INF2080

Context-Free Langugaes

Daniel Lupp

Universitetet i Oslo

1st February 2018

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*

Pumping Lemma revisited

Recall example from last week:

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

Pumping Lemma revisited

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.

Pumping Lemma revisited

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.
What about the following language, for $\Sigma = \{a, b, c\}$:

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

Pumping Lemma revisited

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.
What about the following language, for $\Sigma = \{a, b, c\}$:

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Union of two languages:
- first language: all words of the form $ab^n c^n$

Pumping Lemma revisited

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.
What about the following language, for $\Sigma = \{a, b, c\}$:

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Union of two languages:
- first language: all words of the form $ab^n c^n$
- second language: all Σ^* words that start with either 0 or 2 or more a's.
  $\rightarrow$ L is a disjoint union

Pumping Lemma revisited

Lemma (Pumping Lemma)

If A is a regular language, then there is a number p, called the pumping length, where if w is a word in A of length $\geq p$ then w can be divided into three parts, $w = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

Does L satisfy the pumping lemma?

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$, $|xy| \leq p$.

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Let p be the pumping length.

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$, $|xy| \leq p$.

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Let p be the pumping length.
- Each $w \in L$ is either of the form ab^nc^n or a^kw.

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^i z \in A$ *for every* $i \geq 0$,
2. $|y| > 0$, $|xy| \leq p$.

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = ab^n c^n$, where n is such that $|s| \geq p$.

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$, $|xy| \leq p$.

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = ab^nc^n$, where n is such that $|s| \geq p$.
- choose $x = \varepsilon, y = a, z = b^nc^n$. Then $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$, $|xy| \leq p$.

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = ab^nc^n$, where n is such that $|s| \geq p$.
- choose $x = \varepsilon, y = a, z = b^nc^n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string $xz = b^nc^n = a^0b^nc^n$ is of the form a^kw for $k \neq 1$ and $w \in \Sigma^*$ not starting with a.

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$, $|xy| \leq p$.

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = ab^nc^n$, where n is such that $|s| \geq p$.
- choose $x = \varepsilon, y = a, z = b^nc^n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string $xz = b^nc^n = a^0b^nc^n$ is of the form a^kw for $k \neq 1$ and $w \in \Sigma^*$ not starting with a. $\Rightarrow xz \in L$.

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^i z \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = ab^n c^n$, where n is such that $|s| \geq p$.
- choose $x = \varepsilon, y = a, z = b^n c^n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string $xy^i z = b^n c^n = a^i b^n c^n$ for $i \geq 2$ is of the form $a^k w$ for $k \neq 1$ and $w \in \Sigma^*$ not starting with a.

Pumping Lemma revisited

Lemma (Pumping Lemma, shortened)

If A is a regular, $|w| \geq p$ can be divided into three parts, $w = xyz$, such that

1. $xy^iz \in A$ *for every* $i \geq 0$,
2. $|y| > 0$,
3. $|xy| \leq p$.

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = ab^nc^n$, where n is such that $|s| \geq p$.
- choose $x = \varepsilon, y = a, z = b^nc^n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string $xy^iz = b^nc^n = a^ib^nc^n$ for $i \geq 2$ is of the form a^kw for $k \neq 1$ and $w \in \Sigma^*$ not starting with a. $\Rightarrow xy^iz \in L$.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 0$, choose $x = \varepsilon, y = w_1, z = w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 0$, choose $x = \varepsilon, y = w_1, z = w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The strings xz and $xy^i z$ for $i > 2$ are in Σ^* and don't start with a

Pumping Lemma revisited

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 0$, choose $x = \varepsilon, y = w_1, z = w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The strings xz and xy^iz for $i > 2$ are in Σ^* and don't start with a $\Rightarrow xz, xy^iz \in L$.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 2$, choose $x = \varepsilon, y = aa, z = w_1 w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma revisited

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 2$, choose $x = \varepsilon, y = aa, z = w_1 w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is in Σ^* and doesn't start with a.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 2$, choose $x = \varepsilon, y = aa, z = w_1 w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is in Σ^* and doesn't start with a. $xz \in L$

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 2$, choose $x = \varepsilon, y = aa, z = w_1 w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is in Σ^* and doesn't start with a. $xz \in L$
- The string $xy^i z$ for $i \geq 1$ starts with 2 or more a's, followed by a word $w \in \Sigma^*$ that does not start with an a.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k = 2$, choose $x = \varepsilon, y = aa, z = w_1 w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is in Σ^* and doesn't start with a. $xz \in L$
- The string $xy^i z$ for $i \geq 1$ starts with 2 or more a's, followed by a word $w \in \Sigma^*$ that does not start with an a. $\Rightarrow xy^i z \in L$.

Pumping Lemma revisited

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k \geq 3$, choose $x = \varepsilon, y = a, z = a^{k-1} w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k \geq 3$, choose $x = \varepsilon, y = a, z = a^{k-1} w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is of the form $a^{k-1}w$, where $w \in \Sigma^*$ and doesn't start with a.

