CHAPTER 1

Parsing Deterministic Context-Free Languages

We assume that the reader is familiar with the basic notions of context-free languages and grammars which can be found in classical books such as [**4, 5**]. In particular, we assume that the reader knows the notions of *folding* and *unfolding* of context-free productions we have given in Definition **??** on page ??. For the reader's convenience, we only recall here the following definitions.

Definition 1.0.1. [**Nullable Symbol**] Given a context-free grammar G, we say that a nonterminal symbol A is *nullable* iff $A \rightarrow^*_G \varepsilon$.

Definition 1.0.2. [**Production for a Nonterminal Symbol**] A production of the form $A \rightarrow \beta$, with $A \in V_N$ and $\beta \in (V_T \cup V_N)^*$, is said to be *a production for* (or *of*) *the nonterminal symbol* A.

Unless otherwise specified, V_T, V_N, and S denote respectively, the set of terminal symbols, the set of nonterminal symbols, and the start symbol of the grammars we will consider.

1.1. LL(k) Parsers

The $LL(k)$ *parsers* are algorithms for parsing the languages which are generated by the $LL(k)$ grammars (see Definition 1.1.1). Unless otherwise specified, in this section we assume that V_T, V_N, and S denote, respectively, the set of the terminal symbols, the set of nonterminal symbols, and the start symbol of the $LL(k)$ grammars we consider. In those grammars we assume that:

(i) no useless symbols occur in the productions,
(ii) ε-productions may be present for the start symbol or other symbols of V_N,
(iii) the symbol \$ does not belong to $V_T \cup V_N$, and
(iv) the input string to be parsed is terminated by \$.

Let V denote the set $V_T \cup V_N$.

We begin by giving the formal definition of an $LL(k)$ grammar, for any $k \geq 0$. As indicated on page **??**, given a string w and a natural number $k \geq 0$, we stipulate that:

$$\underline{w}_k = \begin{cases} w & \text{if } |w| \leq k \\ u & \text{if } w = uv \text{ and } |u| = k. \end{cases}$$

Definition 1.1.1. [$LL(k)$ **grammar**] A context-free grammar (possibly with ε-productions) is said to be $LL(k)$ if for any $x, y, z \in V_T^*$ and for any $\alpha, \beta, \gamma \in (V_N \cup V_T)^*$, we have that:

if (1) $S \rightarrow^{*}_{lm} xA\alpha \rightarrow_{lm} x\beta\alpha \rightarrow^{*}_{lm} xy$ *and*

(2) $S \rightarrow^{*}_{lm} xA\alpha \rightarrow_{lm} x\gamma\alpha \rightarrow^{*}_{lm} xz$ *and*

(3) $\underline{y}_k = \underline{z}_k$

then $\beta = \gamma$.

Note that (1) and (2) are two leftmost derivations of two (possibly different) words xy and xz.

Definition 1.1.2. [$LL(k)$ **language**] An $LL(k)$ language is a language such that there exists an $LL(k)$ grammar which generates it.

In what follows we will also need the following definition of a strong $LL(k)$ grammar.

Definition 1.1.3. [**Strong** $LL(k)$ **grammar**] A context-free grammar (possibly with ε-productions) is said to be *strong* $LL(k)$ if for any $x_1, x_2, y, z \in V_T^*$ and for any $\alpha_1, \alpha_2, \beta, \gamma \in (V_N \cup V_T)^*$ we have that:

if (1) $S \rightarrow^{*}_{lm} x_1A\alpha_1 \rightarrow_{lm} x_1\beta\alpha_1 \rightarrow^{*}_{lm} x_1y$ *and*

(2) $S \rightarrow^{*}_{lm} x_2A\alpha_2 \rightarrow_{lm} x_2\gamma\alpha_2 \rightarrow^{*}_{lm} x_2z$ *and*

(3) $\underline{y}_k = \underline{z}_k$

then $\beta = \gamma$.

Note that (1) and (2) are two leftmost derivations of two (possibly different) words x_1y and x_2z.

One can show that any $LL(0)$ grammar or strong $LL(0)$ grammar generates a language $L \subseteq V_T^*$ which is either the empty set of words or it consists of one word only.

In view of this fact, unless otherwise specified, when referring to $LL(k)$ grammars or strong $LL(k)$ grammars, we will assume that k is greater than or equal to 1.

An $LL(k)$ *parser*, also called a *k-predictive parsing algorithm*, is a deterministic pushdown automaton (see Figure 1.1.1) which once initialized, performs its moves according to a table T, called a *parsing table*, as we will indicate.

If the string to parse is the empty string ε, the string given in input to the parsing pushdown automaton is \$. The initial configuration of the stack is the string: $S\,\$$ and S is at the top of the stack. Recall that, unless otherwise specified, we assume that the stack is represented with the top symbol on the left.

Definition 1.1.4. [**Words Accepted by** $LL(k)$ **Parsers**] A word w is accepted by an $LL(k)$ parser, for $k \geq 1$, if the parser, starting from the configuration where: (i) the input tape holds the word $w\$$, (ii) the head of the input tape points to the leftmost symbol of w, and (iii) the stack holds $S\$$, being S the top of the stack, eventually reaches the configuration where: (i) the head of the input tape points to \$, and (ii) the top of the stack is \$.

Contrary to what we will assume for $LR(k)$ parsers (see Section 1.4), here for $LL(k)$ parsers we do not consider *augmented grammars*. Those grammars are defined as follows.

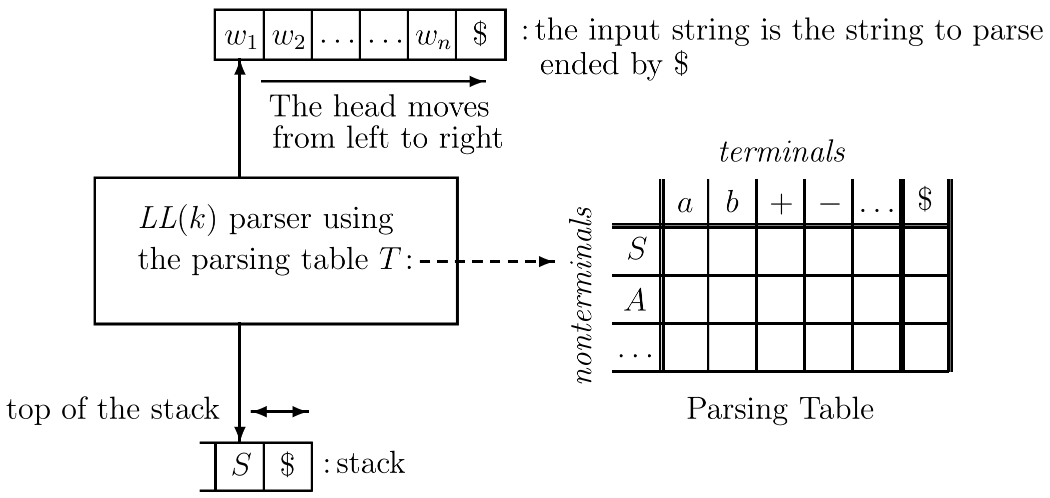
w_1 w_2 . . . . . . w_n \$
: the input string is the string to parse ended by \$
The head moves from left to right
$LL(k)$ parser using the parsing table T:
terminals
a b $+$ $-$ . . . \$
nonterminals
S
A
. . .
Parsing Table
top of the stack
S \$
: stack

