

Chapter 3

Push-Down Automata and Context-Free Languages

In the previous chapter, we studied finite automata, modeling computers without memory. In the next chapter, we study a general model of computers with memory. In the current chapter, we study an interesting class that is in between: a class of automata with memory in the form of a stack, so-called *push-down automata*. The class of languages associated with push-down automata is often called the class of *context-free* languages. This class is important in the study of programming languages, in particular for parsing based on context-free grammars.

3.1 Push-down automata

A rough sketch of the architecture of a DFA or NFA is given in Figure 3.1: the control unit labeled ‘Automaton’ processes the input that is provided, and produces a yes/no answer for the input string processed so far. Yes, if the current state of the control unit is a final state; no, if the current state of the control unit is a non-final state. The only ‘memory’ that is available are the states of the control unit.

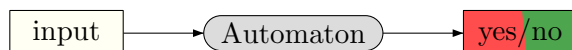


Figure 3.1: Architecture of finite automaton

We next consider an abstract machine model that does have memory, viz. memory in the form of a stack. The stack can be accessed only at the top: something can be added on top of the stack via a push operation, or something can be removed from the top of the stack via a pop operation. Items under the top of the stack cannot be inspected directly. For this reason, the machine model is generally referred to as a *push-down automaton*, PDA for short. We assume that we have means to check if the stack is empty, i.e. to observe that there is no item on (top of) the stack. See Figure 3.2 for the augmented

architecture of a push-down automaton.

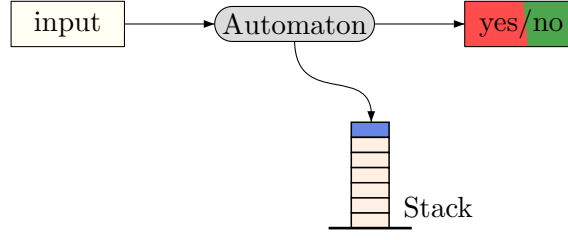


Figure 3.2: Architecture of a push-down automaton

Example 3.1. Let us look at an example of a push-down automaton. In Figure 3.3, we indicate the initial state q_0 as usual. Initially, the stack is assumed to be empty, signaled by the special symbol \emptyset . In state q_0 , we can input a symbol a , thereby replacing the empty stack by the stack containing a single data element 1 while control switches from state q_0 to state q_1 . In the state q_1 , we can either input a symbol a again, replacing the top element 1 by twice the item 1 indicated by 11 (thus effectively adding an item 1 on top), and remain in state q_1 , or alternatively input the symbol b , popping the top item, which is an item 1, by putting nothing in its place as indicated by the string ε , and going to state q_2 . There, we can read in a b repeatedly, popping 1's, but we must terminate and go to final state q_3 if (and only if) the stack has become empty.

We see that this push-down automaton can read any number of a 's (collecting exactly so many 1's on the stack), followed by the input of the same number of b 's followed by termination. As we shall see, the language accepted by the PDA is $\{ a^n b^n \mid n \geq 1 \}$, a non-regular language.

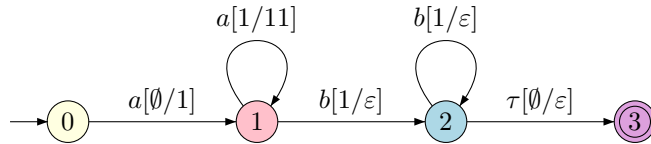


Figure 3.3: Example push-down automaton.

From the example we see that in a PDA we allow internal steps τ , that consume no input symbol. We even allow the stack to be modified during such steps. We also see that the transitions need not to be specified by a function. E.g., a b -transition is missing for state q_0 . Like for NFA, a relation is fine. This also allows to have multiple transitions for given a state, input or τ , and stack.

Definition 3.2 (Push-down automaton). A *push-down automaton* (PDA) is a septuple $P = (Q, \Sigma, \Delta, \emptyset, \rightarrow, q_0, F)$ where

- Q is a finite set of states,
- Σ is a finite input alphabet with $\tau \notin \Sigma$,
- Δ is a finite data alphabet or stack alphabet,
- $\emptyset \notin \Delta$ a special symbol denoting the empty stack,
- $\rightarrow \subseteq Q \times \Sigma_\tau \times \Delta_\emptyset \times \Delta^* \times Q$, where $\Sigma_\tau = \Sigma \cup \{\tau\}$ and $\Delta_\emptyset = \Delta \cup \{\emptyset\}$, is a finite set of *transitions* or *steps*,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states.

We use x and y for strings in Δ^* denoting the content of the stack. For non-empty strings the leftmost symbol indicates the topmost element of the stack. If $(q, a, d, x, q') \in \rightarrow$ with $d \in \Delta$, we write $q \xrightarrow{a[d/x]} q'$, and this means that the machine, when it is in state q and d is the top element of the stack, can consume the input symbol a , replace the top element d by the string x and thereby move to state q' . Likewise, if $(q, a, \emptyset, x, q') \in \rightarrow$ we write $q \xrightarrow{a[\emptyset/x]} q'$ meaning that the machine, when it is in state q and the stack is empty, can consume the input symbol a , put the string x on the stack and thereby move to state q' . In steps $q \xrightarrow{\tau[d/x]} q'$ and $q \xrightarrow{\tau[\emptyset/x]} q'$, no input symbol is consumed, only the stack is modified (if $x \neq d$ and $x \neq \varepsilon$, respectively).

Note the occurrence of Δ_\emptyset vs. Δ^* for the transition relation \rightarrow . As a special case, for a transition of the form $q \xrightarrow{a[\emptyset/\varepsilon]} q'$, the symbol a is read from input, but the stack remains empty.

Example 3.3. Consider again Figure 3.3. The figure depicts a push-down automaton $P = (Q, \Sigma, \Delta, \emptyset, \rightarrow, q_0, \{q_3\})$ with $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, and $\Delta = \{1\}$ as set of states, input alphabet and stack alphabet, respectively, transitions

$$q_0 \xrightarrow{a[\emptyset/1]} q_1, \quad q_1 \xrightarrow{a[1/11]} q_1, \quad q_1 \xrightarrow{b[1/\varepsilon]} q_2, \quad q_2 \xrightarrow{b[1/\varepsilon]} q_2, \quad q_2 \xrightarrow{\tau[\emptyset/\varepsilon]} q_3$$

initial state q_0 , and the singleton $\{q_3\}$ as the set of final states.

Also for PDA we introduce the notion of a configuration for the description of computations. Apart from the current state and the string left on input, we need to keep track of the stack.

Definition 3.4 (Configuration). Let $P = (Q, \Sigma, \Delta, \emptyset, \rightarrow, q_0, F)$ be a push-down automaton. A configuration or instantaneous description (ID) of P is a triple $(q, w, x) \in Q \times \Sigma^* \times \Delta^*$. The relation $\vdash_P \subseteq (Q \times \Sigma^* \times \Delta^*) \times (Q \times \Sigma^* \times \Delta^*)$ is given as follows.

- (i) $(q, aw, dy) \vdash_P (p, w, xy)$ if $q \xrightarrow{a[d/x]} p$

- (ii) $(q, w, dy) \vdash_P (p, w, xy)$ if $q \xrightarrow{\tau[d/x]} p$
- (iii) $(q, aw, \varepsilon) \vdash_P (p, w, x)$ if $q \xrightarrow{a[\emptyset/x]} p$
- (iv) $(q, w, \varepsilon) \vdash_P (p, w, x)$ if $q \xrightarrow{\tau[\emptyset/x]} p$.

The ‘derives’ relation \vdash_P^* is the reflexive and transitive closure of the relation \vdash_P .

In the definition we see, in line with the convention for strings x and y indicating stack content made above, that if $x = d_1 \cdots d_n \in \Delta^*$ then the symbol d_1 will be on top of the stack. The symbol at position i from the top will be d_i , for $1 \leq i \leq n$, and the symbol d_n will be at the bottom of the stack.

Example 3.5. For the push-down automaton of Figure 3.3 we have, e.g.,

$$(q_0, aabb, \varepsilon) \vdash_P (q_1, abb, 1) \vdash_P (q_1, bb, 11) \vdash_P (q_2, b, 1) \vdash_P (q_2, \varepsilon, \varepsilon) \vdash_P (q_3, \varepsilon, \varepsilon)$$

Thus $(q_0, aabb, \varepsilon) \vdash_P^* (q_3, \varepsilon, \varepsilon)$ and also $(q_0, aabb, \varepsilon) \vdash_P^* (q_2, b, 1)$ and $(q_1, abb, 1) \vdash_P^* (q_2, \varepsilon, \varepsilon)$.

For DFA and NFA we know that a derivation is independent of the input that is not touched, see Lemma ??b and Lemma ??. For PDA we can include part of the stack in this consideration. The stack items that are not inspected do not influence the computation. We first consider a one-step derivation, and next generalize to multi-step derivations. Regarding the untouched part of the stack y it is important that it remain covered, since the top element of the stack can be inspected by the PDA. Therefore, in the lemma, the strings x and x_i above y need to be non-empty.

Lemma 3.6. Let $P = (Q, \Sigma, \Delta, \emptyset, \rightarrow, q_0, F)$ be a push-down automaton.

- (a) For $w, w', v \in \Sigma^*$, $q, q' \in Q$, $x, x', y \in \Delta^*$ with $x \neq \varepsilon$, it holds that

$$(q, wv, xy) \vdash_P (q', w'v, x'y) \iff (q, w, x) \vdash_P (q', w', x')$$

- (b) For $n \geq 0$, $w_i \in \Sigma^*$, $q_i \in Q$, $x_i \in \Delta^*$ with $x_i \neq \varepsilon$, for $1 \leq i < n$, $v \in \Sigma^*$ and $y \in \Delta^*$, it holds that

$$\begin{aligned} (q_0, w_0v, x_0y) \vdash_P (q_1, w_1v, x_1y) \vdash_P \dots \vdash_P (q_n, w_nv, x_ny) \\ \iff \\ (q_0, w_0, x_0) \vdash_P (q_1, w_1, x_1) \vdash_P \dots \vdash_P (q_n, w_n, x_n) \end{aligned}$$

Proof. (a) We exploit Definition 3.4. If $(q, wv, xy) \vdash_P (q', w'v, x'y)$ with $x \neq \varepsilon$ then either we have (i) $q \xrightarrow{a[d/\bar{x}]} q'$ and $w = aw'$, $x = dx''$, $x' = \bar{x}x''$ for suitable \bar{x} and x'' (by clause (i) of Definition 3.4), or we have (ii) $q \xrightarrow{\tau[d/\bar{x}]} q'$ and $w = w'$, $x = dx''$, $x' = \bar{x}x''$ for

suitable \bar{x} and x'' (by clause (ii) of Definition 3.4). So, either $(q, w, x) = (q, aw', dx'') \vdash_P (q', w', \bar{x}x'') = (q', w', x')$ or $(q, w, x) = (q, w, dx'') \vdash_P (q', w, \bar{x}x'') = (q', w', x')$.

Reversely, if $(q, w, x) \vdash_P (q', w', x')$ with $x \neq \varepsilon$, then either (i) $q \xrightarrow{a[d/\bar{x}]} q'$ and $w = aw', x = dx'', x' = \bar{x}x''$ for suitable \bar{x} and x'' , or we have (ii) $q \xrightarrow{\tau[d/\bar{x}]} q'$ and $w = w', x = dx'', x' = \bar{x}x''$ for suitable \bar{x} and x'' . It follows that $wv = aw'v$, $xy = dx''y$, $x'y = \bar{x}x''y$, or $wv = w'v$, $xy = dx''y$, $x'y = \bar{x}x''y$. Thus $(q, wv, xy) \vdash_P (q', w'v, x'y)$.

(b) By induction on $n \geq 1$ using part (a). \square

With the notion of an instantaneous description in place and the derivation relation \vdash_P given, we can define the language accepted by a PDA.

Definition 3.7. Let $P = (Q, \Sigma, \Delta, \emptyset, \rightarrow, q_0, F)$ be a push-down automaton. The language $\mathcal{L}(P)$, called the language *accepted* by the push-down automaton P , is given by

$$\{ w \in \Sigma^* \mid \exists q \in F \exists x \in \Delta^*: (q_0, w, \varepsilon) \vdash_P^* (q, \varepsilon, x) \}$$

Note that we require the input string w to be processed completely, and we require q to be a final state of P , i.e. $q \in F$, but the string x can be any stack content.

Example 3.8. Consider once more the push-down automaton P of Figure 3.3. We verify $\mathcal{L}(P) = \{ a^n b^n \mid n \geq 1 \}$ by means of a so-called invariant table.

state q	input w	stack x	
q_0	ε	ε	
q_1	a^n	1^n	$1 \leq n$
q_2	$a^n b^m$	1^{n-m}	$1 \leq m \leq n$
q_3	$a^n b^n$	ε	$1 \leq n$

The first column of the table lists the states in Q . The second column lists the input by which the state is reached (starting from the initial state with empty stack). The third column lists the contents of the stack after the state is reached while reading the input of the second column. The last column provides further constraints and relationships of indices involved. If a triple (q, w, x) is listed, then it holds that $(q_0, w, \varepsilon) \vdash_P^* (q, \varepsilon, x)$.

For the PDA P of Figure 3.3 the first row is obvious: Starting in state q_0 (first column) with empty stack ε (third column) the PDA immediately leaves q_0 . So, P will only be in q_0 when no input is given, i.e., when the input equals ε (second column). The second row captures that P is in state q_1 after reading a^n , for $n > 0$. The first a is read on the transition from q_0 to q_1 , therefore $n > 0$ when in state q_1 . Possibly more a 's are read on execution of the transition from q_1 to itself. In q_1 the stack contains as many symbols 1 as symbols a have been read so far; the index n is the same for the input column displaying a^n and the stack column displaying 1^n . Upon reading a symbol b in state q_1 , hence with the stack non-empty, the PDA moves from state q_1 to state q_2 . More b 's can be read, but no more than the number of symbols 1 on the stack. For

each b read, one symbol 1 is popped off the stack. Thus, if the string b^m is read, the string 1^m is taken from the stack leaving 1^{n-m} on the stack (where $n-m \geq 0$). When the stack has become empty, the transition from q_2 to q_3 becomes enabled. There no further input can be read. As the stack was empty upon leaving q_2 , all symbols 1 must have been popped off. Thus, as many b 's have been read as there were symbols a before.

Now q_3 is the only final state of P . From the table we see that the input read to reach q_3 starting from the initial state q_0 with empty stack is of the form $a^n b^n$ for some $n \geq 1$. Thus $(q_0, w, \varepsilon) \vdash_P^* (q_3, \varepsilon, x)$ iff $w = a^n b^n$ with $n \geq 1$. This proves that $\mathcal{L}(P)$ is the language as claimed.

Example 3.9. Let us construct a push-down automaton for the non-regular language $L = \{ ww^R \mid w \in \{a, b\}^* \}$. The input alphabet is $\{a, b\}$. It is convenient to use a and b as stack symbols as well. So $\Delta = \{a, b\}$. In the initial state, a string is read and put on the stack. At some point the PDA guesses to be half-way the input, right after the string w and before the string w^R . Therefore, the PDA can switch non-deterministically to the second state, where the stack items will be compared with the input one by one. Note that the stack will be read in reversed order now as a stack is last-in first-out. Termination takes place when the stack is empty again. The above is implemented by the push-down automaton P given in Figure 3.4. In this figure we have $d \in \Delta$, so d is either a or b .

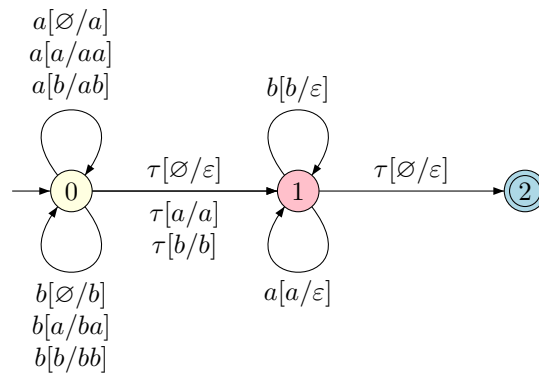
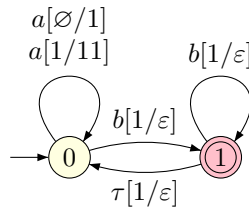
The invariant table belonging to P is as follows. Recall, a triple (q, w, x) is listed in the invariance table iff it holds that $(q_0, w, \varepsilon) \vdash_P^* (q, \varepsilon, x)$.

state q	input w	stack x	
q_0	w	w^R	$w \in \{a, b\}^*$
q_1	wv	u	$w \in \{a, b\}^*, vu = w^R$
q_2	wv	ε	$w \in \{a, b\}^*, v = w^R$

In q_0 the string w that is read, is stored on the stack in reversed order because of the last-in first-out regime of the stack. After the non-deterministic switch from state q_0 to state q_1 which does not affect the input nor the stack, in q_1 a symbol can only be read if it matches the symbol on top of the stack. So the extra input v , which equals what is already popped off the stack, and the remainder of the stack u together form w^R . Since q_1 is left only if the stack is empty, it must be the case in q_2 that $u = \varepsilon$, hence $v = w^R$. Therefore, only words of the form ww^R are accepted. As any string can be read initially, including the empty string ε , we conclude that P indeed accepts L .

Exercises for Section 3.1

Exercise 3.1.1. Consider the following PDA.

Figure 3.4: A PDA accepting $\{ ww^R \mid w \in \{a, b\}^* \}$ 

Compute all maximal derivation sequences, starting from the initial state q_0 , for the following inputs:

- (a) ab ;
- (b) aab ;
- (c) $aaabbb$.

A maximal derivation sequence of a PDA P for a string w is a sequence

$$(q_0, w_0, x_0) \vdash_P (q_1, w_1, x_1) \vdash_P \dots (q_{n-1}, w_{n-1}, x_{n-1}) \vdash_P (q_n, w_n, x_n) \not\vdash_P$$

where $q_0, q_1, \dots, q_{n-1}, q_n$ are states of P with q_0 its initial state, $w_0, w_1, \dots, w_{n-1}, w_n$ strings over the input alphabet of P with w_0 equal to w , and $x_0, x_1, \dots, x_{n-1}, x_n$ strings over the stack alphabet of P with x_0 equal to ε , the empty stack.

Exercise 3.1.2. Construct a push-down automaton and give an invariant table for the following languages over the input alphabet $\Sigma = \{a, b, c\}$. Also provide an invariant table for these PDA.

- (a) $L_1 = \{ a^n b^m c^{n+m} \mid n, m \geq 0 \}$;
- (b) $L_2 = \{ a^{n+m} b^n c^m \mid n, m \geq 0 \}$;
- (c) $L_3 = \{ a^n b^{n+m} c^m \mid n, m \geq 0 \}$;

Exercise 3.1.3. Give a push-down automaton for each of the following languages. Also provide an invariant table for these PDA.

- (a) $L_4 = \{ a^n b^{2n} \mid n \geq 0 \}$;
- (b) $L_5 = \{ a^n b^m \mid m \geq n \geq 1 \}$;
- (c) $L_6 = \{ a^n b^m \mid 2n = 3m + 1 \}$;
- (d) $L_7 = \{ a^n b^m \mid m, n \geq 0, m \neq n \}$.

Exercise 3.1.4. (a) Give a push-down automaton for the language

$$L_8 = \{ w \in \{a, b\}^* \mid \#_a(w) \neq \#_b(w) \}$$

(b) Give a push-down automaton for the language

$$L_9 = \{ w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_b(w) \vee \#_b(w) \neq \#_c(w) \}$$

3.2 Context-free grammars

Consider again the language $L = \{ a^n b^n \mid n > 0 \}$. Clearly $ab \in L$; take $n = 1$. Now, for an arbitrary element $w \in L$, the string awb is also an element of L : if $w = a^n b^n$ for some $n > 0$, then $awb = a^{n+1} b^{n+1}$ and $n+1 > 0$. Thus, writing the symbol S to represent an element of L , we can write

$$S \rightarrow ab \text{ and } S \rightarrow aSb \tag{3.1}$$

We may read $S \rightarrow ab$ as “ S produces ab ”, and $S \rightarrow aSb$ as “ S produces aSb ”. The expressions $S \rightarrow ab$ and $S \rightarrow aSb$ are called *production rules*, or more precisely production rules for S .

Production rules are typically used in a setting with so-called *productions* or *derivations*. If uSv is a string, say with $u, v \in \{S, a, b\}^*$, then we have the productions $uSv \Rightarrow uabv$ and $uSv \Rightarrow uaSbv$, based on the production rules $S \rightarrow ab$ and $S \rightarrow aSb$, respectively. Since the applicability of the production rules does not depend on the context comprised by the strings u and v , but only the symbol S , the production scheme is referred to as *context-free*.

The production rules of Equation (3.1) can be applied to strings containing S repeatedly, yielding sequences of zero, one or more productions or derivations. E.g., we have

$$\begin{aligned} S &\Rightarrow ab \\ S &\Rightarrow aSb \Rightarrow aabb \\ S &\Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb \end{aligned}$$

Such sequences are called production sequences or derivation sequences. We write, e.g., $S \Rightarrow^* ab$, $S \Rightarrow^* aabb$, $S \Rightarrow^* aaabbb$ production for strings in L , but also $aSb \Rightarrow^* aaSbb$ and $aSb \Rightarrow^* aSb$ of strings containing S .

Definition 3.10 (Context-free grammar). A context-free grammar (CFG) is a four-tuple $G = (V, T, R, S)$ where

1. V is a non-empty finite set of variables or non-terminals,
2. T is a finite set of terminals, disjoint from V ,
3. $R \subseteq V \times (V \cup T)^*$ is a finite set of production rules, and
4. $S \in V$ is the start symbol.

We use CFG as shorthand for the notion of a context-free grammar. As we shall see, the set of terminals T corresponds to the input alphabet Σ of a PDA.

If G is a context-free grammar with set of production rules R , we suggestively write $A \rightarrow \alpha$, read “ A produces α ”, if $(A, \alpha) \in R$. Sometimes we write $A \rightarrow_G \alpha$ to stress that the production rule belongs to the grammar G . The production rule is called a production rule for the non-terminal A . If $A \rightarrow \alpha_1, A \rightarrow \alpha_2, \dots, A \rightarrow \alpha_n$ are all the production rules in R for A , we also write $A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$.

Example 3.11. Define $G = (V, T, R, S)$ with $V = \{S\}$, $T = \{a, b\}$ and R consisting of

$$S \rightarrow ab \text{ and } S \rightarrow aSb$$

Then G is a context-free grammar with non-terminal S , terminals a and b and the two production rules above. The start symbol of G is S .

Example 3.12. In view of the shorthand notation given above, the following describes seven production rules:

$$\begin{aligned} S &\rightarrow ABC \\ A &\rightarrow \varepsilon \mid aA \\ B &\rightarrow \varepsilon \mid Bb \\ C &\rightarrow c \mid cC \end{aligned}$$

The rules implicitly define a context-free grammar $G = (V, T, R, S)$. The following conventions apply: (i) Commonly, capital letters indicate non-terminals. Thus, for G we have $V = \{S, A, B, C\}$. Note that the elements of V are precisely the symbols that occur at the left-hand side of the rules above (strictly speaking this is not required). (ii) Usually, lower-case letters indicate terminals. Thus for G we have $T = \{a, b, c\}$. Note that the elements of T are exactly the letters that are not in V and occur at the right-hand side of the rules (again, strictly speaking this is not required). Note, ε indicates the empty string, and hence is not in T , nor in V . (iii) Obviously, the set of production rules R consists of the rules listed. (iv) A customary choice for the start symbol is S which indeed is an element of the set of non-terminals V here.

