Chapter 6
Algebraic Grammar (Context-Free)

Reminder on grammars and algebraic languages

Definition of a Type 2 Grammar:

A grammar $G = (V_T, V_N, S, R)$ is called a context-free grammar (Algebraic or Type 2) if and only if all of its production rules are in the form:

$$A \to B \text{ where } A \in V_N \text{ and } B \in (V_T \cup V_N)^*.$$

Definition of Type 2 Languages (Context-Free or Algebraic):

These are the languages that can be defined by Type 2 grammars.

Note: The set of regular languages is included in the set of algebraic languages.

The syntax tree

Given the use of a single non-terminal symbol on the left-hand side of production grammars, rules in context-free it is always possible to construct a derivation tree for a generated word.

Definition

Let the grammar $G = (V_T, V_N, S, R)$ and let $\omega \in L(G)$. A syntax tree associated with ω is constructed such that:

- The root of the tree is labeled with the axiom;
- The intermediate nodes contain non-terminals;
- The leaves are terminals. The left-to-right reading of the leaves of the tree reconstructs the word to which the tree is associated.

Left derivation (right derivation)

Let $G = (V_T, V_N, S, R)$ be an algebraic grammar. A word w is said to be obtained by left derivation (resp. right derivation) if it is derived from the axiom by always replacing the leftmost non-terminal (resp. the rightmost non-terminal) during the derivations.

Example:

$S \to aAS \mid a$

$A \to ba$

Let $w = abaaabaa$

Left Derivation:

$S \Rightarrow aAS \Rightarrow abaS \Rightarrow abaaAS \Rightarrow abaaabS \Rightarrow abaaabaa$

Right Derivation:

$S \Rightarrow aAS \Rightarrow aAaAS \Rightarrow aAaAa \Rightarrow aAabaa \Rightarrow abaaabaa$

Notion of ambiguity

- **Definition of an ambiguous word:**
  A word ω is said to be ambiguous if and only if there exist two different derivation trees associated with it.

- **Definition of an ambiguous grammar:**
  A grammar G is said to be ambiguous if and only if there exists at least one ambiguous word belonging to L(G).

ambiguity

Definition:

Let G=⟨X,V,P,S⟩ be an algebraic grammar. A word w is said to be ambiguous if and only if there exist more than one left derivation (resp. more than one right derivation) for this word.

Example:

$$E \to E + E \mid E - E \mid E \times E \mid E \div E \mid (E) \mid ID$$
$$ID \to x \mid y$$

Apply to the word **w = x x x + y**

Example 1

- Let G be the grammar that has the following production rules:

$$S \rightarrow S \wedge S \mid S \vee S \mid S \Rightarrow S \mid S \Leftrightarrow S \mid \neg S \mid q \mid p$$

- **Question:** Show that G is ambiguous.

Answer:

The word $p \wedge q \Leftrightarrow p$ is ambiguous because there are two different derivations that allow us to reach it. If we number the rules from 1 to 7, we will have:

$$S \vdash (4)S \Leftrightarrow S \vdash (1)S \wedge S \vdash (7)p \wedge S \vdash (6)p \wedge q \Leftrightarrow p$$

$$S \vdash (1)S \wedge S \vdash (4)p \wedge S \vdash (6)p \wedge q \Leftrightarrow S \vdash (7)p \wedge q \Leftrightarrow p$$

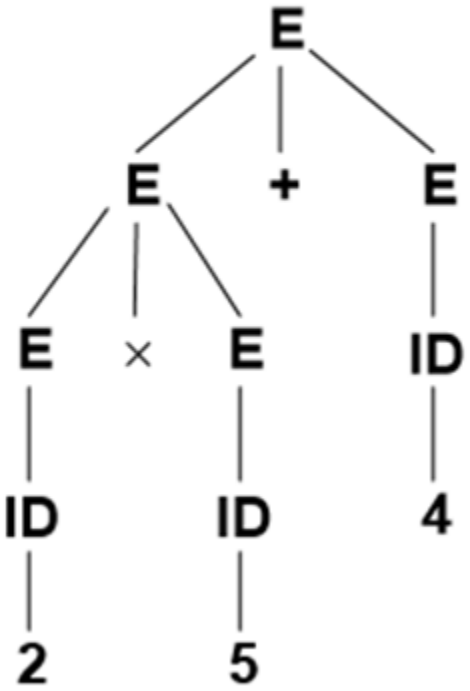
Therefore, the grammar G is ambiguous.

