IN2080

5. Context-Free Languages

Einar Broch Johnsen

University of Oslo

28th February 2023



Department of
Informatics



University of
Oslo

Repetition

- We have looked at one of the simpler computational models: **finite automata**

Repetition

- We have looked at one of the simpler computational models: **finite automata**
- Defined (non)deterministic finite automata (NFAs/DFAs) and the languages they accept: **regular languages**

Repetition

- We have looked at one of the simpler computational models: **finite automata**
- Defined (non)deterministic finite automata (NFAs/DFAs) and the languages they accept: **regular languages**

- Defined **regular expressions**, useful as a shorthand for describing languages

Repetition

- We have looked at one of the simpler computational models: **finite automata**
- Defined (non)deterministic finite automata (NFAs/DFAs) and the languages they accept: **regular languages**

- Defined **regular expressions**, useful as a shorthand for describing languages
- A language L is regular $\leftrightarrow$ there exists a regular expression that describes L

Repetition

- We have looked at one of the simpler computational models: **finite automata**
- Defined (non)deterministic finite automata (NFAs/DFAs) and the languages they accept: **regular languages**

- Defined **regular expressions**, useful as a shorthand for describing languages
- A language L is regular $\leftrightarrow$ there exists a regular expression that describes L

- **Pumping lemma**: a useful tool to determine whether a language is *nonregular*

Context-Free Grammars

Today: Context-free grammars and languages

Context-Free Grammars

Today: Context-free grammars and languages

- Grammars describe the *syntax* of a language; they describe the relationship between the different parts, such as the placement of nouns/verbs in sentences

Context-Free Grammars

Today: Context-free grammars and languages

- Grammars describe the *syntax* of a language; they describe the relationship between the different parts, such as the placement of nouns/verbs in sentences
- Useful for programming languages, specifically compilers and parsers:
  if the grammar of a programming language is available, parsing is very straightforward.

Context-Free Grammars

Recall example from last week:

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

Recall example from last week:

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

We used the pumping lemma to show that this language was *not regular*.

Recall example from last week:

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

We used the pumping lemma to show that this language was *not regular*.

$\rightarrow$ First example of a context-free language

Context-Free Grammars

First example: The language L can be described by a context-free grammar:

$$S \to aSb$$
$$S \to \varepsilon$$

Context-Free Grammars

First example: The language L can be described by a context-free grammar:

$$S \to aSb$$
$$S \to \varepsilon$$

- Every grammar consists of *rules*, which are pairs consisting of one variable (to the left of $\to$) and a string of variables and symbols (to the right of $\to$)

Context-Free Grammars

First example: The language L can be described by a context-free grammar:

$$S \to aSb$$
$$S \to \varepsilon$$

- Every grammar consists of *rules*, which are pairs consisting of one variable (to the left of $\to$) and a string of variables and symbols (to the right of $\to$)
- Every grammar contains a *start variable* (above: variable S).
  Common convention: the first listed variable is the start variable
  (if you choose a different start variable, you must specify!).

Context-Free Grammars

First example: The language L can be described by a context-free grammar:

$$S \to aSb$$
$$S \to \varepsilon$$

- Every grammar consists of *rules*, which are pairs consisting of one variable (to the left of $\to$) and a string of variables and symbols (to the right of $\to$)
- Every grammar contains a *start variable* (above: variable S).
  Common convention: the first listed variable is the start variable
  (if you choose a different start variable, you must specify!).
- Words are generated by starting with the start variable and recursively replacing variables with the righthand side of a rule.