FIGURE 1.1.1. A deterministic pushdown automaton for $LL(k)$ parsing, with $k \geq 1$. The input string is $w_1 w_2 \ldots w_n$. Initially, the stack has two symbols only: (i) S on top of the stack, and (ii) \$ at the bottom of the stack. The string to parse is ended by the symbol \$. The table T we have depicted is for an $LL(1)$ parser. For the $LL(k)$ parsers, with $k > 1$, different tables should be used.

DEFINITION 1.1.5. [**Augmented Context-free Grammars**] Given a context-free grammar $G = \langle V_T, V_N, P, S \rangle$ with start symbol S, its *augmented grammar*, call it $G' = \langle V_T \cup \{\$\}, V_N \cup \{S'\}, P', S' \rangle$, is the grammar obtained by considering: (i) the new start symbol S', (ii) a new terminal symbol \$, and (iii) the extra production $S' \rightarrow S\,\$$, that is, $P' = P \cup \{S' \rightarrow S\,\$\}$. In some cases we will assume that the extra production is $S' \rightarrow S$, instead of $S' \rightarrow S\,\$$, and we will tell the reader when we make that assumption.

Given a context-free grammar G we have that the language $L(G')$ generated by the augmented grammar G' is $L(G)\,\$$, that is, a word w is generated by the grammar G iff the word $w\,\$$ is generated by the augmented grammar G'. If the extra production is $S' \rightarrow S$, instead of $S' \rightarrow S\,\$$, we have that $L(G') = L(G)$, instead of $L(G') = L(G)\,\$$.

Note also that when presenting $LL(k)$ parsers, other textbooks use different conventions. For instance, some authors do *not* use the symbol \$ at the bottom of the stack and do not place \$ at the right end of the input string.

1.2. LL(1) Parsers

In this section we will study the $LL(1)$ parsers and the $LL(1)$ grammars. In these parsers the parsing automaton we have depicted in Figure 1.1.1, makes its moves according to a table T which is a matrix whose rows are labelled by the nonterminals (that is, the elements of V_N), and whose columns are labelled by the terminals (that is, the elements of V_T). In the table T there is also one extra column which is labelled by the symbol \$.

A move of the parsing automaton is either a chop move or an expand move. These moves are specified as follows (see also Figure 1.2.1).

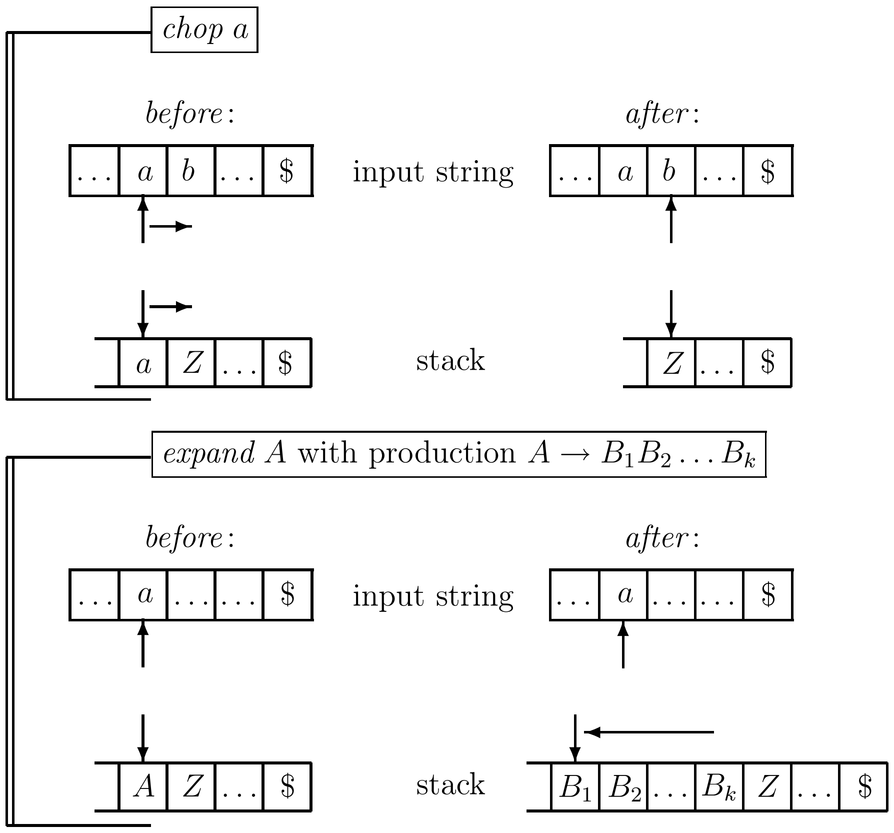

FIGURE 1.2.1. The *chop* and *expand* moves of an $LL(1)$ parser. a and b are symbols in V_T and Z is a symbol in $V_T \cup V_N \cup \{\$\}$.

chop move:

if the input head is pointing at a terminal symbol, say a, and the same symbol a is at the top of the stack, then the input head is moved one cell to the right and the stack is popped;

expand move:

if the input head is pointing at a terminal symbol, say a, and the top of the stack is a nonterminal symbol, say A, then the stack is popped and a new string $\alpha_1\alpha_2\ldots\alpha_n$, with $\alpha_i \in V$ for $i = 1, \ldots, n$, is pushed onto the stack if the production $A \to \alpha_1\alpha_2\ldots\alpha_n$ is at the entry (A, a) of the *parsing table* T (thus, after this move the new top symbol of the stack will be α_1).

In order to explain how to construct the parsing table T of a parsing automaton for the language generated by an $LL(1)$ grammar, we need the following notions of First_1 and Follow_1 (see, for instance, [**1**]).