Example 2

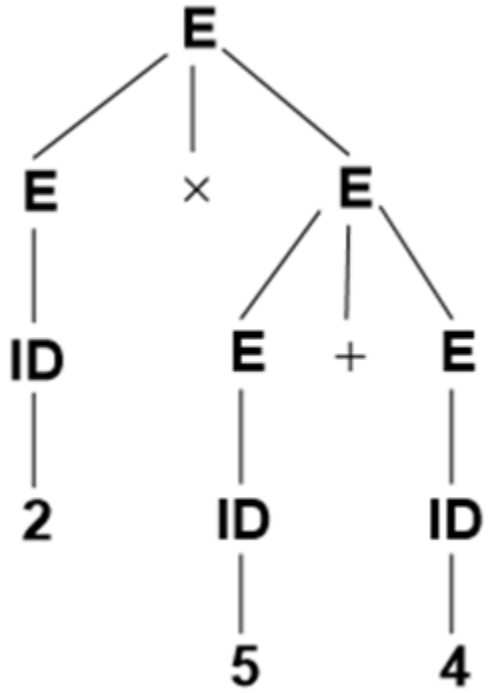
w=2×5+4

$E \rightarrow E+E \mid E-E \mid E\times E \mid E\div E \mid (E) \mid ID$

$ID \rightarrow 1 \mid 2 \mid \ldots \mid 9$



10 + 4 = 14



2 × 9 = 18

Definitions

Productive and Non-Productive Non-Terminals

- A non-terminal A is said to be **productive** if and only if

$$\exists \omega \in V_T^* \text{ such that } A \Rightarrow^* \omega.$$

- A non-terminal A is said to be **non-productive** if and only if

$\forall \omega \in V_T^*$, there is no indirect derivation such that $A \Rightarrow^* \omega$.

Accessible and Inaccessible Non-Terminals

- A non-terminal A is said to be **accessible** if and only if

$\exists \alpha \in (V_T \cup V_N)^*$ such that $S \Rightarrow^* \alpha$ and A appears in α.

- A non-terminal A is said to be **inaccessible** if and only if

$\forall \alpha \in (V_T \cup V_N)^*, S \Rightarrow^* \alpha$, then A does not appear in α.

Definition

- **Reduced Grammar**

  A grammar is said to be reduced if and only if all the non-terminals in its production rules are reachable (accessible) and productive.

- **Note**

  Production rules that contain non-terminals that are non-productive or inaccessible are useless and can be removed without any influence on the language generated by the grammar.

All symbols are accessible and produce something

All symbols are useful

Suppression des symboles inutiles

- Remove symbols that produce nothing and the rules where they appear.

- Remove from G the inaccessible symbols and the productions where they appear.

Unit production

- **Unit Production**

  A production $A \rightarrow B$ is called a **unit production** if and only if A and B are non-terminal symbols.

- **Note:**

  To remove the unit production $A \rightarrow B$, it is enough to add to the production rules of A all the productions of B.

  This removal may lead to the appearance of other unit productions, which is why a recursive algorithm must be applied.

Definition

- **Proper Grammar**

  A grammar is said to be proper if and only if:

  - It is reduced;
  - It does not contain unit productions;
  - The axiom that can generate ε exists, with the condition that it does not appear in any right-hand member of the rules.

- **To eliminate ε-productions**, first determine the set of non-terminals that are derivable to ε (directly or indirectly); then, modify the productions containing these non-terminals to replace in all the left-hand parts of productions the nullable symbols with the empty word, in all possible ways.

Examples

Example 1

1) $S \rightarrow aSb \mid ab$

2) $S \rightarrow aAb \mid \varepsilon$; $A \rightarrow aAb \mid ab$

G is ε-free

Example 2

$S \rightarrow aSb \mid \varepsilon$

G is not ε-free

Make G is ε-free

$S' \rightarrow S \mid \varepsilon$

$S \rightarrow aSb \mid ab$

Example 3

$S \rightarrow AbB$

$A \rightarrow aAb \mid \varepsilon$

$B \rightarrow Ba \mid \varepsilon$

$\Rightarrow$

$S \rightarrow b \mid bB \mid Ab \mid AbB$

$A \rightarrow aAb \mid ab$

$B \rightarrow Ba \mid a$

Example

- Let G be the grammar defined by the following production rules:

$$S \rightarrow AB \mid EaE \qquad E \rightarrow D$$
$$A \rightarrow Aa \mid aB \qquad D \rightarrow dD \mid \varepsilon$$
$$B \rightarrow bB \mid aA$$
$$C \rightarrow AB \mid aS$$

1. Find the language generated by G.
2. Transform G into a reduced grammar.
3. Transform G into a proper grammar.
4. Verify the language found in question 1.

