CHAPTER 6

Reduction Systems, Grammars, and Chomsky Hierarchy

Goal: In this chapter we introduce reduction systems and their use in Theoretical Computer Science in terms of grammars. We will see that certain restrictions on the reduction systems lead to different types of grammars categorized by the Chomsky hierarchy.

6.1. Rewriting and Reduction Systems

Throughout this section, let $\mathbb{A}$ be an arbitrary alphabet. The idea is to replace words over $\mathbb{A}$ with other words according to a set of rules.

Definition 6.1. A *reduction system* over $\mathbb{A}$ is a pair $E = (\mathbb{A}, P)$ with $P \subseteq \mathbb{A}^\star \times \mathbb{A}^\star$. Elements of P are called *productions* or *rewrite rules*. We write $u \to v$ for $(u, v) \in P$.

Consider the following reduction system E_1:

$$E_1 = (\underbrace{\{a, \ldots, z\}}_{\mathbb{A}}, \underbrace{\{(can, must)\}}_{P})$$

The idea is to replace occurrences of the word *can* with the word *must*. We formally introduce these replacements as *derivations* over $\mathbb{A}^\star$.

Definition 6.2. Denote by $\vdash_E \subset \mathbb{A}^\star \times \mathbb{A}^\star$ the *derivable* relation over E. Let $w, v \in \mathbb{A}^\star$. Then $w \vdash_E v$ iff there are words $w_1, w_2 \in \mathbb{A}^\star$ and a production $u \to v \in P$ such that $w = w_1 u w_2$ and $v = w_1 v w_2$.

We may also just write $\vdash$ instead of $\vdash_E$ if the reduction system clear. Sometimes we also write $\to_E$ ($\to$ resp.) following the notation of the production rules. Denote by $\vdash^\star$ the reflexive transitive closure of $\vdash$, as usual. A sequence $w = z_0 \vdash \cdots \vdash z_n = v$ is called *derivation* of size n. Moreover, we find:

i) There is always a derivation of size 0 with $w \vdash^\star w$.
ii) For $w \neq v$ and $w \vdash^\star v$ there is a derivation of size $n > 0$.
iii) If $w \vdash v$, we say w *directly derives* v.
iv) We write $w \nvdash$, if there is no v, such that w directly derives v.

Example 6.1. Consider $E_1 = (\{a, \ldots, z\}, \{(can, must)\})$ like above. The derivations of maximum length replace *all* occurrences of *can* with the string *must*. Note, that there is generally no defined order of replacements.

(1) I can go to work $\vdash_{E_1}$ I must go to work
(2) I can go to work and I can do sports $\vdash_{E_1}$ I must go to work and I can do sports
  or I can go to work and I can do sports $\vdash_{E_1}$ I can go to work and I must do sports
(3) I can go to work and I can do sports $\vdash^\star_{E_1}$ I must go to work and I must do sports

Example 6.2. Let $E_2 = (\mathbb{A}, \{(a, aba)\})$. Under this reduction system, we have infinite derivations. For instance

$$cac \vdash_{E_2} cabac \vdash_{E_2} cababac \vdash_{E_2} \cdots$$

By chance, the replacement of either a in the second step leads to the same word. Consider another word $w = caac$. If we always replace the first (the last) occurrence of a, we get

$$caac \vdash_{E_2} cabaac \vdash_{E_2} cababaac$$
or $$caac \vdash_{E_2} caabac \vdash_{E_2} caababac$$ respectively

However, there are infinitely many other derivations.

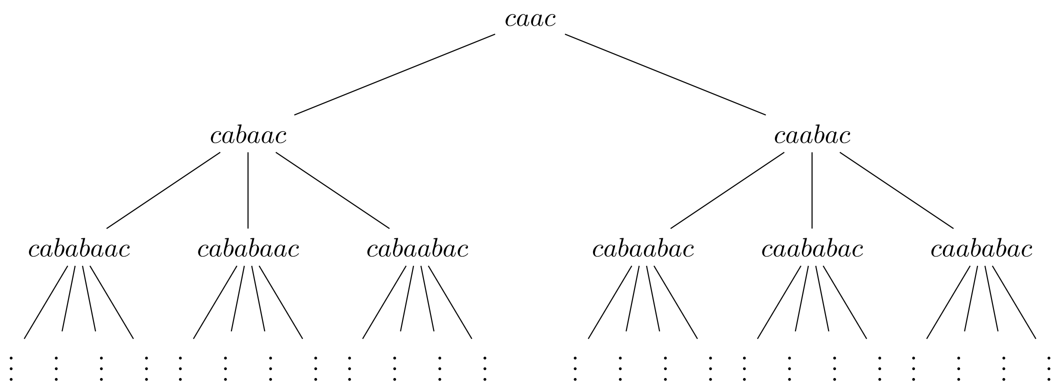

FIGURE 6.1. Derivation tree for $L(caac, E_2)$

Definition 6.3. The language $L(w, E) := \{v \in \mathbb{A} \mid w \vdash_E^\star v\}$ is the language of all derivable words of w over E.