We first introduce the notion of $\mathit{First}_1(x)$. For every $x \in V$, $\mathit{First}_1(x)$ is the set of the *terminal* symbols in V_T beginning any string β such that $x \to^* \beta$, with the stipulation that $\mathit{First}_1(x)$ should have the extra element ε iff either x is ε or x is a nullable nonterminal symbol (see Definition 1.0.1 on page 5). We can compute the value of $\mathit{First}_1(x)$ as indicated by the following definition. Note that for every $x \in V$, the definition of $\mathit{First}_1(x)$ depends, in general, on the definition of $\mathit{First}_1(y)$ for all $y \in V$.

DEFINITION 1.2.1. **[The Set $\mathit{First}_1(x)$ for Symbols]** Let us consider a context-free grammar, possibly with ε-productions. First_1 is a function from $V \cup \{\varepsilon\}$ to the set of all subsets of $V_T \cup \{\varepsilon\}$. For every $x \in V \cup \{\varepsilon\}$, $\mathit{First}_1(x)$ is the smallest set defined as follows:

(i) $\mathit{First}_1(\varepsilon) = \{\varepsilon\}$.

(ii) For $a \in V_T$, $\mathit{First}_1(a) = \{a\}$.

(iii) For $A \in V_N, \mathit{First}_1(A)$ is initialized to $\emptyset$ and modified as we now indicate.
For each production $A \rightarrow B_1 B_2 \ldots B_{k-1} B_k$,
where: (1) $k \geq 0$, and (2) for $i=1,\ldots,k$, $B_i \in V_T \cup V_N$,

		add the elements of $\mathit{First}_1(B_1) - \{\varepsilon\}$ and
if $B_1 \rightarrow^* \varepsilon$	then	add the elements of $\mathit{First}_1(B_2) - \{\varepsilon\}$ and
if $B_1 B_2 \rightarrow^* \varepsilon$	then	add the elements of $\mathit{First}_1(B_3) - \{\varepsilon\}$ and
...		
if $B_1 B_2 \ldots B_{k-1} \rightarrow^* \varepsilon$	then	add the elements of $\mathit{First}_1(B_k) - \{\varepsilon\}$ and
if $B_1 B_2 \ldots B_{k-1} B_k \rightarrow^* \varepsilon$	then	add ε.

Note that if $B_1 B_2 \ldots B_k \rightarrow^* \varepsilon$ then $A \rightarrow^* \varepsilon$ and $\varepsilon \in \mathit{First}_1(A)$. In particular, if $A \rightarrow \varepsilon$ then $\varepsilon \in \mathit{First}_1(A)$.

By definition, we have that for every $x \in V \cup \{\varepsilon\}$, $\mathit{First}_1(x)$ is a subset of $V_T \cup \{\varepsilon\}$. In Definition 1.2.1 we have required that $\mathit{First}_1(x)$ should be the *smallest* set which satisfies Conditions (i)–(iii). This is due to the fact that $\mathit{First}_1(x)$ may depend on itself and this happens, for instance, when in the given grammar there is a production of the form $A \rightarrow AB$.

We can extend to strings the above definition of the function First_1 by defining a new function, also called First_1, from V^* to the set of all subsets of $V_T \cup \{\varepsilon\}$ as follows.

DEFINITION 1.2.2. **[The Set $\mathit{First}_1(\alpha)$ for Strings]** Let us consider a context-free grammar, possibly with ε-productions. First_1 is a function from V^* to the set of all subsets of $V_T \cup \{\varepsilon\}$, such that for all $x \in V$ and $\alpha \in V^*$,

(i) $\mathit{First}_1(\varepsilon) = \{\varepsilon\}$

(ii) $\mathit{First}_1(x\alpha) = \mathit{if}\ x \rightarrow^* \varepsilon\ \mathit{then}\ (\mathit{First}_1(x) - \{\varepsilon\}) \cup \mathit{First}_1(\alpha)$
$\qquad\qquad\qquad \mathit{else}\ \mathit{First}_1(x)$

where $\mathit{First}_1(x)$ is defined as indicated in Definition 1.2.1.

FACT 1.2.3. For any string $\alpha \in V^*$, we have that $\varepsilon \in \mathit{First}_1(\alpha)$ iff $\alpha \rightarrow^* \varepsilon$.

Now we define the set $\mathit{Follow}_1(A)$ for any nonterminal A. By definition, for any nonterminal $A \in V_N$, $\mathit{Follow}_1(A)$ is the smallest subset of $V_T \cup \{\$\}$ which includes every symbol occurring in a sentential form derived from $S\,\$$ immediately to the right of A, that is, for all $\alpha, \beta \in V^*$ and $b \in V_T \cup \{\$\}$,

if $S\,\$ \rightarrow^* \alpha A b \beta$ *then* $b \in \mathit{Follow}_1(A)$.

For every $A \in V_N$, the definition of $\mathit{Follow}_1(A)$ depends, in general, on the definition of $\mathit{Follow}_1(B)$ for all $B \in V_N$, and the definition of $\mathit{First}_1(x)$ for all $x \in V$.

Definition 1.2.4. **[The Set $Follow_1(A)$ for Nonterminals]** Let us consider a context-free grammar with axiom S, possibly with ε-productions. $Follow_1$ is a function from V_N to the set of all subsets of $V_T \cup \{\$\}$ such that for each nonterminal $A \in V_N$, $Follow_1(A)$ is the *smallest* set which satisfies the following conditions:

(i) *if* the nonterminal A is the axiom S *then* $\$ \in Follow_1(A)$;

(ii) *if* there is a production $B \rightarrow \alpha A \beta$ with $\alpha, \beta \in V^*$ and $\beta \rightarrow^* \varepsilon$ *then* $Follow_1(B) \subseteq Follow_1(A)$;

(iii) *if* there is a production $B \rightarrow \alpha A \beta$ with $\alpha \in V^*$ and $\beta \in V^*$, *then* $(First_1(\beta) - \{\varepsilon\}) \subseteq Follow_1(A)$.

We have that for any $A \in V_N$, the empty string ε is *not* an element of $Follow_1(A)$.

Note that Case (ii) holds if there is a production $B \rightarrow \alpha A$ with $\alpha \in V^*$, because in that case β is ε. In order to clarify Condition (ii) of Definition 1.2.4, let us consider the grammar with axiom S and the following productions:

$$S \rightarrow Bb \qquad B \rightarrow A \qquad A \rightarrow Aa \mid b$$

This grammar generates the language ba^*b. We have that $Follow_1(B) = \{b\}$ and $Follow_1(A) = \{a, b\}$ as one can see from the derivation $S \rightarrow Bb \rightarrow Ab \rightarrow Aab \rightarrow bab$.