Example (solution)

- $L(G) = \{d^i a d^j \ / \ i, j \geq 0\}$.
- The non-terminal C is unreachable (non-accessible), A and B are non-productive, so we remove the rules containing A, B, or C and the resulting grammar is:

$S \rightarrow EaEE \rightarrow DD \rightarrow dD \mid \varepsilon$

- The grammar has a unit production $E \rightarrow D$. We remove it and replace all occurrences of E with D, resulting in the following grammar:

$S \rightarrow DaDD \rightarrow dD \mid \varepsilon$

- The grammar is still not clean due to the rule $D \rightarrow \varepsilon$, so we eliminate it. For each occurrence of D on the right-hand side of a rule, we create another rule. We get the clean grammar:

$S \rightarrow DaD \mid aD \mid Da \mid aD \rightarrow dD \mid d$

- The language found is correct, but it is easier to find with the clean grammar.

Chomsky Normal Form

•A grammar $G=(V_T, V_N, S, R)$ is said to be in Chomsky Normal Form (CNF) if and only if all its production rules are of the form:

$$\mathbf{A \rightarrow BC \text{ or } A \rightarrow a \text{ with } A,B,C \in V_N \text{ and } a \in V_T.}$$

Proposition:

- For any algebraic grammar, there exists an equivalent grammar in Chomsky Normal Form.
- The practical advantage of Chomsky Normal Form (CNF) is that the derivation trees are binary trees, which makes it easier to apply tree exploration algorithms.

Chomsky Normal Form"

To obtain a Chomsky Normal Form grammar equivalent to an algebraic grammar G, the following steps are required:

1. Transform the grammar into a proper grammar.
2. For each terminal a, introduce the non-terminal C_a, then add the rule $C_a \to a$.
3. For each rule $A \to a$, with $|\alpha| \geq 2$, replace each terminal by the associated non-terminal.
4. For each rule $A \to \beta$, with $|\beta| \geq 3$, ($\beta = \beta_1\beta_2 \dots \beta_n$), create the non-terminals D_i, then replace the rule with the following rules:
   $A \to \beta_1 D_1$,
   $D_1 \to \beta_2 D_2$, $D_{n-2} \to B_{n-1}B_n$, where $D_i = B_{i+1}B_{i+2} \dots B_n$, with i varying from 1 to $n-2$.

Example

$$S \rightarrow Ba \mid Ab$$
$$A \rightarrow Sa \mid AAb \mid a$$
$$B \rightarrow Sb \mid BBa \mid b$$

$\Rightarrow$

$$S \rightarrow BA_1 \mid AB_1$$
$$A \rightarrow SA_1 \mid AAB_1 \mid A_1$$
$$B \rightarrow SB_1 \mid BBA_1 \mid B_1$$
$$A_1 \rightarrow a$$
$$B_1 \rightarrow b$$

$\Rightarrow$

$$S \rightarrow BA_1 \mid AB_1$$
$$A \rightarrow SA_1 \mid AX_1 \mid a$$
$$B \rightarrow SB_1 \mid BX_2 \mid b$$
$$X_1 \rightarrow AB_1$$
$$X_2 \rightarrow BA_1$$
$$A_1 \rightarrow a$$
$$B_1 \rightarrow b$$

Greibach Normal Form

- An algebraic grammar is in Greibach Normal Form (GNF) if and only if all its production rules are of the form:
  $A \to x\alpha$ or $S \to \varepsilon$, where $x \in V_T$, $\alpha \in V_N^*$, and S is the axiom.

Proposition:

For any algebraic grammar G_1, there exists a grammar G_2 in Greibach Normal Form such that $L(G_2) = L(G_1)$.

Practical Interest of GNF:

The practical advantage of GNF is that with each derivation, we determine an increasingly longer prefix formed solely by terminal symbols.