The derivations of w may be represented as a tree with w as its root node. The children of a node u are the immediately derivable words of u. Figure 6.1 shows the (infinite) derivation tree for Example 6.2.

Example 6.3. Consider the reduction system $E_3 = (\mathbb{A}, \{(ba, ab)\})$ and the word $w = baacba \in \mathbb{A}^\star$. Observe, that there is an n such that all derivations of w under E_3 have a size of at most n. We find $n = 3$ with the following example derivation:

$$baacba \vdash_{E_3} abacba \vdash_{E_3} abacab \vdash_{E_3} aabcab$$

The number $(n+1)$ is also the depth of the derivation tree. Hence, the language $L(w, E_3)$ is finite.

Example 6.4. For $E_4 = (\mathbb{A}, \{(a, b), (a, c), (c, d), (b, e), (d, e)\})$ we see that $L(w, E_4)$ is always finite, independently from the choice of w. An example derivation is

$$a \vdash_{E_4} c \vdash_{E_4} d \vdash_{E_4} e$$

Example 6.5. In Example 6.3 we have seen, that a language $L(w, E)$ is finite, if the corresponding derivation tree is finite. Now we see that the other direction does *not* hold. Let $E_5 = (\mathbb{A}, \{(x, y), (y, x), (x, a), (y, b)\})$. Figure 6.2 shows the infinite derivation tree for $L(x, E_5)$ with the infinite derivation $x \vdash_{E_5} y \vdash_{E_5} x \vdash_{E_5} y \vdash_{E_5} \cdots$. However, $L(x, E_5) = \{x, y, a, b\}$ is obviously a finite language.

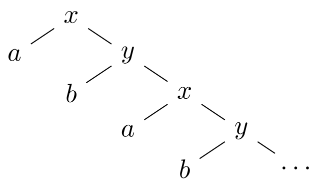

FIGURE 6.2. Derivation tree for $L(x, E_5)$

6.2. Formal Grammars

Formal grammars are reduction systems with additional structure. Some symbols are considered *non-terminal* whereas others are considered *terminal*. The former may be understood as auxiliary symbols for the derivation of a word, where a special symbol S denotes the start symbol. Hence, we are talking about languages $L(S, E_G)$ where E_G is a special reduction system which replaces non-terminals only.

Formal grammars were originally defined by Noam Chomsky in 1959. The difference between non-terminal symbols and terminal symbols has a direct linguistic analogue. Terms like subject, predicate, object correspond to non-terminals while the letters of the English (or any other) alphabet are terminal symbols.

$$\text{[subject] [predicate] [object]} \rightsquigarrow^\star \text{Captain Picard commands the Enterprise}$$

Definition 6.4. A *formal grammar* is a quadruple $G = (N, \Sigma, P, S)$ with

i) an alphabet N of *non-terminals*,
ii) an alphabet Σ of *terminals*,
iii) both alphabets being disjoint ($N \cap \Sigma = \emptyset$),
iv) a set of production rules $P \subseteq (N \cup \Sigma)^+ \times (N \cup \Sigma)^\star$,
v) the left hand-side of a rule containing at least one non-terminal ($(u, v) \in P \Rightarrow u \notin \Sigma^\star$),
vi) and the start symbol $S \in N$.

Observe that $E_G = (N \cup \Sigma, P)$ is a well-defined reduction system (cf. Definition 6.1). We use the notation $u \to v$ for elements of P and the relation $\vdash_{E_G}$ for derivations accordingly. Restricting productions to have at least one non-terminal on the left hand-side has two benefits: first, it prohibits the production of words out of nothing. Second, a word over $\Sigma^\star$ cannot be changed anymore. This motivates the definition of the language $L(G)$ as the set of words consisting of terminal symbols only.