Definition 3.13 (Production, production sequence, language of a CFG). Let $G = (V, T, R, S)$ be a context-free grammar.

- Let $A \rightarrow \alpha \in R$ be a production rule of G . Thus $A \in V$ and $\alpha \in (V \cup T)^*$. Let $\gamma = \beta_1 A \beta_2$ be a string in which A occurs. Put $\gamma' = \beta_1 \alpha \beta_2$. We say that from the string γ the production rule $A \rightarrow \alpha$ produces the string γ' , notation $\gamma \Rightarrow_G \gamma'$.
- A production sequence or derivation is a sequence $(\gamma_i)_{i=0}^n$ such that $\gamma_{i-1} \Rightarrow_G \gamma_i$, for $1 \leq i \leq n$. Often we write

$$\gamma_0 \Rightarrow_G \gamma_1 \Rightarrow_G \dots \Rightarrow_G \gamma_{n-1} \Rightarrow_G \gamma_n$$

The length of this production sequence is n , the number of productions. In case $\gamma = \gamma_0$ and $\gamma' = \gamma_n$ we also write $\gamma \Rightarrow_G^* \gamma'$.

- Let $A \in V$ be a variable of G . The language $\mathcal{L}_G(A)$ generated by G from A is given by

$$\mathcal{L}_G(A) = \{ w \in T^* \mid A \Rightarrow_G^* w \}$$

- The language $\mathcal{L}(G)$, the language generated by the CFG G , consists of all strings of terminals that can be produced from the start symbol S , i.e.

$$\mathcal{L}(G) = \mathcal{L}_G(S)$$

- A language L is called context-free, if there exists a context-free grammar G such that $L = \mathcal{L}(G)$.

Note, with respect to the grammar G , we have $\mathcal{L}(G) = \mathcal{L}_G(S) = \{ w \in T^* \mid S \Rightarrow_G^* w \}$. When the grammar G is clear it may be dropped as a subscript.

Example 3.14. Consider again the grammar G given in Example 3.11. As eluded to at the beginning of this section, $S \Rightarrow_G ab$ and $aaSbb \Rightarrow_G aaabbb$ are productions of G , but also $SS \Rightarrow_G SaSb$ is a production of G . Example production sequences are $S \Rightarrow_G ab$ and $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaabbb$, thus $S \Rightarrow_G^* ab$, and $S \Rightarrow_G^* aaabbb$. Also $S \Rightarrow aSb \Rightarrow aaSbb$ and $SS \Rightarrow_G SaSb \Rightarrow_G abaSb$ are production sequences of G . We claim $\mathcal{L}(G) = \{ a^n b^n \mid n \geq 1 \}$. This can be shown by proving for $L = \{ a^n b^n \mid n \geq 1 \}$ two set inclusions: (i) the inclusion $\mathcal{L}(G) \subseteq L$ by induction on the length of a production sequence, and (ii) the inclusion $L \subseteq \mathcal{L}(G)$ by induction on the parameter n .

Proof of the claim ($\mathcal{L}(G) \subseteq L$) We show by induction on n , if $S \Rightarrow_G^n w$ and $w \in T^*$ then $w \in L$. Basis: $n = 0$. Then $S = w$ while $S \notin T^*$, hence there is nothing to show. Induction step, $n + 1$: Suppose $S \Rightarrow_G^{n+1} w$. We have either $S \Rightarrow_G ab \Rightarrow_G^n w$, in case the production rule $S \rightarrow ab$ was used, or $S \Rightarrow_G aSb \Rightarrow_G^n w$, in case the production rule $S \rightarrow aSb$ was used. In the first case clearly $w \in L$. As to the second case, by Lemma 3.15c below it follows that $w = avb$ for some string $v \in T^*$ and $S \Rightarrow_G^n v$. By induction hypothesis $v \in L$, i.e. $v = a^m b^m$ for some suitable $m \geq 1$. Therefore $w = avb = a^{m+1} b^{m+1}$ and $w \in L$ too.

($L \subseteq \mathcal{L}(G)$) We show by induction on n , $S \Rightarrow_G^* a^n b^n$, for $n \geq 1$. Basis, $n = 1$: Clear, $S \Rightarrow_G ab$ based on the production rule $S \rightarrow ab$. Induction hypothesis, $n + 1$: We have $a^n b^n \in L$. By induction hypothesis it follows that $S \Rightarrow_G^* a^n b^n$. By Lemma 3.15b we obtain $aSb \Rightarrow_G^* aa^n b^n b$. Therefore $S \Rightarrow_G aSb \Rightarrow_G^* a^{n+1} b^{n+1}$, as was to be shown.

In the example above we called to Lemma 3.15 below to split and combine production sequences. This technical lemma summarizes the context independence of the machinery introduced and is used in many situations.

Lemma 3.15. Let $G = (V, T, R, S)$ be a context-free grammar.

- (a) Let $x, x', y, y' \in (V \cup T)^*$. If $x \Rightarrow_G^n x'$ and $y \Rightarrow_G^m y'$ then $xy \Rightarrow_G^{n+m} x'y'$.
- (b) Let $k \geq 1$, $X_1, \dots, X_k \in V \cup T$, $n_1, \dots, n_k \geq 0$, and $x_1, \dots, x_k \in (V \cup T)^*$. If $X_1 \Rightarrow_G^{n_1} x_1, \dots, X_k \Rightarrow_G^{n_k} x_k$, then $X_1 \cdots X_k \Rightarrow_G^n x_1 \cdots x_k$ where $n = n_1 + \dots + n_k$.
- (c) Let $X_1, \dots, X_k \in V \cup T$ and $x \in (V \cup T)^*$. If $X_1 \cdots X_k \Rightarrow_G^n x$ then there exist $n_1, \dots, n_k \geq 0$, and $x_1, \dots, x_k \in (V \cup T)^*$ such that $n = n_1 + \dots + n_k$, $X_1 \Rightarrow_G^{n_1} x_1, \dots, X_k \Rightarrow_G^{n_k} x_k$, and $x = x_1 \cdots x_k$.

Proof. (a) By induction $n+m$, if $x \Rightarrow_G^n x'$ and $y \Rightarrow_G^m y'$ then $xy \Rightarrow_G^{n+m} x'y'$. Basis, $n+m = 0$: Trivial. We have $x = x', y = y'$ and $xy \Rightarrow_G^0 xy$. Induction step, $n+m+1$: If $x \Rightarrow_G \hat{x} \Rightarrow_G^n x'$ and $y \Rightarrow_G^m y'$ we have $x = x_1 A x_2, \hat{x} = x_1 \bar{x} x_2$ for a production rule $A \rightarrow \bar{x}$ of G . By induction hypothesis $\hat{x}y \Rightarrow_G^{n+m} x'y'$. We also have $xy = x_1 A x_2 y \Rightarrow_G x_1 \bar{x} x_2 y = \hat{x}y$. Thus $xy \Rightarrow_G^{n+m+1} x'y'$. If $y \Rightarrow_G \hat{y} \Rightarrow_G^m y'$, then by similar reasoning it follows that $xy \Rightarrow_G x\hat{y}$ and $x\hat{y} \Rightarrow_G^{n+m} x'y'$. From this it follows that $xy \Rightarrow_G^{n+m+1} x'y'$, as was to be shown.

(b) By induction on k , using part (a).

(c) By induction on n , if $X_1 \cdots X_k \Rightarrow_G^n x$ then $X_1 \Rightarrow_G^{n_1} x_1, \dots, X_k \Rightarrow_G^{n_k} x_k$, and $x = x_1 \cdots x_k$ for suitable n_1, \dots, n_k , and x_1, \dots, x_k . Basis, $n = 0$: If $X_1 \cdots X_k \Rightarrow_G^0 x$, then $X_1 \cdots X_k = x$. Choose $x_1 = X_1, \dots, x_k = X_k$ and $n_1, \dots, n_k = 0$. Then clearly $n_1 + \dots + n_k = 0 = n$, $X_i \Rightarrow_G^* x_i$, for $1 \leq i \leq k$, and $x_1 \cdots x_k = X_1 \cdots X_k = x$. Induction step, $n + 1$: Suppose

$$X_1 \cdots X_k \Rightarrow_G X_1 \cdots X_{i-1} Y_1 \cdots Y_\ell X_{i+1} \cdots X_k \Rightarrow_G^n x$$

for some i , $1 \leq i \leq k$, such that $X_i \Rightarrow_G Y_1 \cdots Y_\ell$. By induction hypothesis exist indices $n_1, \dots, n_{i-1}, m_1, \dots, m_\ell, n_{i+1}, \dots, n_k$, and strings $x_1, \dots, x_{i-1}, y_1, \dots, y_\ell, x_{i+1}, \dots, x_k$ such that $X_j \Rightarrow_G^{n_j} x_j$ for $1 \leq j < i$ or $i < j \leq k$, $Y_h \Rightarrow_G^{m_h} y_h$ for $1 \leq h \leq \ell$, and $n = n_1 + \dots + n_{i-1} + m_1 + \dots + m_\ell + n_{i+1} + \dots + n_k$, and $x = x_1 \cdots x_{i-1} y_1 \cdots y_\ell x_{i+1} \cdots x_k$. Put $m = m_1 + \dots + m_\ell$, $n_i = m + 1$, and $x_i = y_1 \cdots y_\ell$. Then we have $n + 1 = n_1 + \dots + n_k$, and

$$X_i \Rightarrow_G Y_1 \cdots Y_\ell \Rightarrow_G^m y_1 \cdots y_\ell = x_i$$

Thus, by part (b), $X_i \Rightarrow_G^{n_i} x_i$. Thus $X_j \Rightarrow_G^{n_j} x_j$ for $1 \leq j \leq k$, including i , and $x = x_1 \cdots x_{i-1} y_1 \cdots y_\ell x_{i+1} \cdots x_k = x_1 \cdots x_k$, as was to be shown. \square

Example 3.16. The so-called parentheses language $L_{()} \subseteq \{(\,,\,)\}^*$ is the language of strings of balanced parentheses: for a string of parentheses $w = b_1 \cdots b_n \in L_{()}$ it holds that for an arbitrary prefix $v \preceq w$, say $v = b_1 \cdots b_m$, $0 \leq m \leq n$, it holds that $\#_{()}(v) \geq \#_{)}(v)$, and for w it holds that $\#_{()}(w) = \#_{)}(w)$. Thus, the i -th right parenthesis occurs

only after the i -th left parenthesis, and for the left parenthesis with number i there is a right parenthesis with number i .

We have $()(()) \in L_{()} by inspection of the string $(_1)_1(2(3)_2)_3$ where we number the left and right parentheses. But $w = ()()$ and $v = (()()$ are not in $L_{() as the annotated versions $(_1)_1)_2(2$ and $(_1(2)_1)_3$ show; for w we have that $)_2$ occurs before $(_2$, while for v we have that there are 3 left parentheses but only 1 right parenthesis. Note that the empty string ε meets the requirements, although it has no parenthesis at all.$$

A possible grammar G for $L_{() is given by the rules$

$$S \rightarrow \varepsilon, S \rightarrow SS, S \rightarrow (S)$$

With respect to G we have the production sequence

$$S \Rightarrow_G SS \Rightarrow_G S(S) \Rightarrow_G S((S)) \Rightarrow_G S(()) \Rightarrow_G (S)(()) \Rightarrow_G ()(())$$

for the string $()(()) \in L_{() , but also the production sequence$

$$S \Rightarrow_G SS \Rightarrow_G (S)S \Rightarrow_G ()S \Rightarrow_G ()(S) \Rightarrow_G ()((S)) \Rightarrow_G ()(())$$

as well as

$$S \Rightarrow_G SS \Rightarrow_G (S)S \Rightarrow_G (S)(S) \Rightarrow_G ()(S) \Rightarrow_G ()((S)) \Rightarrow_G ()(())$$

There are several more derivations for $()(())$ with respect to G .

Definition 3.17. Let $G = (V, T, R, S)$ be a context-free grammar. A production $\gamma \Rightarrow_G \gamma'$ is called a leftmost production if γ' is obtained from γ by application of a production rule of G on the leftmost variable occurring in γ , i.e., $\gamma = wA\beta$, $A \rightarrow \alpha$ a rule of G , $\gamma' = w\alpha\beta$ for some $w \in T^*$, $A \in V$, and $\alpha, \beta \in (V \cup T)^*$. Notation, $\gamma \xRightarrow{\ell}_G \gamma'$. A leftmost derivation of G is a production sequence $(\gamma_i)_{i=0}^n$ of G for which every production is leftmost. Notation, $\gamma \xRightarrow{\ell}_G^* \gamma'$.

Similarly one can define the notion of a rightmost production and a rightmost derivation for a CFG G .

We often write $\gamma \xRightarrow{\ell}_G^* \gamma'$ rather than $\gamma \xRightarrow{\ell}_G^* \gamma'$, if the grammar G is clear from the context. The notion of a leftmost derivation is useful to select a particular production sequence from the many that may produce a string. However, it is not generally the case that there exists precisely one leftmost derivation sequence from a string γ to a string γ' . Although, in a leftmost derivation there is no freedom to select the variable that is used for the production, there may be freedom to select the production rule to use. Consider, for example the grammar

$$S \rightarrow AB, A \rightarrow aa \mid aac, B \rightarrow bb \mid cbb$$

which allows four complete leftmost derivations starting from S , viz.

$$\begin{aligned} S &\xRightarrow{\ell} AB \xRightarrow{\ell} aaB \xRightarrow{\ell} aabb \\ S &\xRightarrow{\ell} AB \xRightarrow{\ell} aaB \xRightarrow{\ell} aacbb \\ S &\xRightarrow{\ell} AB \xRightarrow{\ell} aacB \xRightarrow{\ell} aacbb \\ S &\xRightarrow{\ell} AB \xRightarrow{\ell} aacB \xRightarrow{\ell} aaccbb \end{aligned}$$

Note, there are two leftmost derivations of the string $aacbb$.

In the next section we will show that if there is a production sequence from γ to γ' for some grammar G , then there is also a leftmost derivation from γ to γ' for G , i.e. if $\gamma \Rightarrow_G^* \gamma'$ then $\gamma \xRightarrow_G^* \gamma'$.

Theorem 3.18. If L is a regular language, then L is also context-free.

Proof. Let $D = (Q, \Sigma, \delta, q_0, F)$ be a deterministic automaton that accepts L . Define the grammar $G = (Q, \Sigma, R, q_0)$ where R is given by

$$R = \{ q \rightarrow aq' \mid \delta(q, a) = q' \} \cup \{ q \rightarrow \varepsilon \mid q \in F \}$$

We claim that $L = \mathcal{L}(G)$.

Suppose $w \in L$ and $w = a_1a_2 \dots a_n$. Let $(q_0, a_1a_2 \dots a_n) \vdash_D (q_1, a_2 \dots a_n) \vdash_D \dots \vdash_D (q_{n-1}, a_n) \vdash_D (q_n, \varepsilon)$ with $q_n \in F$ be an accepting transition sequence of D for w . Then we have a production sequence

$$\gamma_0 \Rightarrow_G \gamma_1 \Rightarrow_G \dots \Rightarrow_G \gamma_{n+1}$$

of G for w with $\gamma_0 = q_0$, $\gamma_i = a_1 \dots a_i q_i$ for $0 \leq i \leq n$, and $\gamma_{n+1} = a_1 \dots a_n$. Since $\delta(q_{i-1}, a_i) = q_i$ we have $q_{i-1} \rightarrow a_i q_i$, and hence $\gamma_{i-1} = a_1 \dots a_{i-1} q_{i-1} \Rightarrow_G qa_1 \dots a_{i-1} a_i q_i = \gamma_i$ for $1 \leq i \leq n$ and $\gamma_n = a_1 \dots a_n q_n \Rightarrow_G a_1 \dots a_n = \gamma_{n+1}$. Thus $w \in \mathcal{L}(G)$ and $L \subseteq \mathcal{L}(G)$.

Suppose $w \in \mathcal{L}(G)$. Say $\gamma_0 \Rightarrow_G \gamma_1 \Rightarrow_G \dots \Rightarrow_G \gamma_{n+1}$ for some $n \geq 0$ with $\gamma_0 = q_0$ and $\gamma_{n+1} = w$. Then there exist $a_1, \dots, a_n \in \Sigma$ and $q_0, \dots, q_n \in Q$ such that $\gamma_i = a_1 \dots a_i q_i$, for $0 \leq i \leq n$ and $\gamma_{n+1} = w = a_1 \dots a_n$. Moreover, $\delta(q_{i-1}, a_i) = q_i$, for $1 \leq i \leq n$, and $q_n \in F$. Therefore we have

$$(q_0, a_1a_2 \dots a_n) \vdash_D (q_1, a_2 \dots a_n) \vdash_D \dots \vdash_D (q_{n-1}, a_n) \vdash_D (q_n, \varepsilon) \text{ and } q_n \in F$$

It follows that $w \in L$ and $\mathcal{L}(G) \subseteq L$. □

As we shall see later, the language $L_{()}$ of Example 3.16 is not regular. So, the reverse of Theorem 3.18 does not hold.

Exercises for Section 3.2

Exercise 3.2.1. (Hopcroft, Motwani & Ullman 2001) Consider the context-free grammar G given by the production rules

$$\begin{aligned} S &\rightarrow XbY \\ X &\rightarrow \varepsilon \mid aX \\ Y &\rightarrow \varepsilon \mid aY \mid bY \end{aligned}$$

that generates the language of the regular expression $a^*b(a+b)^*$. Give leftmost and rightmost derivations for the following strings:

- (a) $aabab$;
- (b) $baab$;
- (c) $aaabb$.

Exercise 3.2.2. Consider the context-free grammar G given by the production rules

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow \varepsilon \mid aA \\ B &\rightarrow \varepsilon \mid bB \end{aligned}$$

- (a) Put $L_A = \{a^n \mid n \geq 0\}$. Prove that $\mathcal{L}_G(A) = L_A$.
- (b) Put $L = \{a^n \mid n \geq 0\} \cup \{b^n \mid n \geq 0\}$. Prove that $\mathcal{L}(G) = L$.

Exercise 3.2.3. Give a context-free grammar for each of the following languages and prove them correct.

- (a) $L_1 = \{a^n b^m \mid n, m \geq 0, n \neq m\}$;
- (b) $L_2 = \{a^n b^m c^\ell \mid n, m, \ell \geq 0, n \neq m \vee m \neq \ell\}$;

Exercise 3.2.4. Give a construction, based on the number of operators, that shows that the language of every regular expression can be generated by a context-free grammar.

Exercise 3.2.5. Consider the context-free grammar G given by the production rules

$$S \rightarrow aS \mid Sb \mid a \mid b$$

Put $\mathcal{X} = \{a^n S b^m, a^n b^m \mid n, m \geq 0, n + m \geq 1\}$.

- (a) Prove $S \Rightarrow_G^k x$ implies $x \in \mathcal{X}$, for $x \in \{S, a, b\}^*$.
- (b) Prove that no string $w \in \mathcal{L}(G)$ has a substring ba .

Exercise 3.2.6. Put $L = \{ w \in \{a, b\}^* \mid \#_a(w) = \#_b(w) \}$.

- (a) (Hopcroft, Motwani & Ullman 2001) Consider the context-free grammar G given by the production rules

$$S \rightarrow aSbS \mid bSaS \mid \varepsilon$$

Prove that $\mathcal{L}(G) = L$.

- (b) Give an alternative CFG G' with four production rules such that $\mathcal{L}(G') = L$.

3.3 Chomsky normal form

In the previous section several examples of context-free grammars have been given. In this section we aim to come up with a specific format of a context-free grammar, called Chomsky normal form, that is a convenient starting point for algorithmic purposes. Subsequently, we will discuss (i) how to eliminate ε -productions $A \rightarrow \varepsilon$, (ii) how to eliminate so-called unit rules $A \rightarrow B$, and (iii) how to identify useless variables, all without affecting the language generated by the grammar. These procedures together yield a compact context-free grammar which is then easily converted to Chomsky normal form.

Elimination of ε -productions

The CNF $G = (\{S, A\}, \{a, b\}, \{S \rightarrow A \mid a \mid b, A \rightarrow \varepsilon\}, S)$ generates the language $\mathcal{L}(G) = \{\varepsilon, a, b\}$. However, simply deleting the ε -production $A \rightarrow \varepsilon$ from the set of production rules yields the grammar $G' = (\{S\}, \{a, b\}, \{S \rightarrow a \mid b\}, S)$, which generates a different language, $\mathcal{L}(G') = \{a, b\}$. So, in order to eliminate ε -productions properly, we need to go about it slightly more cautiously. In this respect, the relevant concept is the definition of a nullable variable.

Definition 3.19. A variable A of a context-free grammar G is called nullable if $A \Rightarrow_G^* \varepsilon$. The set of nullable variables of G is denoted by $\text{Null}(G)$.

Example 3.20. Consider the CNF $G = (\{S, A, B, C, D\}, \{a, b, c\}, R, S)$ with productions

$$S \rightarrow AB|c, \quad A \rightarrow \varepsilon, \quad B \rightarrow \varepsilon, \quad C \rightarrow \varepsilon|c, \quad D \rightarrow ab$$

Clearly, the variables A, B and C are nullable. Also the start symbol S is nullable since $S \Rightarrow_G^2 \varepsilon$. The variable D is not nullable. So, $\text{Null}(G) = \{S, A, B, C\}$.

Lemma 3.21. Let $G = (V, T, R, S)$ be a CFG. Define the subsets $Null_i \subseteq V$ by

$$\begin{aligned} Null_0 &= \emptyset \\ Null_{i+1} &= Null_i \cup \{ A \mid A \rightarrow X_1 \cdots X_\ell \in R, \forall j, 1 \leq j \leq \ell: X_j \in Null_i \} \end{aligned}$$

for $i \geq 0$, and put $Null = \bigcup_{i=0}^{\infty} Null_i$. Then it holds that $Null = Null(G)$.

Proof. ($Null \subseteq Null(G)$) It suffices to prove that $Null_i \subseteq Null(G)$ by induction on i . Basis, $i=0$: Clear, since $Null_0 = \emptyset$. Induction step, $i+1$: Suppose $A \in Null_{i+1} \setminus Null_i$. Then there is a rule $A \rightarrow X_1 \cdots X_\ell$ in R with $X_1, \dots, X_\ell \in Null_i$. By induction hypothesis, $X_1, \dots, X_\ell \in (G)$. Thus, $X_j \Rightarrow_G^* \varepsilon$ for $1 \leq j \leq \ell$. Therefore, $A \Rightarrow_G X_1 \cdots X_\ell \Rightarrow_G^* \varepsilon$. Hence $A \in Null(G)$.

($Null(G) \subseteq Null$) It suffices to prove that $A \Rightarrow_G^k \varepsilon$ implies $A \in Null_k$ by induction on k . Basis, $k=0$: Trivial, this situation does not occur. Induction step, $k+1$: Suppose $A \rightarrow_G X_1 \cdots X_\ell \Rightarrow_G^* \varepsilon$. Then $X_j \in V$ and $X_j \Rightarrow_G^{k_j} \varepsilon$ for some $k_j \geq 0$, for $1 \leq j \leq \ell$, such that $k_1 + \cdots + k_\ell = k$. By induction hypothesis, $X_i \in Null_{k_i}$, hence $X_i \in Null_k$. Since $A \rightarrow X_1 \cdots X_\ell$ is a rule of G , it follows that $A \in Null_{k+1}$. \square

Lemma 3.22. Let $G = (V, T, R, S)$ be a CFG. Suppose $B \rightarrow \varepsilon$ is a rule in R . Define the set of rules R' by

$$\begin{aligned} \hat{R} = \{ A \rightarrow \alpha_1 \cdots \alpha_n \mid A \rightarrow X_1 \cdots X_n \in R, \\ \forall i, 1 \leq i \leq n: \alpha_i = X_i \vee (X_i \in Null(G) \wedge \alpha_i = \varepsilon) \} \end{aligned}$$

and the CFG G' by $G' = (V, T, R', S)$ where $R' = \hat{R} \setminus \{ A \rightarrow \varepsilon \mid A \in V \}$. Then it holds that $\mathcal{L}(G) \setminus \{\varepsilon\} = \mathcal{L}(G') \setminus \{\varepsilon\}$.

