COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

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

March 1, 2023

Lecture 17: Context-Free Grammars

Pushdown Automata

Definition

A non-deterministic pushdown automaton (NPDA)
$A = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$, where

Q:	set of states	Σ:	input alphabet
Γ:	stack alphabet	q_0:	start state
$\perp$:	start symbol	F:	set of final states

$$\delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \Gamma^*.$$

Understanding δ

For $q \in Q, a \in \Sigma$ and $X \in \Gamma$, if $\delta(q, a, X) = (p, \gamma)$,
then p is the new state and γ replaces X in the stack.

if $\gamma = \epsilon$ then X is popped.
if $\gamma = X$ then X stays unchanges on the top of the stack.
if $\gamma = \gamma_1\gamma_2 \dots \gamma_k$ then X is replaced by γ_k
and $\gamma_1\gamma_2 \dots \gamma_{k-1}$ are pushed on top of that.

Context-free grammars

Definition

A context-free grammar (CFG) G is given by (V, T, P, S_0), where

Context-free grammars

Definition

A context-free grammar (CFG) G is given by (V, T, P, S_0), where

V is a set of variables (non-terminals/vocabulary),

T is a set of terminal symbols or the alphabet,

P is a set of (rewriting rules) productions, $P \subseteq V \times (V \cup T)^*$,

$S_0 \in V$, a start ("sentence") symbol.

Definition ($\Rightarrow^*$)

Let G be a CFG given by (V, T, P, S_0).

For all $\alpha \in (V \cup T)^*$, we say that $\alpha \Rightarrow^0 \alpha$.

Context-free grammars

Definition

A context-free grammar (CFG) G is given by (V, T, P, S_0), where

- V is a set of variables (non-terminals/vocabulary),
- T is a set of terminal symbols or the alphabet,
- P is a set of (rewriting rules) productions, $P \subseteq V \times (V \cup T)^*$,
- $S_0 \in V$, a start ("sentence") symbol.

Definition ($\Rightarrow^*$)

Let G be a CFG given by (V, T, P, S_0).

- For all $\alpha \in (V \cup T)^*$, we say that $\alpha \Rightarrow^0 \alpha$.
- For all $\alpha, \beta, \gamma \in (V \cup T)^*$,
  - if $\alpha \Rightarrow^{k-1} \beta$ and $\beta \Rightarrow \gamma$

Context-free grammars

Definition

A context-free grammar (CFG) G is given by (V, T, P, S_0), where

V is a set of variables (non-terminals/vocabulary),

T is a set of terminal symbols or the alphabet,

P is a set of (rewriting rules) productions, $P \subseteq V \times (V \cup T)^*$,

$S_0 \in V$, a start ("sentence") symbol.

Definition ($\Rightarrow^*$)

Let G be a CFG given by (V, T, P, S_0).

For all $\alpha \in (V \cup T)^*$, we say that $\alpha \Rightarrow^0 \alpha$.

For all $\alpha, \beta, \gamma \in (V \cup T)^*$,

if $\alpha \Rightarrow^{k-1} \beta$ and $\beta \Rightarrow \gamma$ then $\alpha \Rightarrow^k \gamma$.

For all $\alpha, \beta \in (V \cup T)^*$, we say that $\alpha \Rightarrow^* \beta$, if $\exists k \geq 0$ s.t. $\alpha \Rightarrow^k \beta$.

Language of Arithmetic expressions

Consider the language of arithmetic expressions over binary numbers.

- $10 + 1$
- $(10 + 101) \times 100$

Language of Arithmetic expressions

Consider the language of arithmetic expressions over binary numbers.

- $10 + 1$
- $(10 + 101) \times 100$

$T = \{+, \times, 0, 1, (,)\}$ are the terminals

Language of Arithmetic expressions

Consider the language of arithmetic expressions over binary numbers.

- $10 + 1$
- $(10 + 101) \times 100$

$T = \{+, \times, 0, 1, (,)\}$ are the terminals

$\{B, E\}$ are two non-terminals.