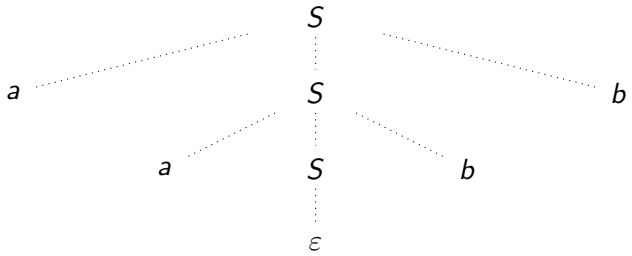
$$S \rightsquigarrow aSb \rightsquigarrow aaSbb \rightsquigarrow aa\varepsilon bb \rightsquigarrow aabb$$

Parse Trees

Derivations of the form

$$S \rightsquigarrow aSb \rightsquigarrow aaSbb \rightsquigarrow aa\varepsilon bb \rightsquigarrow aabb$$

can also be encoded as a parse tree:



Second example:

$$S \to aSa$$
$$S \to bSb$$
$$S \to cSc$$
$$S \to \varepsilon$$

Second example:

$$S \to aSa$$
$$S \to bSb$$
$$S \to cSc$$
$$S \to \varepsilon$$

To simplify notation, you can summarize multiple rules into one line:

$$S \to aSa \mid bSb \mid cSc \mid \varepsilon.$$

Second example:

$$S \to aSa$$
$$S \to bSb$$
$$S \to cSc$$
$$S \to \varepsilon$$

To simplify notation, you can summarize multiple rules into one line:

$$S \to aSa \,|\, bSb \,|\, cSc \,|\, \varepsilon.$$

The symbol | takes on the meaning of "or."

Second example:

$$S \to aSa$$
$$S \to bSb$$
$$S \to cSc$$
$$S \to \varepsilon$$

To simplify notation, you can summarize multiple rules into one line:

$$S \to aSa \mid bSb \mid cSc \mid \varepsilon.$$

The symbol $\mid$ takes on the meaning of "or."

$\to$ This language describes the palindromes of even length over $\{a, b, c\}$.

Context-Free Grammar

Definition (Context-Free Grammar)

A *context-free grammar* is a 4-tuple (V, Σ, R, S) where

1. V is a finite set of *variables*
2. Σ is a finite set disjoint from V of *terminals*
3. R is a finite set of *rules*, where each rule consists of a variable and of a string of variables and terminals, and
4. S is the *start variable*

Context-Free Grammar

Definition (Context-Free Grammar)

A *context-free grammar* is a 4-tuple (V, Σ, R, S) where

1. V is a finite set of *variables*
2. Σ is a finite set disjoint from V of *terminals*
3. R is a finite set of *rules*, where each rule consists of a variable and of a string of variables and terminals, and
4. S is the *start variable*

We call $L(G)$ the language generated by a context-free grammar G.

A language is called a *context-free language* if it is generated by a context-free grammar.

Context-Free Grammar

What can context-free grammars (CFGs) express?

Context-Free Grammar

What can context-free grammars (CFGs) express?

- Regular languages?

Context-Free Grammar

What can context-free grammars (CFGs) express?

- Regular languages?
- Is the class of context-free languages closed under union/intersection/concatenation/complement/Kleene star?

Context-Free Grammar

What can context-free grammars (CFGs) express?

- Regular languages?
- Is the class of context-free languages closed under union/intersection/concatenation/complement/Kleene star?
- Regular languages could be modelled by automata with *finite* memory... what about context-free languages?

Context-Free Grammar

What can context-free grammars (CFGs) express?

- Regular languages?
- Is the class of context-free languages closed under union/intersection/concatenation/complement/Kleene star?
- Regular languages could be modelled by automata with *finite* memory... what about context-free languages?

Answers to these questions over the course of this and the next lecture

RLs and CFLs

Can regular languages be described using context-free grammars?

RLs and CFLs

Can regular languages be described using context-free grammars?

- Given a RL L, there exists some DFA $(Q, \Sigma, \delta, q_0, F)$ that accepts L

RLs and CFLs

Can regular languages be described using context-free grammars?

- Given a RL L, there exists some DFA $(Q, \Sigma, \delta, q_0, F)$ that accepts L
- What if we encode traversing the DFA into grammar rules, i.e., for each transition $\delta(q_1, a) = q_2$ we create a rule $Q_1 \to aQ_2$

RLs and CFLs

Can regular languages be described using context-free grammars?