Proof. First we prove the following claim, for arbitrary $A \in V$ and $w \in T^*$, $w \neq \varepsilon$.

$$A \Rightarrow_G^* w \quad \text{iff} \quad A \Rightarrow_{G'}^* w \quad (3.2)$$

Proof of the claim: (\Rightarrow) By induction on n , for $w \neq \varepsilon$, if $A \Rightarrow_G^n w$ then $A \Rightarrow_{G'}^* w$. Basis, $n = 0$: Trivial. This situation doesn't occur. Induction step, $n + 1$: Suppose $A \rightarrow_G X_1 \cdots X_k$, $X_i \Rightarrow_G^{n_i} w_i$ for some $n_i \geq 0$ and $w_i \in T^*$, for $1 \leq i \leq k$, such that $n_1 + \cdots + n_k = n$ and $w_1 \cdots w_k = w$ (see Lemma 3.15c). Put, for $1 \leq i \leq k$, $\alpha_i = \varepsilon$ if $w_i = \varepsilon$ and $\alpha_i = X_i$ if $w_i \neq \varepsilon$. Note, if $w_i = \varepsilon$ then $X_i \in Null(G)$. Therefore, we have that the rule $A \rightarrow_G X_1 \cdots X_k$ of G induces a rule $A \rightarrow_{G'} \alpha_1 \cdots \alpha_k$ in \hat{R} . Moreover, not all w_i are equal to ε , since $w_1 \cdots w_k = w \neq \varepsilon$. Thus, also $\alpha_1 \cdots \alpha_k \neq \varepsilon$. Hence, $A \rightarrow_{G'} \alpha_1 \cdots \alpha_k$ is a rule of G' . By induction hypothesis, we have $X_i \Rightarrow_{G'}^* w_i$, for $1 \leq i \leq k$ such that $w_i \neq \varepsilon$. Therefore we have $\alpha_i \Rightarrow_{G'}^* w_i$, for $1 \leq i \leq k$, where $\alpha_i \Rightarrow_{G'}^* w_i$ is the empty production sequence $\varepsilon \Rightarrow_{G'}^* \varepsilon$ if $w_i = \varepsilon$. By Lemma 3.15b we obtain $\alpha_1 \cdots \alpha_k \Rightarrow_{G'}^* w_1 \cdots w_k$. Thus,

$$A \Rightarrow_{G'}^* \alpha_1 \cdots \alpha_k \Rightarrow_{G'}^* w_1 \cdots w_k = w$$

which completes the induction step.

(\Leftarrow) By induction on n , for $w \neq \varepsilon$, if $A \Rightarrow_{G'}^n w$ then $A \Rightarrow_G^* w$. Basis, $n = 0$: Trivial. This situation doesn't occur. Induction step, $n + 1$: Suppose $A \rightarrow_{G'} Y_1 \cdots Y_\ell$ for some variables $Y_1, \dots, Y_\ell \in V$ such that $Y_i \Rightarrow_{G'}^{n_i} w_i$ for some $n_i \geq 0$, $w_i \in T^*$, for $1 \leq i \leq \ell$, with $n_1 + \cdots + n_\ell = n$ and $w_1 \cdots w_\ell = w$ (see Lemma 3.15c). Note, $w_i \neq \varepsilon$, for $1 \leq i \leq \ell$, since G' has no ε -productions.

By definition of \hat{R} , there exist, for some $k \geq 0$, variables X_1, \dots, X_k in V and a monotone injection $f : \{1, \dots, \ell\} \rightarrow \{1, \dots, k\}$ such that $A \rightarrow_G X_1 \cdots X_k$ and (i) $Y_i = X_{f(i)}$, for $1 \leq i \leq \ell$, and (ii) $X_j \in \text{Null}(G)$ if $j \notin \text{Img}(f) = \{f(i) \mid 1 \leq i \leq \ell\}$. In the first case, we have $X_{f(i)} \Rightarrow_{G'}^{n_i} w_i \neq \varepsilon$. Thus $X_{f(i)} \Rightarrow_G^* w_i$ by induction hypothesis. In the second case, we have $X_j \Rightarrow_G^* \varepsilon$ since $X_j \in \text{Null}(G)$. Therefore, we can pick $w'_1, \dots, w'_k \in T^*$ such that $X_j \Rightarrow_G^* w'_j$, for $1 \leq j \leq k$, and $w'_1 \cdots w'_k = w_1 \cdots w_\ell = w$: put $w'_j = w_i$ if $f(i) = j \in \{1, \dots, \ell\}$, put $w'_j = \varepsilon$ if $j \notin \text{Img}(f)$. It follows that

$$A \rightarrow_G X_1 \cdots X_k \rightarrow_G w'_1 \cdots w'_k = w_1 \cdots w_\ell = w$$

by Lemma 3.15b, which proves the claim.

Now, let S° be a fresh variable. Put $V^\circ = V \cup \{S^\circ\}$. Define $R^\circ = R' \cup \{S^\circ \rightarrow S\} \cup \{S^\circ \rightarrow \varepsilon \mid S \in \text{Null}(G)\}$. Finally, define G° by $G^\circ = (V^\circ, T, R^\circ, S^\circ)$. We show that $\mathcal{L}(G) = \mathcal{L}(G^\circ)$. We distinguish four cases: (i) Suppose $w \in \mathcal{L}(G)$, $w \neq \varepsilon$. Then $S \Rightarrow_G^* w$, and $S \Rightarrow_{G'}^* w$ by the claim. Since $R' \subseteq R^\circ$ and $S^\circ \Rightarrow_{G^\circ} S$, it follows that $S^\circ \Rightarrow_{G^\circ}^* w$ and $w \in \mathcal{L}(G^\circ)$. (ii) Suppose $w \in \mathcal{L}(G^\circ)$, $w \neq \varepsilon$. Then $S^\circ \Rightarrow_{G^\circ}^* w$ and, by definition of R° , $S \Rightarrow_{G'}^* w$. By the claim it follows that $S \Rightarrow_G^* w$. Hence, $w \in \mathcal{L}(G)$. (iii) If $\varepsilon \in \mathcal{L}(G)$, then $S \Rightarrow_G^* \varepsilon$. Thus, $S \in \text{Null}(G)$, $S^\circ \Rightarrow_{G^\circ} \varepsilon$ and $\varepsilon \in \mathcal{L}(G^\circ)$. (iv) If $\varepsilon \in \mathcal{L}(G^\circ)$, then $S^\circ \Rightarrow_{G^\circ}^* \varepsilon$. Assume $S^\circ \Rightarrow_{G^\circ} S \Rightarrow_{G'}^* \varepsilon$. Then we have $S \Rightarrow_{G'}^* \varepsilon$. But the set of rules R' of G' has no ε -productions, which yields a contradiction. It follows that $S \rightarrow_{G^\circ} \varepsilon$, thus, by definition of R° , $S \in \text{Null}(G)$. We conclude $S \Rightarrow_G^* \varepsilon$ and $\varepsilon \in \mathcal{L}(G)$. \square

Example 3.23. Consider the CNF $G = (\{S, A, B, C\}, \{a, b\}, R, S)$ where

$$S \rightarrow ABC, \quad A \rightarrow aA \mid \varepsilon, \quad B \rightarrow bB \mid \varepsilon, \quad C \rightarrow c \mid \varepsilon,$$

The set of production rules \hat{R} , obtained from R according to the lemma, is given by

$$\begin{aligned} S &\rightarrow ABC \mid AB \mid AC \mid BC \mid A \mid B \mid C \mid \varepsilon \\ A &\rightarrow aA \mid a \mid \varepsilon \\ B &\rightarrow bB \mid b \mid \varepsilon \\ C &\rightarrow c \mid \varepsilon \end{aligned}$$

which yields the following set of productions with non-empty righthand-side

$$\begin{aligned} S &\rightarrow ABC \mid AB \mid AC \mid BC \mid A \mid B \mid C \\ A &\rightarrow aA \mid a \\ B &\rightarrow bB \mid b \\ C &\rightarrow c \end{aligned}$$

The latter CFG has no ε -productions, so its language does not contain ε , although ε was in the language of the original grammar.

Eliminating unit productions

The next transformation on context-free grammars we discuss is the elimination of so-called unit productions. These are productions of the form $A \rightarrow B$, for two variables A and B .

Definition 3.24. A unit production of a context-free grammar $G = (V, T, R, S)$ is a production $A \rightarrow B \in R$ where $A, B \in V$.

The idea is to combine a sequence of unit productions $A_0 \rightarrow A_1, A_1 \rightarrow A_2, \dots, A_{m-1} \rightarrow A_m$ with an other production $A_m \rightarrow \alpha$, where α is not a variable.

Lemma 3.25. Let $G = (V, T, R, S)$ be a CFG. Define the set of productions R' by

$$R' = \{ A \rightarrow \alpha \mid \exists m \geq 0 \exists A_0, \dots, A_m \in V : A = A_0, \\ A_{i-1} \rightarrow A_i \in R \text{ for } 1 \leq i \leq m, A_m \rightarrow \alpha \in R \} \setminus \{ A \rightarrow B \mid A, B \in V \}$$

Define the CFG G' by $G' = (V, T, R', S)$. Then it holds that $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof. We first prove the following claim:

$$A \Rightarrow_G^* w \implies A \Rightarrow_{G'}^* w$$

for $A \in V$ and $w \in T^*$.

(\Rightarrow) We show, if $A \Rightarrow_G^n w$ then $A \Rightarrow_{G'}^* w$, for $A \in V$ and $w \in T^*$, by induction on n . Basis, $n=0$: Trivial. This situation doesn't occur. Induction step, $n+1$: Suppose $A \Rightarrow_G^{n+1} w$. By focussing on the first non-unit production in the derivation of w from A we obtain derivation by

$$A = A_0 \Rightarrow_G A_1 \cdots \Rightarrow_G A_m \Rightarrow_G X_1 \cdots X_k \Rightarrow_G^\ell w$$

for suitable $m \geq 0$, $A_0, \dots, A_m \in V$ such that and $m + 1 + \ell = n + 1$. Then it holds that $A \rightarrow X_1 \cdots X_k \in R'$. Moreover, we can find $\ell_1, \dots, \ell_k \geq 0$ and $w_1, \dots, w_k \in T^*$ such that $X_i \Rightarrow_G^{\ell_i} w_i$ for $1 \leq i \leq k$, $\ell_1 + \dots + \ell_k = \ell$, and $w_1 \cdots w_k = w$. We have either $X_i \in T$ and $X_i = w_i$, or $X_i \in V$ and $X_i \Rightarrow_{G'}^* w_i$ by induction hypothesis. Therefore,

$$A \Rightarrow_{G'}^* X_1 \cdots X_k \Rightarrow_{G'}^* w_1 \cdots w_k = w$$

which proves the claim. From the claim we obtain, if $S \Rightarrow_G^* w$ then $S \Rightarrow_{G'}^* w$. Hence $\mathcal{L}(G) \subseteq \mathcal{L}(G')$.

(\Leftarrow) We show, if $A \Rightarrow_{G'}^* w$ then $A \Rightarrow_G^* w$, for $A \in V$ and $w \in T^*$, by induction on n . Basis, $n=0$: Trivial. This situation doesn't occur. Induction step, $n+1$: Suppose $A \Rightarrow_{G'}^{n+1} w$. Then we have $A \Rightarrow_{G'} X_1 \cdots X_k \Rightarrow_{G'}^n w$. Thus for suitable $n_1, \dots, n_k \geq 0$ and $w_1, \dots, w_k \in T^*$ we have $X_i \Rightarrow_{G'}^{n_i} w_i$, for $1 \leq i \leq k$, $n_1 + \dots + n_k = n$, and $w_1 \cdots w_k = w$. By definition of R' we can find $A_0, \dots, A_m \in V$ for some $m \geq 0$ such that $A = A_0 \Rightarrow_G A_1 \cdots \Rightarrow_G A_m \Rightarrow_G X_1 \cdots X_k$. Also, if $X_i \in T$ then $X_i \Rightarrow_G^* w_i$ since $X_i = w_i$, and if $X_i \in V$ then $X_i \Rightarrow_G^* w_i$ by induction hypothesis. We conclude, $A \Rightarrow_G^* X_1 \cdots X_k \Rightarrow_G^* w_1 \cdots w_k = w$ by Lemma 3.15c, which proves the claim. From the claim we get, if $S \Rightarrow_{G'}^* w$ for some $w \in T^*$ then $S \Rightarrow_G^* w$. Hence $\mathcal{L}(G') \subseteq \mathcal{L}(G)$. \square

Note that a production $A \rightarrow \alpha \in R$, for $\alpha \notin V$, are also in R' : take $m = 0$ in the definition of R' . Also note, a sequence of unit production alone, $A_0 \rightarrow A_1, \dots, A_m \rightarrow B$ for variables A_0, \dots, A_m and B , does not lead to a production in R' .

From the lemma we derive, for every context-free grammar G there exists an equivalent context-free grammar G' without unit productions. Moreover, if G doesn't have ε -productions for variables other than the start symbol, then G' don't need to have such productions either.

Example 3.26. A CFG for arithmetic expressions may have the following productions:

$$\begin{aligned} E &\rightarrow T \mid E + T \\ T &\rightarrow F \mid T * F \\ F &\rightarrow I \mid (E) \\ I &\rightarrow a \mid b \mid c \end{aligned}$$

Elimination of unit productions as prescribed by Lemma 3.25 yields

$$\begin{aligned} E &\rightarrow T * F \mid (E) \mid a \mid b \mid c \mid E + T \\ T &\rightarrow (E) \mid a \mid b \mid c \mid T * F \\ F &\rightarrow a \mid b \mid c \mid (E) \\ I &\rightarrow a \mid b \mid c \end{aligned}$$

which has no unit productions indeed.

Identifying useless variables

The final transformation on context-free grammars we discuss concerns the identification and elimination of so-called useless symbols. A symbol is useless if either (i) it does not generate any terminal string, so doesn't contribute to the language of the grammar, or (ii) it is never produced by the start symbol, so not encountered in the production of a word of the language.

Definition 3.27. Let $G = (V, T, R, S)$ be a CNF. Let $X \in V \cup T$ be a symbol of G .

- (a) X is *generating* if $X \Rightarrow_G^* w$ for some $w \in T^*$.
- (b) X is called *reachable* if $S \Rightarrow_G^* \alpha X \beta$ for some strings $\alpha, \beta \in (V \cup T)^*$.
- (c) X is called *useful* if X is both generating and reachable.

Note that a terminal $a \in T$ of a grammar $G = (V, T, R, S)$ is always a generating symbol, since we always have $a \Rightarrow_G^* a$ and $a \in T^*$. Similarly trivial, the start symbol S of G is reachable, since $S \Rightarrow \alpha S \beta$ for $\alpha, \beta = \varepsilon$.

We first focus on finding generating symbols. We construct the set *Gen* containing these symbols by iteration.

Lemma 3.28. Let $G = (V, T, R, S)$ be a CNF. Define the sets $Gen_i \subseteq V \cup T$, for $i \geq 0$, by $Gen_0 = T$ and $Gen_{i+1} = Gen_i \cup \{ A \in V \mid \exists A \rightarrow \alpha \in R: \alpha \in Gen_i^* \}$. Put $Gen = \bigcup_{i=0}^{\infty} Gen_i$. Then it holds that

$$X \in Gen \iff X \text{ is generating}$$

for all symbols $X \in V \cup T$.

Proof. (\Rightarrow) We prove by induction on i , if $X \in Gen_i$ then X is generating. Basis, $i = 0$: Clear, each terminal $a \in T = Gen_0$ is generating.

Induction step, $i + 1$: Suppose $X \in Gen_{i+1} \setminus Gen_i$. Pick a rule $A \rightarrow \alpha \in R$ such that $X = A$ and $\alpha \in Gen_i^*$. Say, $\alpha = X_1 \cdots X_n$ for $n \geq 0$, $X_1, \dots, X_n \in V \cup T$. Since $X_1, \dots, X_n \in Gen_i$ we have that X_1, \dots, X_n are generating by induction hypothesis. Pick $w_1, \dots, w_n \in T^*$ such that $X_k \Rightarrow_G^* w_k$ for $1 \leq k \leq n$. Then, by Lemma ??,

$$X \Rightarrow_G X_1 \cdots X_n \Rightarrow_G^* w_1 \cdots w_n$$

Because $w_1 \cdots w_n \in T^*$ it follows that the symbol X is generating, as was to be shown.

(\Leftarrow) If X is generating, we can find $i \geq 0$ and $w \in T^*$ such that $X \Rightarrow_G^i w$. We prove $X \in Gen_i$ by induction on i . Basis, $i = 0$: We have $X = w \in T^*$, hence $X \in T$. Therefore, $X \in Gen_0$.

Induction step, $i + 1$: Since $X \Rightarrow_G^{i+1}$ we can find a rule $A \rightarrow X_1 \cdots X_n \in R$, where $n \geq 0$, $X_1, \dots, X_n \in V \cup T$, such that $A = X$ and $X_k \Rightarrow_G^{k_i} w_k$ for $0 \leq k_i \leq i$ and $w_k \in T^*$ for $1 \leq k \leq n$, again with appeal to Lemma ??. By induction hypothesis, $X_k \in Gen_{k_i}$ for $1 \leq k \leq n$. Because of rule $A \rightarrow X_1 \cdots X_n$ of G and $A = X$ it follows that $X \in Gen_{i+1}$, as was to be shown. \square

Note, the set of symbols Gen satisfies $Gen = Gen \cup \{ A \in V \mid \exists A \rightarrow \alpha \in R: \alpha \in Gen^* \}$. We use $Gen(G)$ to denote the set of generating symbols of a CFG G .

Theorem 3.29. Let $G = (V, T, R, S)$ be a CNF such that $\mathcal{L}(G) \neq \emptyset$. Let the set $Gen \subseteq V \cup T$ be as given by Lemma 3.28. Define the CNF $G' = (V', T, R', S)$ by $V' = V \cap Gen$ and $R' = \{ A \rightarrow \alpha \in R \mid A \in Gen, \alpha \in Gen^* \}$. Then it holds that $\mathcal{L}(G) = \mathcal{L}(G')$, and G' has generating symbols only.

Proof. Since $\mathcal{L}(G) \neq \emptyset$ it holds that $S \Rightarrow_G^* w$ for some $w \in T^*$. Hence, S is generating, and $S \in V' = V \cap Gen$.

Since $R' \subseteq R$ we have $\mathcal{L}(G') \subseteq \mathcal{L}(G)$. To show the reverse, $\mathcal{L}(G) \subseteq \mathcal{L}(G')$, we claim:

$$\text{if } X \Rightarrow_G^k w \text{ then } X \Rightarrow_{G'}^* w \tag{3.3}$$

for all $X' \in V' \cup T$, $k \geq 0$, and $w \in T^*$. *Proof of the claim* Induction on k . Basis, $k = 0$: Clear, if $X \Rightarrow_G^0 w$ then $X = w$ and $X \Rightarrow_{G'}^0 w$ by definition of $\Rightarrow_{G'}^0$.

Induction step, $k + 1$: Suppose $X \Rightarrow_G X_1 \cdots X_n \Rightarrow_G^k w$ for a rule $X \rightarrow X_1 \cdots X_n \in R$, where $X_1, \dots, X_n \in V \cup T$, $n \geq 0$. We can find $w_1, \dots, w_n \in T^*$, $0 \leq k_1, \dots, k_n \leq k$ such that $X_i \Rightarrow_G^{k_i} w_i$ for $1 \leq i \leq n$, and $w_1 \cdots w_n = w$. By induction hypothesis, $X_i \Rightarrow_{G'}^* w_i$,

$1 \leq i \leq n$. Note, $X_1 \cdots X_n \in (V \cup T)^* \cap \text{Gen}$. Since $X \Rightarrow_G^{k+1} w \in T^*$ we have that $X \in V \cap \text{Gen}$, thus $X \rightarrow X_1 \cdots X_n \in R'$. It follows that $X \Rightarrow_{G'}^* X_1 \cdots X_n \Rightarrow_{G'}^* w_1 \cdots w_n = w$. This proves the claim.

It holds that $S \in V'$. From the claim, instantiating X by S , we obtain, if $S \Rightarrow_G^* w$ then $S \Rightarrow_{G'}^* w$ for $w \in T^*$. Thus, if $w \in \mathcal{L}(G)$ then $w \in \mathcal{L}(G')$, i.e. $\mathcal{L}(G) \subseteq \mathcal{L}(G')$. \square

With the help of Lemma 3.28 and Theorem 3.29 we can construct a CFG with generating variables only for every CFG that has a non-empty language.

Example 3.30. Put $G = (\{S, A, B, C\}, \{a, b\}, R, S)$ where the set of productions R is given by

$$S \rightarrow AB \mid AC, \quad A \rightarrow BC \mid a, \quad C \rightarrow \varepsilon$$

Then we have

$$\begin{aligned} \text{Gen}_0 &= \{a, b\} \\ \text{Gen}_1 &= \{a, b\} \cup \{A, C\} = \{A, C, a, b\} \\ \text{Gen}_2 &= \{A, C, a, b\} \cup \{S, A, C\} = \{S, A, C, a, b\} \end{aligned}$$

It follows that $\text{Gen} = \{S, A, C, a, b\}$. The construction of the CNF $G' = (V', T, R', S)$, as given by Theorem 3.29 yields $V' = \{S, A, C\}$ and $R' = \{S \rightarrow AC, A \rightarrow a, C \rightarrow \varepsilon\}$. We have that $\mathcal{L}(G) = \mathcal{L}(G') = \{a\}$.

We proceed by defining a procedure to eliminate non-reachable symbols from a CNF. If the starting CNF is non-empty and has generating symbols only, the resulting CNF has only symbols that are both generating and reachable, i.e. symbols that are useful.

Lemma 3.31. Let $G = (V', T, R', S)$ be a CNF. Define the sets $\text{Reach}_i \subseteq V' \cup T$, for $i \geq 0$, by $\text{Reach}_0 = \{S\}$ and

$$\begin{aligned} \text{Reach}_{i+1} &= \text{Reach}_i \cup \{ X \in V' \cup T \mid \\ &\quad \exists A \in \text{Reach}_i \exists \alpha, \beta \in (V' \cup T)^*: A \rightarrow \alpha X \beta \in R' \} \end{aligned}$$

Put $\text{Reach} = \bigcup_{i=0}^{\infty} \text{Reach}_i$. Then it holds that

$$X \in \text{Reach} \iff X \text{ is reachable}$$

for all symbols $X \in V' \cup T$.

Proof. (\Rightarrow) By induction on i , if $X \in \text{Reach}_i$ then X is reachable. Basis, $i = 0$: Since $S \Rightarrow_{G'}^* \alpha S \beta$ for $\alpha, \beta = \varepsilon$ we have that the start symbol S is reachable. Induction step, $i + 1$: Suppose $X \in \text{Reach}_{i+1} \setminus \text{Reach}_i$. Pick $A \rightarrow \gamma \in R'$ such that $A \in \text{Reach}_i$ and $\gamma = \alpha X \beta$ for some $\alpha, \beta \in (V' \cup T)^*$. By induction hypothesis, the variable A is reachable, i.e. $S \Rightarrow_{G'}^* \alpha' A \beta'$ for suitable $\alpha', \beta' \in (V' \cup T)^*$. Then also $S \Rightarrow_{G'}^* \alpha' \alpha X \beta \beta'$, thus X is reachable, as was to be shown.

