Theory of Formal Languages and Automata

Lecture 15

Mahdi Dolati

Sharif University of Technology

Fall 2023

November 14, 2023

Outline

- Some Decidable Properties
- Deterministic PDAs

Decidable Properties

- THEOREM 8.6

Given a context-free grammar $G = (V, T, S, P)$, there exists an algorithm for deciding whether or not $L(G)$ is empty.

Proof: For simplicity, assume that $\lambda \notin L(G)$. Slight changes have to be made in the argument if this is not so. We use the algorithm for removing useless symbols and productions. If S is found to be useless, then $L(G)$ is empty; if not, then $L(G)$ contains at least one element. ■

Decidable Properties

- There is no such algorithm to determine whether two context-free grammars generate the same language.

Deterministic PDA

- Definition:

A pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ is said to be deterministic if it is an automaton as defined in Definition 7.1, subject to the restrictions that, for every $q \in Q, a \in \Sigma \cup \{\lambda\}$ and $b \in \Gamma$,

1. $\delta(q, a, b)$ contains at most one element,

2. if $\delta(q, \lambda, b)$ is not empty, then $\delta(q, c, b)$ must be empty for every $c \in \Sigma$.

Deterministic PDA

- A language L is said to be a deterministic context-free language if and only if there exists a DPDA M such that L = L (M).

Deterministic PDA

- Example:

EXAMPLE 7.10

The language

$$L = \{a^n b^n : n \geq 0\}$$

is a deterministic context-free language. The pda $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{0, 1\}, \delta, q_0, 0, \{q_0\})$ with

$$\begin{aligned}
\delta(q_0, a, 0) &= \{(q_1, 10)\}, \\
\delta(q_1, a, 1) &= \{(q_1, 11)\}, \\
\delta(q_1, b, 1) &= \{(q_2, \lambda)\}, \\
\delta(q_2, b, 1) &= \{(q_2, \lambda)\}, \\
\delta(q_2, \lambda, 0) &= \{(q_0, \lambda)\}
\end{aligned}$$

accepts the given language. It satisfies the conditions of Definition 7.3 and is therefore deterministic.

Deterministic PDA

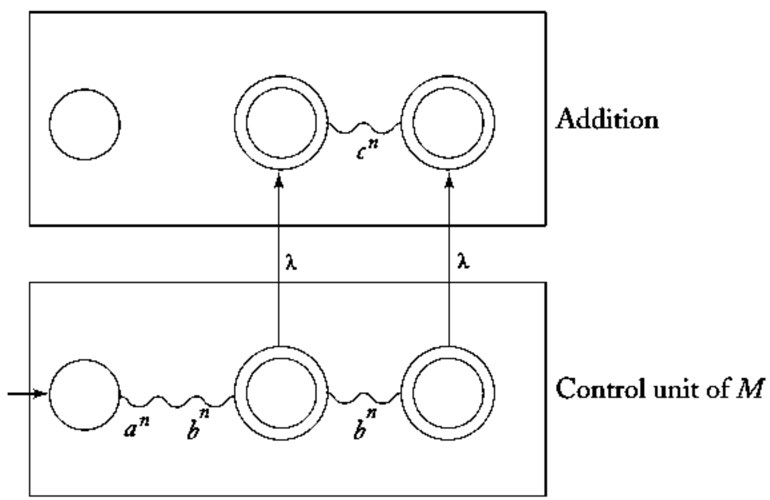
- **There are context-free languages that are not deterministic.**
- Example:
  - Consider $L_1 = \{a^n b^n : n \geq 0\}$ and $L_2 = \{a^n b^{2n} : n \geq 0\}$
  - We know $L = L_1 \cup L_2$ is CF
  - Observe that there is no information available at the beginning of any $w \in L$ string by which the choice can be made deterministically about whether $w \in L_1$ or $w \in L_2$

Deterministic PDA

- If $L = L_1 \cup L_2$ is a **deterministic** CF, then the following language is CF:

$$\widehat{L} = L \cup \{a^n b^n c^n : n \geq 0\}$$

- **Proof**: Assume DPDA M for L is available.
- Thus, we know state after reading $a^n b^n$.
- Copy M and replace all b with c.
- Consider the following PDA:



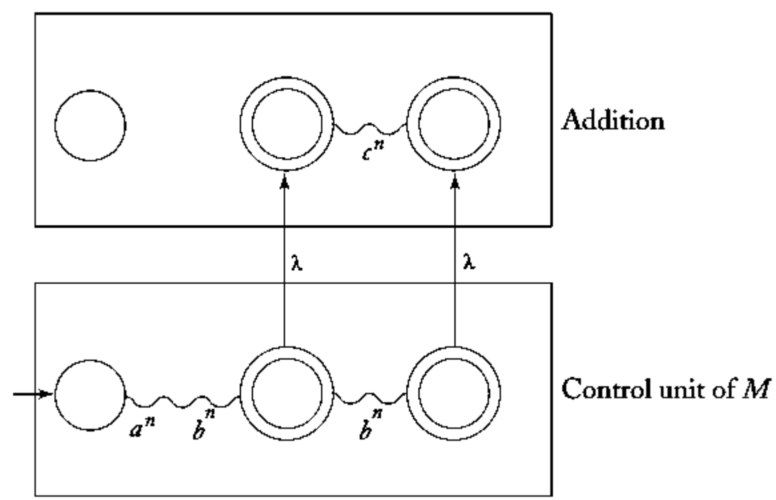
Deterministic PDA

- If $L = L_1 \cup L_2$ is a **deterministic** CF, then the following language is CF:

$$\widehat{L} = L \cup \{a^n b^n c^n : n \geq 0\}$$

- **Proof**:
- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with states $Q = \{q_0, q_1, ..., q_n\}$
- Construct $\widehat{M} = \left(\widehat{Q}, \Sigma, \Gamma, \delta \cup \widehat{\delta}, z, \widehat{F}\right)$ by

$$\widehat{Q} = Q \cup \{\widehat{q}_0, \widehat{q}_1, ..., \widehat{q}_n\},$$
$$\widehat{F} = F \cup \{\widehat{q}_i : q_i \in F\},$$

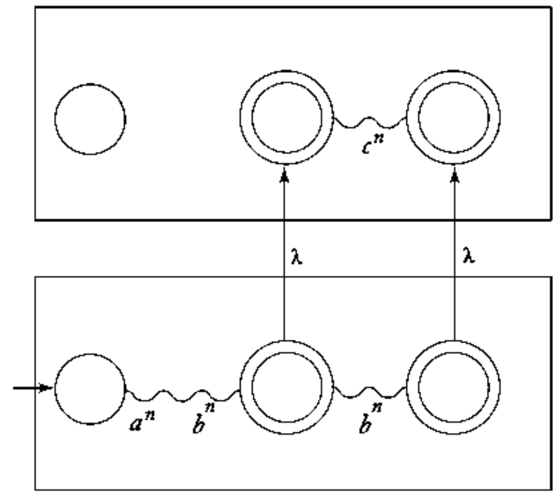


Deterministic PDA

- If $L = L_1 \cup L_2$ is a **deterministic** CF, then the following language is CF:

$$\widehat{L} = L \cup \{a^n b^n c^n : n \geq 0\}$$

- **Proof**:
- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with states $Q = \{q_0, q_1, ..., q_n\}$
- Construct $\widehat{M} = \left(\widehat{Q}, \Sigma, \Gamma, \delta \cup \widehat{\delta}, z, \widehat{F}\right)$ by



$q_f \in F, s \in \Gamma$ ⟹ $\widehat{\delta}(q_f, \lambda, s) = \{(\widehat{q}_f, s)\}$,

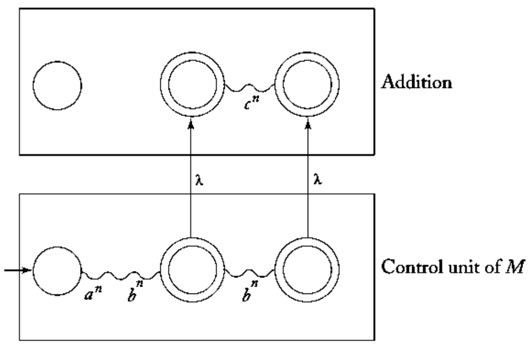
$\delta(q_i, b, s) = \{(q_j, u)\}$, $q_i \in Q, s \in \Gamma, u \in \Gamma^*$ ⟹ $\widehat{\delta}(\widehat{q}_i, c, s) = \{(\widehat{q}_j, u)\}$,