- Given a RL L, there exists some DFA $(Q, \Sigma, \delta, q_0, F)$ that accepts L
- What if we encode traversing the DFA into grammar rules, i.e., for each transition $\delta(q_1, a) = q_2$ we create a rule $Q_1 \to aQ_2$
- The variables of our grammar correspond to the states in Q, with Q_0 as the start variable.

RLs and CFLs

Can regular languages be described using context-free grammars?

- Given a RL L, there exists some DFA $(Q, \Sigma, \delta, q_0, F)$ that accepts L
- What if we encode traversing the DFA into grammar rules, i.e., for each transition $\delta(q_1, a) = q_2$ we create a rule $Q_1 \to aQ_2$
- The variables of our grammar correspond to the states in Q, with Q_0 as the start variable.
- How do we deal with accept states?

RLs and CFLs

Can regular languages be described using context-free grammars?

- Given a RL L, there exists some DFA $(Q, \Sigma, \delta, q_0, F)$ that accepts L
- What if we encode traversing the DFA into grammar rules, i.e., for each transition $\delta(q_1, a) = q_2$ we create a rule $Q_1 \to aQ_2$
- The variables of our grammar correspond to the states in Q, with Q_0 as the start variable.
- How do we deal with accept states?
  $\rightsquigarrow$ For each $q_i \in F$, add rule $Q_i \to \varepsilon$

RLs and CFLs

Can regular languages be described using context-free grammars?

- Given a RL L, there exists some DFA $(Q, \Sigma, \delta, q_0, F)$ that accepts L
- What if we encode traversing the DFA into grammar rules, i.e., for each transition $\delta(q_1, a) = q_2$ we create a rule $Q_1 \to aQ_2$
- The variables of our grammar correspond to the states in Q, with Q_0 as the start variable.
- How do we deal with accept states?
  $\rightsquigarrow$ For each $q_i \in F$, add rule $Q_i \to \varepsilon$

Theorem

Every regular language is context-free.

Properties of CFLs

Can we build larger CFGs from smaller CFGs?

Properties of CFLs

Can we build larger CFGs from smaller CFGs?

- Closure under union/concatenation/Kleene star?

Properties of CFLs

Can we build larger CFGs from smaller CFGs?

- Closure under union/concatenation/Kleene star?
  $\rightsquigarrow$ Yes, next lecture!

Properties of CFLs

Can we build larger CFGs from smaller CFGs?

- Closure under union/concatenation/Kleene star?
  $\rightsquigarrow$ Yes, next lecture!
- Closure under complement/intersection?

Properties of CFLs

Can we build larger CFGs from smaller CFGs?

- Closure under union/concatenation/Kleene star?

  $\rightsquigarrow$ Yes, next lecture!

- Closure under complement/intersection?

  $\rightsquigarrow$ No, but we need to know more before we can determine if a language is not context-free...

Ambiguity

- Consider the grammar

$$E \to E + E \mid E \times E \mid (E) \mid a$$

Ambiguity

- Consider the grammar
$$E \to E + E \mid E \times E \mid (E) \mid a$$
- Here: the alphabet is $\{a, +, \times, (,)\}$.

Ambiguity

- Consider the grammar
$$E \to E + E \mid E \times E \mid (E) \mid a$$
- Here: the alphabet is $\{a, +, \times, (,)\}$.

  $\to$ The grammar describes arithmetic expressions over a

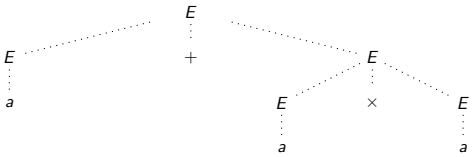
Ambiguity

- Consider the grammar
$$E \to E + E \mid E \times E \mid (E) \mid a$$
- Here: the alphabet is $\{a, +, \times, (,)\}$.