- B is a non-terminal for binary numbers.
- E is a non-terminal for expressions.

Language of Arithmetic expressions

Consider the language of arithmetic expressions over binary numbers.

- $10 + 1$
- $(10 + 101) \times 100$

$T = \{+, \times, 0, 1, (,)\}$ are the terminals

$\{B, E\}$ are two non-terminals.

- B is a non-terminal for binary numbers.
- E is a non-terminal for expressions.

E is the start symbol.

Language of Arithmetic expressions

Consider the language of arithmetic expressions over binary numbers.

- $10 + 1$
- $(10 + 101) \times 100$

$T = \{+, \times, 0, 1, (,)\}$ are the terminals

$\{B, E\}$ are two non-terminals.

- B is a non-terminal for binary numbers.
- E is a non-terminal for expressions.

E is the start symbol.

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \rightarrow 1$
- $B \rightarrow B0$
- $B \rightarrow B1$

- $E \rightarrow B$
- $E \rightarrow E + E$
- $E \rightarrow E \times E$
- $E \rightarrow (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$
- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies E \times E + B \implies$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$$E \implies E + E \implies E + B \implies E \times E + B \implies E \times B + B$$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies E \times E + B \implies E \times B + B \implies E \times 1 + B$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies E \times E + B \implies E \times B + B \implies$
$E \times 1 + B \implies B \times 1 + B \implies B \times 1 + B \implies$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies E \times E + B \implies E \times B + B \implies$
$E \times 1 + B \implies B \times 1 + B \implies B \times 1 + B \implies B0 \times 1 + B \implies$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$
- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies E \times E + B \implies E \times B + B \implies E \times 1 + B \implies B \times 1 + B \implies B \times 1 + B \implies B0 \times 1 + B \implies B0 \times 1 + B1 \implies$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$
- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies E \times E + B \implies E \times B + B \implies$
$E \times 1 + B \implies B \times 1 + B \implies B \times 1 + B \implies B0 \times 1 + B \implies$
$B0 \times 1 + B1 \implies 10 \times 1 + B1 \implies$

Grammar for Arithmetic expressions

$G_{arith} = (\{E, B\}, \{+, \times, 0, 1, (,)\}, P, E)$

- $B \to 1$
- $B \to B0$
- $B \to B1$
- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$

$E \implies E + E \implies E + B \implies E \times E + B \implies E \times B + B \implies E \times 1 + B \implies B \times 1 + B \implies B \times 1 + B \implies B0 \times 1 + B \implies B0 \times 1 + B1 \implies 10 \times 1 + B1 \implies 10 \times 1 + 11$

Too many choices for derivation

- We can expand on any non-terminal

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \rightarrow 1$
- $B \rightarrow B0$
- $B \rightarrow B1$

- $E \rightarrow B$
- $E \rightarrow E + E$
- $E \rightarrow E \times E$
- $E \rightarrow (E)$

Lets look at the derivation of $10 \times 1 + 11$:
$E \implies E + E$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:
$E \implies E + E \implies E \times E + E$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:

$E \implies E + E \implies E \times E + E \implies B \times E + E$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:

$E \implies E + E \implies E \times E + E \implies B \times E + E \implies B0 \times E + E$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \rightarrow 1$
- $B \rightarrow B0$
- $B \rightarrow B1$

- $E \rightarrow B$
- $E \rightarrow E + E$
- $E \rightarrow E \times E$
- $E \rightarrow (E)$

Lets look at the derivation of $10 \times 1 + 11$:

$E \implies E + E \implies E \times E + E \implies B \times E + E \implies B0 \times E + E \implies 10 \times E + E$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:

$E \implies E + E \implies E \times E + E \implies B \times E + E \implies B0 \times E + E \implies 10 \times E + E \implies 10 \times B + E$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:

$E \implies E + E \implies E \times E + E \implies B \times E + E \implies B0 \times E + E \implies 10 \times E + E \implies 10 \times B + E \implies 10 \times 1 + E$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$
- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:

$E \implies E + E \implies E \times E + E \implies B \times E + E \implies B0 \times E + E \implies$
$10 \times E + E \implies 10 \times B + E \implies 10 \times 1 + E \implies 10 \times 1 + B$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:

$E \implies E + E \implies E \times E + E \implies B \times E + E \implies B0 \times E + E \implies 10 \times E + E \implies 10 \times B + E \implies 10 \times 1 + E \implies 10 \times 1 + B \implies 10 \times 1 + B1$

Too many choices for derivation

- We can expand on any non-terminal
- Inherently there is some non-determinism.
- We can limit the derivation moves and have predictable sequence of derivations, i.e., making derivations deterministic.
- Idea: Always derive on the left-most non-terminal.

- $B \to 1$
- $B \to B0$
- $B \to B1$

- $E \to B$
- $E \to E + E$
- $E \to E \times E$
- $E \to (E)$

Lets look at the derivation of $10 \times 1 + 11$:
$E \implies E + E \implies E \times E + E \implies B \times E + E \implies B0 \times E + E \implies 10 \times E + E \implies 10 \times B + E \implies 10 \times 1 + E \implies 10 \times 1 + B \implies 10 \times 1 + B1 \implies 10 \times 1 + 11$

Words extending at many places

Words extending at many places

- In regular languages, words are extended at the end depending on the finite information collected on the word so far.
- In CFLs, words are extended at unboundedly many points, which gives CFLs more power.
- To understand the above intuition, we view the words in derivations as tree.

Parse Trees

- A CFG provide a structure to a string.

Parse Trees

- A CFG provide a structure to a string.
- Such structure assigns meaning to a string, and hence a unique structure is really important in several applications, e.g. compilers

Parse Trees

- A CFG provide a structure to a string.
- Such structure assigns meaning to a string, and hence a unique structure is really important in several applications, e.g. compilers
- Parse trees are a successful data-structures to represent and store such structures

Parse Trees

- A CFG provide a structure to a string.
- Such structure assigns meaning to a string, and hence a unique structure is really important in several applications, e.g. compilers
- Parse trees are a successful data-structures to represent and store such structures
- Let's review the Tree terminology:
  - A tree is a directed acyclic graph (DAG) where every node has at most incoming edge.
  - Edge relationship as parent-child relationship
  - Every node has at most one parent, and zero or more children
  - We assume an implicit order on children ("from left-to-right")
  - There is a distinguished root node with no parent, while all other nodes have a unique parent
  - There are some nodes with no children called leaves—other nodes are called interior nodes
  - Ancestor and descendent relationships are closure of parent and child relationships, resp

Parse trees in a CFG

- Given a grammar $G = (V, T, P, S)$, the parse trees associated with G has the following properties:

Parse trees in a CFG

- Given a grammar $G = (V, T, P, S)$, the parse trees associated with G has the following properties:
  - Each interior node is labeled by a variable in V.

Parse trees in a CFG

- Given a grammar $G = (V, T, P, S)$, the parse trees associated with G has the following properties:
  - Each interior node is labeled by a variable in V.
  - Each leaf is either a variable, terminal, or ϵ. However, if a leaf is ϵ it is the only child of its parent.

Parse trees in a CFG

- Given a grammar $G = (V, T, P, S)$, the parse trees associated with G has the following properties:
  - Each interior node is labeled by a variable in V.
  - Each leaf is either a variable, terminal, or ϵ. However, if a leaf is ϵ it is the only child of its parent.
  - If an interior node is labeled A and has children labeled $X_1, X_2, ..., X_k$ from left-to-right, then

    $$A \to X_1 X_2 \dots X_k$$

    is a production is P. Only time X_i can be ϵ is when it is the only child of its parent, i.e. corresponding to the production $A \to \epsilon$.
- **Exercise:** Give parse tree representation of examples seen so far.

Ambiguity and Interpretation

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.
Not good for building compilers.