Note also that, as a particular instance of Case (iii), we have that if there is a production $B \rightarrow \alpha A x \beta$ with $\alpha, \beta \in V^*$, $x \in V$, and $x \not\rightarrow^* \varepsilon$ then $First_1(x) \subseteq Follow_1(A)$. This is a consequence of Fact 1.2.3 on the preceding page.

Case (iii) may hold either when $\beta \rightarrow^* \varepsilon$ or when $\beta \not\rightarrow^* \varepsilon$.

Finally, note that in the $LL(k)$ parsing we do *not* consider augmented grammars (see Definition 1.1.5) and thus, in particular, we consider neither the new start symbol S' nor the extra production $S' \rightarrow S\,\$$.

Condition (i) of Definition 1.2.4 is motivated by the fact that we have stipulated that the string given in input to the parsing automaton, is the string to be parsed followed by the symbol \$. We have that $Follow_1(S) = \{\$\}$ if S does *not* occur on the right hand side of any production.

In the case of an $LL(1)$ parser the table T is constructed as follows.

Algorithm 1.2.5.
Construction of the table T for an $LL(1)$ parser given an $LL(1)$ grammar G.

For each production $A \rightarrow \alpha$ of G with $A \in V_N$ and $\alpha \in (V_T \cup V_N)^*$,

Rule (i): if $a \in First_1(\alpha)$ then we place $A \rightarrow \alpha$ in row A and column a of the table T, and

Rule (ii): if $\varepsilon \in First_1(\alpha)$ then for each $b \in Follow_1(A)$ we place $A \rightarrow \alpha$ in row A and column b of the table T. Note that, in particular, Rule (ii) is applied if $\alpha = \varepsilon$, because in that case $\varepsilon \in First_1(\alpha)$.

Once the parsing table has been constructed we can use the deterministic pushdown automaton of Figure 1.1.1 as an automaton for parsing the string w. Initially, the stack of the automaton has the string $S\,\$$ (with S as top symbol) and the input string is $w\,\$$.

We have the following properties which we state without proof.

If while the parsing automaton is working, it should use an entry without production then the input string *does not belong* to the language generated by the given grammar.

If the parsing automaton is pointing to the symbol \$ in the input string and the top of the stack is \$ (that is, it may make a chop move of \$), then we accept the given string w, that is, w *belongs* to the language generated by the given grammar G.

Note that given any context-free grammar G, we can construct a parsing table T as indicated above and then we can use the pushdown automaton as indicated in Figure 1.1.1 for parsing words. If an entry of the parsing table T turns out to be multiply defined (that is, more than one production has to be placed in the same entry) then the given grammar is not an $LL(1)$ grammar.

We state without proof the following theorem.

THEOREM 1.2.6. [**$LL(1)$ Grammars and $LL(1)$ Languages**] The $LL(1)$ grammars are those which generate languages that can be parsed by a deterministic pushdown automaton as the one of Figure 1.1.1, with the parsing table constructed as indicated by Algorithm 1.2.5.

The following result which we also state without proof, characterizes the $LL(1)$ grammars and the strong $LL(1)$ grammars.

THEOREM 1.2.7. [**$LL(1)$ Grammars and Strong $LL(1)$ Grammars**] (i) A context-free grammar G is $LL(1)$ iff for every pair of distinct productions $A \rightarrow \alpha$ and $A \rightarrow \beta$ with $\alpha \neq \beta$, and for every string $w\,A\,\sigma$ such that $S \rightarrow^*_{lm} w\,A\,\sigma$ with $w \in V_T^*$ and $\sigma \in V^*$, we have that:

$First_1(\alpha\,\sigma) \cap First_1(\beta\,\sigma) \;=\; \emptyset.$

(ii) A context-free grammar G is *strong* $LL(1)$ iff for every pair of distinct productions $A \rightarrow \alpha$ and $A \rightarrow \beta$ with $\alpha \neq \beta$, we have that:

$First_1(\alpha\, Follow_1(A)) \cap First_1(\beta\, Follow_1(A)) \;=\; \emptyset.$

As a consequence of this theorem we have the following fact.

FACT 1.2.8. The notions of an $LL(1)$ grammar and a strong $LL(1)$ grammar coincide.

EXAMPLE 1.2.9. [**An $LL(1)$ Grammar and Its $LL(1)$ Parsing Table**] Let us consider the grammar G whose productions are:

$S \rightarrow aAb \mid b$
$A \rightarrow a \quad\; \mid bSA$

The axiom is S. We have that:

$First_1(aAb) = \{a\},\;\; First_1(b) = \{b\},\;\; First_1(a) = \{a\},\;\; First_1(bSA) = \{b\}.$

The parsing table is:

	a	b	\$
S	$S \rightarrow aAb$	$S \rightarrow b$	
A	$A \rightarrow a$	$A \rightarrow bSA$	

input string:	$\underset{\blacktriangle}{a}\,b\,b\,a\,b\,\$$		$\underset{\blacktriangle}{a}\,b\,b\,a\,b\,\$$		$a\,\underset{\blacktriangle}{b}\,b\,a\,b\,\$$		$a\,\underset{\blacktriangle}{b}\,b\,a\,b\,\$$		$a\,b\,\underset{\blacktriangle}{b}\,a\,b\,\$$		$a\,b\,\underset{\blacktriangle}{b}\,a\,b\,\$$		$a\,b\,b\,\underset{\blacktriangle}{a}\,b\,\$$		$a\,b\,b\,\underset{\blacktriangle}{a}\,b\,\$$		$a\,b\,b\,a\,\underset{\blacktriangle}{b}\,\$$		$a\,b\,b\,a\,b\,\underset{\blacktriangle}{\$}$
	(1)	$\xrightarrow{\text{expand } S}$	(2)	$\xrightarrow{\text{chop } a}$	(3)	$\xrightarrow{\text{expand } A}$	(4)	$\xrightarrow{\text{chop } b}$	(5)	$\xrightarrow{\text{expand } S}$	(6)	$\xrightarrow{\text{chop } b}$	(7)	$\xrightarrow{\text{expand } A}$	(8)	$\xrightarrow{\text{chop } a}$	(9)	$\xrightarrow{\text{chop } b}$	(10)
stack:	$\underset{\blacktriangle}{S}\,\$$		$\underset{\blacktriangle}{a}\,A\,b\,\$$		$\underset{\blacktriangle}{A}\,b\,\$$		$\underset{\blacktriangle}{b}\,S\,A\,b\,\$$		$\underset{\blacktriangle}{S}\,A\,b\,\$$		$\underset{\blacktriangle}{b}\,A\,b\,\$$		$\underset{\blacktriangle}{A}\,b\,\$$		$\underset{\blacktriangle}{a}\,b\,\$$		$\underset{\blacktriangle}{b}\,\$$		$\underset{\blacktriangle}{\$}$

FIGURE 1.2.2. *LL*(1) parsing of the string $a\,b\,b\,a\,b\,\$$. The given grammar has the following productions: $S \rightarrow a\,A\,b \mid b,\ \ A \rightarrow a \mid b\,S\,A$. The black triangle ▲ indicates the symbol at hand in the input string and the top of the stack (which is always the leftmost symbol of the stack).