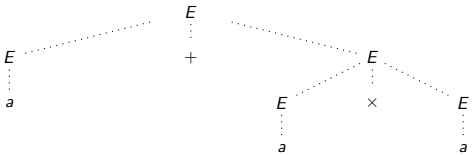
  $\to$ The grammar describes arithmetic expressions over a

- What does the parse tree for the string $a + a \times a$ look like?

Ambiguity

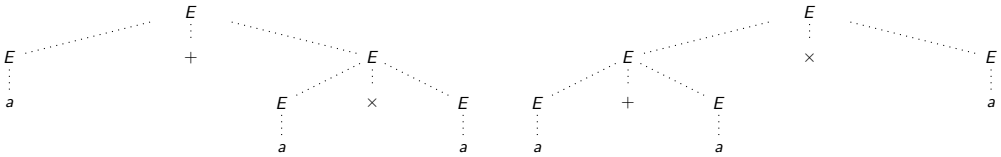


Ambiguity



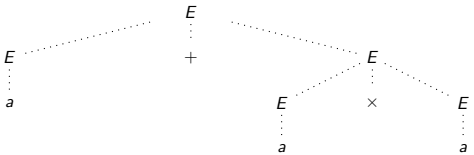
Intuitively corresponds to $a + (a \times a)$

Ambiguity

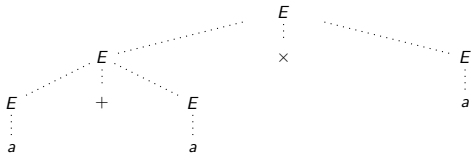


Intuitively corresponds to $a + (a \times a)$

Ambiguity

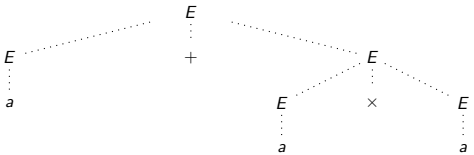


Intuitively corresponds to $a + (a \times a)$

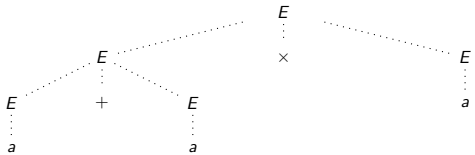


Intuitively corresponds to $(a + a) \times a$

Ambiguity



Intuitively corresponds to $a + (a \times a)$



Intuitively corresponds to $(a + a) \times a$

This is called *ambiguity*

Ambiguity

- . . . just having multiple possible derivations does not mean that a grammar is ambiguous.

Ambiguity

- . . . just having multiple possible derivations does not mean that a grammar is ambiguous.
- Two derivations could look different, yet "structurally" the same: apply the same rules to the same variables, yet in a different order.

Ambiguity

- . . . just having multiple possible derivations does not mean that a grammar is ambiguous.
- Two derivations could look different, yet "structurally" the same: apply the same rules to the same variables, yet in a different order.

$$E \rightsquigarrow E + E \rightsquigarrow E + E \times E \rightsquigarrow a + E \times E \rightsquigarrow a + a \times E \rightsquigarrow a + a \times a$$

$$E \rightsquigarrow E + E \rightsquigarrow a + E \rightsquigarrow a + E \times E \rightsquigarrow a + a \times E \rightsquigarrow a + a \times a$$

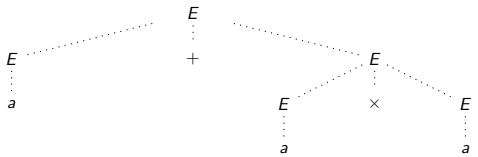
Ambiguity

- . . . just having multiple possible derivations does not mean that a grammar is ambiguous.
- Two derivations could look different, yet "structurally" the same: apply the same rules to the same variables, yet in a different order.

$$E \rightsquigarrow E + E \rightsquigarrow E + E \times E \rightsquigarrow a + E \times E \rightsquigarrow a + a \times E \rightsquigarrow a + a \times a$$

$$E \rightsquigarrow E + E \rightsquigarrow a + E \rightsquigarrow a + E \times E \rightsquigarrow a + a \times E \rightsquigarrow a + a \times a$$

- Both have the same parse tree!



