Parsing

Context-Free Grammars (CFG)

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2019

Table of contents

Grammar Formalisms (again)

Type 1/2/3 grammars

A type 0 grammar is called

- **context-sensitive** (or of **type 1**) if for all productions $\alpha \to \beta$, $|\alpha| \leq |\beta|$ holds. The only exception is $S \to \varepsilon$ which is allowed if S does not appear in any righthand side.
- **context-free** (or of **type 2**) if for all productions $\alpha \to \beta$, $\alpha \in N$.
- **regular** (or of **type 3**) if for all productions $\alpha \to \beta$, $\alpha \in N$ and $\beta \in T^*$ or $\beta = \beta' X$ with $\beta' \in T^*$, $X \in N$.

The type 1/2/3 languages are the languages generated by the corresponding grammars.

The hierarchy of the type 0, 1, 2 and 3 languages is called the **Chomsky Hierarchy**.

Grammar Formalisms (again)

Type 3 grammar

Grammar for L(das (rote|grüne)* Auto (von Otto| ε))

NP $\rightarrow$ Det N1 N1 $\rightarrow$ rote N1 N1 $\rightarrow$ grüne N1 N1 $\rightarrow$ Auto
N1 $\rightarrow$ Auto PP PP $\rightarrow$ von N2 N2 $\rightarrow$ Otto

Type 2 grammar

Grammar for $\{a^n b^m (cd)^n d \mid n, m \geq 0\}$

$\mathrm{S} \rightarrow \mathrm{T}\, d$ $\quad \mathrm{T} \rightarrow a\, \mathrm{T}\, cd$ $\quad \mathrm{T} \rightarrow \mathrm{U}$ $\quad \mathrm{U} \rightarrow b\, \mathrm{U}$ $\quad \mathrm{U} \rightarrow \varepsilon$

Type 1 grammar

Grammar for $\{a^n b^n c^n \mid n \geq 1\}$

$\mathrm{S} \rightarrow \mathrm{T}\,\mathrm{E}$ $\quad \mathrm{T} \rightarrow a\,\mathrm{T}\,\mathrm{B}$ $\quad \mathrm{T} \rightarrow \varepsilon$ $\quad \mathrm{B} \rightarrow b\,\mathrm{C}$
$\mathrm{C}\, b \rightarrow b\,\mathrm{C}$ $\quad \mathrm{C}\,\mathrm{E} \rightarrow \mathrm{E}\, c$ $\quad b\,\mathrm{E} \rightarrow b$

CFG

CFG

A **context-free grammar** (CFG) is a tuple $G = \langle N, T, P, S \rangle$ such that:

- N and T are disjoint alphabets, the nonterminals and terminals
- $S \in N$ is the start symbol
- P is a set of productions of the form $A \to \beta$ with $A \in N, \beta \in (N \cup T)^*$

Any $\beta \in (N \cup T)^*$ with $S \stackrel{*}{\Rightarrow} \beta$ is called a **sentential form**.

CFG

CFG parse tree

A tree t is a **parse tree** for a CFG $G = \langle N, T, P, S\rangle$ iff

- each node in t is labeled with an $\alpha \in N \cup T \cup \{\varepsilon\}$
- the root label is S
- if there is a node with label A that has n daughters labeled (from left to right) $\alpha_1, \ldots, \alpha_n$, then $A \to \alpha_1 \ldots \alpha_n \in P$
- if a node has label ε, it is a leaf and the unique daughter of its mother node

$S \overset{*}{\Rightarrow} \beta$ in G iff there is a parse tree for G with yield β.

CFG

Languages generated by a CFG

Let $G = \langle N, T, P, S\rangle$ be a CFG

- The **tree language** is the set of all parse trees with root label S and all leaves labelled with $a \in T \cup \{\varepsilon\}$.
- The **string language** $L(G)$ of G is the set $\{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$ where
  1. for $w, w' \in (N \cup T)^*$: $w \Rightarrow w'$ iff there is a $A \to \beta \in P$ and there are $v, u \in (N \cup T)^*$ such that $w = vAu$ and $w' = v\beta u$.
  2. $\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.
- A **derivation** of a word $w \in T^*$ is a sequence

$$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow w$$

of derivation steps leading to w.

CFG

For a single parse tree, there might be more than one corresponding derivation.

Derivations

CFG $G_{a,b} = \langle \{S, A, B\}, \{a, b\}, P, S \rangle$ with productions

$$S \to AB \mid BA \qquad A \to a \mid aS \mid bAA \qquad B \to b \mid bS \mid aBB$$

(This CFG generates the language $\{w \in \{a, b\}^+ \mid |w|_a = |w|_b\}$.)