In Figure 1.2.2 we have depicted the sequence of the input string and stack configurations while parsing the string $a\,b\,b\,a\,b\,\$$. (That sequence from configuration (1) to configuration (10) is divided into three subsequences to be read from left to right.) The black triangle indicates the symbols at hand in the input string and the top of the stack (as usual, in every stack configuration the top of the stack is the leftmost symbol).
Now, in the last configuration of Figure 1.2.2 we have that both the top of the stack and the input head are pointing to the symbol \$. Thus, the given string $a\,b\,b\,a\,b$ belongs to the language generated by the grammar G. □

EXAMPLE 1.2.10. [*LL*(1) **Parsing of Arithmetic Expressions**] Let us consider the context free grammar G with axiom E and the following productions:

$$\begin{array}{ll} E \rightarrow E + T & \mid T \\ T \rightarrow T \times F & \mid F \\ F \rightarrow (E) & \mid a \end{array}$$

	a	(	)	+	$\times$	\$
E	$E \to T\,\widetilde{E}$	$E \to T\,\widetilde{E}$				
$\widetilde{E}$			$\widetilde{E} \to \varepsilon$	$\widetilde{E} \to +T\,\widetilde{E}$		$\widetilde{E} \to \varepsilon$
T	$T \to F\,\widetilde{T}$	$T \to F\,\widetilde{T}$				
$\widetilde{T}$			$\widetilde{T} \to \varepsilon$	$\widetilde{T} \to \varepsilon$	$\widetilde{T} \to \times F\,\widetilde{T}$	$\widetilde{T} \to \varepsilon$
F	$F \to a$	$F \to (E)$				

FIGURE 1.2.3. $LL(1)$ parsing table for the grammar with axiom E and productions: $E \to T\widetilde{E}$, $\widetilde{E} \to \varepsilon \mid +T\widetilde{E}$, $T \to F\widetilde{T}$, $\widetilde{T} \to \varepsilon \mid \times F\widetilde{T}$, $F \to (E) \mid a$.

This grammar is left recursive and thus, it cannot be $LL(k)$ for any $k \geq 0$ (see [**2**, page 344]). If we want to use an $LL(k)$ parsing algorithm we have to look for an equivalent grammar which is *not* left recursive. One such context free grammar is the following grammar $\widetilde{G}$ (with ε-productions):

$$\begin{array}{ll} E \to T\,\widetilde{E} & \widetilde{E} \to \varepsilon \mid +T\,\widetilde{E} \\ T \to F\,\widetilde{T} & \widetilde{T} \to \varepsilon \mid \times F\,\widetilde{T} \\ F \to (E) \mid a & \end{array}$$

The grammar $\widetilde{G}$ is an $LL(1)$ grammar as shown by the $LL(1)$ parsing table shown in Figure 1.2.3. In order to construct that parsing table, we have first to compute the following sets:

$$\begin{array}{ll} \mathit{First}_1(\varepsilon) = \{\varepsilon\} & \\ \mathit{First}_1(T\,\widetilde{E}) = \{(, a\} & \mathit{First}_1(+T\,\widetilde{E}) = \{+\} \\ \mathit{First}_1(F\,\widetilde{T}) = \{(, a\} & \mathit{First}_1(\times F\,\widetilde{T}) = \{\times\} \\ \mathit{First}_1((E)) = \{(\} & \mathit{First}_1(a) = \{a\} \\ \mathit{Follow}_1(E) = \{), \$\} & \mathit{Follow}_1(\widetilde{E}) = \{), \$\} \\ \mathit{Follow}_1(T) = \{+,), \$\} & \mathit{Follow}_1(\widetilde{T}) = \{+,), \$\} \end{array}$$

Figure 1.2.4 on the following page shows the sequence of input string and stack configurations while parsing the arithmetic expression a (thus, the input is the string $a\,\$$) according to the grammar $\widetilde{G}$. □

With reference to Example 1.2.10 note that the following grammar H, which is not left recursive and has no ε-productions, is *not* $LL(1)$:

$$\begin{array}{ll} E \to T \mid T\,\widetilde{E} & \widetilde{E} \to +T \mid +T\,\widetilde{E} \\ T \to F \mid F\,\widetilde{T} & \widetilde{T} \to \times F \mid \times F\,\widetilde{T} \\ F \to (E) \mid a & \end{array}$$

This grammar H can be obtained from the given grammar G by eliminating the left recursion (thus, the grammar H is equivalent to grammar G and $\widetilde{G}$). To show that

input string:	$a\,\$$ ▲		$a\,\$$ ▲		$a\,\$$ ▲		$a\,\$$ ▲
	(1)	$\xrightarrow{\text{expand } E}$	(2)	$\xrightarrow{\text{expand } T}$	(3)	$\xrightarrow{\text{expand } F}$	(4)
stack:	$E\,\$$ ▲		$T\widetilde{E}\,\$$ ▲		$F\widetilde{T}\widetilde{E}\,\$$ ▲		$a\widetilde{T}\widetilde{E}\,\$$ ▲

	$a\,\$$ ▲		$a\,\$$ ▲		$a\,\$$ ▲
$\xrightarrow{\text{chop } a}$	(5)	$\xrightarrow{\text{expand } \widetilde{T}}$	(6)	$\xrightarrow{\text{expand } \widetilde{E}}$	(7)
	$\widetilde{T}\widetilde{E}\,\$$ ▲		$\widetilde{E}\,\$$ ▲		$\$$ ▲

FIGURE 1.2.4. $LL(1)$ parsing of the string $a\,\$$ according to the grammar $\widetilde{G}$ with axiom E and productions: $E \to T\widetilde{E}$, $\widetilde{E} \to \varepsilon \mid +T\widetilde{E}$, $T \to F\widetilde{T}$, $\widetilde{T} \to \varepsilon \mid \times F\widetilde{T}$, $F \to (E) \mid a$. The black triangle ▲ indicates the symbol at hand in the input string and the top of the stack (which is always the leftmost symbol of the stack).

