COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

March 15, 2023

Lecture 20: CFLs: Closure Properties

Recap

- New model: NPDA = NFA + Stack.
- Context-free Languages: those languages accepted by NPDAs.

$$L_{0,1} = \{0^n 1^n \mid n \in \mathbb{N}\}$$

$$PAL = \{ww^R \mid w \in \Sigma^*\}$$

- Algebraic way to define CFLs: Context-free Grammars.

Recap

- New model: NPDA = NFA + Stack.
- Context-free Languages: those languages accepted by NPDAs.

$$L_{0,1} = \{0^n 1^n \mid n \in \mathbb{N}\}$$

$$PAL = \{ww^R \mid w \in \Sigma^*\}$$

- Algebraic way to define CFLs: Context-free Grammars.
- CFG - Chomsky Normal Form, Parse trees, Ambiguity.
- Pumping Lemma for CFLs.

Recap

- New model: NPDA = NFA + Stack.
- Context-free Languages: those languages accepted by NPDAs.

$$L_{0,1} = \{0^n 1^n \mid n \in \mathbb{N}\}$$

$$PAL = \{ww^R \mid w \in \Sigma^*\}$$

- Algebraic way to define CFLs: Context-free Grammars.
- CFG - Chomsky Normal Form, Parse trees, Ambiguity.
- Pumping Lemma for CFLs.

$$L_{0,1,2} = \{0^n 1^n 2^n \mid n \in \mathbb{N}\}$$

$$EQ = \{ww \mid w \in \Sigma^*\}$$

Recap

- New model: NPDA = NFA + Stack.
- Context-free Languages: those languages accepted by NPDAs.

$$L_{0,1} = \{0^n 1^n \mid n \in \mathbb{N}\}$$

$$PAL = \{ww^R \mid w \in \Sigma^*\}$$

- Algebraic way to define CFLs: Context-free Grammars.
- CFG - Chomsky Normal Form, Parse trees, Ambiguity.
- Pumping Lemma for CFLs.

$$L_{0,1,2} = \{0^n 1^n 2^n \mid n \in \mathbb{N}\}$$

$$EQ = \{ww \mid w \in \Sigma^*\}$$

- Equivalence of CFG and NPDA.

Properties of CFLs

Theorem

Context-free languages are closed under the following operations:

1. *Union*
2. *Concatenation*
3. *Kleene closure*
4. *Homomorphism*
5. *Substitution*
6. *Inverse-homomorphism*
7. *Reverse*

Properties of CFLs

Theorem

Context-free languages are closed under the following operations:

1. *Union*
2. *Concatenation*
3. *Kleene closure*
4. *Homomorphism*
5. *Substitution*
6. *Inverse-homomorphism*
7. *Reverse*

What about complementation? Intersection?

Properties of CFLs

Theorem

Context-free languages are not closed under complement and intersection.

Properties of CFLs

Theorem

Context-free languages are not closed under complement and intersection.

Proof.

$$L_1 = \{0^n 1^n 2^m \mid n, m \geq 0\}$$
$$L_2 = \{0^m 1^n 2^n \mid n, m \geq 0\}$$

Properties of CFLs

Theorem

Context-free languages are not closed under complement and intersection.

Proof.

$$L_1 = \{0^n 1^n 2^m \mid n, m \geq 0\}$$
$$L_2 = \{0^m 1^n 2^n \mid n, m \geq 0\}$$

What is $L_1 \cap L_2$?

Properties of CFLs

Theorem

Context-free languages are not closed under complement and intersection.

Proof.

$$L_1 = \{0^n 1^n 2^m \mid n, m \geq 0\}$$
$$L_2 = \{0^m 1^n 2^n \mid n, m \geq 0\}$$

What is $L_1 \cap L_2$?
$L = \{0^n 1^n 2^n \mid n \geq 0\}$
Hence CFLs are not closed under intersection. □

Properties of CFLs

Theorem

Context-free languages are not closed under complement and intersection.

Proof.

$$L_1 = \{0^n 1^n 2^m \mid n, m \geq 0\}$$
$$L_2 = \{0^m 1^n 2^n \mid n, m \geq 0\}$$

What is $L_1 \cap L_2$?
$L = \{0^n 1^n 2^n \mid n \geq 0\}$
Hence CFLs are not closed under intersection. □

Exercise 1: What if L_1 is regular and L_2 is context-free?
Exercise 2: Prove closure under homomorphisms and inverse homomorphisms.