This allows the construction of pushdown automata from grammars more easily, and consequently, syntax analyzers that are easily implementable.

Construction of the GNF of a grammar

- **Input:** G reduced and proper
- **Output:** G' in GNF

Let G be an algebraic grammar, reduced and proper.

Definitions:

- A non-terminal symbol A is said to be **left-recursive** if there exists at least one production rule in P such that:
  $A \to A\alpha$ ($A \in V_N$, $\alpha \in (V_N \cup V_T)^*$).
- A grammar is said to be **left-recursive** if there exists at least one left-recursive non-terminal.

Algorithm for eliminating left recursion

- Apply the following transformation to all production rules until all direct and hidden left recursions are eliminated.

If $(A \rightarrow A\alpha_i/\beta_i) \in P$ with $\alpha_i, \beta_i, \varnothing \in (V_N \cup V_T)^*$ and $\beta_i \neq A\varnothing$

Then $(A \rightarrow \beta_i/\beta_i A')$ and $A' \rightarrow \alpha_i/\alpha_i A'$.

Example: Simple Left Recursion

The rule $A \rightarrow AAS/aS/b$ is transformed into:

$A \rightarrow aS/b/aSA'/bA'$

$A' \rightarrow AS/ASA'$

Example: Hidden Left Recursion

Let the grammar defined by the following set P:

$S \to AA/a$
$A \to SS/b$

Transformation:

Replace S in $A \to SS/b$, we get $A \to AAS/aS/b$ (already transformed)

$A \to aS/b/aSA'/bA'$
$A' \to AS/ASA'$

Thus, the resulting grammar (non-left-recursive) is defined by the following set P':

$S \to AA/a$
$A \to aS/b/aSA'/bA'$
$A' \to AS/ASA'$

Definition

- We say that $A < B$ if and only if $A \to B\alpha$ and $B \in V; \alpha \in (X \cup V)^*$.
- **Obtaining the GNF:**

  a) Establish the linear order.

  b) Substitutions of non-terminals.

Example

$S \to AA/a$

$A \to aS/b/aSA'/bA'$

$A' \to AS/ASA'$

We have $S < A$ and $A' < A$, so this means that the rule for A is in GNF.

Substitutions:

- From A in S:
  $S \to aS/bA'/aSA'A/bA'A/a$ [F1]
- From A in A':
  $A' \to aSS/bS/aS'A/bS'A'/aS'A'A/bA'SAA'$ [F2]

In the GNF grammar of G, G' is such that:

$[F1], A \to aS/b/aSA'/bA'$, [F2]

Algebraic Languages

- **Proposition:**
  The class of algebraic languages is closed under union, concatenation, mirror, and iteration.
- **Note:**
  The class of algebraic languages is not closed under intersection and complement.
- **Theorem:**
  The class of languages recognized by pushdown automata is equal to the class of languages generated by algebraic grammars.

Double Star Lemma (Ogden's Lemma)"

For every algebraic language L, there exists an integer k such that

for every word $w \in L$ with length $|w| \geq k$,

there is a factorization of $w = \alpha\mu\beta\nu\delta$ with $|\mu\nu| > 0$ and $|\mu\beta\nu| < k$ such that:

$$\forall n \in \mathbb{N}, \alpha^n \mu^n \beta^n \nu^n \delta \in L$$

Example

Let's prove that $L = \{a^n b^n c^n, n \geq 0\}$ is not an algebraic language.

Suppose that the language L is algebraic and w is some word in L.

There exists $n \geq 0$ such that $w = d^n b^n c^n$, and it can be written in the form $w = \alpha\mu\beta\nu\gamma$, where the property $\alpha\beta^k\nu^k \in L$ for all $k \geq 0$ must hold.

It is enough to find a k that does not satisfy the property, depending on the chosen factorization.

Let us take $\alpha = d^k$; $\mu = b^n$; $\beta = c^n$; $\nu = \varepsilon$.

We form $\alpha\beta^k\nu^k$ for $k = 2$ for example:
$\alpha^2\beta^2\nu^2 = d^2b^2c^2 \neq d^n b^n c^n$, thus $d^n b^n c^n \notin L$.

This leads to a contradiction, and thus the hypothesis is false. Therefore, the language $\{a^n b^n c^n, n \geq 0\}$ is not algebraic.