INF2080

Context-Free Langugaes

Daniel Lupp

Universitetet i Oslo

1st February 2016

Department of Informatics

University of Oslo

Repetition

- We've looked at one of the simpler computational models: finite automata

Repetition

- We've 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've 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've 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've 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 as a useful tool for determining 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 try to describe the relationship of all the parts to one another, such as placement of nouns/verbs in sentences

Context-Free Grammars

Today: Context-free grammars and languages

- grammars describe the *syntax* of a language; they try to describe the relationship of all the parts to one another, such as 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:

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

Context-Free Grammars

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.

Context-Free Grammars

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:

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

Context-Free Grammars

First example:

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

- Every grammar consists of *rules*, which are a pair 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:

$$S \rightarrow aSb$$
$$S \rightarrow \varepsilon$$

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

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

- Every grammar consists of *rules*, which are a pair 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.

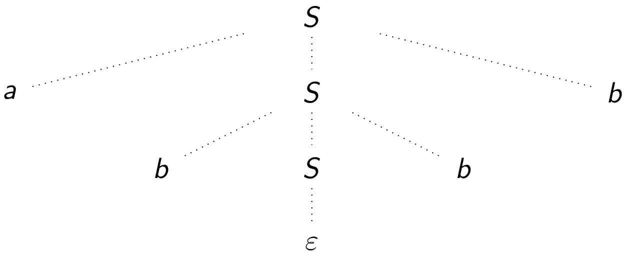
$$S \rightsquigarrow aSb \rightsquigarrow aaSbb \rightsquigarrow aa\varepsilon bb \rightsquigarrow aabb$$

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:



Context-Free Grammars

Second example:

$$
\begin{aligned}
S &\rightarrow aSa \\
S &\rightarrow bSb \\
S &\rightarrow cSc \\
S &\rightarrow \varepsilon
\end{aligned}
$$

Context-Free Grammars

Second example:

$$
\begin{aligned}
S &\rightarrow aSa\\
S &\rightarrow bSb\\
S &\rightarrow cSc\\
S &\rightarrow \varepsilon
\end{aligned}
$$

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

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

Second example:

$$\begin{aligned} S &\rightarrow aSa \\ S &\rightarrow bSb \\ S &\rightarrow cSc \\ S &\rightarrow \varepsilon \end{aligned}$$

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

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

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

Context-Free Grammars

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.”
$\to$ 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*, each consisting of a variable and of a string of variables and terminals
4. and 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*, each consisting of a variable and of a string of variables and terminals
4. and S is the *start variable*

We call $L(G)$ the language generated by a context-free grammar. A language is called a *context-free language* if it is generated by a context-free grammar.

So what can context-free grammars (CFGs) express?

Context-Free Grammar

So what can context-free grammars (CFGs) express?

- Regular languages?

Context-Free Grammar

So what can context-free grammars (CFGs) express?

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

Context-Free Grammar

So what can context-free grammars (CFGs) express?

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

Context-Free Grammar

So what can context-free grammars (CFGs) express?

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

Answers to these over the course of this and next lecture (and group sessions)

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 $Q_1 \rightarrow aQ_2$
- 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 $Q_1 \rightarrow aQ_2$
- 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 $Q_1 \rightarrow aQ_2$
- 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 $Q_i \rightarrow \varepsilon$

Theorem

Every regular language is context-free.

Properties of CFLs

Closure under union/concatanation/Kleene star?

Properties of CFLs

Closure under union/concatanation/Kleene star?
$\rightsquigarrow$ Yes, group sessions!

Properties of CFLs

Closure under union/concatanation/Kleene star?

$\rightsquigarrow$ Yes, group sessions!

Closure under complement/intersection?