Input $w = ab$.

The two derivations for w are
$S \Rightarrow AB \Rightarrow aB \Rightarrow ab$ and $S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$

($|w|_a$ gives the number of occurrences of a in w.)

Leftmost and rightmost derivations

A derivation is called a

- **leftmost** derivation iff, in each derivation step, a production is applied to the leftmost non-terminal of the already derived sentential form
- **rightmost** derivation iff, in each derivation step, a production is applied to the rightmost non-terminal of the already derived sentential form

In the preceding example, the first derivation was a leftmost derivation and the second a rightmost derivation.

CFG

For a single word w, there might even be more than one parse tree.

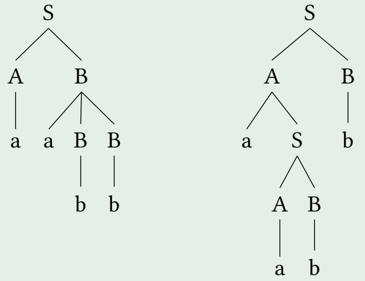
Ambiguous grammars

A CFG giving more than one parse tree for some word w is called **ambiguous**.

Ambiguous grammar

Consider again $G_{a,b}$ ($S \to AB \mid BA$, $A \to a \mid aS \mid bAA$, $B \to b \mid bS \mid aBB$)

The two parse trees for *aabb* are



CFG

Some languages are such that their structure is necessarily ambiguous. Natural languages are probably such cases.

Inherently ambiguous

A CFL L is called **inherently ambiguous** if each CFG G with $L = L(G)$ is ambiguous.

Inherently ambiguous language

$$L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

For words of the form $a^k b^k c^k d^k$ one cannot tell wich of the two patterns is the right structure. Both are possible.

Removing useless symbols

An important grammar clean-up one has to do sometimes is the removal of symbols (non-terminals or terminals) that cannot occur in any derivation of a word in the string language.

Useful/useless symbols

Let $G = \langle N, T, P, S \rangle$ be a CFG. An $X \in N \cup T$ is called

- **useful** if there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$ with $w \in T^*$
- **useless** otherwise

Removing useless symbols

- Non-terminals X can be useless because they do not allow to derive a terminal string, i.e., there is no derivation $X \stackrel{*}{\Rightarrow} w$ with $w \in T^*$.
- Non-terminals and terminals X can be useless because they cannot be reached from the start symbol, i.e., there is no derivation $S \stackrel{*}{\Rightarrow} \alpha X$.

Useless symbols

1. $G = \langle\{S, A, B, C\}, \{a, b, c\}, P, S\rangle$ with
   $P = \{S \to AB \,|\, aS, A \to aaAc \,|\, c, B \to aCc\}$
   Useless: B and C since from them one cannot derive any terminal string.
2. $G = \langle\{S, A, B, C\}, \{a, b, c\}, P, S\rangle$ with
   $P = \{S \to AB \,|\, aS, A \to aaAc \,|\, c, B \to ac, C \to b\}$
   Useless: C and b since they are not reachable from S.

Removing useless symbols

To eliminate all useless symbols, two things need to be done.

1. All $X \in N$ need to be eliminated that cannot lead to a terminal sequence.
   This can be done recursively: Starting from the terminals and following the productions from right to left, the set of all symbols leading to terminals can be computed recursively.
   Productions containing symbols that are not in this set are eliminated.
2. In the resulting CFG, the unreachable symbols need to be eliminated.
   This is done starting from S and applying productions. Each time, the symbols from the right-hand sides are added.
   Again, productions containing non-terminals or terminals that are not in the set are eliminated.

Eliminating ε-rules

Let $G = \langle N, T, P, S \rangle$. A production of the form $A \rightarrow \varepsilon$ is called an ε-production.

The following holds:

For each CFG G, there is a CFG G' without ε-productions such that $L(G') = L(G) \setminus \{\varepsilon\}$.

Removing ε-productions

$G = \langle \{S, T\}, \{a, b, c, d\}, P, S \rangle$ with
$P = \{S \rightarrow aSb \mid aTb, T \rightarrow cTd \mid \varepsilon\}$

Equivalent ε-free CFG:
$G = \langle \{S, T\}, \{a, b, c, d\}, P, S \rangle$ with
$P = \{S \rightarrow aSb \mid aTb \mid ab, T \rightarrow cTd \mid cd\}$

Eliminating ε-rules

In order to eliminate ε-productions, we

- Compute the set $N_\varepsilon = \{A \mid A \overset{*}{\Rightarrow} \varepsilon\}$ recursively
  1. $N_\varepsilon := \{A \in N \mid A \Rightarrow \varepsilon\}$
  2. For all A with $A \to \alpha$, $\alpha \in N_\varepsilon^*$: add A to N_ε
  3. Repeat 2. until N_ε does not change any more
- Delete the ε-productions and for each $A \to X_1 \ldots X_n$:
  add all productions one can obtain by deleting some $X_j \in N_\varepsilon$ from the right-hand side as long as one does not delete all $X_1, \ldots, X_n$.