(\Leftarrow) If a symbol $X \in V' \cup T$ is reachable, we have $S \Rightarrow_{G'}^k \alpha X \beta$ for suitable strings $\alpha, \beta \in (V' \cup T)^*$ and some $k \geq 0$. We prove, if $S \Rightarrow_{G'}^k \alpha X \beta$ then $X \in \text{Reach}$ by induction on k . Basis, $k = 0$: We have $X = S$ and $X \in \text{Reach}_0 \subseteq \text{Reach}$ by definition.

Induction step, $k + 1$: Suppose $S \Rightarrow_{G'}^k \gamma \Rightarrow_{G'} \alpha X \beta$. Then we can find $A \in V'$, $\alpha', \alpha'', \beta', \beta'' \in (V' \cup T)^*$ such that $\gamma = \alpha' A \beta'$, $A \rightarrow \alpha'' X \beta'' \in R'$ and $\alpha = \alpha' \alpha''$, $\beta = \beta' \beta''$. Note, the symbol A is reachable. Therefore, by induction hypothesis, we have $A \in Reach$, say $A \in Reach_i$ for some $i \geq 0$. Since $A \rightarrow \alpha'' X \beta'' \in R'$, it follows that $X \in Reach_{i+1}$ and hence $X \in Reach$. \square

Now we have a means to identify reachable symbols, we are able to transform a CFG with generating symbols into a CFG without useless symbols.

Theorem 3.32. Let $G' = (V', T, R', S)$ be a CNF with generating symbols only such that $\mathcal{L}(G') \neq \emptyset$. Let the set $Reach \subseteq V \cup T$ be as given by Lemma 3.31. Define the CNF $G'' = (V'', T, R'', S)$ by $V'' = V' \cap Reach$ and $R'' = \{A \rightarrow \gamma \in R' \mid A \in Reach, \gamma \in Reach^*\}$. Then it holds that $\mathcal{L}(G') = \mathcal{L}(G'')$, and G'' has useful symbols only.

Proof. Since $R'' \subseteq R'$ we have $\mathcal{L}(G'') \subseteq \mathcal{L}(G')$. In order to show $\mathcal{L}(G') \subseteq \mathcal{L}(G'')$ we first prove the following claim:

$$\text{if } X \Rightarrow_{G'}^n w \text{ then } X \Rightarrow_{G''}^* w \quad (3.4)$$

for all $X \in Reach(G')$ $n \geq 0$, and $w \in T^*$.

Proof of the claim Induction on n . Basis, $n = 0$: We have $X = w$ thus $X \Rightarrow_{G''}^* w$. Induction step, $k + 1$: Suppose $X \Rightarrow_{G'}^{n+1} w$. Pick symbols $X_1, \dots, X_k \in V'' \cup T$, $w_1, \dots, w_k \in T^*$, $n_1, \dots, n_k \geq 0$ such that $X \rightarrow X_1 \cdots X_k \in R'$, $X_i \Rightarrow_{G'}^{n_i} w_i$ for $1 \leq i \leq k$, $n_1 + \dots + n_k = n$ and $w_1 \cdots w_k = w$. Since $X \in Reach(G')$ we have $X_1, \dots, X_k \in Reach(G')$, thus $X \rightarrow X_1 \cdots X_k \in R''$ and $X_i \Rightarrow_{G''}^* w_i$ for $1 \leq i \leq k$ by induction on n . We conclude $X \Rightarrow_{G''}^* w$, as was to be shown.

Since $S \in Reach(G')$ we obtain $\mathcal{L}(G') \subseteq \mathcal{L}(G'')$ by instantiating Equation (3.4) for S . Let $A \in V''$ be a variable of G'' . Then $A \in Gen(G')$ by construction. Thus $A \Rightarrow_{G'}^* w$ for some $w \in T^*$. By Equation (3.4) also $A \Rightarrow_{G''}^* w$. Hence $A \in Gen(G'')$. It is left to the reader to verify $S \Rightarrow_{G''}^* \alpha A \beta$ implies $A \in Reach(G'')$, for $A \in V''$, $\alpha, \beta \in (V \cup T)^*$. Having this, we see for $A \in V''$ that both $A \in Gen(G'')$ and $A \in Reach(G'')$. Hence, each variable of G'' is useful. \square

Example 3.33. Consider the CFG $G = (\{S, A, B, C\}, \{a, b, c, d\}, R, S)$ where R is the set of productions given by

$$S \rightarrow A \mid d, \quad A \rightarrow AB, \quad B \rightarrow \varepsilon, \quad C \rightarrow S \mid c$$

Then we have

$$\begin{aligned} Reach_0 &= \{S\} \\ Reach_1 &= \{S\} \cup \{A, d\} = \{S, A, d\} \\ Reach_2 &= \{S, A, d\} \cup \{A, d, A, B\} = \{S, A, B, d\} \\ Reach_3 &= \{S, A, B, d\} \cup \{A, d, A, B\} = \{S, A, B, d\} \end{aligned}$$

Thus $Reach(G) = \{S, A, B, d\}$. Restriction to reachable symbols, or rather reachable variables, yields the CFG $G' = (\{S, A, B\}, \{a, b, c, d\}, R', S)$ where the set of production R' is given by

$$S \rightarrow A \mid d, \quad A \rightarrow AB, \quad B \rightarrow \varepsilon$$

Since $\mathcal{L}(G') = \{d\}$ we must have $\mathcal{L}(G) = \{d\}$ too, by the theorem.

With the use of Lemma 3.28 up to Theorem 3.32 we can devise a procedure to construct from a CFG G , with non-empty language, an equivalent CFG G'' with useful variables only. Hence deleting productions that do not contribute to the production of any sequence in the language of G . Starting from $G = (V, T, R, S)$ with $\mathcal{L}(G) \neq \emptyset$, the procedure is as follows:

1. Compute $\text{Gen}(G)$.
2. Put $G' = (V', T, R', S)$ with set of variables $V' = V \cap \text{Gen}(G)$ and set of productions $R' = \{A \rightarrow \gamma \in R \mid \gamma \in \text{Gen}(G)^*\}$.
3. Compute $\text{Reach}(G')$.
4. Put $G'' = (V'', T, R'', S)$ with set of variables $V'' = V' \cap \text{Reach}(G')$ and set of productions $R'' = \{A \rightarrow \gamma \in R' \mid A \in \text{Reach}(G')\}$.

Combination of the results above yields that $\mathcal{L}(G'') = \mathcal{L}(G)$ and G'' has useful symbols only.

The order in which we eliminate non-generating symbols and non-reachable variables is important. We need to consider generating symbols first and reachable variables next. Consider the CFG $G = (\{S, A, B, C\}, \{a, b, c\}, R, S)$ with set of productions R given by

$$S \rightarrow AB \mid c, \quad A \rightarrow a, \quad C \rightarrow c$$

Doing things in the right order, we have $\text{Gen}(G) = \{S, A, C, a, b, c\}$. Thus, $G' = (\{S, A, C\}, \{a, b, c\}, R', S)$ with set of productions R' given by

$$S \rightarrow c, \quad A \rightarrow a, \quad C \rightarrow c$$

Since $\text{Reach}(G') = \{S\}$ we obtain $G'' = (\{S\}, \{a, b, c\}, R'', S)$ with set of productions R'' consisting of one production

$$S \rightarrow c$$

However, doing things in the reversed order, we have $\text{Reach}(G) = \{S, A, B, c\}$. Hence, $G' = (\{S, A, B\}, \{a, b, c\}, R', S)$ with set of productions R' given by

$$S \rightarrow AB \mid c, \quad A \rightarrow a$$

Next, $\text{Gen}(G') = \{S, A, a, b, c\}$. Therefore now $G'' = (\{S, A\}, \{a, b, c\}, R'', S)$ with set of productions R'' given by

$$S \rightarrow c, \quad A \rightarrow a$$

a grammar which still contains a useless symbol, viz. the variable A .

Chomsky normal form

So far we have seen transformations to eliminate ε -productions, unit productions, and useless symbols from a context-free grammar. We push it a little further though, by bringing a CFG into a normal form, viz. Chomsky normal form.

Definition 3.34. A context-free grammar $G = (V, T, R, S)$ is in Chomsky normal form if for all rules $A \rightarrow \alpha \in R$ either (i) there exist variables $B, C \in V$ such that $\alpha = BC$, or (ii) there exists a terminal $a \in T$ such that $\alpha = a$, or (iii) $A = S$ and $\alpha = \varepsilon$.

The constructions we have seen above help to focus on the production of strings in the language of a CFG. As a next step in bringing a grammar in Chomsky normal form we describe how to adapt a CFG such that the righthand-side of a production is either a terminal or a string of two or more variables. For this we need to introduce additional variables and production. However, we gain that we can assume that the CFG is in a standard form, which comes in handy in various situations.

Theorem 3.35. Let $G = (V, T, R, S)$ be a CNF such that $\gamma \in T$ or $|\gamma| \geq 2$, for all $A \rightarrow \gamma \in R$. Put $U = \{a \in T \mid \exists A \rightarrow \gamma \in R: |\gamma| \geq 2 \wedge a \in \gamma\}$. Let $W = \{B_a \mid a \in U\}$ be a fresh set of variables, i.e. $W \cap V = \emptyset$ and $W \cap T = \emptyset$. Define the CNF $G' = (V', T, R', S)$ by $V' = V \cup W$ and

$$\begin{aligned} R' = & \{A \rightarrow \gamma \mid \gamma \in T\} \cup \{B_a \rightarrow a \mid a \in U\} \cup \\ & \{A \rightarrow Y_1 \cdots Y_k \mid \exists A \rightarrow X_1 \cdots X_k \in R \exists i, 1 \leq i \leq k: \\ & (\exists a \in U: X_i = a \wedge Y_i = B_a) \vee (X_i \in V \wedge Y_i = X_i)\} \end{aligned}$$

Then it holds that $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof. ($\mathcal{L}(G) \subseteq \mathcal{L}(G')$) We first prove, by induction on n , if $A \Rightarrow_G^n w$ then $A \Rightarrow_{G'}^* w$, for all $w \in T^*$. Basis, $n = 0$: Trivial. Induction step, $n + 1$: Suppose $A \Rightarrow_G X_1 \cdots X_k \Rightarrow_G^n w$, $X_i \Rightarrow_G^{n_i} w_i$, $n_1 + \cdots + n_k = n$, and $w_1 \cdots w_k = w$. If $X_1 \cdots X_k \in T$, then $w = X_1 \cdots X_k$ and $A \Rightarrow_{G'} w \in R'$. If $X_1 \cdots X_k \notin T$, put $Y_i = X_i$ if $X_i \in V$, and $Y_i = B_a$ if $X_i = a$ for some $a \in T$. Then we have $A \Rightarrow_{G'} Y_1 \cdots Y_k \Rightarrow_{G'}^* X_1 \cdots X_k$. By induction hypothesis, $X_i \Rightarrow_{G'}^* w_i$, for $1 \leq i \leq k$. Thus $A \Rightarrow_{G'}^* w$, as was to be shown.

($\mathcal{L}(G') \subseteq \mathcal{L}(G)$) Note, each righthand-side of R' is either a terminal or a string of two or more variables. We first prove

$$A \Rightarrow_{G'}^n w \text{ implies } A \Rightarrow_G^* w \quad (3.5)$$

for $A \in V$, $n \geq 0$, $w \in T^*$, by induction on n . Basis, $n = 0$: Trivial. Induction step, $n + 1$: Suppose $A \Rightarrow_{G'} Y_1 \cdots Y_k \Rightarrow_{G'}^n w$, $Y_i \Rightarrow_{G'}^{n_i} w_i$ for $1 \leq i \leq k$ and $n_1 + \cdots + n_k = n$, $w_1 \cdots w_k = w$, for $k \geq 2$, $Y_1, \dots, Y_k \in V \cup W$, $n_1, \dots, n_k \geq 0$, and $w_1, \dots, w_k \in T^*$. If $Y_i \in V$ we put $X_i = Y_i$; if $Y_i \in W$ we put $X_i = w_i \in T$. We have that $A \rightarrow X_1 \cdots X_k$ is a production rule of G . Moreover, $X_i \Rightarrow_G^* w_i$: if $X_i \in V$ then this is by induction hypothesis; if $X_i = w_i$ then is always the case. We conclude $A \Rightarrow_G X_1 \cdots X_k \Rightarrow_G^* w_1 \cdots w_k = w$, which proves the induction step.

If we take S for A in Equation (3.5), we see $\mathcal{L}(G') \subseteq \mathcal{L}(G)$, as was to be shown. \square

Theorem 3.36. Let $G = (V, T, R, S)$ be a CNF such that $\gamma \in T$ or $\gamma \in V^*$ and $|\gamma| \geq 2$, for all $A \rightarrow \gamma \in R$. Put

$$R_1 = \{ A \rightarrow \gamma \mid |\gamma| = 1 \}, R_2 = \{ A \rightarrow \gamma \mid |\gamma| = 2 \}, R_3 = \{ A \rightarrow \gamma \mid |\gamma| \geq 3 \}$$

Choose, for each production rule $r : A^r \rightarrow B_1^r \cdots B_k^r$ in R_3 , a fresh set of variables $\{C_1^r, \dots, C_{k-2}^r\}$ such that $V_r \cap V = \emptyset$ and $V_r \cap V_{r'} = \emptyset$ for two different rules $r, r' \in R_3$. Put

$$V' = V \cup \bigcup \{ V_r \mid r \in R_3 \}$$

$$R'_3 = \{ A^r \rightarrow C_1^r B_k^r, C_1^r \rightarrow C_2^r B_{k-1}^r, \dots, C_{k-3}^r \rightarrow C_{k-2}^r B_3^r, C_{k-2}^r \rightarrow B_1^r B_2^r \}$$

and $R' = R_1 \cup R_2 \cup R'_3$. Define $G' = (V', T, R', S)$. Then it holds that $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof. ($\mathcal{L}(G) \subseteq \mathcal{L}(G')$) We first prove, by induction on n ,

$$A \Rightarrow_G^n w \text{ implies } A \Rightarrow_{G'}^* w \quad (3.6)$$

for $A \in V$, $n \geq 0$, and $w \in T^*$.

Basis, $n = 0$: Trivial. Induction step, $n + 1$: Suppose $A \Rightarrow_G^{n+1} w$. If the first production is of the form $A \rightarrow a \in R_1$ for some $a \in T$, then $k = 0$, $w = a$, and also $A \Rightarrow_G^* w$. If the first production is of the form $A \rightarrow B_1 B_2 \in R_2$ then $B_1 \Rightarrow_G^{n_1} w_1$, $B_2 \Rightarrow_G^{n_2} w_2$ for suitable $n_1, n_2 \geq 0$, $w_1, w_2 \in T^*$ such that $n_1 + n_2 = n$ and $w_1 w_2 = w$. Then we have $A \rightarrow B_1 B_2 \in R'$ and $B_1 \Rightarrow_{G'}^* w_1$, $B_2 \Rightarrow_{G'}^* w_2$ by induction hypothesis. Hence, $A \Rightarrow_{G'}^* B_1 B_2 \Rightarrow_{G'}^* w_1 w_2 = w$. Finally, if the first production is of the form $A \rightarrow B_1 \cdots B_k \in R_3$ for $k \geq 3$, then exist $n_1, \dots, n_k \geq 0$, $w_1, \dots, w_k \in T^*$ such that $B_i \Rightarrow_G^{n_i} w_i$ for $1 \leq i \leq k$, $n_1 + \dots + n_k = n$, and $w_1 \cdots w_k = w$. Moreover, by construction of R'_3 , we have $A \Rightarrow_{G'}^* B_1 \cdots B_k$. By induction hypothesis, $B_i \Rightarrow_{G'}^* w_i$ for $1 \leq i \leq k$. Hence $A \Rightarrow_{G'}^* B_1 \cdots B_k \Rightarrow_{G'}^* w_1 \cdots w_k = w$. Taking S for A in Equation (3.6) shows $\mathcal{L}(G) \subseteq \mathcal{L}(G')$.

($\mathcal{L}(G') \subseteq \mathcal{L}(G)$) We first prove, by induction on n ,

$$A \Rightarrow_{G'}^n w \text{ implies } A \Rightarrow_G^* w \quad (3.7)$$

for $A \in V$, $n \geq 0$, and $w \in T^*$.

Basis, $n = 0$: In this case there is nothing to prove. Induction step, $n + 1$: If the first production is of the form $A \rightarrow a \in R_1$, then $w = a$ and also $A \Rightarrow_{G'}^* w$. If the first production is of the form $A \rightarrow B_1 B_2 \in R_2$, then $B_1 \Rightarrow_G^{n_1} w_1$, $B_2 \Rightarrow_G^{n_2} w_2$ for some $n_1, n_2 \geq 0$, $w_1, w_2 \in T^*$ such that $n_1 + n_2 = n$ and $w_1 w_2 = w$. We have $A \rightarrow B_1 B_2 \in R$, and $B_i \Rightarrow_G^* w_i$, $i = 1, 2$, by induction hypothesis. Therefore, $A \Rightarrow_G B_1 B_2 \Rightarrow_G^* w_1 w_2 = w$. If the first production is of the form $A \rightarrow C_1 B_k \in R'_3$, then exists a production $A \rightarrow B_1 \cdots B_k \in R_3$ from which $A \rightarrow C_1 B_k$ is derived. Moreover, without loss of generality, we can assume that the derivation $A \Rightarrow_{G'}^n w$ is left-most. Thus $A \Rightarrow_{G'}^{k-1} B_1 \cdots B_k \Rightarrow_{G'}^{n-k+1} w$ by construction. Also $B_i \Rightarrow_{G'}^{n_i} w_i$ for $1 \leq i \leq k$, for suitable $n_1, \dots, n_k \geq 0$, $w_1, \dots, w_k \in T^*$ such that $n_1 + \dots + n_k = n - k + 1$ and $w_1 \cdots w_k = w$. By induction hypothesis $B_i \Rightarrow_G^* w_i$ for $1 \leq i \leq k$. We conclude $A \Rightarrow_G B_1 \cdots B_k \Rightarrow_G^* w_1 \cdots w_k = w$. Again by taking S for A , now in Equation (3.7), we see $\mathcal{L}(G') \subseteq \mathcal{L}(G)$. \square

Example 3.37. Consider the CNF given by

$$S \rightarrow ASB \mid \varepsilon \quad A \rightarrow aAS \mid a \mid \varepsilon \quad B \rightarrow bS \mid A \mid bb$$

Elimination of ε -productions yields the grammar

$$S \rightarrow ASB \mid SB \mid AS \mid S \mid \varepsilon \quad A \rightarrow aAS \mid aS \mid a \quad B \rightarrow bS \mid bb$$

Elimination of unit rules yields the grammar

$$S \rightarrow ASB \mid SB \mid AS \mid \varepsilon \quad A \rightarrow aAS \mid aS \mid a \quad B \rightarrow bS \mid bb$$

Introduction of auxilliary variables yields the grammar

$$\begin{array}{lll} S \rightarrow X_1B \mid SB \mid AS \mid \varepsilon & X_1 \rightarrow SB \\ A \rightarrow X_aX_2 \mid X_aS \mid a & X_a \rightarrow a & X_2 \rightarrow AS \\ B \rightarrow X_bS \mid bb & X_b \rightarrow b \end{array}$$

Exercises for Section 3.3

Exercise 3.3.1. Consider the CNF $G = (\{S, A, B, C, D, E\}, \{a, b\}, R, S)$ where the set of production rules R is given by

$$S \rightarrow A \mid C, \quad A \rightarrow BB, \quad B \rightarrow A \mid \varepsilon, \quad C \rightarrow AD, \quad D \rightarrow bC, \quad E \rightarrow \varepsilon$$

Give an equivalent CNF $G' = (V', \{a, b\}, R', S)$ with useful symbols only.

Exercise 3.3.2. Consider the CNF $G = (\{S, A, B, C\}, \{a, b, c\}, R, S)$ where the set of production rules R is given by

$$S \rightarrow A, \quad A \rightarrow BC \mid a, \quad B \rightarrow \varepsilon \mid b, \quad C \rightarrow \varepsilon \mid c$$

Give an equivalent CNF $G' = (V', \{a, b, c\}, R', S)$ without ε -productions for variables different from S and without unit rules.

Exercise 3.3.3. Consider the CNF $G = (\{S, A, B\}, \{a, b\}, R, S)$ where the set of production rules R is given by

$$S \rightarrow AAA \mid B, \quad A \rightarrow aA \mid B, \quad B \rightarrow \varepsilon$$

Give an equivalent CNF $G' = (V', \{a, b, c\}, R', S)$ which is in Chomsky normal form.

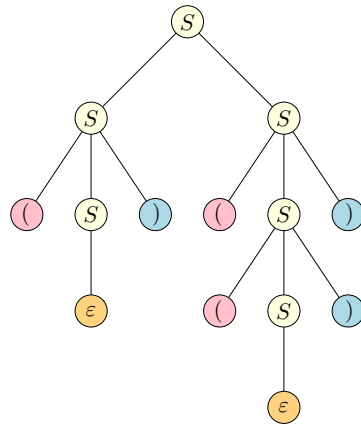


Figure 3.5: A parse tree representing three (and more) production sequences

3.4 Parse trees

In the previous section, in Example 3.16, we saw various production sequences for the same string. Basically, the particular production sequences are not very different. It is the order in which the various rules are applied that varies.

If we visualize the three derivations given in Example 3.16

$$\begin{aligned}
 S &\Rightarrow_G SS \Rightarrow_G S(S) \Rightarrow_G S((S)) \Rightarrow_G S(()) \Rightarrow_G (S)(()) \Rightarrow_G ()(()) \\
 S &\Rightarrow_G SS \Rightarrow_G (S)S \Rightarrow_G ()S \Rightarrow_G ()(S) \Rightarrow_G ()((S)) \Rightarrow_G ()(()) \\
 S &\Rightarrow_G SS \Rightarrow_G (S)S \Rightarrow_G (S)(S) \Rightarrow_G ()(S) \Rightarrow_G ()((S)) \Rightarrow_G ()(())
 \end{aligned}$$

as a rooted tree, we obtain in all three cases the picture given in Figure 3.5. The end result is the same, although the tree is built up differently. In fact, the order of independent productions is abstracted away. Only the causal order remains. Note, an internal node, i.e. a node that is not a leaf, together with its children corresponds to a production rule. A leaf labeled ε has no siblings.

We formalize the above idea in the notion of parse tree for a context-free grammar.

Definition 3.38. Let $G = (V, T, R, S)$ be a context-free grammar. The collection \mathcal{PT}_G of parse trees of G , a set of rooted node-labeled trees, is given as follows.

- A single root tree $[X]$ with root labeled X is a parse tree for G if X is a variable or terminal of G .
- A two-node tree $[A \rightarrow \varepsilon]$, with root labeled A and leaf labeled ε is a parse tree for G if $A \in V$ and $A \rightarrow \varepsilon$ is a production rule of G .
- If PT_1, PT_2, \dots, PT_k are k parse trees for G with roots X_1, X_2, \dots, X_k , respectively, and $A \rightarrow X_1X_2 \cdots X_k$ is a production rule of G , then the tree $[A \rightarrow PT_1, PT_2, \dots, PT_k]$ with root labeled A and subtrees of the root PT_1, PT_2, \dots, PT_k is a parse tree for G .