Ambiguity

- We are interested in structurally different derivations, i.e., two derivations of the same word that, given a predefined order of derivation, are different

Ambiguity

- We are interested in structurally different derivations, i.e., two derivations of the same word that, given a predefined order of derivation, are different

Definition

A *leftmost derivation* of a string replaces, in each derivation step, the leftmost variable. Then a string is derived *ambiguously* over a grammar G if it has two or more *leftmost derivations* over G.

Ambiguity

- We are interested in structurally different derivations, i.e., two derivations of the same word that, given a predefined order of derivation, are different

Definition

A *leftmost derivation* of a string replaces, in each derivation step, the leftmost variable. Then a string is derived *ambiguously* over a grammar G if it has two or more *leftmost derivations* over G.

- If $L(G)$ contains a string that is derived ambiguously, we say that G is ambiguous.

Ambiguity

- We are interested in structurally different derivations, i.e., two derivations of the same word that, given a predefined order of derivation, are different

Definition

A *leftmost derivation* of a string replaces, in each derivation step, the leftmost variable. Then a string is derived *ambiguously* over a grammar G if it has two or more *leftmost derivations* over G.

- If $L(G)$ contains a string that is derived ambiguously, we say that G is ambiguous.
- Remember we had two parse trees for $a + a \times a$ generated from

$$E \to E + E \mid E \times E \mid (E) \mid a$$

Are they both leftmost derivations?

Chomsky Normal Form

- Context-free languages have a nice property:

  Every CFL can be described by a CFG in *Chomsky Normal Form*:

Chomsky Normal Form

- Context-free languages have a nice property:

  Every CFL can be described by a CFG in *Chomsky Normal Form*:

Definition

A grammar is in *Chomsky Normal Form* if every rule is of the form:

$$A \to BC$$
$$A \to a$$

where a is any terminal, A is any variable, B, C are any variables except the start variable S. In addition the rule $S \to \varepsilon$ is permitted.

Definition

A grammar is in *Chomsky Normal Form* if every rule is of the form:

$$A \to BC$$
$$A \to a$$

where a is any terminal, A is any variable, B, C are any variables except the start variable S. In addition the rule $S \to \varepsilon$ is permitted.

Proof sketch: Given an arbitrary grammar G.

- First, add new start variable S_0 and new rule $S_0 \to S$ to G.

Definition

A grammar is in *Chomsky Normal Form* if every rule is of the form:

$$A \to BC$$
$$A \to a$$

where a is any terminal, A is any variable, B, C are any variables except the start variable S. In addition the rule $S \to \varepsilon$ is permitted.

Proof sketch: Given an arbitrary grammar G.

- First, add new start variable S_0 and new rule $S_0 \to S$ to G.
- Then, remove all rules $A \to \varepsilon$, followed by all "unit" rules $A \to B$.

Definition

A grammar is in *Chomsky Normal Form* if every rule is of the form:

$$A \to BC$$
$$A \to a$$

where a is any terminal, A is any variable, B, C are any variables except the start variable S. In addition the rule $S \to \varepsilon$ is permitted.

Proof sketch: Given an arbitrary grammar G.

- First, add new start variable S_0 and new rule $S_0 \to S$ to G.
- Then, remove all rules $A \to \varepsilon$, followed by all "unit" rules $A \to B$.
  For each such occurence of A in the righthand side of a rule, add a new rule with ε (resp. B) substituted for A (see examples on next slide).

Definition

A grammar is in *Chomsky Normal Form* if every rule is of the form:

$$A \to BC$$
$$A \to a$$

where a is any terminal, A is any variable, B, C are any variables except the start variable S. In addition the rule $S \to \varepsilon$ is permitted.

Proof sketch: Given an arbitrary grammar G.

- First, add new start variable S_0 and new rule $S_0 \to S$ to G.
- Then, remove all rules $A \to \varepsilon$, followed by all "unit" rules $A \to B$.
  For each such occurence of A in the righthand side of a rule, add a new rule with ε (resp. B) substituted for A (see examples on next slide).
- Finally, split all rules with more than 3 righthandside symbols into multiple rules containing only 2 symbols.