Computational questions

- **Membership:** Given a CFG G, and a string w, does $w \in L(G)$?

Computational questions

- **Membership:** Given a CFG G, and a string w, does $w \in L(G)$?

Membership

$$
\begin{aligned}
S &\rightarrow AB \mid BA \mid SS \mid AC \mid BD \\
A \rightarrow a &\qquad B \rightarrow b \\
C \rightarrow SB &\qquad D \rightarrow SA
\end{aligned}
$$

$$
\begin{array}{ccccccccccccc}
 & a & & a & & b & & b & & a & & b & \\
0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6
\end{array}
$$

Membership

$$
\begin{array}{rcl}
S & \rightarrow & AB \mid BA \mid SS \mid AC \mid BD \\
A \rightarrow a & & B \rightarrow b \\
C \rightarrow SB & & D \rightarrow SA
\end{array}
$$

$$
\begin{array}{ccccccccccccc}
| & a & | & a & | & b & | & b & | & a & | & b & | \\
0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6
\end{array}
$$

Membership

$$
\begin{aligned}
S &\rightarrow AB \mid BA \mid SS \mid AC \mid BD \\
A \rightarrow a \qquad & B \rightarrow b \\
C \rightarrow SB \qquad & D \rightarrow SA
\end{aligned}
$$

$$
\begin{array}{ccccccc}
 & a & a & b & b & a & b \\
0 & 1 & 2 & 3 & 4 & 5 & 6
\end{array}
$$

0						
–	1					
–	–	2				
–	–	–	3			
–	–	–	–	4		
–	–	–	–	–	5	
–	–	–	–	–	–	6

Membership

$$
\begin{aligned}
S &\rightarrow AB \mid BA \mid SS \mid AC \mid BD \\
A \rightarrow a \quad & \quad B \rightarrow b \\
C \rightarrow SB \quad & \quad D \rightarrow SA
\end{aligned}
$$

$$|a|a|b|b|a|b|$$
$$0 \;\; 1 \;\; 2 \;\; 3 \;\; 4 \;\; 5 \;\; 6$$

0						
A	1					
–	A	2				
–	–	B	3			
–	–	–	B	4		
–	–	–	–	A	5	
–	–	–	–	–	B	6

Membership

$$
\begin{array}{rcl}
S & \to & AB \mid BA \mid SS \mid AC \mid BD \\
A \to a & & B \to b \\
C \to SB & & D \to SA
\end{array}
$$

$$
\begin{array}{ccccccccccccc}
| & a & | & a & | & b & | & b & | & a & | & b & | \\
0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6
\end{array}
$$

$$
\begin{array}{ccccccc}
0 & & & & & & \\
A & 1 & & & & & \\
\varnothing & A & 2 & & & & \\
- & S & B & 3 & & & \\
- & - & \varnothing & B & 4 & & \\
- & - & - & S & A & 5 & \\
- & - & - & - & S & B & 6
\end{array}
$$

Membership

$$S \rightarrow AB \mid BA \mid SS \mid AC \mid BD$$

$$A \rightarrow a \qquad B \rightarrow b$$

$$C \rightarrow SB \qquad D \rightarrow SA$$

$$\begin{array}{ccccccccccccc} | & a & | & a & | & b & | & b & | & a & | & b & | \\ 0 & & 1 & & 2 & & 3 & & 4 & & 5 & & 6 \end{array}$$

$$\begin{array}{ccccccc}
0 & & & & & & \\
A & 1 & & & & & \\
\varnothing & A & 2 & & & & \\
\varnothing & S & B & 3 & & & \\
- & C & \varnothing & B & 4 & & \\
- & - & - & S & A & 5 & \\
- & - & - & - & S & B & 6
\end{array}$$

Membership

$$S \rightarrow AB \mid BA \mid SS \mid AC \mid BD$$

$$A \rightarrow a \qquad B \rightarrow b$$

$$C \rightarrow SB \qquad D \rightarrow SA$$

$$|a|a|b|b|a|b|$$
$$0\ 1\ 2\ 3\ 4\ 5\ 6$$

0						
A	1					
$\varnothing$	A	2				
$\varnothing$	S	B	3			
S	C	$\varnothing$	B	4		
D	S	$\varnothing$	S	A	5	
S	C	$\varnothing$	C	S	B	6