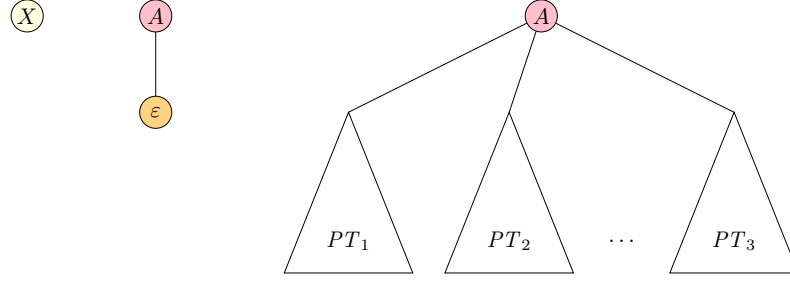


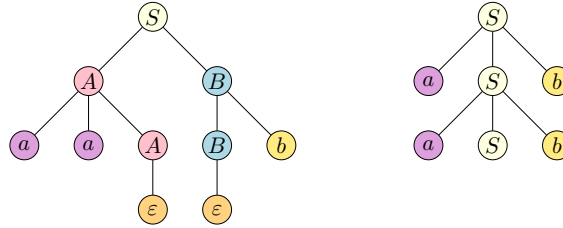
Figure 3.6: Parse tree formation

The yield function $yield : \mathcal{PT}_G \rightarrow (V \cup T)^*$ with respect to G is given by

- $yield([X]) = X$ for $X \in V \cup T$,
- $yield([A \rightarrow \varepsilon]) = \varepsilon$ for $A \in V$,
- $yield([A \rightarrow PT_1, PT_2, \dots, PT_k]) = yield(PT_1) \cdot yield(PT_2) \cdot \dots \cdot yield(PT_k)$ for $A \in V, PT_1, \dots, PT_k \in \mathcal{PT}_G$.

A parse tree PT for G is called complete if $yield(PT) \in T^*$.

For a context-free grammar G , we use $\mathcal{PT}_G(A)$ to denote the set of all complete parse trees of G with root labeled A .

Figure 3.7: Complete parse tree yield aab ; incomplete parse tree, yield $aaSbb$.

Next we relate derivations of a context-free grammar and its parse trees.

Theorem 3.39. Let $G = (V, T, R, S)$ be a context-free grammar. For $A \in V$ and $w \in T^*$, if $A \Rightarrow_G^* w$ then $w = yield(PT)$ for some complete parse tree PT of G with root A .

Proof. We prove, for $A \in V$ and $w \in T^*$,

$$A \Rightarrow_G^n w \quad \text{implies} \quad \exists PT \in \mathcal{PT}_G(A) : w = yield(PT)$$

by induction on n .

Basis, $n = 0$: Since $A \notin T^*$ there is nothing to prove. Induction step, $n + 1$: Suppose $A \Rightarrow_G X_1 \cdots X_k \Rightarrow_G^n w$ for a production rule $A \rightarrow X_1 \cdots X_k$ of G . By Lemma 3.15 we can find strings $w_1, \dots, w_k \in T^*$ such that $X_i \Rightarrow_G^* w_i$, for $1 \leq i \leq k$, and $w_1 \cdots w_k = w$. If $X_i \in T$, for $1 \leq i \leq k$, we have $X_i = w_i$ and we put $PT_i = [X_i]$. If $X_i \in V$, for $1 \leq i \leq k$, we obtain from the induction hypothesis a parse tree PT_i of G with root X_i and yield w_i . Consider the parse tree $PT = [A \rightarrow PT_1, \dots, PT_k]$ obtained from the juxtaposition of the parse trees PT_1, \dots, PT_k and putting the variable A on top. Since $A \rightarrow X_1 \cdots X_k \in R$ this is indeed a parse tree of G , it has root A and is such that

$$\text{yield}(PT) = \text{yield}(PT_1) \cdots \text{yield}(PT_k) = w_1 \cdots w_k = w$$

This was to be shown. \square

Corollary 3.40. For a context-free grammar G with start symbol S it holds that $\mathcal{L}(G) = \{ \text{yield}(PT) \mid PT \in \mathcal{PT}_G(S) \}$.

Proof. Direct from the definition of $\mathcal{L}(G)$ and the theorem. \square

We next show that if a string of terminals is the yield of a parse tree, then the string can be obtained by a leftmost derivation. For the proof of this result we need the following lemma.

Lemma 3.41. Let G be a context-free grammar.

- $X \xRightarrow_G^* \alpha$ implies $wX\beta \xRightarrow_G^* w\alpha\beta$ for all $X \in (V \cup T)$, $\alpha, \beta \in (V \cup T)^*$, and $w \in T^*$.
- $w\alpha \xRightarrow_G^* w\beta$ implies $\alpha \xRightarrow_G^* \beta$ for all $w \in T^*$, and $\alpha, \beta \in (V \cup T)^*$.

Proof. Left as an exercise. \square

We now turn to the announced result that relates parse trees and leftmost derivations. In its proof we make use of the height of a parse tree. This is the usual height of a tree, i.e. the maximal number of edges of a path from the root to a leaf.

Theorem 3.42. Let G be a context-free grammar. Suppose G has a parse tree PT with root labeled A and yield $w \in T^*$. Then there exists a left-most derivation $A \xRightarrow_G^* w$.

Proof. By induction on the height of the parse tree PT .

Basis, $n = 1$: Then PT has root labeled A and either one leaf labeled ε and $w = \varepsilon$, or k leaves for some $k \geq 1$ labeled a_1, \dots, a_k and $w = a_1 \cdots a_k$. In both cases, $A \rightarrow w$ is a production rule of G . Since $A \rightarrow w$ we also have $A \xRightarrow_G^* w$.

Induction step, $n + 1$: Then PT has the form $[A \rightarrow PT_1, \dots, PT_k]$, for some $k \geq 1$, each PT_i a parse tree of height at most n say with root labeled X_i and yield w_i , $1 \leq i \leq k$, and $A \rightarrow X_1 \cdots X_k$ a production rule of G . Moreover, $w = w_1 \cdots w_k$.

We claim the existence of a left-most derivation $A \xRightarrow_G^* w_1 \cdots w_i X_{i+1} \cdots X_k$ for $0 \leq i \leq k$. We prove this claim by induction on i . Basis, $i=0$: Since G has a production rule

$A \rightarrow X_1 \cdots X_k$, we also have $A \xRightarrow{\ell}_G^* X_1 \cdots X_k$, and hence we are done. Induction step, $i + 1$: By induction hypothesis for i we have $A \xRightarrow{\ell}_G^* w_1 \cdots w_i X_{i+1} \cdots X_k$. By induction hypothesis for n we have $X_{i+1} \xRightarrow{\ell}_G^* w_{i+1}$. With the help of Lemma 3.41 we obtain that

$$A \xRightarrow{\ell}_G^* w_1 \cdots w_i X_{i+1} X_{i+2} \cdots X_k \xRightarrow{\ell}_G^* w_1 \cdots w_i w_{i+1} X_{i+2} \cdots X_k$$

This proves the claim.

By choosing $i = k$ in the claim, we obtain $A \xRightarrow{\ell}_G^* w_1 \cdots w_k$, i.e., $A \xRightarrow{\ell}_G^* w$, as was to be shown. \square

We summarize the results of this section as the following theorem, which states that for a context-free grammar, the set of strings obtained by arbitrary derivations, the set of strings that arise as the yield of a parse tree, and the set of strings obtained by leftmost derivations are all the same set.

Theorem 3.43. Let $G = (V, T, R, S)$ be a context-free grammar. Then it holds that

$$\mathcal{L}(G) = \{ \text{yield}(PT) \in T^* \mid PT \in \mathcal{PT}_G(S) \} = \{ w \in T^* \mid S \xRightarrow{\ell}_G^* w \}$$

Proof. From Corollary 3.40 we obtain

$$\mathcal{L}(G) \subseteq \{ \text{yield}(PT) \in T^* \mid PT \in \mathcal{PT}_G(S) \}$$

By Theorem 3.42 we have

$$\{ \text{yield}(PT) \in T^* \mid PT \in \mathcal{PT}_G(S) \} \subseteq \{ w \in T^* \mid S \xRightarrow{\ell}_G^* w \}$$

By definition of $\mathcal{L}(G)$

$$\{ w \in T^* \mid S \xRightarrow{\ell}_G^* w \} \subseteq \mathcal{L}(G)$$

It follows that all set inclusions are equalities, as was to be shown. \square

Exercises for Section 3.4

Exercise 3.4.1. Consider again the the grammar of Exercise 3.2.1 with production rules

$$\begin{aligned} S &\rightarrow XbY \\ X &\rightarrow \varepsilon \mid aX \\ Y &\rightarrow \varepsilon \mid aY \mid bY \end{aligned}$$

Provide parse trees for this grammar with yield $aabab$, $baab$, and $aaabb$.

A context-free grammar G is called *ambiguous* if there exist two different complete parse trees PT_1 and PT_2 of G such that $\text{yield}(PT_1) = \text{yield}(PT_2)$. Otherwise G is called *unambiguous*.

Exercise 3.4.2. (a) Show that the grammar G given by the production rules

$$S \rightarrow AB \mid aaB \quad A \rightarrow a \mid Aa \quad B \rightarrow b$$

is ambiguous.

- (b) Provide an unambiguous grammar G' that generates the same language as G . Argue why G' is unambiguous and why $\mathcal{L}(G') = \mathcal{L}(G)$.

Exercise 3.4.3. (a) Show that the grammar G given by the production rules

$$S \rightarrow \varepsilon \mid aSbS \mid bSaS$$

is ambiguous.

- (b) Provide an unambiguous grammar G' that generates the same language as G . Argue why G' is unambiguous and why $\mathcal{L}(G') = \mathcal{L}(G)$.

3.5 The class of context-free languages

In this section we establish that a language is generated by a context-free grammar precisely when it is accepted by a push-down automaton. We introduce the notion of acceptance by empty stack for a PDA and obtain the main result in three steps: (i) from CFG to PDA, (ii) from PDA to PDA accepting on empty stack, (iii) from PDA accepting on empty stack to CFG.

Theorem 3.44. Let $G = (V, T, R, S)$ be a context-free grammar. Then there exists a push-down automaton P such that $\mathcal{L}(P) = \mathcal{L}(G)$.

Proof. Define $P = (\{q_0, q_1, q_2\}, T, V \cup T, \emptyset, \rightarrow, q_0, \{q_2\})$ with

- (1) $q_0 \xrightarrow{\tau[\emptyset/S]} q_1$
- (2) $q_1 \xrightarrow{a[a/\varepsilon]} q_1$ for all $a \in T$ (matching step)
- (3) $q_1 \xrightarrow{\tau[\emptyset/\varepsilon]} q_2$
- (4) $q_1 \xrightarrow{\tau[A/\alpha]} q_1$ for all $A \rightarrow \alpha \in R$ (production step)

We will prove $\mathcal{L}(P) = \mathcal{L}(G)$. For this we need to claim, for all $\gamma \in (V \cup T)^*$ and $w \in T^*$,

$$\gamma \xRightarrow{*}_G w \quad \text{iff} \quad (q_1, w, \gamma) \vdash_P^* (q_1, \varepsilon, \varepsilon) \quad (3.8)$$

Proof of the claim: (\Rightarrow) We show by induction on n , $\gamma \xRightarrow{n}_G w$ implies $(q_1, w, \gamma) \vdash_P^* (q_1, \varepsilon, \varepsilon)$. Basis, $n = 0$: Then we have $\gamma = w$. By clause (2) in the definition of P we

have $(q_1, w, \gamma) = (q_1, w, w) \vdash_P^* (q_1, \varepsilon, \varepsilon)$. Induction step, $n+1$: Suppose $\gamma \xRightarrow{\ell}_G \gamma' \xRightarrow{\ell}_G^n w$. Then $\gamma = vA\beta$ for suitable $v \in T^*$, $A \in V$, and $\beta \in (V \cup T)^*$. Then we have $\gamma' = v\alpha\beta$ for some $A \rightarrow_G \alpha \in R$. Since $\gamma' \xRightarrow{\ell}_G^n w$ we have that $w = vu$ and $\alpha\beta \xRightarrow{\ell}_G^n u$, see Lemma 3.41. Therefore

$$\begin{aligned} (q_1, w, \gamma) &= (q_1, vu, vA\beta) \\ &\vdash_P^* (q_1, u, A\beta) \quad \text{by clause (2)} \\ &\vdash_P (q_1, u, \alpha\beta) \quad \text{by clause (4), } A \rightarrow_G \alpha \\ &\vdash_P^* (q_1, \varepsilon, \varepsilon) \quad \text{by induction hypothesis, } \alpha\beta \xRightarrow{\ell}_G^n u \end{aligned}$$

as was to be shown.

(\Leftarrow) By induction on n , if $(q_1, w, \gamma) \vdash_P^n (q_1, \varepsilon, \varepsilon)$ then $\gamma \xRightarrow{\ell}_G^* w$. Basis, $n = 0$: Then we have $w = \varepsilon$ and $\gamma = \varepsilon$. Clearly $\varepsilon \xRightarrow{\ell}_G^* \varepsilon$.

Induction step, $n+1$: Suppose

$$(q_1, w, \gamma) \vdash_P (q_1, w', \gamma') \vdash_P^n (q_1, \varepsilon, \varepsilon)$$

We distinguish two cases. Case I, $\gamma = A\beta$: We then have $(q_1, w, \gamma) \vdash_P (q_1, w', \gamma')$ by clause (4) for a rule $A \rightarrow_G \alpha$. Thus $w = w'$ and $\gamma' = \alpha\beta$. By induction hypothesis $\gamma' \xRightarrow{\ell}_G^* w'$, i.e. $\alpha\beta \xRightarrow{\ell}_G^* w$. Thus $\gamma = A\beta \xRightarrow{\ell}_G \alpha\beta \xRightarrow{\ell}_G^* w$. Case II, $\gamma = a\beta$. We then have $(q_1, w, \gamma) \vdash_P (q_1, w', \gamma')$ by clause (2). Thus $w = aw'$ and $\gamma = a\gamma'$. Since $(q_1, w', \gamma') \vdash_P^n (q_1, \varepsilon, \varepsilon)$ we have $\gamma' \xRightarrow{\ell}_G^* w'$ by induction hypothesis. Using Lemma 3.41 it follows that $a\gamma' \xRightarrow{\ell}_G^* aw'$, i.e. $\gamma \xRightarrow{\ell}_G^* w$.

We finally prove $\mathcal{L}(P) = \mathcal{L}(G)$. Suppose $w \in \mathcal{L}(P)$. Thus $(q_0, w, \varepsilon) \vdash_P^* (q_2, \varepsilon, \beta)$ for some string $\beta \in (V \cup T)^*$. By definition of P it follows that

$$(q_0, w, \varepsilon) \vdash_P (q_1, w, S) \vdash_P^* (q_1, \varepsilon, \varepsilon) \vdash_P (q_2, \varepsilon, \varepsilon)$$

Since $(q_1, w, S) \vdash_P^* (q_1, \varepsilon, \varepsilon)$ we have by claim (3.8) $S \xRightarrow{\ell}_G^* w$. Thus $w \in \mathcal{L}(G)$. Reversely, suppose $w \in \mathcal{L}(G)$. Thus $S \xRightarrow{\ell}_G^* w$. Again by claim (3.8) we obtain $(q_1, w, S) \vdash_P^* (q_1, \varepsilon, \varepsilon)$. Adding the derivation steps $(q_0, w, \varepsilon) \vdash_P (q_1, w, S)$ and $(q_1, \varepsilon, \varepsilon) \vdash_P (q_2, \varepsilon, \varepsilon)$ we see $(q_0, w, \varepsilon) \vdash_P^* (q_2, \varepsilon, \varepsilon)$. Thus $w \in \mathcal{L}(P)$. \square

Next we introduce the idea of a PDA accepting on empty stack.

Definition 3.45 (Push-down automaton accepting on empty stack). A push-down automaton accepting on empty stack, also called an empty-stack automaton, is a six-tuple $P = (Q, \Sigma, \Delta, D, \rightarrow_P, q_0)$ where

1. Q is a finite set of states,
2. Σ is a finite input alphabet,
3. Δ is a finite data alphabet or stack alphabet,
4. $D \in \Delta$ is the stack bottom symbol,

5. $\rightarrow_P \subseteq Q \times \Sigma_\tau \times \Delta \times \Delta^* \times Q$, where $\Sigma_\tau = \Sigma \cup \{\tau\}$, is a finite set of transitions or steps,
6. $q_0 \in Q$ is the initial state.

Notation and the notion of a configuration extend from a PDA as given by Definition 3.2 to a PDA accepting on empty stack. Note, a PDA accepting on empty stack has no transition if the stack is empty. The idea is that, when the stacks becomes empty the PDA blocks and accepts the string read so far. As a consequence we need to adapt the stack in the start situation. Instead of an empty stack, a PDA accepting on empty stack starts with a stack consisting of the stack bottom symbol D only. No final states need to be identified. Honoring its name, this variant of PDAs accepts on the empty stack. Therefore we have

Definition 3.46. Let $P = (Q, \Sigma, \Delta, D, \rightarrow_P, q_0)$ be a PDA accepting on empty stack. The language accepted by P on empty stack, notation $\mathcal{N}(P)$, is given by

$$\{ w \in \Sigma^* \mid \exists q \in Q: (q_0, w, D) \vdash_P^* (q, \varepsilon, \varepsilon) \}$$

A PDA accepting on empty stack may terminate in any state. Acceptance or non-acceptance is decided by the stack being empty or not.

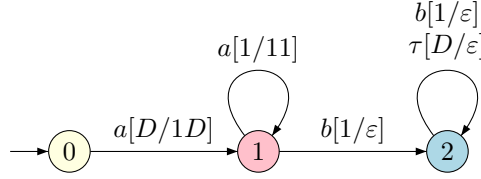


Figure 3.8: Example push-down automaton accepting on empty stack

Example 3.47. Figure 3.8 displays a push-down automaton P accepting on empty stack. We have $P = (Q, \Sigma, \Delta, D, \rightarrow_P, q_0)$ with $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Delta = \{1, D\}$ now including the special symbol D that is the stack bottom, and a transition relation given by

$$q_0 \xrightarrow{a[D/1D]}_P q_1, \quad q_1 \xrightarrow{a[1/11]}_P q_1, \quad q_1 \xrightarrow{b[1/\varepsilon]}_P q_2, \quad q_2 \xrightarrow{b[1/\varepsilon]}_P q_2, \quad q_2 \xrightarrow{\tau[D/\varepsilon]}_P q_2$$

Compared to the standard PDA given in Figure 3.3 there is no final state q_3 . In fact, there is no final state at all. In the transitions of Figure 3.8 there is no mentioning of the empty stack using \emptyset . Testing for empty stack can now be done by testing for the new stack symbol D on top (or rather at the bottom) of the stack.

Theorem 3.48. For every push-down automaton P , accepting on final state, exists a push-down automaton P' accepting on empty stack such that $\mathcal{L}(P) = \mathcal{N}(P')$.

Proof. Suppose $P = (Q, \Sigma, \Delta, \emptyset, \rightarrow_P, q_0, F)$. Define P' by

$$P' = (Q \cup \{q_{end}\}, \Sigma, \Delta \cup \{D\}, D, \rightarrow_{P'}, q_0)$$

with a new state q_{end} , a new stack symbol D , and with a transition relation $\rightarrow_{P'}$ given by

$$\begin{aligned} q &\xrightarrow{a[d/x]}_{P'} q' && \text{if } q \xrightarrow{a[d/x]}_P q' \\ q &\xrightarrow{a[D/xD]}_{P'} q' && \text{if } q \xrightarrow{a[\emptyset/x]}_P q' \\ q &\xrightarrow{\tau[d/x]}_{P'} q' && \text{if } q \xrightarrow{\tau[d/x]}_P q' \\ q &\xrightarrow{\tau[D/xD]}_{P'} q' && \text{if } q \xrightarrow{\tau[\emptyset/x]}_P q' \\ q &\xrightarrow{\tau[d/d]}_{P'} q_{end} && \text{for } q \in F \text{ and } d \in \Delta \cup \{D\} \\ q_{end} &\xrightarrow{\tau[d/\varepsilon]}_{P'} q_{end} && \text{for } d \in \Delta \\ q_{end} &\xrightarrow{\tau[D/\varepsilon]}_{P'} q_{end} \end{aligned}$$

It holds that $(q, w, x) \vdash_P (q', w', x')$ iff $(q, w, xD) \vdash_{P'} (q', w', x'D)$ for $w \in \Sigma^*$, $q, q' \in Q$, and $x, x' \in \Delta^*$. From this it follows that $(q_0, w, \varepsilon) \vdash_P^* (q, \varepsilon, x)$ for some $q \in F$ iff $(q_0, w, D) \vdash_{P'}^* (q, \varepsilon, xD)$ for some $q \in F$ iff $(q_0, w, D) \vdash_{P'}^* (q_{end}, \varepsilon, \varepsilon)$, since q_{end} can only be reached in P' via a final state of P . We conclude $w \in \mathcal{L}(P)$ iff $w \in \mathcal{N}(P')$, and $\mathcal{L}(P) = \mathcal{N}(P')$ as was to be shown. \square

The above theorem connects the two types of push-down automata in one direction. The next result connects a push-down automaton accepting on empty stack with a context-free grammar.

Theorem 3.49. Let $P = (Q, \Delta, \Sigma, \rightarrow, q_0, D)$ be a push-down automaton accepting on empty stack. There exists a context-free grammar G such that $\mathcal{L}(G) = \mathcal{N}(P)$.

Proof. We first define the grammar G . Its terminal alphabet T is the input alphabet Σ of P . The variable alphabet V contains a start symbol S and variables of the form $[qdq']$ where q and q' are states in Q and d is a stack symbol in Δ . Thus

$$V = \{S\} \cup \{[qdq''] \mid q, q'' \in Q, d \in \Delta\}$$

The intuition for a variable $[qdq'']$ is that P' in state q , for any stack content x , can reach by processing the symbol d on top of the stack dx the state q'' with stack content x . So, the net change of state is from q to q'' , the net change of the stack is that d is popped.

The rules of G are of three types

- $S \rightarrow [q_0 D q']$ for all $q' \in Q$
- $[qdq''] \rightarrow a[p_0 d_1 p_1][p_1 d_2 p_2] \cdots [p_{k-1} d_k p_k]$ if $q \xrightarrow{a[d/d_1 \cdots d_k]} q'$
with $p_0, \dots, p_k \in Q$ such that $p_0 = q'$ and $p_k = q''$

- $[q d q''] \rightarrow [p_0 d_1 p_1][p_1 d_2 p_2] \cdots [p_{k-1} d_k p_k]$ if $q \xrightarrow{\tau[d/d_1 \cdots d_k]} q'$
with $p_0, \dots, p_k \in Q$ such that $p_0 = q'$ and $p_k = q''$

In order to show the equality of $\mathcal{L}(G)$ and $\mathcal{N}(P)$ we prove

$$[q d q'] \Rightarrow_G^* w \quad \text{iff} \quad (q, w, d) \vdash_P^* (q', \varepsilon, \varepsilon) \quad (3.9)$$

(\Rightarrow) We prove, by induction on n , that $[q d q'] \Rightarrow_G^n w$ implies $(q, w, d) \vdash_P^* (q', \varepsilon, \varepsilon)$.