CNF - Example

Grammar:

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

First, add a new start variable:

CNF - Example

Grammar:

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

First, add a new start variable:

$$S_0 \to S$$
$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \\
A &\to B \mid S \\
B &\to b \mid \varepsilon
\end{aligned}
$$

Then, remove $B \to \varepsilon$:

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \\
A &\to B \mid S \\
B &\to b \mid \varepsilon
\end{aligned}
$$

Then, remove $B \to \varepsilon$:

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \\
A &\to B \mid \varepsilon \mid S \\
B &\to b
\end{aligned}
$$

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \\
A &\to B \mid \varepsilon \mid S \\
B &\to b
\end{aligned}
$$

Then, remove $A \to \varepsilon$:

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid aB \mid a \\
A &\to B \mid \varepsilon \mid S \\
B &\to b
\end{aligned}
$$

Then, remove $A \to \varepsilon$:

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid SA \mid AS \mid S \mid aB \mid a \\
A &\to S \mid B \\
B &\to b
\end{aligned}
$$

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid SA \mid AS \mid S \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

Then remove $S \to S$:

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid SA \mid AS \mid S \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

Then remove $S \to S$:

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

Remove unit rule $S_0 \to S$:

CNF - Example

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

Remove unit rule $S_0 \to S$:

$$
\begin{aligned}
S_0 &\to ASA \mid SA \mid AS \mid aB \mid a \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

CNF - Example

$$
\begin{aligned}
S_0 &\to ASA \mid SA \mid AS \mid aB \mid a \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

and continue to remove the unit rules $A \to S$, etc....

CNF - Example

$$
\begin{aligned}
S_0 &\to ASA \mid SA \mid AS \mid aB \mid a \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

and continue to remove the unit rules $A \to S$, etc....

But how to convert, say, $S \to ASA$ into rules with only two symbols on the right?

CNF - Example

$$
\begin{aligned}
S_0 &\to ASA \mid SA \mid AS \mid aB \mid a \\
S &\to ASA \mid SA \mid AS \mid aB \mid a \\
A &\to B \mid S \\
B &\to b
\end{aligned}
$$

and continue to remove the unit rules $A \to S$, etc....

But how to convert, say, $S \to ASA$ into rules with only two symbols on the right?
$\rightsquigarrow$ Introduce help variables!

$$
\begin{aligned}
& S \to ASA \\
\rightsquigarrow\ & S \to AA_1, A_1 \to SA
\end{aligned}
$$

Properties of CFLs

Context-free languages: Closure under union/concatenation/Kleene star?

Properties of CFLs

Context-free languages: Closure under union/concatenation/Kleene star?

Let the grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ generate L_1, L_2 respectively.

Properties of CFLs

Context-free languages: Closure under union/concatenation/Kleene star?

Let the grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ generate L_1, L_2 respectively.

Union:

- Create grammar $G_{L_1 \cup L_2} = (V, \Sigma, R, S)$ that generates all words $w \in L_1 \cup L_2$.

Properties of CFLs

Context-free languages: Closure under union/concatenation/Kleene star?

Let the grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ generate L_1, L_2 respectively.

Union:

- Create grammar $G_{L_1 \cup L_2} = (V, \Sigma, R, S)$ that generates all words $w \in L_1 \cup L_2$.
- Create new start variable S.
- $V = V_1 \cup V_2 \cup \{S\}$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$, and
- $R = R_1 \cup R_2 \cup \{S \to S_1 \mid S_2\}$.

CFL Union: Example

$$S_1 \to aS_1b \mid \varepsilon \qquad \cup \qquad S_2 \to aS_2a \mid bS_2b \mid cS_2c \mid \varepsilon$$

CFL Union: Example

$S_1 \to aS_1b \mid \varepsilon \quad \cup \quad S_2 \to aS_2a \mid bS_2b \mid cS_2c \mid \varepsilon$

$$S \to S_1 \mid S_2$$
$$S_1 \to aS_1b \mid \varepsilon$$
$$S_2 \to aS_2a \mid bS_2b \mid cS_2c \mid \varepsilon$$

Properties of CFLs: Concatenation

Let the grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ generate L_1, L_2 respectively.