Deterministic PDA

- If $L = L_1 \cup L_2$ is a **deterministic** CF, then the following language is CF:

$$\widehat{L} = L \cup \{a^n b^n c^n : n \geq 0\}$$

- **Proof**:



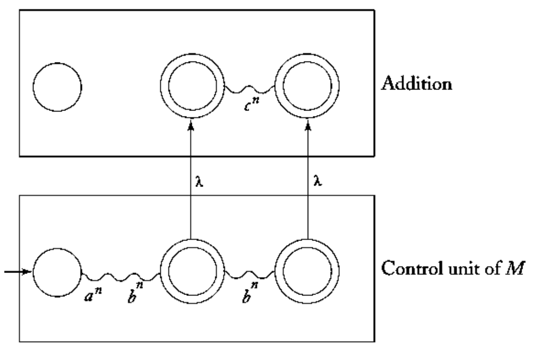
- Show $\widehat{M}$ accepts $a^n b^n c^n$, $a^n b^n$, and $a^n b^{2n}$
- Show $\widehat{M}$ accepts no other string

Deterministic PDA

- If $L = L_1 \cup L_2$ is a **deterministic** CF, then the following language is CF:

$$\widehat{L} = L \cup \{a^n b^n c^n : n \geq 0\}$$

- **Proof**:



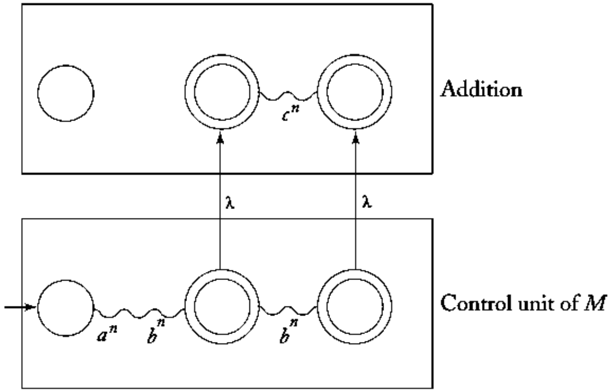
- Show $\widehat{M}$ accepts $a^n b^n c^n$, $a^n b^n$, and $a^n b^{2n}$
- Show $\widehat{M}$ accepts no other string
  - Exercise

Deterministic PDA

- If $L = L_1 \cup L_2$ is a **deterministic** CF, then the following language is CF:

$$\widehat{L} = L \cup \{a^n b^n c^n : n \geq 0\}$$

- **Proof**:



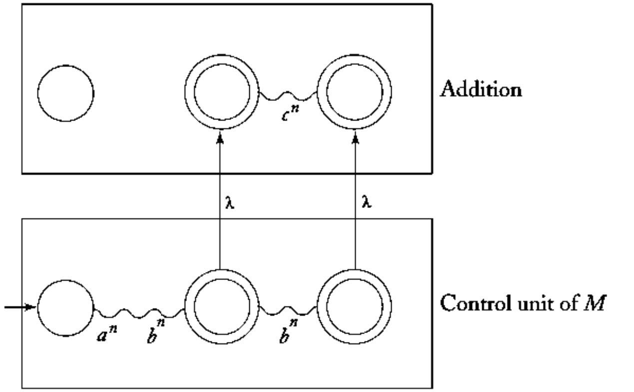
- Show $\widehat{M}$ accepts $a^n b^n c^n$, $a^n b^n$, and $a^n b^{2n}$:

- M accepts $a^n b^n$: $(q_0, a^n b^n, z) \overset{*}{\underset{M}{\Rightarrow}} (q_i, \lambda, u)$, $\quad q_i \in F$.
- M is deterministic: $(q_0, a^n b^{2n}, z) \overset{*}{\underset{M}{\Rightarrow}} (q_i, \lambda, u)$
- Also, $(q_i, b^n, u) \overset{*}{\underset{M}{\Rightarrow}} (q_j, \lambda, u_1)$, $\quad q_j \in F$.
- And, by construction $(\hat{q}_i, c^n, u) \overset{*}{\underset{\widehat{M}}{\Rightarrow}} (\hat{q}_j, \lambda, u_1) \rightarrow \widehat{M}$ accepts $a^n b^n c^n$