Basis, $n = 1$: If $[q d q'] \Rightarrow_G w$ then $[q d q'] \rightarrow w$ is a rule of G . So, $w = a$ for some $a \in \Sigma$ and $q \xrightarrow{a[d/\varepsilon]} q'$ or $w = \varepsilon$ and $q \xrightarrow{\tau[d/\varepsilon]} q'$. Thus $(q, a, d) \vdash_P (q', \varepsilon, \varepsilon)$ or $(q, \varepsilon, d) \vdash_P (q', \varepsilon, \varepsilon)$ and hence $(q, a, d) \vdash_P^* (q', \varepsilon, \varepsilon)$ or $(q, \varepsilon, d) \vdash_P^* (q', \varepsilon, \varepsilon)$.

Induction step, $n + 1$: We have either have

$$[q d q'] \Rightarrow_G a[p_0 d_1 p_1] \cdots [p_{k-1} d_k p_k] \Rightarrow_G^n w$$

for some rule $[q d q'] \rightarrow a[p_0 d_1 p_1] \cdots [p_{k-1} d_k p_k]$ of G , or

$$[q d q'] \Rightarrow_G [p_0 d_1 p_1] \cdots [p_{k-1} d_k p_k] \Rightarrow_G^n w$$

for some rule $[q d q'] \rightarrow [p_0 d_1 p_1] \cdots [p_{k-1} d_k p_k]$ of G . We consider the first case, leaving the second case to the reader.

By Lemma 3.15 we have $w = aw_1 \dots w_k$ where $[p_{i-1} d_i p_i] \Rightarrow_G^* w_i$, $1 \leq i \leq k$. Thus, by induction hypothesis, $(p_{i-1}, w_i, d_i) \vdash_P^* (p_i, \varepsilon, \varepsilon)$. By Lemma 3.6 we obtain

$$(p_{i-1}, w_i w_{i+1} \cdots w_k, d_i d_{i+1} \cdots d_k) \vdash_P^* (p_i, w_{i+1} \cdots w_k, d_{i+1} \cdots d_k)$$

for $1 \leq i \leq k$. By putting these k derivations together we obtain that

$$(p_0, w_1 \cdots w_k, d_1 \cdots d_k) \vdash_P^* (p_k, \varepsilon, \varepsilon)$$

Since $(q, aw, d) = (q, aw_1 \cdots w_k, d) \vdash_P (p_0, w_1 \cdots w_k, d_1 \cdots d_k)$ we conclude $(q, aw, d) \vdash_P^* (q', \varepsilon, \varepsilon)$.

(\Leftarrow) We prove, by induction on n , $(q, w, d) \vdash_P^n (q', \varepsilon, \varepsilon)$ implies $[q d q'] \Rightarrow_G^* w$. Basis, $n = 1$: If $(q, w, d) \vdash_P (q', \varepsilon, \varepsilon)$, we have $w = a$ for some transition $q \xrightarrow{a[d/\varepsilon]} q'$ or $w = \varepsilon$ for some transition $q \xrightarrow{\tau[d/\varepsilon]} q'$. Thus G has the production rule $[q d q'] \rightarrow a$ or $[q d q'] \rightarrow \varepsilon$ and $[q d q'] \Rightarrow_G^* w$. Induction step, $n + 1$: Suppose

$$(q, w, d) \vdash_P (r, v, d_1 \cdots d_k) \vdash_P^n (q', \varepsilon, \varepsilon)$$

based on the transition $q \xrightarrow{a[d/d_1 \cdots d_k]} r$. As all of d_1, \dots, d_k are successively popped of the stack in this derivation, we can find states $r_0, \dots, r_k \in Q$, strings $v_0, \dots, v_k \in \Sigma^*$ such that $r_0 = r$, $v_0 = v$, and

$$(r_0, v_0, d_1 \cdots d_k) \vdash_P^* (r_1, v_1, d_2 \cdots d_k) \vdash_P^* \cdots \vdash_P^* (r_{k-1}, v_{k-1}, d_k) \vdash_P^* (r_k, v_k, \varepsilon) \vdash_P^* (q', \varepsilon, \varepsilon)$$

Note, since P terminates on empty stack, $r_k = q'$, $v_k = \varepsilon$. By Lemma 3.6 we obtain

$$\begin{aligned} (r_0, v_0 \setminus v_1, d_1) \vdash_P^* (r_1, \varepsilon, \varepsilon), \quad (r_1, v_1 \setminus v_2, d_2) \vdash_P^* (r_2, \varepsilon, \varepsilon), \\ \dots, \quad (r_{k-1}, v_{k-1} \setminus v_k, d_k) \vdash_P^* (r_k, \varepsilon, \varepsilon) \end{aligned}$$

If $x \succ y$ then $x \setminus y$ is a prefix of x such that $x = (y \setminus x)y$. Since the length of these derivations does not exceed n , it follows from the induction hypothesis that

$$[r_0 d_1 r_1] \Rightarrow_G^* v_0 \setminus v_1, \quad [r_1 d_2 r_2] \Rightarrow_G^* v_1 \setminus v_2 \quad \dots, \quad [r_{k-1} d_k r_k] \Rightarrow_G^* v_{k-1} \setminus v_k$$

Hence, by Lemma 3.6,

$$[r_0 d_1 r_1][r_1 d_2 r_2] \cdots [r_{k-1} d_k r_k] \Rightarrow_G^* (v_0 \setminus v_1)(v_1 \setminus v_2) \cdots (v_{k-1} \setminus v_k)$$

Since $(v_0 \setminus v_1)(v_1 \setminus v_2) \cdots (v_{k-1} \setminus v_k) = v_0 \setminus v_k = v \setminus \varepsilon = v$ we conclude

$$[q d q'] \Rightarrow_G^* a[r_0 d_1 r_1][r_1 d_2 r_2] \cdots [r_{k-1} d_k r_k] \Rightarrow_G^* av = w$$

The case where

$$(q, w, d) \vdash_P (r, w, d_1 \cdots d_k) \vdash_P^n (q', \varepsilon, \varepsilon)$$

based on the transition $q \xrightarrow{\tau[d/d_1 \cdots d_k]} r$ is left to the reader. This proves the claim.

With the property of Equation (3.9) in place, we finish the proof as follows: $w \in \mathcal{L}(G)$ iff $S \Rightarrow_G^* w$ iff $[q_0 D q'] \Rightarrow_G^* w$ for some $q' \in Q$ iff $(q_0, w, D) \vdash_P (q', \varepsilon, \varepsilon)$ for some $q' \in Q$ iff $w \in \mathcal{N}(P)$. \square

Combination of the three theorems above yields that context-free grammars and push-down automata characterize the same languages.

Corollary 3.50. Let L be a language. The following three statements are equivalent.

- (i) $L = \mathcal{L}(P)$ for a push-down automaton P accepting on final state;
- (ii) $L = \mathcal{N}(P')$ for a push-down automaton P' accepting on empty stack;
- (iii) $L = \mathcal{L}(G)$ for a context-free grammar G .

Proof. [(i) \Rightarrow (ii)] This is Theorem 3.48. [(ii) \Rightarrow (iii)] This is Theorem 3.49. [(iii) \Rightarrow (i)] This is Theorem 3.44. \square

3.6 Properties of the class of context-free languages

In this section we establish a number of positive and negative results for the class of context-free languages. We start off with closure properties that hold for regular languages and show that these hold for context-free languages too.

We will make use of the following observation: Suppose G_1 and G_2 are two context-free grammars for two languages L_1 and L_2 , respectively. Say $G_i = (V_i, \Sigma, R_i, S_i)$ for $1 \leq i \leq 2$. Then we can assume that G_1 and G_2 have different variables, i.e. $V_1 \cap V_2 = \emptyset$, applying a renaming of variables if needed.

Theorem 3.51. Let L_1 , L_2 and L be context-free languages over an alphabet Σ .

- (a) The language $L_1 \cup L_2$, the union of L_1 and L_2 , is also a context-free language.
- (b) The language $L_1 \cdot L_2 = \{ w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2 \}$, the concatenation of L_1 and L_2 , is also a context-free language.
- (c) The language $L^* = \{ w_1 \cdots w_n \mid n \geq 0, w_1, \dots, w_n \in L \}$, the Kleene closure of L , is also a context-free language.

Proof. (a) Suppose $L_i = \mathcal{L}(G_i)$, $G_i = (V_i, \Sigma, R_i, S_i)$ for $1 \leq i \leq 2$. We can assume $V_1 \cap V_2 = \emptyset$. Consider the context-free grammar $G' = (V', \Sigma, R', S')$ where

$$V' = \{S'\} \cup V_1 \cup V_2 \quad \text{and} \quad R' = \{S' \rightarrow S_1, S' \rightarrow S_2\} \cup R_1 \cup R_2$$

for a fresh variable S' not in V_1 or V_2 . We claim $\mathcal{L}(G') = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$.

If $w \in \mathcal{L}(G')$, then $S' \Rightarrow_{G'}^* w$. By definition of G' , either it holds that

$$S' \Rightarrow_{G'} S_1 \Rightarrow_{G_1}^* w \quad \text{or} \quad S' \Rightarrow_{G'} S_2 \Rightarrow_{G_2}^* w$$

In both cases, the derivation $S_i \Rightarrow_{G'}^* w$ of G' is also a derivation $S_i \Rightarrow_{G_i}^* w$ of G_i , since G_1 and G_2 have disjoint sets of variables. Thus $w \in \mathcal{L}(G_1)$ or $w \in \mathcal{L}(G_2)$, and hence $w \in \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$. We conclude $\mathcal{L}(G') \subseteq \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$.

If $w \in \mathcal{L}(G_1)$, then $S_1 \Rightarrow_{G_1}^* w$. By definition of G' , we then also have $S' \Rightarrow_{G'} S_1 \Rightarrow_{G'}^* w$. Therefore $w \in \mathcal{L}(G')$. It follows that $\mathcal{L}(G_1) \subseteq \mathcal{L}(G')$. Likewise we can derive $\mathcal{L}(G_2) \subseteq \mathcal{L}(G')$. We conclude $\mathcal{L}(G_1) \cup \mathcal{L}(G_2) \subseteq \mathcal{L}(G')$. Hence $\mathcal{L}(G_1) \cup \mathcal{L}(G_2) = \mathcal{L}(G')$.

(b) Consider the context-free grammar $G' = (V', \Sigma, R', S')$ where

$$V' = \{S'\} \cup V_1 \cup V_2 \quad \text{and} \quad R' = \{S' \rightarrow S_1 S_2\} \cup R_1 \cup R_2$$

for a fresh variable S' not in V_1 or V_2 . We claim $\mathcal{L}(G') = \mathcal{L}(G_1) \cdot \mathcal{L}(G_2)$. If $w \in \mathcal{L}(G')$, then $S' \Rightarrow_{G'}^* w$. By definition of G' , it holds that

$$S' \Rightarrow_{G'} S_1 S_2 \Rightarrow_{G'}^* w$$

By Lemma 3.15, we can find two words $w_1, w_2 \in \Sigma^*$ such that

$$S_1 \Rightarrow_{G_1}^* w_1, \quad S_2 \Rightarrow_{G_2}^* w_2, \quad \text{and} \quad w_1 w_2 = w$$

Since $V_1 \cap V_2 = \emptyset$, we then also have $S_1 \Rightarrow_{G_1}^* w_1$ and $S_2 \Rightarrow_{G_2}^* w_2$. Thus $w_1 \in \mathcal{L}(G_1)$ and $w_2 \in \mathcal{L}(G_2)$. Hence $w = w_1 w_2 \in \mathcal{L}(G_1) \cdot \mathcal{L}(G_2)$ and $\mathcal{L}(G') \subseteq \mathcal{L}(G_1) \cdot \mathcal{L}(G_2)$.

If $w \in \mathcal{L}(G_1) \cdot \mathcal{L}(G_2)$, we can write $w = w_1 w_2$ for some $w_1 \in \mathcal{L}(G_1)$ and $w_2 \in \mathcal{L}(G_2)$. Thus $S_1 \Rightarrow_{G_1}^* w_1$ and $S_2 \Rightarrow_{G_2}^* w_2$. By definition of G' we then also have $S_1 \Rightarrow_{G'}^* w_1$ and $S_2 \Rightarrow_{G'}^* w_2$. Therefore, by Lemma 3.15, it follows that $S_1 S_2 \Rightarrow_{G'}^* w_1 w_2$. Hence,

$$S \Rightarrow_{G'} S_1 S_2 \Rightarrow_{G'}^* w_1 w_2 = w$$

and thus $w \in \mathcal{L}(G')$. We conclude that $\mathcal{L}(G_1) \cdot \mathcal{L}(G_2) \subseteq \mathcal{L}(G')$.

(c) Consider the context-free grammar $G' = (V', \Sigma, R', S')$ where

$$V' = \{S'\} \cup V \quad \text{and} \quad R' = \{S' \rightarrow \varepsilon, S' \rightarrow SS'\} \cup R$$

for a fresh variable S' not in V . We claim $\mathcal{L}(G') = \mathcal{L}(G)^*$. If $w \in \mathcal{L}(G')$, then $S' \Rightarrow_{G'}^* w$. By Theorem 3.43 we can assume $S' \xRightarrow{\ell}_{G'}^* w$. By definition of G' , assuming $k+1$ production sequences for S' , we then have

$$\begin{aligned} S' &\xRightarrow{\ell}_{G'} SS' \xRightarrow{\ell}_G^* \\ w_1 S' &\xRightarrow{\ell}_{G'} w_1 SS' \xRightarrow{\ell}_G^* \cdots \xRightarrow{\ell}_G^* \\ w_1 \cdots w_{k-1} S' &\xRightarrow{\ell}_{G'} w_1 \cdots w_{k-1} SS' \xRightarrow{\ell}_G^* \\ w_1 \cdots w_{k-1} w_k S' &\xRightarrow{\ell}_{G'} w_1 \cdots w_{k-1} w_k = w \end{aligned}$$

for suitable strings $w_1, \dots, w_k \in \Sigma^*$. By definition of G' and Lemma 3.15, we then also have $S \Rightarrow_G^* w_i$ for $1 \leq i \leq k$. Thus $w_1, \dots, w_k \in \mathcal{L}(G)$. Hence $w = w_1 \cdots w_k \in \mathcal{L}(G)^*$ and $\mathcal{L}(G') \subseteq \mathcal{L}(G)^*$.

If $w \in \mathcal{L}(G)^*$, then $w = w_1 \cdots w_k$ for suitable $k \geq 0$ and $w_1, \dots, w_k \in \mathcal{L}(G)$. We have $S \Rightarrow_G^* w_i$ for $1 \leq i \leq k$. Therefore, by Lemma 3.15, we have

$$w_1 \cdots w_{i-1} S' \Rightarrow_{G'} w_1 \cdots w_{i-1} SS' \Rightarrow_{G'}^* w_1 \cdots w_{i-1} w_i S'$$

for $1 \leq i \leq k$. Combining these derivations with $w_1 \cdots w_k S' \Rightarrow_{G'} w_1 \cdots w_k$, and using $w = w_1 \cdots w_k$ this yields $S' \Rightarrow_{G'}^* w$. Thus $w \in \mathcal{L}(G')$ and hence $\mathcal{L}(G)^* \subseteq \mathcal{L}(G')$. \square

At the end of the section, see Theorem 3.59, we will show that the intersection of two context-free languages need not to be context-free again. So, the class of context-free languages is not closed under intersection. However, a restricted closure property for intersection does hold.

Theorem 3.52. For a context-free language L_1 and a regular language L_2 , it holds that the intersection $L_1 \cap L_2$ is a context-free language.

Proof. Let $P = (Q_1, \Sigma, \Delta, \emptyset, \rightarrow_P, q_0^1, F_1)$ be a push-down automaton accepting L_1 , and let $D = (Q_2, \Sigma, \delta, q_0^2, F_2)$ be a deterministic finite automaton accepting L_2 . Note automaton D has no τ -transitions. We construct a ‘product’ push-down automaton

$$P' = (Q_1 \times Q_2, \Sigma, \Delta, \emptyset, \rightarrow_{P'}, (q_0^1, q_0^2), F_1 \times F_2)$$

where the transition relation $\rightarrow_{P'}$ is defined by

$$\begin{aligned} (q_1, q_2) &\xrightarrow{a[d/x]}_{P'} (q'_1, q'_2) \quad \text{iff} \quad q_1 \xrightarrow{a[d/x]}_P q'_1 \quad \text{and} \quad \delta(q_2, a) = q'_2 \\ (q_1, q_2) &\xrightarrow{\tau[d/x]}_{P'} (q'_1, q_2) \quad \text{iff} \quad q_1 \xrightarrow{\tau[d/x]}_P q'_1 \end{aligned}$$

for $q_1, q'_1 \in Q_1$, $q_2, q'_2 \in Q_2$, $a \in \Sigma$, $d \in \Delta_\emptyset$, and $x \in \Delta^*$. Thus, either the PDA component and the DFA component simultaneously process input and update their

state, or the PDA component makes a τ -move, while the DFA component doesn't move. We will show $\mathcal{L}(P') = \mathcal{L}(P) \cap \mathcal{L}(D)$. We first claim

$$\begin{aligned} ((q_1, q_2), w, z) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z') \quad \text{iff} \\ (q_1, w, z) \vdash_P^* (\bar{q}_1, \varepsilon, z') \text{ and } (q_2, w) \vdash_D^* (\bar{q}_2, \varepsilon) \end{aligned} \quad (3.10)$$

for states $q_1, \bar{q}_1 \in Q_1$, $q_2, \bar{q}_2 \in Q_2$, $w \in \Sigma^*$ and $z, z' \in \Delta^*$.

By induction on n , if $((q_1, q_2), w, z) \vdash_{P'}^n ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$, then $(q_1, w, z) \vdash_P^* (\bar{q}_1, \varepsilon, z')$ and $(q_2, w) \vdash_D^* (\bar{q}_2, \varepsilon)$. Basis, $n = 0$: Then we have $w = \varepsilon$, $q_1 = \bar{q}_1$, $q_2 = \bar{q}_2$, and $z = z'$. Therefore $(q_1, w, z) \vdash_{P'}^* (\bar{q}_1, \varepsilon, z')$ and $(q_2, w) \vdash_D^* (\bar{q}_2, \varepsilon)$. Induction step, $n + 1$: Suppose

$$((q_1, q_2), w, z) \vdash_{P'} ((\tilde{q}_1, \tilde{q}_2), \tilde{w}, \tilde{z}) \vdash_{P'}^n ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$$

Then either

- (i) input is processed: $w = a\tilde{w}$, $z = dy$, $\tilde{z} = xy$ and $(q_1, q_2) \xrightarrow{a[d/x]}_{P'} (\tilde{q}_1, \tilde{q}_2)$ because $q_1 \xrightarrow{a[d/x]}_P \tilde{q}_1$ and $\delta(q_2, a) = \tilde{q}_2$, or
- (ii) a τ -move is made: $w = \tilde{w}$, $z = dy$, $\tilde{z} = xy$, $q_2 = \tilde{q}_2$, and $(q_1, q_2) \xrightarrow{\tau[d/x]}_{P'} (\tilde{q}_1, q_2)$ because $q_1 \xrightarrow{\tau[d/x]}_P \tilde{q}_1$.

Thus either (i) $w = a\tilde{w}$, $(q_1, w, z) \vdash_P ((\tilde{q}_1, \tilde{w}, \tilde{z})$ and $q_2 \xrightarrow{a}_D \tilde{q}_2$, or (ii) $w = \tilde{w}$, $q_2 = \tilde{q}_2$, and $(q_1, w, z) \vdash_P ((\tilde{q}_1, \tilde{w}, \tilde{z})$. By induction hypothesis, $(\tilde{q}_1, \tilde{w}, \tilde{z}) \vdash_{P'}^* (\bar{q}_1, \varepsilon, z')$ and $(\tilde{q}_2, \tilde{w}) \vdash_D^* (\bar{q}_2, \varepsilon)$. Therefore, in both cases $(q_1, w, z) \vdash_{P'}^* (\bar{q}_1, \varepsilon, z')$ and $(q_2, w) \vdash_D^* (\bar{q}_2, \varepsilon)$.

By induction on n , if $(q_1, w, z) \vdash_P^n (\bar{q}_1, \varepsilon, z')$ and $(q_2, w) \vdash_D^* (\bar{q}_2, \varepsilon)$, then it holds that $((q_1, q_2), w, z) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$. Note, the index n for the derivation in P and the superscript $*$ for the derivation in D . Basis, $n = 0$: Then we have $w = \varepsilon$, $q_1 = \bar{q}_1$, $z = z'$, and also $q_2 = \bar{q}_2$, since D has no ε -transitions. Thus $((q_1, q_2), w, z) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$.

Induction step, $n + 1$: Suppose $(q_1, w, z) \vdash_{P'} ((\tilde{q}_1, \tilde{w}, \tilde{z}) \vdash_{P'}^n (\bar{q}_1, \varepsilon, z')$. We distinguish two cases. Case (i): $w = a\tilde{w}$, $z = dy$, $q_1 \xrightarrow{a[d/x]}_P \tilde{q}_1$, and $\tilde{z} = xy$. Since $w = a\tilde{w}$ and D is deterministic, we have $(q_2, w) \vdash_D ((\tilde{q}_2, \tilde{w}) \vdash_D^* (\bar{q}_2, \varepsilon)$ for a unique state $\tilde{q}_2 \in Q_2$. Therefore, by construction of P' , $(q_1, q_2) \xrightarrow{a[d/x]}_{P'} (\tilde{q}_1, \tilde{q}_2)$. From $(\tilde{q}_1, \tilde{w}, \tilde{z}) \vdash_{P'}^n (\bar{q}_1, \varepsilon, z')$ and $(\tilde{q}_2, \tilde{w}) \vdash_D^* (\bar{q}_2, \varepsilon)$ and the induction hypothesis it follows that $((\tilde{q}_1, \tilde{q}_2), \tilde{w}, \tilde{z}) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$. Therefore $((q_1, q_2), w, z) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$ in this case. Case (ii): $w = \tilde{w}$, $z = dy$, $q_1 \xrightarrow{\tau[d/x]}_P \tilde{q}_1$, and $\tilde{z} = xy$. By construction of P' we have $(q_1, q_2) \xrightarrow{\tau[d/x]}_{P'} (\tilde{q}_1, q_2)$. Thus, $((q_1, q_2), w, z) \vdash_{P'} ((\tilde{q}_1, q_2), w, \tilde{z})$. From $(\tilde{q}_1, w, \tilde{z}) \vdash_{P'}^n (\bar{q}_1, \varepsilon, z')$, $(q_2, w) \vdash_D^* (\bar{q}_2, \varepsilon)$, and the induction hypothesis it follows that $((\tilde{q}_1, q_2), w, \tilde{z}) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$. Therefore, $((q_1, q_2), w, z) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z')$ also in this case, which concludes the induction step.

To show $\mathcal{L}(P') = \mathcal{L}(P) \cap \mathcal{L}(D)$ we reason as follows: Suppose $w \in L_1 \cap L_2$. Thus, $w \in L_1$, hence $(q_0^1, w, \varepsilon) \vdash_P^* (\bar{q}_1, \varepsilon, z)$ for some $\bar{q}_1 \in F_1$ and $z \in \Delta^*$, and $w \in L_1$, hence $(q_0^2, w) \vdash_D^* (\bar{q}_2, \varepsilon)$ for some $\bar{q}_2 \in F_2$. By the claim (3.10), implication right to

left, it follows that $((q_0^1, q_0^2), w, \varepsilon) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z)$ while $(\bar{q}_1, \bar{q}_2) \in F_1 \times F_2$. Therefore $w \in \mathcal{L}(P')$.