the grammar H is indeed not $LL(1)$, let us consider the following grammar which is a simplified version of H. This grammar has axiom T and the productions:

$$T \to F \mid F\,\widetilde{T} \qquad \widetilde{T} \to \times F \mid \times\, F\,\widetilde{T}$$
$$F \to (T) \mid a$$

We have that:

$$\mathit{First}_1(F) = \mathit{First}_1(F\,\widetilde{T}) = \{(, a\} \qquad \mathit{First}_1(\times F) = \mathit{First}_1(\times F\,\widetilde{T}) = \{\times\}$$
$$\mathit{First}_1((T)) = \{(\} \qquad \mathit{First}_1(a) = \{a\}$$

We get the following parsing table (there is no need to compute the *Follow*$_1$ sets because ε does not occur in any of the *First*$_1$ sets):

	a	(	)	$\times$	$\$$
T	$T \to F$ $T \to F\,\widetilde{T}$	$T \to F$ $T \to F\,\widetilde{T}$			
$\widetilde{T}$				$T \to \times F$ $T \to \times F\,\widetilde{T}$	
F	$F \to a$	$F \to (T)$			

□

EXAMPLE 1.2.11. [**A Grammar Which is Not** $LL(1)$] Let us consider the grammar G whose axiom is S and whose productions are:

$S \to \varepsilon \quad | \; a\,b\,A$
$A \to S\,a\,a \; | \; b$

We have that:

$First_1(\varepsilon) = \{\varepsilon\}$ $\qquad$ $First_1(a\,bA) = \{a\}$
$First_1(S\,a\,a) = \{\varepsilon, a\}$ $\qquad$ $First_1(b) = \{b\}$
$Follow_1(S) = \{\$, a\}$ $\qquad$ $Follow_1(A) = \{\$, a\}$

The parsing table is:

	a	b	$\$$
S	$S \to a\,bA$ $S \to \varepsilon$		$S \to \varepsilon$
A	$A \to S\,a\,a$	$A \to b$	$A \to S\,a\,a$

Since in that parsing table for the symbol S on the top of the stack and the input symbol a, there are two entries, we have that the given grammar is *not* $LL(1)$. □

1.3. LL(k) Parsers (for k ≥ 1)

In this section we consider the general case of $LL(k)$ parsers and $LL(k)$ grammars with $k \geq 1$.

We first need the following definition which is a generalization of Definition 1.2.1.

DEFINITION 1.3.1. [**The Set** $First_k(\alpha)$] Let us consider a context-free grammar, possibly with ε-productions. For $k \geq 1$, $First_k$ is a function from V^* to the set of all subsets of $V_T^0 \cup V_T^1 \cup \ldots \cup V_T^k$. Given a string $\alpha \in V^*$, we have that:

$$First_k(\alpha) =_{def} \{w \mid (\alpha \to^* w\,\beta \text{ and } w \in V_T^k \text{ and } \beta \in V_T^*) \text{ or } (\alpha \to^* w \text{ and } w \in V_T^i \text{ for some } 0 \leq i < k)\}.$$

Note that for any $k \geq 0$, if $\alpha \to^* \varepsilon$ then $\varepsilon \in First_k(\alpha)$.

We will also use the following binary operation on languages.

DEFINITION 1.3.2. [**k-bounded Concatenation**] Given two languages L_1 and L_2, their *k-bounded concatenation* $\odot_k$, for $k \geq 1$, is defined as follows:

$$L_1 \odot_k L_2 = \{w \mid (|w| \leq k \text{ and } w \in L_1 \cdot L_2) \text{ or } (|w| = k \text{ and } \exists z\; wz \in L_1 \cdot L_2)\}.$$

where the concatenation operation $\cdot$ on languages is defined as usual (see Chapter **??**).

For instance, given the languages $L_1 = \{\varepsilon,\, abb\}$ and $L_2 = \{b,\, bab\}$, we have that:

$L_1 \cdot L_2 = \{b,\, bab,\, abbb,\, abbbab\}$ and

$L_1 \odot_2 L_2 = \{b,\, ba,\, ab\}$.

Note that in $L_1 \odot_k L_2$ with $k \geq 1$, there may be words whose length is less than k.

Now we describe the $LL(k)$ parsing process for $k \geq 1$.

Supposed we are given a string v to be parsed and an $LL(k)$ grammar G. Suppose also that we have parsed the proper prefix p of the given string v, that is, $v = p\,u$ for some $u \in V_T^+$ and the input head is pointing to the leftmost symbol of u. We assume that the sentential form we have generated so far is: $p\,A\,z$, for some $z \in V^*$, that is, all symbols in p have been chopped and the stack is holding $A\,z\,\$$ with the top of the stack pointing to A.

The production which has to be applied by the parser for expanding the nonterminal A is uniquely determined by:

(i) A itself,
(ii) the leftmost k symbols of z (or z itself if $|z| < k$), and
(iii) the leftmost k symbols of u (or u itself if $|u| < k$).

In order to determine the production for expanding A, the parser uses a parsing table which is constructed as we now indicate from some other tables.

Let T be one of these other tables. Each of these tables is parameterized by (i) a nonterminal, and (ii) a language subset of $V_T^*\,\$^{0,1}$. A table T with parameters the nonterminal B and the language L, will also be denoted by $T[B, L]$.

A table $T[B, L]$ can be depicted as a matrix whose rows have three components (or columns):

(i) the first one is a word in $V_T^*\,\$^{\,0,1}$,

(ii) the second one is a production for B, and

(iii) the third one is a *list of sets of words* which depends on the production for B which is the second component and also on L. Each set in the list is called a *local follow sets* for B.

The first and the second components (or columns) of $T[B, L]$ are constructed by considering every production for B. For each of this production, say $B \rightarrow \alpha$, we will construct some rows of the table $T[B, L]$ as follows. We first compute the set of words:

$First_k(\alpha) \odot_k L$

and then we construct a row of the table $T[B, L]$ for each word in $First_k(\alpha) \odot_k L$ as follows. Let us consider one of these words, say w. The corresponding row is of the form:

w	$B \rightarrow \alpha$	M

where M is a list of set of words constructed as follows:

(i) *if* $\alpha \in V_T^*$ *then* $M = [\,]$, and

(ii) *if* α is of the form: $x_0\,B_1 x_1 \ldots\ B_m x_m$, with $m \geq 0$, and for $h = 0, \ldots, m$, $x_h \in V_T^*$, and for $i = 1, \ldots, m$, $B_i \in V_N$, *then* $M = [Y_1, \ldots, Y_m]$, where for $i = 1, \ldots, m$,

$Y_i = First_k(x_i\,B_{i+1} x_{i+1} \ldots\ B_m x_m) \odot_k L$.