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.
Not good for building compilers.
$10 \times 1 + 11$ can be 101 (binary 5) or 1000 (binary 8) based on whether you multiply first or add first.

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.
Not good for building compilers.
$10 \times 1 + 11$ can be 101 (binary 5) or 1000 (binary 8) based on whether you multiply first or add first.

- Time flies like an arrow.
- Fruit flies like a banana.

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.

Not good for building compilers.

$10 \times 1 + 11$ can be 101 (binary 5) or 1000 (binary 8) based on whether you multiply first or add first.

- Time flies like an arrow.
- Fruit flies like a banana.

Question: Is it possible to remove ambiguity from a grammar?

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.
Not good for building compilers.
$10 \times 1 + 11$ can be 101 (binary 5) or 1000 (binary 8) based on whether you multiply first or add first.

- Time flies like an arrow.
- Fruit flies like a banana.

Question: Is it possible to remove ambiguity from a grammar? **Answer:** Not in general! (coming up in a month)

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.
Not good for building compilers.
$10 \times 1 + 11$ can be 101 (binary 5) or 1000 (binary 8) based on whether you multiply first or add first.

- Time flies like an arrow.
- Fruit flies like a banana.

Question: Is it possible to remove ambiguity from a grammar? **Answer:** Not in general! (coming up in a month) Can you remove ambiguity from specific grammars?

Ambiguity and Interpretation

Definition

yield of a parse tree = word formed by all the leaves from left to right.

Definition

A CFG G is called ambiguous if there is a word $w \in L(G)$ such that there are two parse trees that yield w.

Ambiguity leads to multiple interpretations of the word.
Not good for building compilers.
$10 \times 1 + 11$ can be 101 (binary 5) or 1000 (binary 8) based on whether you multiply first or add first.

- Time flies like an arrow.
- Fruit flies like a banana.

Question: Is it possible to remove ambiguity from a grammar? **Answer:** Not in general! (coming up in a month) Can you remove ambiguity from specific grammars? **Exercise:** Come up with an unambiguous grammar for the Arithmetic Expression parsing

Chomsky normal form

Definition

A context-free grammar is said to be in Chomsky normal form if every rule is of the form

Chomsky normal form

Definition

A context-free grammar is said to be in Chomsky normal form if every rule is of the form

$$A \to BC$$

Chomsky normal form

Definition

A context-free grammar is said to be in Chomsky normal form if every rule is of the form

$$A \rightarrow BC$$

$$A \rightarrow a$$

Chomsky normal form

Definition

A context-free grammar is said to be in Chomsky normal form if every rule is of the form

$$A \to BC$$

$$A \to a$$

where $a \in T$, $A, B, C \in V$, neither B nor C is the start variable, i.e. start variabe does not appear on the right of any rule.

Chomsky normal form

Definition

A context-free grammar is said to be in Chomsky normal form if every rule is of the form

$$A \to BC$$

$$A \to a$$

where $a \in T$, $A, B, C \in V$, neither B nor C is the start variable, i.e. start variabe does not appear on the right of any rule. Moreover, epsilon does not appear on the right of any rule except as $S \to \epsilon$.

Chomsky normal form

Definition

A context-free grammar is said to be in Chomsky normal form if every rule is of the form

$$A \to BC$$

$$A \to a$$

where $a \in T$, $A, B, C \in V$, neither B nor C is the start variable, i.e. start variabe does not appear on the right of any rule. Moreover, epsilon does not appear on the right of any rule except as $S \to \epsilon$.

Lemma

Any context-free grammar G can be converted into another context-free grammar G'

Chomsky normal form

Definition

A context-free grammar is said to be in Chomsky normal form if every rule is of the form

$$A \to BC$$

$$A \to a$$

where $a \in T$, $A, B, C \in V$, neither B nor C is the start variable, i.e. start variabe does not appear on the right of any rule. Moreover, epsilon does not appear on the right of any rule except as $S \to \epsilon$.

Lemma

Any context-free grammar G can be converted into another context-free grammar G' such that $L(G) = L(G')$ and G' is in the Chomsky normal form.