Removing unary rules

Let $G = \langle N, T, P, S \rangle$. For $A, B \in N$, a production of the form $A \to B$ is called a unary production

For each CFL that does not contain ε-rules, a CFG without unary productions can be found.

Removing unary rules

$G = \langle \{S, T\}, \{a, b, c, d\}, P, S \rangle$ with $P = \{S \to aSb \mid T, T \to cTd \mid cd\}$

Equivalent CFG without unary rules: $G = \langle \{S, T\}, \{a, b, c, d\}, P, S \rangle$ with $P = \{S \to aSb \mid cTd \mid cd, T \to cTd \mid cd\}$

Elimination of unary productions for a CFG without ε-productions:

1. For all $A \overset{*}{\Rightarrow} B$ and all $B \to \beta, \beta \notin N$:
   add $A \to \beta$ to the set of productions
2. Delete all unary productions

Chomsky Normal Form

A normal form of a grammar formalism F is a further restriction on the grammars in F that does not affect the set of generated string languages.

There are two important normal forms for CFGs.

CFG normal forms

A CFG $G = \langle N, T, P, S \rangle$ for a language without ε-rules is

1. in **Chomsky normal form** iff all productions have either the form $A \to BC$ or $A \to a$ with $A, B, C \in N$, $a \in T$
2. in **Greibach normal form** iff all productions have the form $A \to a\alpha$ with $a \in T$, $\alpha \in N^*$

Chomsky Normal Form

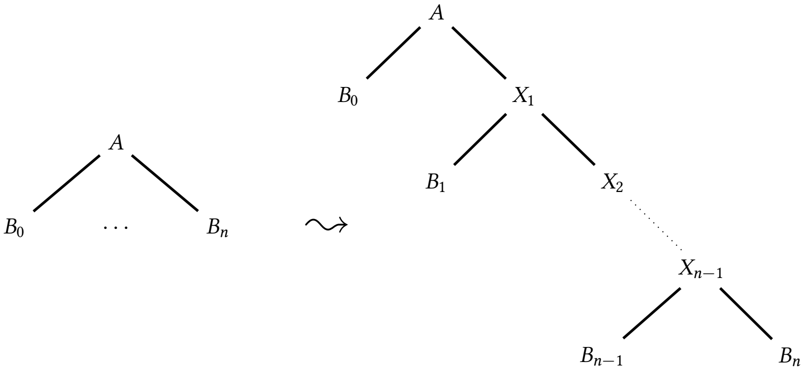
For each CFL L without ε, there is a CFG in Chomsky normal form with $L = L(G)$.

Construction of an equivalent CFG in CNF for a given CFG $G = \langle N, T, P, S \rangle$

1. Eliminate useless symbols
2. Eliminate ε-productions
3. Eliminate unary productions
4. For each $a \in T$: introduce new non-terminal T_a, replace a by T_a in all right-hand sides of length > 1 and add production $T_a \to a$ to P
5. For each production $A \to B_0 \dots B_n$ introduce new non-terminals $X_1, \dots, X_{n-1}$ and replace production $A \to B_0 \dots B_n$ with

$$A \to B_0 X_1 \qquad X_1 \to B_1 X_2 \qquad \dots \qquad X_{n-1} \to B_{n-1} B_n$$

Chomsky Normal Form



Chomsky Normal Form

Transformation to CNF

$G = \langle \{S, C\}, \{a, b, c, \}, \{S \to aSbC \mid ab, C \to c \mid cC\}, S \rangle$.

1. Introducing new preterminals:
   $G = \langle \{S, C\}, \{a, b, c, \}, P, S \rangle$ with productions
   $S \to T_a S T_b C \mid T_a T_b, C \to c \mid T_c C, T_a \to a, T_b \to b, T_c \to c$

2. Binarization:
   $G = \langle \{S, C\}, \{a, b, c, \}, P, S \rangle$ with productions
   $S \to T_a X_1 \mid T_a T_b, X_1 \to S X_2, X_2 \to T_b C, C \to c \mid T_c C,$
   $T_a \to a, T_b \to b, T_c \to c$

Greibach Normal Form

For each CFL L without ε, there is a CFG in Greibach normal form with $L = L(G)$.

