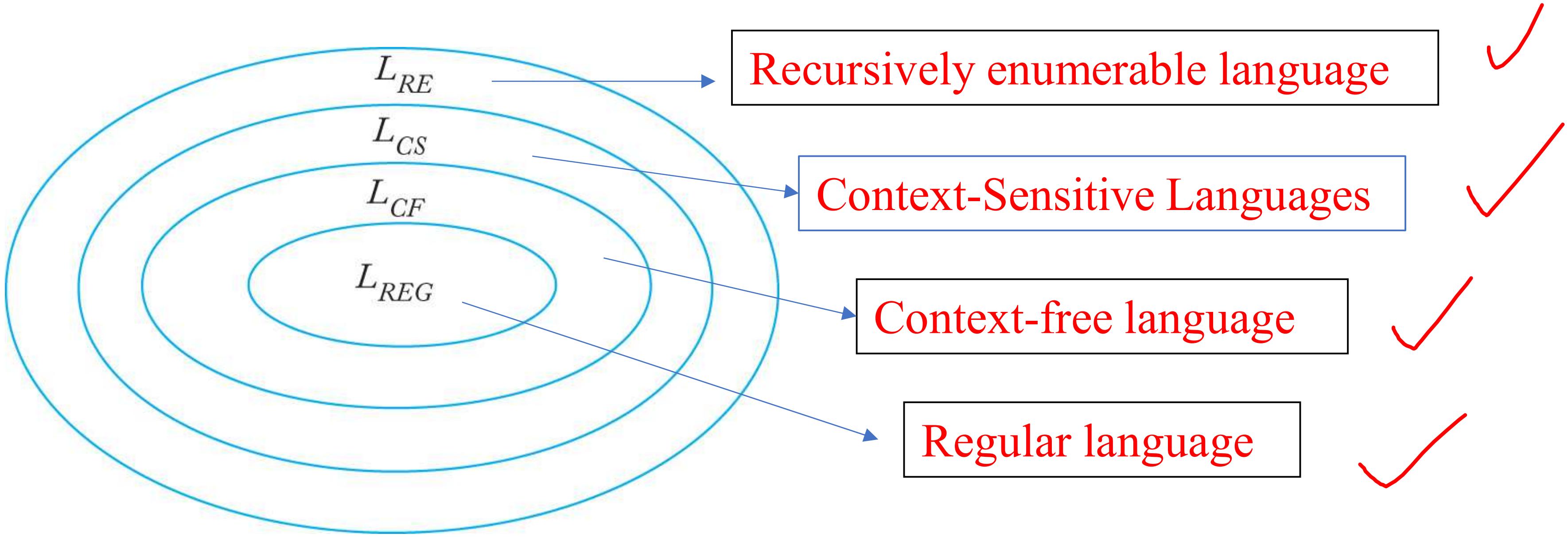
- Theorem 11.8 states that, for every context-sensitive language L not including λ, there is a linear bounded automaton that recognizes L
- Theorem 11.9 states that, if a language L is accepted by a linear bounded automaton M, then there is a context-sensitive grammar that generates L
- These two theorems establish the result that context-sensitive grammars generate exactly the family of languages accepted by linear bounded automata, the context-sensitive languages

Relationship Between Recursive and Context-Sensitive Languages

- Theorem 11.10 states that every context-sensitive language is recursive
- Theorem 11.11 maintains that some recursive languages are not context-sensitive
- These two theorems help establish a hierarchical relationship among the various classes of automata and languages:
  - Linear bounded automata are less powerful than Turing machines
  - Linear bounded automata are more powerful than pushdown automata

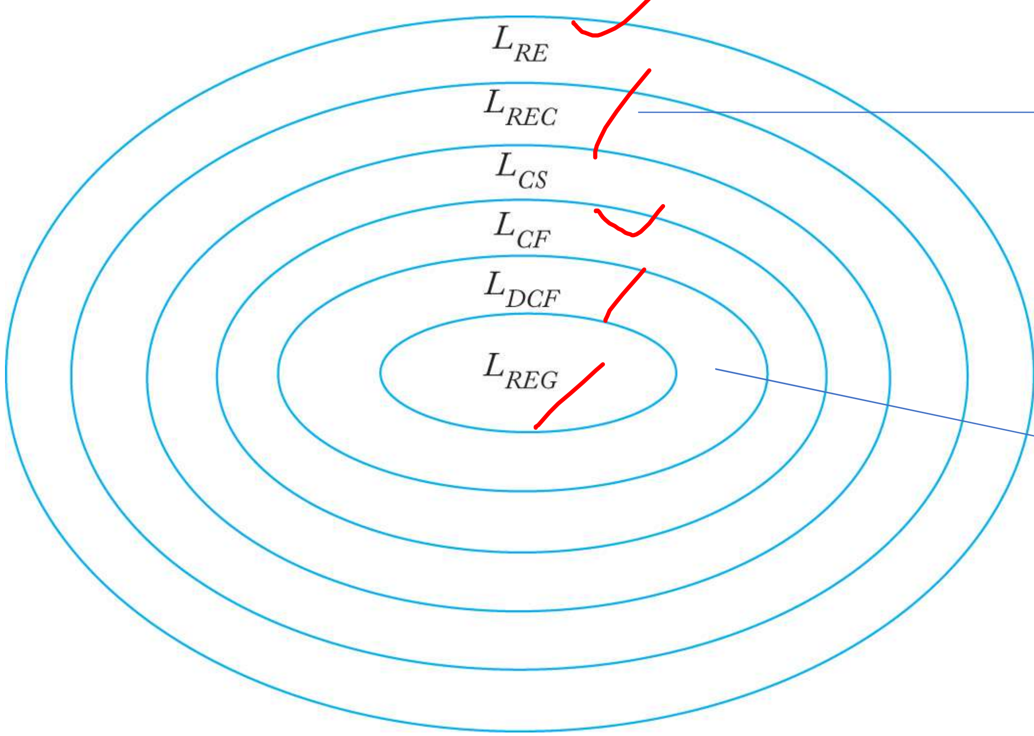
The Chomsky Hierarchy

- The linguist Noam Chomsky summarized the relationship between language families by classifying them into four language types, type 0 to type 3
- This classification, which became known as the *Chomsky Hierarchy*, is illustrated as below



An Extended Hierarchy

- We have studied additional language families and their relationships to those in the Chomsky Hierarchy
- By including deterministic context-free languages and recursive languages, we obtain the extended hierarchy as below



Recursive languages
Turing machine that halts on any inputs.

Deterministic context-free languages
DPDA

A Closer Look at the Family of Context-Free Languages

The following figure illustrates the relationships among various subsets of the family of context-free languages: regular (L_{REG}), linear (L_{LIN}), deterministic context-free (L_{DCF}), and nondeterministic context-free (L_{CF})