Properties of CFLs

Closure under union/concatanation/Kleene star?
$\rightsquigarrow$ Yes, group sessions!
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

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

- 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 $\{a, +, \times, (,)\}$.
  $\to$ arithmetic expressions over a

Ambiguity

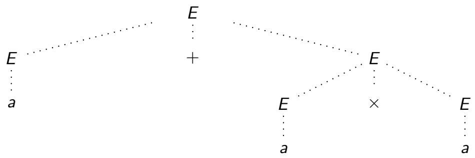
- Consider the grammar

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

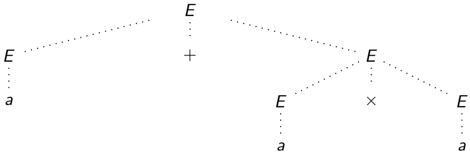
- Here: the alphabet is $\{a, +, \times, (,)\}$.
  $\to$ arithmetic expressions over a

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

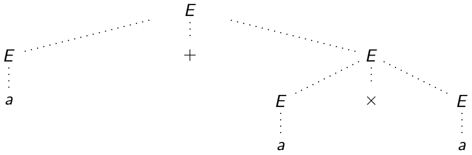
Ambiguity

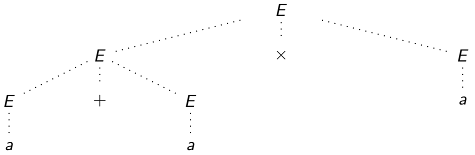


Ambiguity

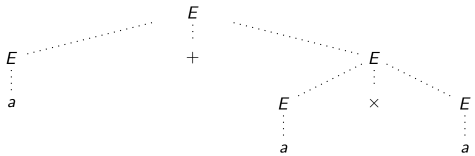


Intuitively corresponds to $a + (a \times a)$

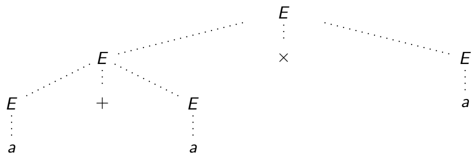




Intuitively corresponds to $a + (a \times a)$

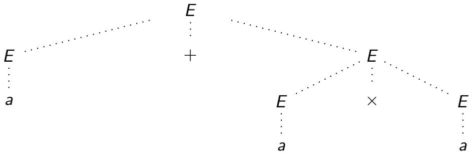


Intuitively corresponds to $a + (a \times a)$

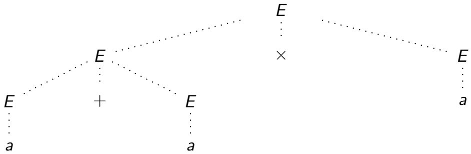


Intuitively corresponds to $(a + a) \times a$

Ambiguity



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



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

This is called *ambiguity*

Ambiguity

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

Ambiguity

- But 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

- But 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.
- 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

- But 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.
- 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

- But 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.
- 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.

Chomsy 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 that are not the start variable. 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 that are not the start variable. 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 that are not the start variable. 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:

$$A \rightarrow BC$$
$$A \rightarrow a$$

where a is any terminal, A is any variable, B, C are any variables that are not the start variable. In addition the rule $S \rightarrow \varepsilon$ is permitted.

Proof sketch: Given an arbitrary grammar G. First, add new start variable S_0 and new rule $S_0 \rightarrow S$ to G.Then, remove all rules $A \rightarrow \varepsilon$, followed by all "unit" rules $A \rightarrow B$. 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 that are not the start variable. 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 new start variable:

CNF - Example

Grammar;

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

First, add new start variable:

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

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

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

Then, remove $B \rightarrow \varepsilon$:

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

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

$$S_0 \to S$$
$$S \to ASA \mid SA \mid AS \mid S \mid aB \mid a$$
$$A \to B \mid S$$
$$B \to b$$

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

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

Remove unit rule $S_0 \rightarrow S$:

CNF - Example

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

Remove unit rule $S_0 \rightarrow S$:

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

CNF - Example

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

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

CNF - Example

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

and you would continue to remove the unit rules $A \rightarrow S$, etc....But how to convert, say, $S \rightarrow ASA$ into rules with only two symbols on the right?

CNF - Example

$$S_0 \to ASA \mid SA \mid AS \mid aB \mid a$$
$$S \to ASA \mid SA \mid AS \mid aB \mid a$$
$$A \to B \mid S$$
$$B \to b$$

and you would 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!

$$S \to ASA$$
$$\rightsquigarrow S \to AA_1, A_1 \to SA$$

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 time: how can finite automata be enriched so as to accept context-free languages?