Chomsky normal form

Step 1: Add a new start symbol

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \rightarrow S$

Step 2: Remove ϵ rules.

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

If $R \to uAvAw$ is a rule

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

If $R \to uAvAw$ is a rule then delete the rule and add $R \to uvAw \mid uAvw \mid uvw$.

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

If $R \to uAvAw$ is a rule then delete the rule and add $R \to uvAw \mid uAvw \mid uvw$.

If $R \to A$ is a rule

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \rightarrow S$

Step 2: Remove ϵ rules.

Suppose $A \rightarrow \epsilon$ is a rule and A is not the start symbol.

If $R \rightarrow uAv$ is a rule then delete the rule and add $R \rightarrow uv$ to the rules.

If $R \rightarrow uAvAw$ is a rule then delete the rule and add $R \rightarrow uvAw \mid uAvw \mid uvw$.

If $R \rightarrow A$ is a rule then delete the rule and add $R \rightarrow \epsilon$

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

If $R \to uAvAw$ is a rule then delete the rule and add $R \to uvAw \mid uAvw \mid uvw$.

If $R \to A$ is a rule then delete the rule and add $R \to \epsilon$ unless $R \to \epsilon$ was already removed.

Step 3: Remove unit rules.

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

If $R \to uAvAw$ is a rule then delete the rule and add $R \to uvAw \mid uAvw \mid uvw$.

If $R \to A$ is a rule then delete the rule and add $R \to \epsilon$ unless $R \to \epsilon$ was already removed.

Step 3: Remove unit rules.

If $A \to B$ is a rule and if $B \to u$ appears

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

If $R \to uAvAw$ is a rule then delete the rule and add $R \to uvAw \mid uAvw \mid uvw$.

If $R \to A$ is a rule then delete the rule and add $R \to \epsilon$ unless $R \to \epsilon$ was already removed.

Step 3: Remove unit rules.

If $A \to B$ is a rule and if $B \to u$ appears

then remove $A \to B$ and add $A \to u$.

Chomsky normal form

Step 1: Add a new start symbol

$S_0 \to S$

Step 2: Remove ϵ rules.

Suppose $A \to \epsilon$ is a rule and A is not the start symbol.

If $R \to uAv$ is a rule then delete the rule and add $R \to uv$ to the rules.

If $R \to uAvAw$ is a rule then delete the rule and add $R \to uvAw \mid uAvw \mid uvw$.

If $R \to A$ is a rule then delete the rule and add $R \to \epsilon$ unless $R \to \epsilon$ was already removed.

Step 3: Remove unit rules.

If $A \to B$ is a rule and if $B \to u$ appears

then remove $A \to B$ and add $A \to u$.

Chomsky normal form

Step 1: Add a new start symbol

Chomsky normal form

Step 1: Add a new start symbol

Step 2: Remove ϵ rules.

Chomsky normal form

Step 1: Add a new start symbol

Step 2: Remove ϵ rules.

Step 3: Remove unit rules.

Chomsky normal form

Step 1: Add a new start symbol

Step 2: Remove ϵ rules.

Step 3: Remove unit rules.

Step 4: Put the rest of the rules in the proper form.

If $A \to u_1 u_2 \ldots u_k$ is a rule

Chomsky normal form

Step 1: Add a new start symbol

Step 2: Remove ϵ rules.

Step 3: Remove unit rules.

Step 4: Put the rest of the rules in the proper form.

If $A \to u_1 u_2 \dots u_k$ is a rule, where $k \geq 3$ and $u_i \in V \cup T$

Chomsky normal form

Step 1: Add a new start symbol

Step 2: Remove ϵ rules.

Step 3: Remove unit rules.

Step 4: Put the rest of the rules in the proper form.

If $A \to u_1 u_2 \dots u_k$ is a rule, where $k \geq 3$ and $u_i \in V \cup T$

Remove this rule and add the following rules:

$A \to u_1 A_1$, $A_1 \to u_2 A_2 \dots A_{k-2} \to u_{k-1} u_k$.