Suppose $w \in \mathcal{L}(P')$. Then $((q_0^1, q_0^2), w, \varepsilon) \vdash_{P'}^* ((\bar{q}_1, \bar{q}_2), \varepsilon, z)$ for some states $\bar{q}_1 \in F_1$, $\bar{q}_2 \in F_2$ and $z \in \Delta^*$. By the claim (3.10), implication left to right, it follows that $(q_0^1, w, \varepsilon) \vdash_P^* (\bar{q}_1, \varepsilon, z)$ and $(q_0^2, w) \vdash_D^* (\bar{q}_2, \varepsilon)$. Thus $w \in \mathcal{L}(P)$ and $w \in \mathcal{L}(D)$, hence $w \in \mathcal{L}_1 \cap \mathcal{L}_2$. \square

Example 3.53. Consider the language

$$L = \{ w \in \{a, b\}^* \mid \#_a(w) = \#_b(w), w \text{ has no substring } abba \}$$

It holds that L is a context-free language. We will use Theorem 3.52 to verify this.

Choose $L_1 = \{ w \in \{a, b\}^* \mid \#_a(w) = \#_b(w) \}$ as context-free language and $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has no substring } abba \}$ as regular language. L_2 is regular as it is the complement of the language $\{ w \in \{a, b\}^* \mid w \text{ has a substring } abba \}$, i.e. the complement of the language of the regular expression $(a + b)^* abba (a + b)^*$. Clearly, $L = L_1 \cap L_2$, hence L is context-free being the intersection of a context-free language and a regular language.

We next consider a counterpart of the Pumping Lemma for regular languages, Theorem ???. Also Theorem 3.54 can be used to prove negative results. Here, regarding a language not being context-free.

Theorem 3.54 (Pumping Lemma for context-free languages). Let L be a context-free language over an alphabet Σ . There exists a bound $m > 0$ such that each $w \in L$ with $|w| \geq m$ can be written as $w = uvxyz$ with $u, v, x, y, z \in \Sigma^*$, $vy \neq \varepsilon$, $|vxy| \leq m$, and for all $k \geq 0$: $uv^kxy^kz \in L$.

Proof. Let $G = (V, \Sigma, R, S)$ be a context-free grammar that generates L . Let $\kappa = \max\{s \mid A \rightarrow X_1 \cdots X_s \in R\}$. Put $m = \kappa^{\#V+1}$.

Pick $w \in L$ with $|w| \geq m$. Let PT be a parse tree for w , with root labeled S , and a minimal number of nodes.

If the height of PT is h , then $|w| \leq \kappa^h$. Since $|w| \geq m = \kappa^{\#V+1}$ we have $h \geq \#V + 1$. So PT has a path from root to a leaf of length h . This path has $h + 1$ nodes of which h nodes are labeled with a variable. Since $h \geq \#V + 1$ there is a variable A that occurs more than once on the path. See Figure 3.9.

Let x be the yield of the parse tree PT_{low} with the lowest occurrence of A on the path as its root. Let v and y be such that vxy is the yield of the parse tree PT_{high} with the second to lowest occurrence of A on the path as its root. Let u and z be such that $uvxyz$ is the yield of PT .

If we replace in PT the subtree PT_{high} by PT_{low} , we obtain a parse tree PT' with root S and yield uxz . Hence $uxz \in L$. If we replace in PT the subtree PT_{low} by PT_{high} , we obtain a parse tree with root S and yield uv^2xw^2z . Hence $uv^2xw^2z \in L$. If we iterate this k times we get a parse tree with root S and yield uv^kxw^kz . Hence $uv^kxw^kz \in L$.

To prove that $vy \neq \varepsilon$ we argue that if $vy = \varepsilon$ then $v = \varepsilon$ and $y = \varepsilon$. Thus the parse tree PT' above would be a parse tree with root S , yield $uxz = uvzyz = w$, but with a smaller number of nodes. This contradicts the minimality of PT . So $vy \neq \varepsilon$.

Finally, to establish that $|vxy| \leq m$ we argue that we can arrange that the height of parse tree PT_{high} is at most $\#V + 1$ by taking the root node of PT_{high} the lowest node for which there is a node labeled with the same variable down the path. It follows that PT_{high} has at most $\kappa^{\#V+1} = m$ leaves. Since vxy is the yield of PT_{high} , we have $|vxy| \leq m$. \square

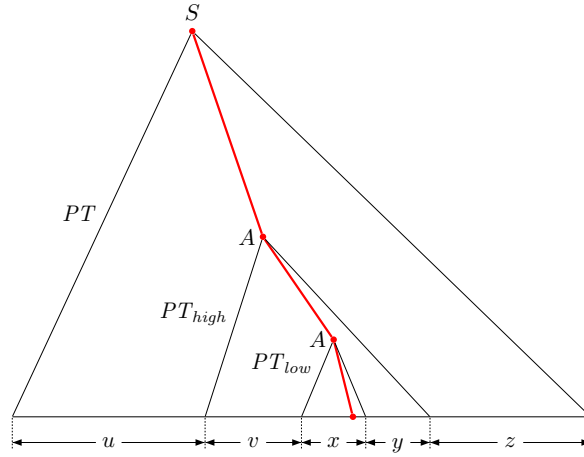


Figure 3.9: Repeating variable A in minimal parse tree PT with yield $uvxyz$.

Example 3.55. The language $L = \{ a^n b^n c^n \mid n \geq 0 \}$ is not context-free. We apply the Pumping Lemma. Choose $m > 0$ arbitrarily. Put $w = a^m b^m c^m \in L$. Then $|w| \geq m$. Suppose we u, v, x, y, z are such that $w = uvxyz$, $vy \neq \varepsilon$. Since $|vy| \leq |vxy| \leq m$, the string vy contains only a 's, only a 's and b 's, only b 's, only b 's and c 's, or only c 's. But then $w' = uv^2xy^2z$ has only more a 's, only more a 's and b 's, only more b 's, only more b 's and c 's, or only more c 's, respectively. So, the number of a 's, b 's and c 's in w' is not in balance and $w' \notin L$. We conclude that there is no $m > 0$ that fulfills the requirements of the Pumping Lemma. Hence L is not context-free.

Example 3.56. The language $L = \{ a^n \mid n \text{ is prime} \}$ is not context-free. We apply the Pumping Lemma to show this: Choose $m > 0$ arbitrarily. Let p be a prime number $p \geq m$. Put $w = a^p \in L$. Then $|w| \geq m$. Suppose we u, v, x, y, z are such that $w = uvxyz$, $vy \neq \varepsilon$ and $uv^n xy^n z \in L$ for each $n \geq 0$. Put $q = |vy|$ and $r = |uxz|$. Note $q > 0$. We have that $nq + r$ is prime, for each $n \geq 0$. Consider $n = 2q + r + 2$. It holds that $nq + r = (2q + r + 2)q + r = 2q^2 + rq + 2q + r = (q + 1)(2q + r)$. Since both $q + 1 > 1$ and $2q + r > 1$, it follows that $r + nq$ is not prime for this choice of n . Contradiction. So, no $m > 0$ meets the requirements of the Pumping Lemma. Hence L is not a context-free language.

Example 3.57. Although $\{ww^R \mid w \in \{a, b\}^*\}$ is a context-free language, the language $L = \{ww \mid w \in \{a, b\}^*\}$ is not context-free. To see this, we apply the Pumping Lemma. Let $m > 0$ be arbitrary. Consider the string $a^m b^m a^m b^m \in L$. Pick u, v, x, y, z such that $a^m b^m a^m b^m = uvxyz$, $vy \neq \varepsilon$, and $|vxy| \leq m$.

Since $|vxy| \leq m$ and $|vy| > 0$, leaving out v and y from $a^m b^m a^m b^m$ results in a string uxz of the form

$$a^\ell b^m a^m b^m, \quad a^m b^\ell a^m b^m, \quad a^m b^m a^\ell b^m, \quad \text{or} \quad a^m b^m a^m b^\ell$$

where $0 \leq \ell < m$, or

$$a^\ell b^k a^m b^m, \quad a^m b^\ell a^k b^m, \quad \text{or} \quad a^m b^m a^\ell b^k$$

where $0 \leq k, \ell$, and $k < m$ or $\ell < m$. But, these possibilities cannot equal ww for some string $w \in \{a, b\}^*$. This is because for ww to match, w must start with an a and end in a b . Therefore, both the first group of a 's and b 's equals w and the second group of a 's and b 's equals w . However, because of the restrictions on ℓ and k , in all cases this does not hold.

So no choice of u, v, x, y, z can be right. Hence, no m exists that meets the requirements of the Pumping Lemma. So, L is not a context-free language.

Example 3.58. The language $L = \{w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$ is not context-free. Suppose L is context-free. Then, by Lemma 3.52, $L \cap a^* b^* c^*$ is context-free too. But $L \cap a^* b^* c^* = \{a^n b^n c^n \mid n \geq 0\}$ and is not context-free as shown in Example 3.55 above.

Theorem 3.59. The class of context-free languages is not closed under intersection and complement.

Proof. The languages $L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$ and $L_2 = \{a^n b^m c^m \mid n, m \geq 0\}$ are both context-free. However, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ which is not context-free according to Example 3.55.

The union of two context-free languages is context-free again, according to Theorem 3.51. If the complement L^C of a context-free language L would be context-free in general, then also

$$L_1 \cap L_2 = (L_1^C \cup L_2^C)^C$$

would be context-free. But it isn't, as was just shown. \square

Theorem 3.60. Let $L \subseteq \Sigma^*$ be a CFL represented by a CFG $G = (V, \Sigma, R, S)$ in Chomsky normal form. Let $w \in \Sigma^*$ be a string over Σ . Then it can be decided if $w \in L$ or not.

Proof. Let $w = a_1 \cdots a_n \in \Sigma^*$ where $n \geq 0$. If $n = 0$, then we have $w \in L$ iff $S \rightarrow \varepsilon \in R$. Suppose $n > 0$. Define the sets $V_{i,k} \subseteq V$, for $1 \leq i \leq k \leq n$ as follows

$$\begin{aligned} V_{i,i} &= \{A \in V \mid A \rightarrow a_i \in R\} \quad \text{for } 1 \leq i \leq n \\ V_{i,k} &= \bigcup \{A \in V \mid \exists A \rightarrow BC \in R \exists j, i \leq j < k: B \in V_{i,j} \wedge C \in V_{j+1,k}\} \\ &\quad \text{for } 1 \leq i < k \leq n \end{aligned}$$

The intuition for $V_{i,k}$ is that each variable in $V_{i,k}$ can produce, in one or more steps, the substring $a_i \cdots a_k$ of w . With $V_{1,n}$ defined as above, decide $w \in L$ if $S \in V_{1,n}$; decide $w \notin L$ if $S \notin V_{1,n}$.

To establish the correctness of the above decision procedure, we prove

$$A \Rightarrow_G^* a_i \cdots a_k \iff A \in V_{i,k}$$

for $A \in V$ and $1 \leq i \leq k \leq n$, by induction on $k - i$. Basis, $k - i = 0$: Direct from the definition of $V_{i,i}$, for $1 \leq i \leq n$. Induction step, $k - i > 0$: (\Rightarrow) If $A \Rightarrow_G^* a_i \cdots a_k$ then we must have $A \Rightarrow_G BC \Rightarrow_G^* a_i \cdots a_k$ for some $B, C \in V$, since G is in Chomsky normal form. Thus we can find j , $i \leq j < k$, such that $B \Rightarrow_G^* a_i \cdots a_j$ and $C \Rightarrow_G^* a_{j+1} \cdots a_k$. Note $j - i < k - i$ and $k - (j + 1) < k - i$. By induction hypothesis we find $B \in V_{i,j}$, $C \in V_{j+1,k}$. Therefore, $A \in V_{i,k}$ by definition of $V_{i,k}$. (\Leftarrow) If $A \in V_{i,k}$, where $k > i$, we can choose a production $A \rightarrow BC \in R$ and an index j , $i \leq j < k$ such that $B \in V_{i,j}$, $C \in V_{j+1,k}$. Note again $j - i < k - i$ and $k - (j + 1) < k - i$. Thus, by induction hypothesis, $B \Rightarrow_G^* a_i \cdots a_j$, $C \Rightarrow_G^* a_{j+1} \cdots a_k$. Hence, we have $A \Rightarrow_G BC \Rightarrow_G^* a_i \cdots a_j a_{j+1} \cdots a_k = a_i \cdots a_k$, which concludes the induction step. \square

Example 3.61 (HMU Example 7.34). Consider the CFG G with variables S, A, B, C , terminals $0, 1$, start symbol S , and production rules

$$S \rightarrow AB \mid BC \quad A \rightarrow BA \mid 0 \quad B \rightarrow CC \mid 1 \quad C \rightarrow AB \mid 0$$

Note that G is in Chomsky normal form. Consider the string 10010. We verify, using the algorithm described above, whether $10010 \in \mathcal{L}(G)$.

We set up a table of 5 columns, one for each position of 10010, and six rows, the bottom row will initially store the letters of the string. The cell on the i -th row and k -th column, for $1 \leq i, k \leq 5$, will hold the set $V_{i,i+k-1}$, as defined by the algorithm, provided $i \leq i+k-1 \leq 5$. Otherwise the cell is left empty. We put the i th letter of 10010 at the cell in column i of row 6. See Figure 3.10 on the left for a symbolic representation.

The actual values of the sets V_{ik} inductively built up from row 5 going up to row 1. We put a variable in $V_{1,1}$ and $V_{4,4}$ if it can directly produce 1. Thus, $V_{1,1} = V_{4,4} = \{B\}$. Likewise $V_{2,2}$, $V_{3,3}$, and $V_{5,5}$ are set to $\{A, C\}$, since both A and C have a production rule with righthand-side 0. This describes row 5 of the table at the right of Figure 3.10.

Now, by definition, $V_{1,2}$ contains variables that have a righthand-side yielding a variable in $V_{1,1}$ and in $V_{2,2}$, respectively. Since $S \rightarrow BC$ and $B \in V_{1,1}$, $C \in V_{2,2}$, we put S in $V_{1,2}$. Because A has the production BA , $B \in V_{1,1}$, and $A \in V_{2,2}$, we also put A in $V_{1,2}$. The variable B is not put into $V_{1,2}$; for its production $B \rightarrow CC$ we have $C \notin V_{1,1}$. Also C is not added to $V_{1,2}$. (It would, if it had a production $C \rightarrow BA$.) The sets $V_{2,3}$, $V_{3,4}$, and $V_{4,5}$ are established similarly.

To see if S has to be put in the set $V_{1,3}$, we check (i) for the production $S \rightarrow AB$ if $A \in V_{1,1} = \{B\}$ and $B \in V_{2,3} = \{B\}$, or $A \in V_{1,2} = \{S, A\}$ and $B \in V_{3,3} = \{A, C\}$, and (ii), for the production $S \rightarrow BC$ if $B \in V_{1,1} = \{B\}$ and $C \in V_{2,3} = \{B\}$, or $B \in V_{1,2} = \{S, A\}$ and $C \in V_{3,3} = \{A, C\}$. Since none of the case hold S is not added to $V_{1,3}$. Also the other variable are not added to $V_{1,3}$. Hence, it remains empty.

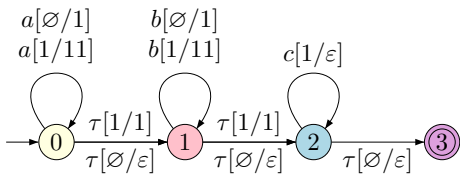
Answers to exercises of Chapter 3

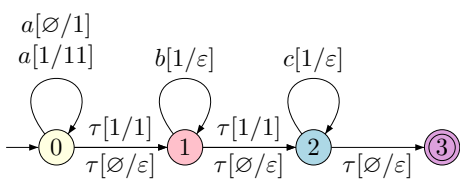
Answers for exercises of Section 3.1

Answer to Exercise 3.1.1

- (a) One maximal derivation for (q_0, ab, ε) :
 $(q_0, ab, \varepsilon) \vdash_P (q_0, b, 1) \vdash_P (q_1, \varepsilon, \varepsilon) \not\vdash_P$
- (b) One maximal derivation for (q_0, abb, ε) :
 $(q_0, abb, \varepsilon) \vdash_P (q_0, bb, 1) \vdash_P (q_1, b, \varepsilon) \not\vdash_P$
- (c) One maximal derivation for (q_0, aab, ε) :
 $(q_0, aab, \varepsilon) \vdash_P (q_0, ab, 1) \vdash_P (q_0, b, 11) \vdash_P (q_1, \varepsilon, 1) \vdash_P (q_0, \varepsilon, \varepsilon) \not\vdash_P$
- (d) Three maximal derivations for $(q_0, aaabbb, \varepsilon)$:
- $$(q_0, aaabbb, \varepsilon) \vdash_P (q_0, aaabbb, 1) \vdash_P (q_0, abbb, 11) \vdash_P (q_0, bbb, 111) \vdash_P (q_1, bb, 11) \vdash_P (q_1, b, 1) \vdash_P (q_1, \varepsilon, \varepsilon) \not\vdash_P$$
- $$(q_0, aaabbb, \varepsilon) \vdash_P (q_0, aaabbb, 1) \vdash_P (q_0, abbb, 11) \vdash_P (q_0, bbb, 111) \vdash_P (q_1, bb, 11) \vdash_P (q_1, b, 1) \vdash_P (q_0, b, \varepsilon) \not\vdash_P$$
- $$(q_0, aaabbb, \varepsilon) \vdash_P (q_0, aaabbb, 1) \vdash_P (q_0, abbb, 11) \vdash_P (q_0, bbb, 111) \vdash_P (q_1, bb, 11) \vdash_P (q_0, bb, 1) \vdash_P (q_1, b, \varepsilon) \not\vdash_P$$

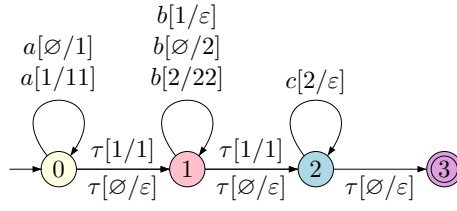
Answer to Exercise 3.1.2

- (a)
- 

state q	input w	stack x	
q_0	a^n	1^n	$n \geq 0$
q_1	$a^n b^m$	1^{n+m}	$n, m \geq 0$
q_2	$a^n b^m c^\ell$	$1^{n+m-\ell}$	$0 \leq \ell \leq n+m$
q_3	$a^n b^m c^\ell$	ε	$n+m-\ell = 0$
-
- (b)
- 

state q	input w	stack x	
q_0	a^ℓ	1^ℓ	$\ell \geq 0$
q_1	$a^\ell b^n$	$1^{\ell-n}$	$0 \leq n \leq \ell$
q_2	$a^\ell b^n c^m$	$1^{\ell-n-m}$	$0 \leq m \leq \ell-n$
q_3	$a^\ell b^n c^m$	ε	$\ell-n-m = 0$

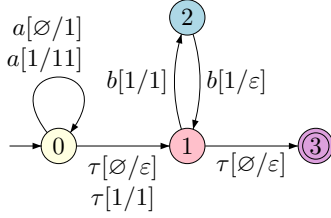
(c)



state q	input w	stack x	
q_0	a^n	1^n	$n \geq 0$
q_1	$a^n b^\ell$	$1^{n-\ell}$	$0 \leq \ell \leq n$
	$a^n b^{n+k}$	2^k	$k \geq 0$
q_2	$a^n b^{n+k} c^m$	2^{k-m}	$0 \leq m \leq k$
q_3	$a^n b^{n+k} c^m$	ε	$k = m$

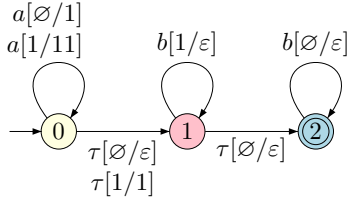
Answer to Exercise 3.1.3

(a)



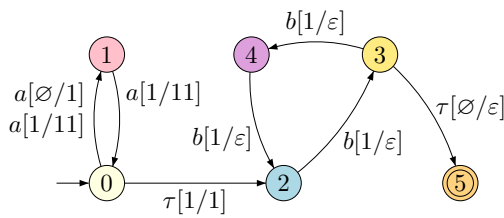
state q	input w	stack x	
q_0	a^n	1^n	$n \geq 0$
q_1	$a^n b^m$	1^{n+m}	$n, m \geq 0$
q_2	$a^n b^m c^\ell$	$1^{n+m-\ell}$	$0 \leq \ell \leq n+m$
q_3	$a^n b^m c^\ell$	ε	$n+m-\ell = 0$

(b)



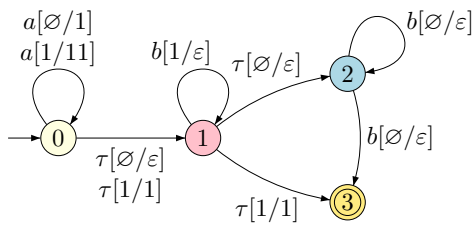
state q	input w	stack x	
q_0	a^n	1^n	$n \geq 0$
q_1	$a^n b^m$	1^{n+m}	$0 \leq m \leq n$
q_2	$a^n b^{m+\ell}$	ε	$n = m, \ell \geq 0$

(c)



state q	input w	stack x	
q_0	a^{2n}	1^{2n}	$n \geq 0$
q_1	a^{2n+1}	1^{2n+1}	$n \geq 0$
q_2	$a^{2n} b^{3m}$	1^{2n-3m}	$0 \leq 3m \leq 2n$
q_3	$a^{2n} b^{3m+1}$	$1^{2n-(3m+1)}$	$0 \leq 3m+1 \leq 2n$
q_4	$a^{2n} b^{3m+2}$	$1^{2n-(3m+2)}$	$0 \leq 3m+2 \leq 2n$
q_5	$a^{2n} b^{3m+1}$	ε	$2n = 3m+1$

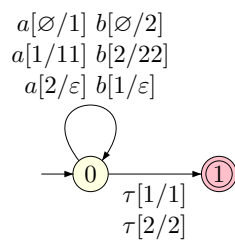
(d)



state q	input w	stack x	
q_0	a^n	1^n	$n \geq 0$
q_1	$a^n b^m$	1^{n-m}	$0 \leq m \leq n$
q_2	$a^n b^{m+\ell}$	ε	$n-m=0, \ell \geq 0$
q_3	$a^n b^{m+\ell+1}$	ε	$n-m=0, \ell \geq 0$
	$a^n b^m$	1^{n-m}	$n-m > 0$

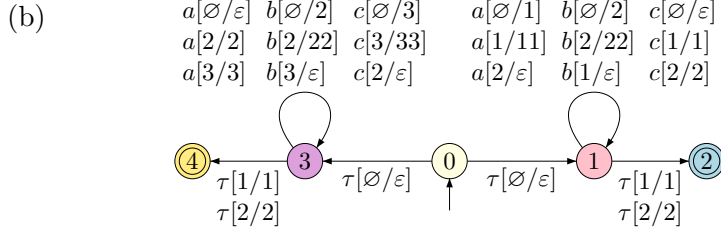
Answer to Exercise 3.1.4

(a)



state q	input w	stack x	
q_0	w	1^n	$\#_a(w) - \#_b(w) = n, n \geq 0$
	w	2^m	$\#_b(w) - \#_a(w) = m, m \geq 0$
q_1	w	1^n	$\#_a(w) - \#_b(w) = n, n > 0$
	w	2^m	$\#_b(w) - \#_a(w) = m, m > 0$

A computation arrives at the only accepting state q_1 , precisely when either $\#_a(w) > \#_b(w)$ or $\#_b(w) > \#_a(w)$ for input w . Thus, $\#_a(w) \neq \#_b(w)$. Hence, the language accepted by the PDA is L_8 .



state q	input w	stack x	
q_0	w	ε	
q_1	w	1^n	$\#_a(w) - \#_b(w) = n, n \geq 0$
	w	2^m	$\#_b(w) - \#_a(w) = m, m \geq 0$
q_2	w	1^n	$\#_a(w) - \#_b(w) = n, n > 0$
	w	2^m	$\#_b(w) - \#_a(w) = m, m > 0$
q_3	w	2^m	$\#_b(w) - \#_c(w) = m, m \geq 0$
	w	3^ℓ	$\#_c(w) - \#_b(w) = \ell, \ell \geq 0$
q_4	w	2^m	$\#_b(w) - \#_c(w) = m, m > 0$
	w	3^ℓ	$\#_c(w) - \#_b(w) = \ell, \ell > 0$

The PDA is a non-deterministic combination of two PDA similar to the one of item (a). The left part, with accepting state q_4 accepts $L_9^1 = \{ w \in \{a, b, c\}^* \mid \#_b(w) \neq \#_c(w) \}$. The right part, with accepting state q_2 , accepts $L_9^2 = \{ w \in \{a, b, c\}^* \mid \#_a(w) \neq \#_b(w) \}$. Thus, the PDA itself accepts $L_9^1 \cup L_9^2 = L_9$.