The first table to be constructed should be $T[S, \{\$\}]$.

When constructing tables, we also maintain a set of tables to be constructed. Thus, initially, that set of tables to be constructed has exactly one element which is the table $T[S, \{\$\}]$.

At the end of the construction of a table, say $T[B, L]$, we update that set of tables as follows. For each row of the table $T[B, L]$ of the form:

w	$B \rightarrow x_0\,B_1 x_1 \ldots\ B_m x_m$	$[Y_1, \ldots, Y_m]$

where: (i) $w \in V_T^*$, (ii) $m \geq 1$, (iii) for $h = 0, \ldots, m$, $x_h \in V_T^*$, and (iv) for $i = 1, \ldots, m$, $B_i \in V_N$, we add to that set of tables the m tables $T[B_i, Y_i]$, for $i = 1, \ldots, m$.

The process of constructing tables terminates when we have constructed all the tables which occur in the set of tables to be constructed.

At that point we can construct the parsing table for the $LL(k)$ parsing for $k \geq 1$. In the parsing table the rows are indexed by the parameters of the tables we have constructed (thus, the rows are as many as those tables), and the columns are indexed by the words in $V^0\,\$ \cup V^1\,\$ \cup \ldots \cup V^{k-1}\,\$ \cup V^k$ (of course, $V^0\,\$$ is equal to $\{\$\}$). The entry of the parsing table in row $[B_i, Y_i]$ and in column w is the second component of row with index w of table $T[B_i, Y_i]$.

Thus,
(i) the index $[B_i, Y_i]$ of each row of the parsing table is the symbol B_i of the top of the stack together with the string Y_i of the symbols below the top, and
(ii) the index w of each column of the parsing table is the string of symbols to the right of the input head (which is pointing to the leftmost symbol of w).

Now we will give an example of the construction of the tables in the case of the $LL(2)$ parsing for a particular $LL(2)$ grammar G. This example will clarify the general rules we have given above for the case of the $LL(k)$ parsing, for any $k \geq 1$.

Example 1.3.3. Let us consider the following $LL(2)$ grammar G with productions:

$$\begin{array}{ll} S \rightarrow aAaa & \mid bAba \\ A \rightarrow b & \mid \varepsilon \end{array}$$

Construction of the initial table $T[S, \{\$\}]$.
For S we have the two productions: $S \rightarrow aAaa$ and $S \rightarrow bAba$. For each of them, say $S \rightarrow \alpha$, we have compute the set of words:

$First_2(\alpha) \odot_2 \{\$\}$.

Thus, we have:

$First_2(aAaa) \odot_2 \{\$\} = \{ab, aa\} \odot_2 \{\$\} = \{ab, aa\}$
$First_2(bAba) \odot_2 \{\$\} = \{bb\} \odot_2 \{\$\} = \{bb\}$

From these values we get the three rows of the table $T[S, \{\$\}]$. We have that:

the first row is	aa,	$S \rightarrow aAaa$,	$L1$,
the second row is	aa,	$S \rightarrow aAaa$,	$L2$,
the third row is	bb,	$S \rightarrow bAba$,	$L3$,

where the lists of set of words $L1$, $L2$, and $L3$ are defined as follows:

$L1 = [First_2(aa) \odot_2 \{\$\}] = [\{aa\}]$

(Note that aa is the word in V_T^* which follows A in the right hand side of the production $S \rightarrow aAaa$, and there is only one nonterminal symbol in $aAaa$).

$L2 = [First_2(aa) \odot_2 \{\$\}] = [\{aa\}]$
$L3 = [First_2(ba) \odot_2 \{\$\}] = [\{ba\}]$

(Note that ba is the word in V_T^* which follows A in the right hand side of the production $S \rightarrow bAba$.)

Thus, we get the following table $T[S, \{\$\}]$:

$T[S, \{\$\}]$:

aa	$S \rightarrow aAaa$	$[\{aa\}]$
ab	$S \rightarrow aAaa$	$[\{aa\}]$
bb	$S \rightarrow bAba$	$[\{ba\}]$

Having constructed this table, we insert in the set of tables to be constructed the following two tables:

$T[A, \{aa\}]$ and $T[A, \{ba\}]$.

Note that either the first row or the second row of table $T[S, \{\$\}]$ forces us to insert table $T[A, \{aa\}]$ in the set of tables to be constructed.

Construction of the table $T[A, \{aa\}]$.
For A we have the two productions: $A \rightarrow b$ and $A \rightarrow \varepsilon$. Thus, we have that:

$\mathit{First}_2(b) \odot_2 \{aa\} = \{ba\}$
$\mathit{First}_2(\varepsilon) \odot_2 \{aa\} = \{aa\}$

From these values we get the following two rows:

the first row is: $ba,\quad A \rightarrow b,\quad M1,$
the second row is: $aa,\quad A \rightarrow \varepsilon,\quad M2.$

where the lists $M1$ and $M2$ of sets of words are both empty because in the right hand sides of the productions $A \rightarrow b$ and $A \rightarrow \varepsilon$ there are no nonterminal symbols. Thus, we get the following table $T[A, \{aa\}]$:

$T[A, \{aa\}]$:

ba	$A \rightarrow b$	$[\,]$
aa	$A \rightarrow \varepsilon$	$[\,]$

After the construction of this table we do *not* add any new table to the set of tables to be constructed because in the right hand sides of the productions $A \rightarrow b$ and $A \rightarrow \varepsilon$ there are no nonterminal symbols.