Deterministic PDA

- If $L = L_1 \cup L_2$ is a **deterministic** CF, then the following language is CF:

$$\widehat{L} = L \cup \{a^n b^n c^n : n \geq 0\}$$

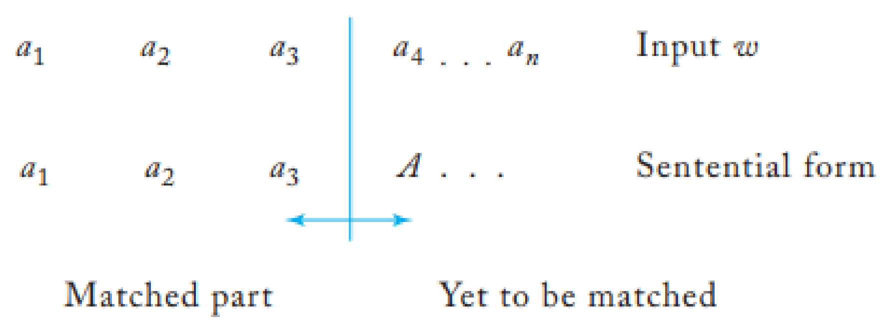


- **Proof**:

- Thus, $\mathrm{L}(\widehat{M}) = \hat{L}$.
- It is a contradiction as $\hat{L}$ is not CF.

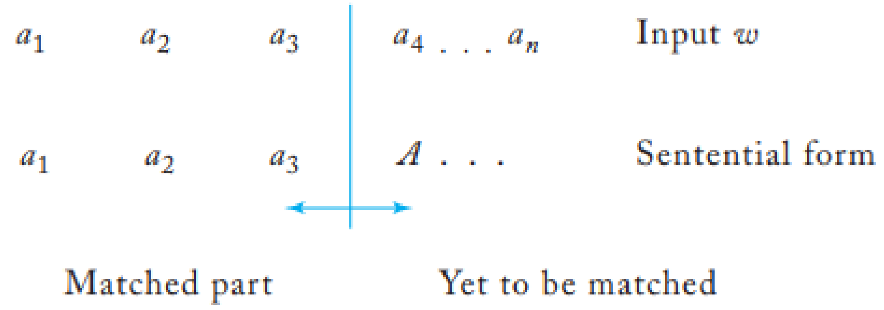
GRAMMARS FOR DETERMINISTIC CONTEXT-FREE LANGUAGES

- We can parse DCFLs efficiently,
- We can follow the transitions of a DPDA with no backtracking or parallelism,
  - What about ε-transition?
- Design of deterministic grammars,
  - Good for compilers,
- Suppose we are parsing **top-down**, attempting to find the leftmost derivation of a particular sentence.
- We scan the input w from left to right, while developing a sentential form whose terminal prefix matches the prefix of w up to the currently scanned symbol.
- Know which rule to use?



GRAMMARS FOR DETERMINISTIC CONTEXT-FREE LANGUAGES

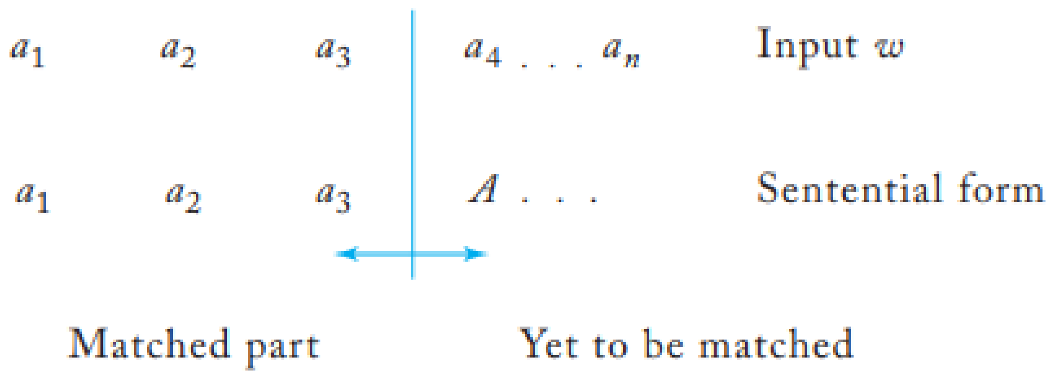
- We can parse DCFLs efficiently,
- We can follow the transitions of a DPDA with no backtracking or parallelism,
  - What about ε-transition?
- Design of deterministic grammars,
  - Good for compilers,
- Suppose we are parsing top-down, attempting to find the leftmost derivation of a particular sentence.
- We scan the input w from left to right, while developing a sentential form whose terminal prefix matches the prefix of w up to the currently scanned symbol.
- Know which rule to use?
- In case of s-grammars there is one rule at each step.
  - Too restrictive.