Construction of the CFG in GNF for a given CFG $G = \langle N, T, P, S \rangle$

1. Eliminate useless symbols
2. Eliminate unary productions
3. Eliminate ε-productions
4. Left-recursion is eliminated, i.e., a grammar is constructed that does not allow derivations of the form $A \stackrel{+}{\Rightarrow} A\alpha$

Greibach Normal Form

Elimination of left-recursion

- Assume the set of non-terminals to be ordered, i.e. $N = \{A_1, \ldots, A_m\}$
- Construct a CFG with $j > i$ if $A_i \rightarrow A_j\gamma$:

  For all A_i, $1 \leq i \leq m$, steps (I) and (II) are done.

(I) Transformation such that $j \geq i$ if $A_i \rightarrow A_j\gamma$:

Replace all productions $A_i \rightarrow A_j\gamma$ with $j < i$ with new productions obtained from replacing A_j with all right-hand sides of A_j-productions. Do this until the condition holds for all A_i-productions.

Greibach Normal Form

(II) Elimination of left-recursive productions $A_i \to A_i\alpha$:

Add a new non-terminal B and replace

$$A_i \to A_i\alpha_1,\ \ldots,\ A_i \to A_i\alpha_r,\ A_i \to \beta_1,\ \ldots,\ A_i \to \beta_s$$

with

$$A_i \to \beta_i,\ A_i \to \beta_i B \quad \forall i,\ 1 \leq i \leq s$$

and

$$B \to \alpha_i,\ B \to \alpha_i B \quad \forall i,\ 1 \leq i \leq r$$

(Left recursion is turned into right recursion)

In the resulting grammar, for all $A_i \stackrel{+}{\Rightarrow} A_j\alpha$, $i < j$ holds.

Greibach Normal Form

Transformation to GNF

$G = \langle N, T, P, S \rangle,\ N = \{A_1, A_2, A_3, B\},\ T = \{a, b\}$

$$A_1 \to A_2A_3 \quad A_2 \to A_3A_1 \mid b \quad A_3 \to A_1A_2 \mid a$$
$$A_1 \to A_2A_3 \quad A_2 \to A_3A_1 \mid b \quad A_3 \to A_2A_3A_2 \mid a$$
$$A_1 \to A_2A_3 \quad A_2 \to A_3A_1 \mid b \quad A_3 \to A_3A_1A_3A_2 \mid bA_3A_2 \mid a$$

Replace $A_3 \to A_3A_1A_3A_2 \mid bA_3A_2 \mid a$ by

$$A_3 \to bA_3A_2 \qquad A_3 \to a \qquad A_3 \to bA_3A_2B \quad A_3 \to aB$$
$$B \to A_1A_3A_2 \quad B \to A_1A_3A_2B$$

The resulting grammar does not allow left-recursive derivations.

Greibach Normal Form

Lexicalization

- Now, two more steps are necessary to obtain the grammar in GNF:

5. For $i = m - 1$ to $i = 1$: Replace all $A_i \rightarrow A_j\beta$ with all productions obtained by replacing A_j with the right-hand side of a A_j-production

   Then, do the same for all B-productions where B is one of the symbols introduced in step (II)

6. For all productions $A \rightarrow \alpha a\beta$, $\alpha \neq \varepsilon$: replace a with a new non-terminal T_a and add a production $T_a \rightarrow a$ to P.

Greibach Normal Form

Transformation to GNF (cont'd)

$$A_1 \to A_2 A_3$$
$$A_2 \to A_3 A_1 \mid b$$
$$A_3 \to bA_3A_2 \mid a \mid bA_3A_2B \mid aB$$
$$B \to A_1A_3A_2 \mid A_1A_3A_2B$$

Replace

1. $A_2 \to A_3A_1$ by
   $A_2 \to bA_3A_2A_1 \mid aA_1 \mid bA_3A_2BA_1 \mid aBA_1$
2. $A_1 \to A_2A_3$ by
   $A_1 \to bA_3A_2A_1A_3 \mid \ldots \mid aBA_1A_3 \mid bA_3$
3. $B \to A_1A_3A_2$ by
   $B \to bA_3A_2A_1A_3A_3A_2 \mid \ldots \mid bA_3A_3A_2$
4. $B \to A_1A_3A_2B$ by
   $B \to bA_3A_2A_1A_3A_3A_2B \mid \ldots \mid bA_3A_3A_2B$