Concatenation:

- Create grammar $G_{L_1 L_2} = (V, \Sigma, R, S)$ that accepts all words $w = w_1 w_2$, where $w_1 \in L_1$ and $w_2 \in L_2$.

Properties of CFLs: Concatenation

Let the grammars $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ generate L_1, L_2 respectively.

Concatenation:

- Create grammar $G_{L_1 L_2} = (V, \Sigma, R, S)$ that accepts all words $w = w_1 w_2$, where $w_1 \in L_1$ and $w_2 \in L_2$.
- New start variable S
- $V = V_1 \cup V_2 \cup \{S\}$,
- $\Sigma = \Sigma_1 \cup \Sigma_2$, and
- $R = R_1 \cup R_2 \cup \{S \to S_1 S_2\}$.

CFL Concatenation: Example

$S_1 \to aS_1b \mid \varepsilon$

$S_2 \to aS_2a \mid bS_2b \mid cS_2c \mid \varepsilon$

CFL Concatenation: Example

$S_1 \to aS_1b \mid \varepsilon$

$S_2 \to aS_2a \mid bS_2b \mid cS_2c \mid \varepsilon$

$$S \to S_1S_2$$
$$S_1 \to aS_1b \mid \varepsilon$$
$$S_2 \to aS_2a \mid bS_2b \mid cS_2c \mid \varepsilon$$

Properties of CFLs: Kleene star

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ generate language L_1.

Kleene star:

- Create grammar $G = (V, \Sigma, R, S)$ that generates all words in L_1^*.

Properties of CFLs: Kleene star

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ generate language L_1.

Kleene star:

- Create grammar $G = (V, \Sigma, R, S)$ that generates all words in L_1^*.
- New start variable S

Properties of CFLs: Kleene star

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ generate language L_1.

Kleene star:

- Create grammar $G = (V, \Sigma, R, S)$ that generates all words in L_1^*.
- New start variable S
- $V = V_1 \cup \{S\}$,
- $\Sigma = \Sigma_1$,
- $R = R_1 \cup \{S \to \varepsilon, S \to S_1 S\}$,

Properties of CFLs: Kleene star

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ generate language L_1.

Kleene star:

- Create grammar $G = (V, \Sigma, R, S)$ that generates all words in L_1^*.
- New start variable S
- $V = V_1 \cup \{S\}$,
- $\Sigma = \Sigma_1$,
- $R = R_1 \cup \{S \to \varepsilon, S \to S_1 S\}$,

Example:

$$S_1 \to aS_1b \mid \varepsilon$$

$$S \to \varepsilon \mid S_1 S$$
$$S_1 \to aS_1b \mid \varepsilon$$

Properties of CFLs

Are CFLs closed under complement/intersection?

Properties of CFLs

Are CFLs closed under complement/intersection?

$\rightsquigarrow$ No, but we need to know more before we can determine if a language is not context-free (more soon!)

CNF

- Thus, we see how all CFGs can be converted to CFGs in CNF.

CNF

- Thus, we see how all CFGs can be converted to CFGs in CNF.
- Useful property to have, both for practical purposes and theoretical work: knowing what the grammar looks like can be very beneficial (we will see an example next week)

CNF

- Thus, we see how all CFGs can be converted to CFGs in CNF.
- Useful property to have, both for practical purposes and theoretical work: knowing what the grammar looks like can be very beneficial (we will see an example next week)
- **Next week:**

CNF

- Thus, we see how all CFGs can be converted to CFGs in CNF.
- Useful property to have, both for practical purposes and theoretical work: knowing what the grammar looks like can be very beneficial (we will see an example next week)
- **Next week:**
  - How can finite automata be enriched to accept context-free languages?