GRAMMARS FOR DETERMINISTIC CONTEXT-FREE LANGUAGES

- LL grammars:
  - L: The input is scanned from left to right,
  - L: Leftmost derivative is contructed.

- Predict which rule to use by scanning a symbol plus a finite number of symbols following it.
  - Every s-grammar is an LL grammar.



GRAMMARS FOR DETERMINISTIC CONTEXT-FREE LANGUAGES

- LL grammars:
  - LL (k) grammar: We can uniquely identify the correct production by scanning the current symbol and a “lookahead” of the next k −1 symbols.

EXAMPLE 7.12

The grammar

$$S \rightarrow aSb|ab$$

is not an s-grammar, but it is an LL grammar. In order to determine which production is to be applied, we look at two consecutive symbols of the input string. If the first is an a and the second a b, we must apply the production $S \rightarrow ab$. Otherwise, the rule $S \rightarrow aSb$ must be used.

GRAMMARS FOR DETERMINISTIC CONTEXT-FREE LANGUAGES

- The following grammar (the language of properly nested parenthesis structures) is not an LL(k) grammar for any k.

$$S \to SS \,|\, aSb \,|\, ab$$

- No matter how many symbols you examine, you can not decide which rule to use:
  - ((((((((((((((….((()))….)))))))))))
  - ((((((((((((((….))))))))