Pumping Lemma revisited

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k \geq 3$, choose $x = \varepsilon, y = a, z = a^{k-1} w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is of the form $a^{k-1}w$, where $w \in \Sigma^*$ and doesn't start with a. $xz \in L$

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k \geq 3$, choose $x = \varepsilon, y = a, z = a^{k-1} w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is of the form $a^{k-1}w$, where $w \in \Sigma^*$ and doesn't start with a. $xz \in L$
- The string $xy^i z$ for $i \geq 1$ is of the form $a^{k+i-1}w$ where $w \in \Sigma^*$ that does not start with an a.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

- Assume $s = a^k w_1 w_2 \cdots w_n$, for $k \neq 1$ and $w \in \Sigma^*$ not starting with a, where n, k are such that $|s| \geq p$.
- if $k \geq 3$, choose $x = \varepsilon, y = a, z = a^{k-1} w_2 \cdots w_n$. Then $|y| > 0$ and $|xy| \leq p$.
- The string xz is of the form $a^{k-1} w$, where $w \in \Sigma^*$ and doesn't start with a. $xz \in L$
- The string $xy^i z$ for $i \geq 1$ is of the form $a^{k+i-1} w$ where $w \in \Sigma^*$ that does not start with an a. $\Rightarrow xy^i z \in L$.

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

can be pumped!! Does that mean L is regular?

Pumping Lemma revisited

$$L = \{ab^nc^n \mid n \geq 0\} \cup \{a^kw \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

can be pumped!! Does that mean L is regular?

- If L is regular, then so is $L \cap ab\Sigma^*$ (recall: regular languages are closed under intersection).

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

can be pumped!! Does that mean L is regular?

- If L is regular, then so is $L \cap ab\Sigma^*$ (recall: regular languages are closed under intersection).
- $L \cap ab\Sigma^* = \{ab^n c^n \mid n \geq 1\}$

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

can be pumped!! Does that mean L is regular?

- If L is regular, then so is $L \cap ab\Sigma^*$ (recall: regular languages are closed under intersection).
- $L \cap ab\Sigma^* = \{ab^n c^n \mid n \geq 1\}$
- Exercise: show that this language is nonregular! (analogous to proof for $a^n b^n$)
- So L is nonregular...is this a counter-example to the pumping lemma?

Pumping Lemma revisited

$$L = \{ab^n c^n \mid n \geq 0\} \cup \{a^k w \mid k \neq 1, \text{ and } w \in \Sigma^* \text{ doesn't start with } a\}$$

can be pumped!! Does that mean L is regular?

- If L is regular, then so is $L \cap ab\Sigma^*$ (recall: regular languages are closed under intersection).
- $L \cap ab\Sigma^* = \{ab^n c^n \mid n \geq 1\}$
- Exercise: show that this language is nonregular! (analogous to proof for $a^n b^n$)
- So L is nonregular...is this a counter-example to the pumping lemma? No, pumping lemma is not an if and only if statement!

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

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:

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

- 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!).

Context-Free Grammars

First example:

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

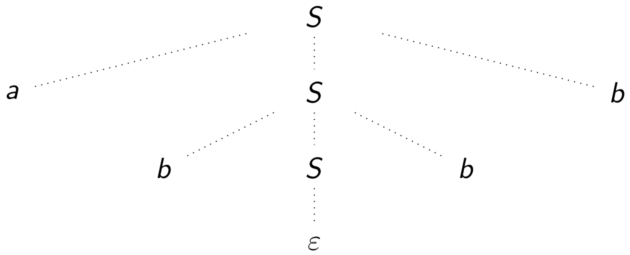
- Every grammar consists of *rules*, which are a pair consisting of one variable (to the left of $\to$) and a string of variables and symbols (to the right of $\to$)
- Every grammar contains a *start variable* (above: variable S). Common convention: the first listed variable is the start variable (if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

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

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:

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

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

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

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:

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

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.

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 $Q_1 \rightarrow aQ_2$

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 $Q_1 \rightarrow aQ_2$
- 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? Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ be two grammars that generate L_1, L_2 respectively.

Properties of CFLs

Closure under union/concatanation/Kleene star? Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ be two grammars that generate L_1, L_2 respectively.
Union:

- create grammar $G_{L_1 \cup L_2}$ that generates all words $w \in L_1 \cup L_2$.

Properties of CFLs

Closure under union/concatanation/Kleene star? Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ be two grammars that generate L_1, L_2 respectively.
Union:

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

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

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

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

Properties of CFLs: Concatanation

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

- 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 $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and $G_2 = (V_2, \Sigma_2, R_2, S_2)$ be two grammars that 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 Concatanation: Example

$S_1 \to aS_1b \mid \varepsilon$

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

CFL Concatenation: Example

$S_1 \rightarrow aS_1b \mid \varepsilon$

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

$$
\begin{aligned}
S &\rightarrow S_1S_2 \\
S_1 &\rightarrow aS_1b \mid \varepsilon \\
S_2 &\rightarrow aS_2a \mid bS_2b \mid cS_2c \mid \varepsilon
\end{aligned}
$$

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^*.
- $V = V_1$,
- $\Sigma = \Sigma_1$,
- $R = R_1 \cup \{S_1 \rightarrow \varepsilon, S_1 \rightarrow S_1 S_1\}$,
- $S = S_1$.