Definition 6.5. Let $G = (N, \Sigma, P, S)$ be a grammar. A *sentential form* of G is any word w of terminals and non-terminals over $(N \cup \Sigma)^\star$. If w does not contain non-terminals (i.e. $w \in \Sigma^\star$), the word itself is also called *terminal*. The language $L(G)$ produced by G is defined as follows:

$$L(G) = \{w \in L(S, E_G) \mid w \text{ is terminal}\} \text{ where } E_G = (N \cup \Sigma, P)$$

Example 6.6. Let $G_1 = (N_1, \Sigma_1, P_1, S_1)$ be a grammar with

$$\begin{aligned} N_1 &= \{S_1\} \\ \Sigma_1 &= \{a, b\} \\ P_1 &= \{S_1 \to \varepsilon, S_1 \to aS_1b\} \end{aligned}$$

Figure 6.3 shows the derivation tree for $L(G_1) = L(S_1, E_{G_1})$. The derivations to the left use the

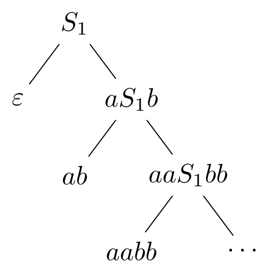

FIGURE 6.3. Derivation tree for $L(G_1)$

rule $S_1 \to \varepsilon$ and the derivations to the right use the rule $S_1 \to aS_1b$. The terminal words are exactly the leafs of the derivation tree ($\varepsilon, ab, aabb, \dots$). Hence, the language of G_1 is $L(G_1) = \{a^nb^n \mid n \in \mathbb{N}\}$.

Grammars are very useful to describe desired structural properties of a text. Most programming languages are described by using formal grammars. They define the syntax of a valid program source code. The following example shows, how a simple grammar can be used to describe the correct usage of parenthesis.

Example 6.7. Consider the language L_{Dyck} with

$$L_{Dyck} = \{w \in \{(,)\}^\star \mid |w|_(= |w|_) \wedge \forall u \in \mathsf{prefix}(w) : |u|_(\geq |u|_)\}$$

where $|w|_a$ denotes the amount of occurrences of a in w and $\mathsf{prefix}(w)$ denotes the set of words x such that there is a word y over the same alphabet with $xy = w$, as usual. We may construct the following grammar $G_2 = (N_2, \Sigma_2, P_2, S_2)$ with

$$\begin{aligned} N_2 &= \{S_2\} \\ \Sigma_2 &= \{(,)\} \\ P_2 &= \{S_2 \to \varepsilon, S_2 \to (S_2), S_2 \to S_2S_2\} \end{aligned}$$

Indeed, $L(G_2) = L_{Dyck}$ (the proof is left as an exercise).

6.3. Chomsky Hierarchy

Hitherto, grammars may contain production rules with arbitrary concatenations of terminals and non-terminals on the left hand-side, as long as there is at least one non-terminal. In this section we introduce some further restrictions on the structure of the production rules, which we will find having a direct influence on the structure of the words produced by the grammar. The grammars as well as their corresponding languages are thereby divided into classes which leads to the *Chomsky hierarchy.* While having several different classes of grammars, we essentially find only four language classes: Chomsky Type-0 (unrestricted) to Type-3 (most restricted). We start by defining restrictions for the productions of a grammar.

Definition 6.6. A production $u \to v \in P$ for some grammar $G = (N, \Sigma, P, S)$ is called

i) left regular/left linear if $u \in N$ and $v \in N\Sigma^\star \cup \Sigma^\star$
The left hand-side consists of exactly one non-terminal and the right hand-side contains at most one non-terminal which is required to be the leftmost symbol.
ii) right regular/right linear if $u \in N$ and $v \in \Sigma^\star N \cup \Sigma^\star$
The left hand-side consists of exactly one non-terminal and the right hand-side contains at most one non-terminal which is required to be the rightmost symbol.
iii) context-free if $u \in N$.
The left hand-side consists of exactly one non-terminal and the right hand-side is unrestricted (in particular v may be ε).
iv) context-sensitive if $u = xYz$ and $v = xwz$ with $Y \in N, x, z, w \in (N \cup \Sigma)^\star$ and $w \neq \varepsilon$
One non-terminal Y on the left hand-side in the context of x and z is replaced with the non-empty word w.
v) monotone if $|u| \leq |v|$
The production does not shorten the word.