The Cocke-Kasami-Younger Algorithm

```
for i := 0 to n − 1 do                        /* strings of length 1 first */
  begin
    T_{i,i+1} := ∅;                           /* initialize to ∅ */
    for A → a a production of G do
      if a = x_{i,i+1} then T_{i,i+1} := T_{i,i+1} ∪ {A}
  end;
for m := 2 to n do                            /* for each length m ≥ 2 */
  for i := 0 to n − m do                      /* for each substring */
    begin                                                /* of length m */
      T_{i,i+m} := ∅;                         /* initialize to ∅ */
      for j := i + 1 to i + m − 1 do          /* for all ways to break */
        for A → BC a production of G do              /* up the string */
          if B ∈ T_{i,j} ∧ C ∈ T_{j,i+m}
            then T_{i,i+m} := T_{i,i+m} ∪ {A}
    end;
```

Computational questions

- **Membership:** Given a CFG G, and a string w, does $w \in L(G)$?
- **Membership':** Given an arbitrary string w, does there exist a CFG G such that $w \in L(G)$?

Computational questions

- **Membership:** Given a CFG G, and a string w, does $w \in L(G)$?
- **Membership':** Given an arbitrary string w, does there exist a CFG G such that $w \in L(G)$?
- **Emptiness/Universality:** Given CFG G is $L(G) = \emptyset$ or $L(G) = \Sigma^*$?

Computational questions

- **Membership:** Given a CFG G, and a string w, does $w \in L(G)$?
- **Membership':** Given an arbitrary string w, does there exist a CFG G such that $w \in L(G)$?
- **Emptiness/Universality:** Given CFG G is $L(G) = \emptyset$ or $L(G) = \Sigma^*$?
- **Minimisation:** Given CFG G does there exist a CFG G' with smaller number of variables/productions such that $L(G) = L(G')$?

Computational questions

- **Membership:** Given a CFG G, and a string w, does $w \in L(G)$?
- **Membership':** Given an arbitrary string w, does there exist a CFG G such that $w \in L(G)$?
- **Emptiness/Universality:** Given CFG G is $L(G) = \emptyset$ or $L(G) = \Sigma^*$?
- **Minimisation:** Given CFG G does there exist a CFG G' with smaller number of variables/productions such that $L(G) = L(G')$?
- **Language Equivalence:** Given CFG G_1, G_2, does $L(G_1) = L(G_2)$?

Computational questions

- **Membership:** Given a CFG G, and a string w, does $w \in L(G)$?
- **Membership':** Given an arbitrary string w, does there exist a CFG G such that $w \in L(G)$?
- **Emptiness/Universality:** Given CFG G is $L(G) = \emptyset$ or $L(G) = \Sigma^*$?
- **Minimisation:** Given CFG G does there exist a CFG G' with smaller number of variables/productions such that $L(G) = L(G')$?
- **Language Equivalence:** Given CFG G_1, G_2, does $L(G_1) = L(G_2)$?
- **Unambiguity:** Given CFG G is $L(G)$ unambiguous?

Other possible variants

- 2-NPDA vs NPDA?

Other possible variants

- 2-NPDA vs NPDA?
- Do there exist grammars that capture them?

Other possible variants

- 2-NPDA vs NPDA?
- Do there exist grammars that capture them?
- There exist 2-NPDA that can accept $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$.

Other possible variants

- 2-NPDA vs NPDA?
- Do there exist grammars that capture them?
- There exist 2-NPDA that can accept $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$.
- What about machines with 2 stacks?
- What about all these machines with k pointers on the input tape?

Other possible variants

- 2-NPDA vs NPDA?
- Do there exist grammars that capture them?
- There exist 2-NPDA that can accept $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$.
- What about machines with 2 stacks?
- What about all these machines with k pointers on the input tape?
- PDA/Grammars with weights for each transitions? (useful in NLP)

Other possible variants

- 2-NPDA vs NPDA?
- Do there exist grammars that capture them?
- There exist 2-NPDA that can accept $\{0^n 1^n 2^n \mid n \in \mathbb{N}\}$.
- What about machines with 2 stacks?
- What about all these machines with k pointers on the input tape?
- PDA/Grammars with weights for each transitions? (useful in NLP)

Is there a machine-independent notion of computation?