Chomsky normal form

Step 1: Add a new start symbol

Step 2: Remove ϵ rules.

Step 3: Remove unit rules.

Step 4: Put the rest of the rules in the proper form.

If $A \to u_1 u_2 \dots u_k$ is a rule, where $k \geq 3$ and $u_i \in V \cup T$

Remove this rule and add the following rules:

$A \to u_1 A_1$, $A_1 \to u_2 A_2 \dots A_{k-2} \to u_{k-1} u_k$.

If $u_i \in T$, moreover replace each u_i with a variable U_i and add $U_i \to u_i$.

Non context-free languages

- How do we show that there is no PDA accepting L?

Non context-free languages

- How do we show that there is no PDA accepting L?
- Pumping Lemma for CFL!

Non context-free languages

- How do we show that there is no PDA accepting L?
- Pumping Lemma for CFL!
- Intuition: Consider $\{a^n b^n c^n j \mid n \geq 0\}$. Can you accept this with a PDA, i.e., NFA with one stack?

Non context-free languages

- How do we show that there is no PDA accepting L?
- Pumping Lemma for CFL!
- Intuition: Consider $\{a^n b^n c^n j \mid n \geq 0\}$. Can you accept this with a PDA, i.e., NFA with one stack?
- Indeed, this looks like $\{a^n b^n \mid n \geq 0\}$ which we proved cant be accepted by any NFA.

Non context-free languages

- How do we show that there is no PDA accepting L?
- Pumping Lemma for CFL!
- Intuition: Consider $\{a^n b^n c^n j \mid n \geq 0\}$. Can you accept this with a PDA, i.e., NFA with one stack?
- Indeed, this looks like $\{a^n b^n \mid n \geq 0\}$ which we proved cant be accepted by any NFA.
- Remember, pumping lemma for regular languages was a property of regular languages. What can we do for CFLs?

Weakness of CFGs

Let us start by considering CFG in CNF for $L_{a,b} = \{a^n b^n \mid n \geq 1$:

Weakness of CFGs

Let us start by considering CFG in CNF for $L_{a,b} = \{a^n b^n \mid n \geq 1$:

$$S \rightarrow AC|AB, C \rightarrow SB, A \rightarrow a, B \rightarrow b$$

Weakness of CFGs

Let us start by considering CFG in CNF for $L_{a,b} = \{a^n b^n \mid n \geq 1$:

$$S \to AC|AB, C \to SB, A \to a, B \to b$$

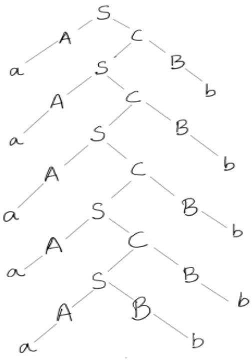
Consider a parse tree for $a^5b^5 \in L_{a,b}$

Weakness of CFGs

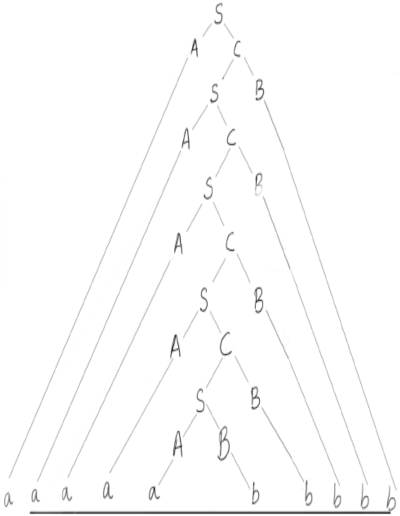
Let us start by considering CFG in CNF for $L_{a,b} = \{a^n b^n \mid n \geq 1$:

$$S \to AC|AB, C \to SB, A \to a, B \to b$$

Consider a parse tree for $a^5 b^5 \in L_{a,b}$



Weakness of CFGs



Pumping Lemma for CFLs