Example 6.8. Consider $G = (\{S, X\}, \{a, b\}, P, S)$ The following table shows some examples for productions in P with their respective properties:

Production	Properties
$S \to \varepsilon$	regular (linear) and context-free, but neither context-sensitive nor monotone
$X \to a$	has all of the above five properties
$S \to aX$	right regular (right linear), context-free, context-sensitive, and monotone
$S \to Sb$	left regular (left linear), context-free, context-sensitive, and monotone
$S \to aSb$	context-free, context-sensitive, and monotone
$aSb \to aXaXb$	context-sensitive, and monotone
$aXb \to bSXa$	monotone and noting else
$aXb \to Xa$	has non of the above five properties

We may derive some relations between the above properties. For instance, regular productions are also context-free. This immediately follows from the definition. Context-free productions with $v \neq \varepsilon$ are also context-sensitive with an empty context $x = z = \varepsilon$ and every context-sensitive production is necessarily monotone. Now we extend the characterization of productions rules to the whole grammar.

Definition 6.7. A grammar $G = (N, \Sigma, P, S)$ is called

i) left regular/left linear if *all* productions are left regular/left linear

ii) right regular/right linear if *all* productions are right regular/right linear
iii) context-free if *all* productions are context-free
iv) context-sensitive if *all* productions are context-sensitive except $S \to \varepsilon$ (then, S must not occur on the right hand-side of any other production)
v) monotone (or noncontracting) if *all* productions are monotone except $S \to \varepsilon$ (then, S must not occur on the right hand-side of any other production)

For context-sensitive and monotone we need the exception for $S \to \varepsilon$ to allow the production of the empty word. Observe, that without this exception we have $\varepsilon \notin L(G)$ for all context-sensitive and monotone grammars G. However, S must not simultaneously occur on the right hand-side of a production, because this would break monotonicity. For instance the rule $aXb \to aSb$ in combination with $S \to \varepsilon$ would allow for $aXb \vdash^\star ab$ with $|aXb| \not\leq |ab|$, despite all production rules obeying that restriction.

Up to now we can categorize a *grammar* to be (left/right) regular, context-free, context-sensitive or monotone. We change perspective and perform a similar categorization for *languages*. Please note, that for grammars we argued about *all* productions. For instance, a grammar is context-free, if *all* productions are context-free. However, for languages it shall suffice that *there is* a grammar G, such that G produces that language. There may still be other grammars of different types producing the same language.

Definition 6.8. Let Σ be an alphabet. A language $L \subseteq \Sigma^\star$ is called

i) left regular/left linear if there is a left regular/left linear grammar G with $L = L(G)$
ii) right regular/right linear, regular or Chomsky Type-3 if there is a right regular/right linear grammar G with $L = L(G)$
iii) context-free or Chomsky Type-2 if there is a context-free grammar G with $L = L(G)$
iv) context-sensitive or Chomsky Type-1 if there is a context-sensitive grammar G with $L = L(G)$
v) noncontracting if there is a monotone grammar G with $L = L(G)$
vi) Chomsky Type-0 if there is *any* grammar G with $L = L(G)$

Example 6.9. Consider $\Sigma = \{a, b\}$ and the language $L_3 = \{a^n b \mid n \in \mathbb{N}\}$. This language can be produced by the right regular grammar $G_3 = (N_3, \Sigma, P_3, S)$ with

$$\begin{aligned} N_3 &= \{S\} \\ P_3 &= \{S \to aS, S \to b\} \end{aligned}$$

Hence, L_3 is right regular. But L_3 is also left regular according to the following left regular grammar $G_3' = (N_3', \Sigma, P_3', S)$:

$$\begin{aligned} N_3' &= \{A, S\} \\ P_3' &= \{S \to Ab, A \to Aa, A \to \varepsilon\} \end{aligned}$$

The language L_3 is also context-free, context-sensitive – this is an easy exercise.

Example 6.10. Recall the grammar G_1 with $L_1 = L(G_1) = \{a^n b^n \mid n \in \mathbb{N}\}$. Since both productions $S_1 \to \varepsilon$ and $S \to aS_1b$ are context-free, L_1 is a context-free language. However, from the previous chapter we already know that there cannot exist a regular grammar for L_1.

Example 6.11. Let $\Sigma = \{a, b, c\}$ and $L_4 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$. This language is noncontracting (or monotonic) because we find the following monotone grammar $G_4 = (N_4, \Sigma, P_4, S)$:

$$\begin{array}{rll} & N_4 = \{S, R, B\} & \\ P_4: & S \to \varepsilon & S \to R \\ & R \to aRBc & R \to abc \\ & cB \to Bc & bB \to bb \end{array}$$

An example derivation could be $S \vdash R \vdash aRBc \vdash aabcBc \vdash aabBcc \vdash aabbcc$. Later we will see, that L_4 is even context-sensitive, but not context-free.