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^*.
- $V = V_1$,
- $\Sigma = \Sigma_1$,
- $R = R_1 \cup \{S_1 \to \varepsilon, S_1 \to S_1 S_1\}$,
- $S = S_1$.

Example:

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

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

Properties of CFLs

Closure under complement/intersection?

Properties of CFLs

Closure under complement/intersection?

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

(next week)

Ambiguity

- Consider the grammar

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

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

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

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

Ambiguity

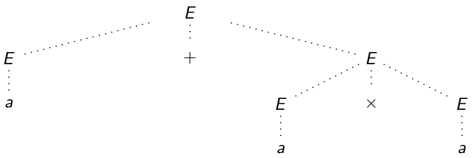
- Consider the grammar

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

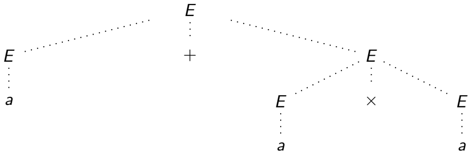
- Here: the alphabet is $\{a, +, \times, (,)\}$.
  $\to$ arithmetic expressions over a

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

Ambiguity

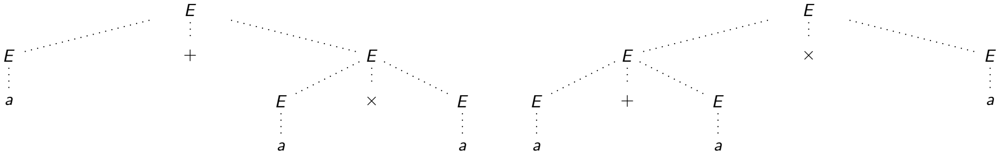


Ambiguity



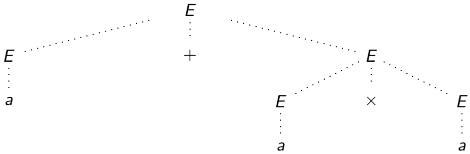
Intuitively corresponds to $a + (a \times a)$

Ambiguity

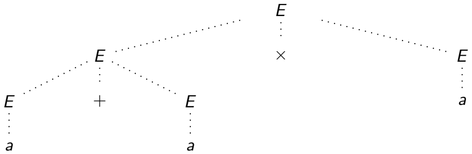


Intuitively corresponds to $a + (a \times a)$

Ambiguity

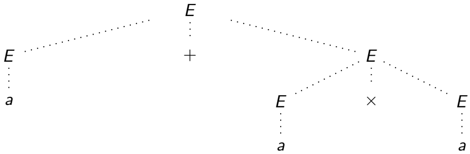


Intuitively corresponds to $a + (a \times a)$

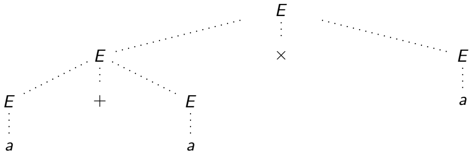


Intuitively corresponds to $(a + a) \times a$

Ambiguity



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



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

This is called *ambiguity*

Ambiguity

- 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.

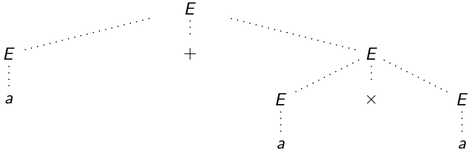
$E \rightsquigarrow E + E \rightsquigarrow E + E \times E \rightsquigarrow a + E \times E \rightsquigarrow a + a \times E \rightsquigarrow a + a \times a$

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

- 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.

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

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.

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:

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

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.

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:

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

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).

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;

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

First, add new start variable:

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

CNF - Example

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

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

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

Then, remove $A \to \varepsilon$:

CNF - Example

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

Then, remove $A \to \varepsilon$:

$$S_0 \to S$$
$$S \to ASA \mid {\color{blue}SA} \mid {\color{blue}AS} \mid {\color{blue}S} \mid aB \mid a$$
$$A \to S \mid B$$
$$B \to b$$

CNF - Example

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

Then remove $S \rightarrow S$:

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 $S \to S$:

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

CNF - Example

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

Remove unit rule $S_0 \to S$:

CNF - Example

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

Remove unit rule $S_0 \rightarrow S$:

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

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

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

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

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

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? $\rightsquigarrow$ introduce help variables!

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

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

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)
- how can finite automata be enriched so as to accept context-free languages? $\rightarrow$ next week!