Answers for exercises of Section 3.2

Answer to Exercise 3.2.1

- a $S \xRightarrow{\ell}_G XbY \xRightarrow{\ell}_G aXbY \xRightarrow{\ell}_G aaXbY \xRightarrow{\ell}_G aabY \xRightarrow{\ell}_G aabaY \xRightarrow{\ell}_G aababY \xRightarrow{\ell}_G aabab$;
 $S \xRightarrow{r}_G XbY \xRightarrow{r}_G XbaY \xRightarrow{r}_G XbabY \xRightarrow{r}_G Xbab \xRightarrow{r}_G aXbab \xRightarrow{r}_G aaXbab \xRightarrow{r}_G aabab$
- (b) $S \xRightarrow{\ell}_G XbY \xRightarrow{\ell}_G bY \xRightarrow{\ell}_G baY \xRightarrow{\ell}_G baaY \xRightarrow{\ell}_G baabY \xRightarrow{\ell}_G baab$;
 $S \xRightarrow{r}_G XbY \xRightarrow{r}_G XbaY \xRightarrow{r}_G XbaaY \xRightarrow{r}_G XbaabY \xRightarrow{r}_G Xbaab \xRightarrow{r}_G baab$
- (c) $S \xRightarrow{\ell}_G XbY \xRightarrow{\ell}_G aXbY \xRightarrow{\ell}_G aaXbY \xRightarrow{\ell}_G aaaXbY \xRightarrow{\ell}_G aaabY \xRightarrow{\ell}_G aaabbY \xRightarrow{\ell}_G aaabb$;
 $S \xRightarrow{r}_G XbY \xRightarrow{r}_G XbbY \xRightarrow{r}_G Xbb \xRightarrow{r}_G aXbb \xRightarrow{r}_G aaXbb \xRightarrow{r}_G aaaXbb \xRightarrow{r}_G aaabb$

Answer to Exercise 3.2.2

- (a) ($\mathcal{L}_G(A) \subseteq L_A$) We prove $A \Rightarrow_G^n w$ implies $w \in L_A$, for all $w \in \{a, b\}^*$, by induction on n . Basis, $n = 0$: There is nothing to show, $S \notin \{a, b\}^*$. Induction step, $k + 1$: If $A \Rightarrow_G^{n+1} w$, then $A \Rightarrow_G \varepsilon \Rightarrow_G^n w$ or $A \Rightarrow_G aA \Rightarrow_G^n w$. Thus $w = \varepsilon$ or $w = aw'$ where $A \Rightarrow_G^n w'$ (by Lemma 3.15). In the first case we are done, $\varepsilon \in L_A$. In the second case we have $w' \in L_A$, i.e. $w' = a^m$ for some $m \geq 0$. But then it holds that $w = aw' = aa^m = a^{m+1}$.
- ($L_A \subseteq \mathcal{L}_G(A)$) By induction on n , $a^n \in \mathcal{L}_G(A)$. Basis, $n = 0$: We have $A \Rightarrow_G \varepsilon$. Induction step, $n + 1$: By induction hypothesis $a^n \in \mathcal{L}_G(A)$, i.e. $A \Rightarrow_G^* a^n$. Therefore, by Lemma 3.15, $aA \Rightarrow_G^* aa^n$. Thus, since $A \rightarrow_G aA$, we obtain $A \Rightarrow_G aA \Rightarrow_G^* aa^n = a^{n+1}$.
- (b) Similar to part (a) one shows $\mathcal{L}_G(B) = \{b^{mkern1mu n} \mid n \geq 0\}$. Note, $L = \mathcal{L}_G(A) \cup \mathcal{L}_G(B)$. We have, for $w \in \{a, b\}^*$, $w \in \mathcal{L}(G)$ iff $S \Rightarrow_G^* w$ iff $A \Rightarrow_G^* w$ or $B \Rightarrow_G^* w$ iff $w \in \mathcal{L}_G(A)$ or $w \in \mathcal{L}_G(B)$ iff $w \in \mathcal{L}_G(A) \cup \mathcal{L}_G(B)$ iff $w \in L$. Thus $\mathcal{L}(G) = L$.

Answer to Exercise 3.2.3

- (a) Consider CFG G_1 given by

$$S \rightarrow aSb \mid A \mid B, \quad A \rightarrow a \mid aA, \quad B \rightarrow b \mid bB$$

We have $\mathcal{L}_{G_1}(A) = \{a^n \mid n \geq 1\}$ and $\mathcal{L}_{G_1}(B) = \{b^n \mid n \geq 1\}$ with a proof similar to that of Exercise 3.2.2. Thus $v \in \mathcal{L}_{G_1}(A)$ iff $v = a^n$ for some $n \geq 1$, and $u \in \mathcal{L}_{G_1}(B)$ iff $u = b^n$ for some $n \geq 1$.

($\mathcal{L}(G_1) \subseteq L$) Suppose $S \Rightarrow_{G_1}^* w$ for $w \in \{a, b\}^*$. Then $S \Rightarrow_{G_1}^n a^n Sb^n \Rightarrow_{G_1} a^n Ab^n \Rightarrow_{G_1}^* w$ for some $n \geq 0$, or $S \Rightarrow_{G_1}^n a^n Sb^n \Rightarrow_{G_1} a^n Bb^n \Rightarrow_{G_1}^* w$ for some $n \geq 0$. In the first case we have $w = a^n vb^n$ and $A \Rightarrow_{G_1}^* v$. In the second case we have $w = a^n ub^n$ and $B \Rightarrow_{G_1}^* u$. So, $w = a^n vb^n$ and $v = a^m$ for some $n \geq 0$, $m \geq 1$, or $w = a^n ub^n$ and $u = b^m$ for some $n \geq 0$, $m \geq 1$. Hence $w = a^{n+m}b^n$ or $w = a^n b^{n+m}$ for some $n \geq 0$, $m \geq 1$. In either case, $w \in L_1$.

($L_1 \subseteq \mathcal{L}(G_1)$) If $w \in L_1$ then $w = a^{n+m}b^n$ or $w = a^n b^{n+m}$ for some $n \geq 0$, $m \geq 1$. Suppose $w = a^{n+m}b^n$ where $n \geq 0$, $m \geq 1$. We have $S \Rightarrow_{G_1}^n a^n Sb^n \Rightarrow_{G_1} a^n Ab^n \Rightarrow_{G_1}^{m-1} a^n a^{m-1} Ab^n \Rightarrow_{G_1} a^n a^{m-1} ab^n = a^{n+m}b^n$, thus $a^{n+m}b^n \in \mathcal{L}(G_1)$. Similarly, if $w = a^n b^{n+m}$ for some $n \geq 0$, $m \geq 1$, then $w \in \mathcal{L}(G_1)$.

- (b) Define the CFG G_2 by

$$\begin{aligned} S &\rightarrow DC \mid AE \\ A &\rightarrow \varepsilon \mid aA, \quad B \rightarrow \varepsilon \mid bB, \quad C \rightarrow \varepsilon \mid cC \\ D &\rightarrow aDb \mid aA \mid bB, \quad E \rightarrow bEc \mid bB \mid cC \end{aligned}$$

One can show

$$\begin{aligned} \mathcal{L}_{G_2}(A) &= \{a^n \mid n \geq 0\} \\ \mathcal{L}_{G_2}(C) &= \{c^n \mid n \geq 0\} \\ \mathcal{L}_{G_2}(D) &= \{a^n b^m \mid n, m \geq 0, n \neq m\} \\ \mathcal{L}_{G_2}(E) &= \{b^m c^\ell \mid m, \ell \geq 0, m \neq \ell\} \end{aligned}$$

by proofs similar to the proofs given for Exercise 3.2.2 and for part (a). We have

$$\begin{aligned}\mathcal{L}(G) &= \mathcal{L}_{G_2}(D)\mathcal{L}_{G_2}(C) \cup \mathcal{L}_{G_2}(A)\mathcal{L}_{G_2}(E) \\ &= \{a^n b^m c^\ell \mid n, m, \ell \geq 0, n \neq m\} \cup \{a^n b^m c^\ell \mid n, m, \ell \geq 0, m \neq \ell\} \\ &= \{a^n b^m c^\ell \mid n, m, \ell \geq 0, n \neq m \vee m \neq \ell\}\end{aligned}$$

Answer to Exercise 3.2.4 A CFG for **0** has no production rules; a CFG for **1** has one production rule, viz. $S \rightarrow \varepsilon$; a CFG for the regular expression **a**, for $a \in \Sigma$, has one production rule, viz. $S \rightarrow a$.

Suppose $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ are CFG such that $\mathcal{L}(G_1) = \mathcal{L}(r_1)$ and $\mathcal{L}(G_2) = \mathcal{L}(r_2)$ for two regular expressions r_1 and r_2 . Assume $V_1 \cap V_2 = \emptyset$ and the variable S is fresh, i.e. $S \notin V_1 \cup V_2$. Put $G_{1+2} = (\{S\} \cup V_1 \cup V_2, \Sigma, \{S \rightarrow S_1, S \rightarrow S_2\} \cup R_1 \cup R_2, S)$. Then $\mathcal{L}(G_{1+2}) = \mathcal{L}(r_1 + r_2)$.

Let G_1, G_2, r_1, r_2 and S be as above. Put $G_{1.2} = (\{S\} \cup V_1 \cup V_2, \Sigma, \{S \rightarrow S_1 S_2\} \cup R_1 \cup R_2, S)$. Then $\mathcal{L}(G_{1.2}) = \mathcal{L}(r_1 \cdot r_2)$.

Suppose $G = (V, \Sigma, R, S)$ is a CFG such that $\mathcal{L}(G) = \mathcal{L}(r)$ for a regular expression r . Let S' be a fresh variable, i.e. $S' \notin V$. Put $G_* = (\{S'\} \cup V, \Sigma, \{S' \rightarrow \varepsilon, S' \rightarrow SS'\} \cup R, S')$. Then $\mathcal{L}(G_*) = \mathcal{L}(r^*)$.

Answer to Exercise 3.2.5 We prove, if $S \Rightarrow_G^k x$ then $x \in \mathcal{X}$ by induction on k . Basis, $k = 0$: Then we have $x = S$ and $S \in \mathcal{X}$. Induction step, $k + 1$: if $S \Rightarrow_G aS \Rightarrow_G^k x$, then $x = ax'$ and $x' \in \mathcal{X}$ by induction hypothesis. It follows that $x \in \mathcal{X}$. Similarly, if $S \Rightarrow Sb \Rightarrow_G^k x$ then $x = x'b$ and $x' \in \mathcal{X}$ by induction hypothesis, and it follows that $x \in \mathcal{X}$. If $S \Rightarrow_G a \Rightarrow_G^k x$ then $x = a \in \mathcal{X}$; if $S \Rightarrow_G b \Rightarrow_G^k x$ then $x = b$ and $x \in \mathcal{X}$. If $S \Rightarrow_G^* w$ and $w \in \{a, b\}$ we have $w = a^n b^m$ for some $n, m \geq 0$ with $n + m \geq 1$. So, w has no substring ba .

Answer to Exercise 3.2.6

- (a) An inductive argument shows $S \Rightarrow_G^* x$ implies $\#_a(x) = \#_b(x)$ for $x \in \{S, a, b\}^*$. Thus, $\mathcal{L}(G) \subseteq L$. For the reverse, we prove $w \in L$ implies $w \in \mathcal{L}(G)$ by induction on $|w|$. Basis, $|w| = 0$: We have $w = \varepsilon$ and $S \Rightarrow_G \varepsilon$. Induction step, $|w| > 0$: Suppose w starts with a . Then we can find strings $u, v \in \{a, b\}^*$ such that $w = aubv$, $\#_a(u) = \#_b(u)$ and $\#_a(v) = \#_b(v)$. Note, $|u|, |v| < |w|$. Thus, by induction hypothesis, $S \Rightarrow_G^* u$, $S \Rightarrow_G^* v$. Therefore, $S \Rightarrow_G^* aSbS \Rightarrow_G^* aubS \Rightarrow_G^* aubv = w$. If w starts with b a similar argument applies.

- (b) The CFG G' given by $S \rightarrow \varepsilon \mid SS \mid aSb \mid bSa$ also generates L .

Answers to exercises of Section 3.3

Answer to Exercise 3.3.1 The generating symbols of G are S, A, B, E of which E is not reachable. The production rules that remain are

$$S \rightarrow A \quad A \rightarrow BB \quad B \rightarrow A \mid \varepsilon$$

Answer to Exercise 3.3.2 Elimination of unit rules gives the grammar

$$S \rightarrow BC \mid a \quad A \rightarrow BC \mid a \quad B \rightarrow \varepsilon \mid b \quad C \rightarrow \varepsilon \mid c$$

Expanding RHS to eliminate ε -productions for non-start symbols gives

$$S \rightarrow BC \mid B \mid C \mid a \mid \varepsilon \quad A \rightarrow BC \mid B \mid C \quad B \rightarrow b \quad C \rightarrow c$$

Answer to Exercise 3.3.3 Elimination of unit rules and ε -productions gives the grammar

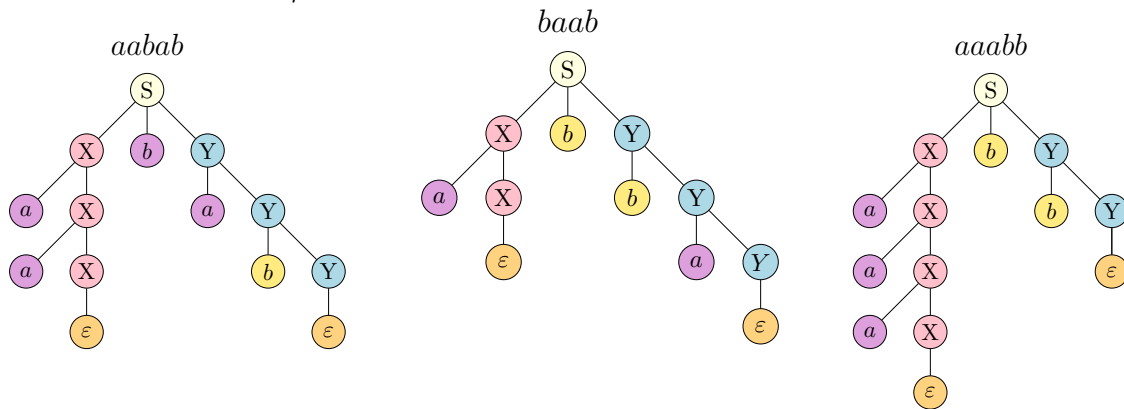
$$S \rightarrow AAA \mid AA \mid aA \mid a \mid \varepsilon \quad A \rightarrow aA \mid a$$

Using auxilliary variables we obtain the CFG in CNF below.

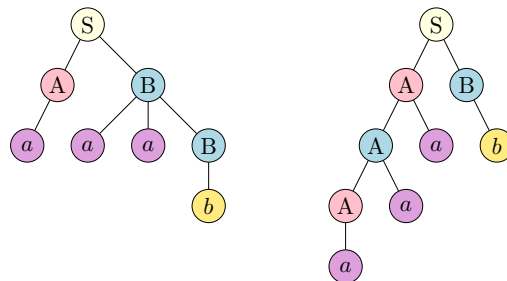
$$S \rightarrow AX_1 \mid AA \mid X_aA \mid a \quad X_1 \rightarrow AA \quad X_a \rightarrow a \quad A \rightarrow X_aA \mid a$$

Answers to exercises of Section 3.4

Answer to Exercise 3.4.1



Answer to Exercise 3.4.2 (a) The following two complete parse trees are different but both have yield *aaab*:



(b) The variable B can only yield the string b . Thus the grammar

$$S \rightarrow Ab \mid aab \quad A \rightarrow a \mid Aa$$

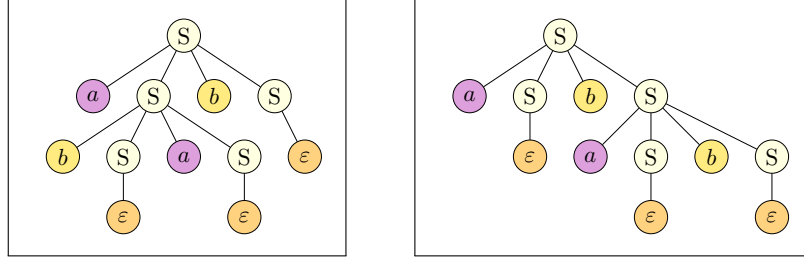
has a grammar that is equivalent to the language $\mathcal{L}(G)$ of G . The string aab can also be produced using the other three production rules, hence we can leave the production $S \rightarrow aab$ out without changing the associated language. We obtain the grammar G' with production rules

$$S \rightarrow Ab \quad A \rightarrow a \mid Aa$$

Each sentential form of G' has at most one variable. Therefore, G' is an unambiguous grammar.

Answer to Exercise 3.4.3

(a) The following two complete parse trees are different but both have yield $abab$:



(b) The following CFG generates strings with an equal number of a 's and b 's. The intuition is that S generates strings with as many a 's as b 's, A generates strings with one more a than b 's, and B generates strings with one more b than a 's.

$$S \rightarrow aB \mid bA \mid \varepsilon \quad A \rightarrow aS \mid bAA \quad B \rightarrow bS \mid aBB$$

When producing left-most a string w from the start symbol S , at any moment exactly one product rule can be applied when w should be yielded. Sentential forms are of the shape wx with $w \in \{a, b\}^*$, $x \in \{S, A, B\}^*$.

Answers to exercises of Section 3.6

Answer to Exercise 3.6.1 Let $G = (V, \Sigma, R, S)$ be a CFG such that $\mathcal{L}(G) = L$. Construct the CFG $G' = (V, \Sigma, R', S)$ by putting $R' = \{A \rightarrow \alpha^R \mid A \rightarrow \alpha \in R\}$. Then one may prove

$$A \Rightarrow_G^* w \iff A \Rightarrow_{G'}^* w^R$$

From this it follows that $\mathcal{L}(G') = L^R$, and that L^R is a context-free language.

We first prove the following claim:

$$A \Rightarrow_G^n w \text{ implies } A \Rightarrow_{G'}^* w^R \quad (3.11)$$

for $A \in V$, $w \in \Sigma^*$, by induction on n . Basis, $n = 0$: Trivial. Induction step, $n+1$: If $A \Rightarrow_G^{n+1} w$, we can find $k \geq 0$, $X_1, \dots, X_k \in V \cup \Sigma$, $w_1, \dots, w_k \in \Sigma^*$, and $n_1, \dots, n_k \geq$

0 such that $A \rightarrow_G X_1 \cdots X_k$, $X_i \Rightarrow_G^{n_i} w_i$ for $1 \leq i \leq k$, $n_1 + \cdots + n_k = n$, and $w_1 \cdots w_k = w$. We have $A \rightarrow_{G'} X_k \cdots X_1$ by construction of G' , and $X_i \Rightarrow_{G'}^* w_i$ for $1 \leq i \leq k$, by induction hypothesis. It follows that $A \Rightarrow_{G'}^* w_k^R \cdots w_1^R$. Since $w^R = (w_1 \cdots w_k)^R = w_k^R \cdots w_1^R$, we have $A \Rightarrow_{G'}^* w^R$. This proves half of the claim (3.11). The other implication is similar.

From Equation (3.11) we derive $S \Rightarrow_G^* w$ iff $S \Rightarrow_{G'}^* w^R$. Therefore $\mathcal{L}(G') = \mathcal{L}(G)^R$. Hence, $L^R = \mathcal{L}(G')$ and L^R is a context-free language.

Answer to Exercise 3.6.2 Consider $L_1 = \{ a^n b^n c^m \mid n, m \geq 0 \}$ and $L_2 = \{ a^n b^m c^\ell \mid n, m, \ell \geq 0, m \neq \ell \}$.

Answer to Exercise 3.6.3 Let $m > 0$ be arbitrary. Consider the string $w = a^{2^m} \in L$. Pick strings $u, v, x, y, z \in \Sigma^*$ such that $w = uvxyz$, $|vxy| \leq m$, and $vy \neq \varepsilon$. We have $vy = a^\ell$ for some ℓ , $1 \leq \ell \leq m$. Thus $2^{m^2} = |w| < |uv^2xy^2z| = 2^m + \ell \leq 2^m + m < 2^{m+1}$. Hence, $uv^2xy^2z \notin L$. It follows by the Pumping Lemma for context-free languages that L is not context-free.

Answer to Exercise 3.6.4 Let $m > 0$ be arbitrary. Consider the string $w = ba^m bba^m bba^m b \in L$. Pick strings $u, v, x, y, z \in \Sigma^*$ such that $w = uvxyz$, $|vxy| \leq m$, and $vy \neq \varepsilon$. We have $vy = a^\ell$ for some ℓ , $1 \leq \ell \leq m$. Otherwise would pumping of v and y change the number of b 's unbalancedly. But if vy consists of a 's only, the number of a 's in one of the blocks can be made unbalanced. It follows by the Pumping Lemma for context-free languages that L is not context-free.

Answer to Exercise 3.6.5 Let $m > 0$ be arbitrary. Consider the string $w = 0^m 10^{2^m} 10^{3^m} \in L$. Pick strings $u, v, x, y, z \in \Sigma^*$ such that $w = uvxyz$, $|vxy| \leq m$, and $vy \neq \varepsilon$. We have $vy = 0^\ell$ for some ℓ , $1 \leq \ell \leq m$. Otherwise would pumping of v and y change the number of 1's. But if vy consists of 0's only, it is a substring of the prefix 0^m , of the string $10^{2^m} 1$, or of the suffix 0^{3^m} . In all these cases, pumping of v and y disturbs the required pattern for being a member of L . It follows by the Pumping Lemma for context-free languages that L is not context-free.

Answer to Exercise 3.6.6 Let $m > 0$ be arbitrary. Consider the string $w = a^m b^m c^m \in L$. Pick strings $u, v, x, y, z \in \Sigma^*$ such that $w = uvxyz$, $|vxy| \leq m$, and $vy \neq \varepsilon$. If vy contains no c 's, then uxz does not meet the requirement of L . The number of a 's or the number of b 's, or both, is strictly less than the number of c 's. If vy contains a c 's, then vxy is a substring of $b^m c^m$. Thus pumping of v and y gives a string strictly less a 's than c 's, also failing the requirement of L . It follows by the Pumping Lemma for context-free languages that L is not context-free.