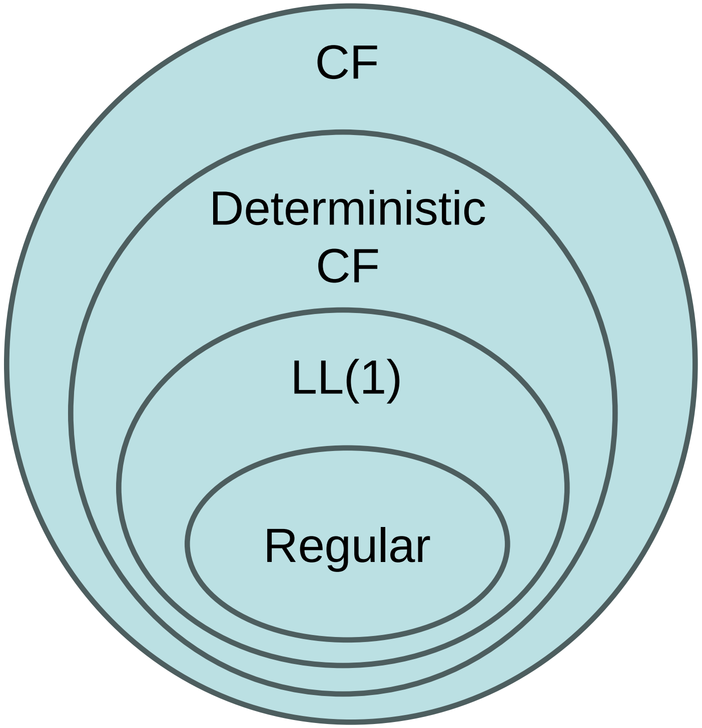
GRAMMARS FOR DETERMINISTIC CONTEXT-FREE LANGUAGES

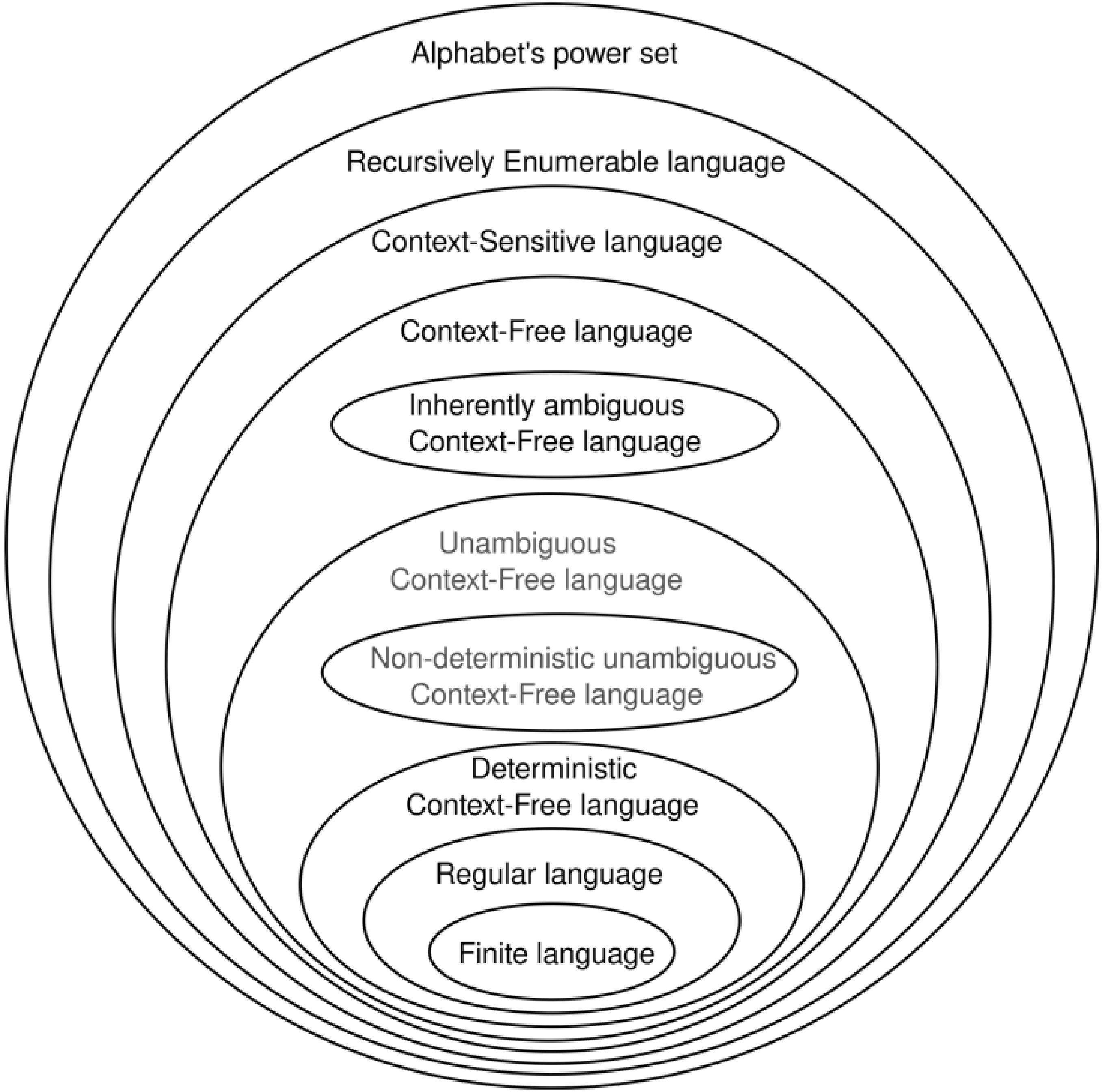
- The following grammar (the language of properly nested parenthesis structures) is not an LL(k) grammar for any k.

$$S \rightarrow SS \,|aSb|\, ab$$

- Following is an LL equivalent:

$$S_0 \rightarrow aSbS,$$
$$S \rightarrow aSbS|\lambda$$


CF
Deterministic
CF
LL(1)
Regular


Alphabet's power set
Recursively Enumerable language
Context-Sensitive language
Context-Free language
Inherently ambiguous
Context-Free language
Unambiguous
Context-Free language
Non-deterministic unambiguous
Context-Free language
Deterministic
Context-Free language
Regular language
Finite language