Construction of the table $T[A, \{ba\}]$.
For A we have the two productions: $A \rightarrow b$ and $A \rightarrow \varepsilon$. Thus, we have that:

$\mathit{First}_2(b) \odot_2 \{ba\} = \{bb\}$
$\mathit{First}_2(\varepsilon) \odot_2 \{ba\} = \{ba\}$

From these values we get the following two rows:

the first row is: $bb,\quad A \rightarrow b,\quad N1,$
the second row is: $ba,\quad A \rightarrow \varepsilon,\quad N2.$

where the lists $N1$ and $N2$ of sets of words are both empty because in the right hand sides of the productions $A \rightarrow b$ and $A \rightarrow \varepsilon$ there are no nonterminal symbols. Thus, we get the following table $T[A, \{ba\}]$:

$T[A, \{ba\}]$:

bb	$A \rightarrow b$	$[\,]$
ba	$A \rightarrow \varepsilon$	$[\,]$

	$a\,a$	$a\,b$	$b\,a$	$b\,b$	$a\,\$$	$b\,\$$	$\$$
$[S,\ \{\$\}]$	$S \rightarrow aAaa$	$S \rightarrow aAaa$		$S \rightarrow bAba$			
$[A,\ \{aa\}]$	$A \rightarrow \varepsilon$		$A \rightarrow b$				
$[A,\ \{ba\}]$			$A \rightarrow \varepsilon$	$A \rightarrow b$			

FIGURE 1.3.1. The $LL(2)$ parsing table for the grammar with axiom S and productions: $S \rightarrow aAaa \mid bAba$ and $A \rightarrow b \mid \varepsilon$.

After the construction of this table we do *not* add any new table to the set of tables to be constructed because in the right hand sides of the productions $A \rightarrow b$ and $A \rightarrow \varepsilon$ there are no nonterminal symbols.

Construction of the parsing table.
Having constructed the tables $T[S, \{\$\}]$, $T[A, \{aa\}]$, and $T[A, \{ba\}]$, we can construct the parsing table depicted in Figure 1.3.1. That table is used for the $LL(2)$ parsing of the given grammar.
In Figure 1.3.2 below we have depicted the sequence of the input string and stack configurations when parsing the string $b\,b\,a\,\$$. The symbol ▲ indicates the positions of the input head and the top of the stack (as usual, in every stack configuration the top of the stack is the leftmost symbol).

Note that the expansion of S (see Figure 1.3.2) is done using the production $S \rightarrow bAba$ because: (i) below S on the stack there is \$, and (ii) the symbols in the position of the input head and at its right are $b\,b$ (see the entry at row $[S, \{\$\}]$ and column $b\,b$ of the parsing table depicted in Figure 1.3.1). The expansion of A is done using the production $A \rightarrow \varepsilon$ because: (i) below A on the stack there are the symbols $b\,a$, and (ii) the symbols in the position of the input head and at its right are $b\,a$ (see the entry at row $[A, \{ba\}]$ and column $b\,a$ of the parsing table depicted in Figure 1.3.1).

In the final configuration we have that both the top of the stack and the input head are pointing to the symbol \$. Thus, the given string $b\,b\,a$ belongs to the language generated by the grammar G. □

Now we define for any nonterminal A and for any $k \geq 1$, the set $Follow_k(A)$.

DEFINITION 1.3.4. **[The Set $Follow_k(A)$]** Let us consider a context-free grammar, possibly with ε-productions. For any $k \geq 1$, $Follow_k$ is a function from V_N to the set of all subsets of $V^*\$^{0,1}$. For any $A \in V_N$, $Follow_k(A)$ is the *smallest* set such that for any string $w \in (V^0\$ \cup V^1\$ \cup V^2\$ \cup \ldots \cup V^{k-1}\$ \cup V^k)$,

if $S\,\$ \rightarrow^* \alpha A \beta$ for some $\alpha \in V^*$ and $\beta \in V^*\$$ and $w \in First_k(\beta)$
then $w \in Follow_k(A)$.

Thus, by definition, for every nonterminal $A \in V_N$, $Follow_k(A)$ is a set of words w in $V^0\$ \cup V^1\$ \cup V^2\$ \cup \ldots \cup V^{k-1}\$ \cup V^k$ which occur in a sentential form derived from $S\,\$$ immediately to the right of A.

We have that for any $k \geq 1$ and $A \in V_N$, the empty string ε is *not* an element of $Follow_k(A)$.

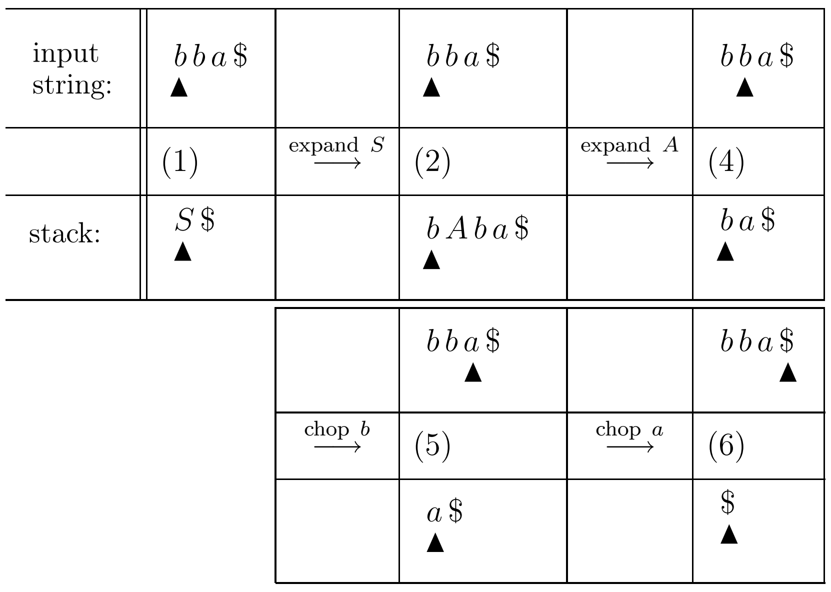

FIGURE 1.3.2. $LL(2)$ parsing of the string $b\,b\,a\,\$$. The given grammar has the following productions: $S \rightarrow aAaa \mid bAba$, $A \rightarrow b \mid \varepsilon$. The black triangle indicates the symbol at hand in the input string and the top of the stack (which is always the leftmost symbol).

Note that contrary to Definition 1.2.4 where we have given an algorithm for the construction of the set $\mathit{Follow}_1(A)$, in the above Definition 1.3.4 we do *not* provide an algorithm for constructing the set $\mathit{Follow}_k(A)$ starting from the productions of the given $LL(k)$ grammar, for any $k \geq 1$.

The following result which we state without proof, characterizes the $LL(k)$ grammars and the strong $LL(k)$ grammars for any $k \geq 1$ [**2**, page 342].

THEOREM 1.3.5. [$LL(k)$ **Grammars and Strong** $LL(k)$ **Grammars**] (i) A context-free grammar G is $LL(k)$ iff for every pair of distinct productions $A \rightarrow \alpha$ and $A \rightarrow \beta$ with $\alpha \neq \beta$, and for every string $w\,A\,\sigma$ such that $S \rightarrow^*_{lm} w\,A\,\sigma$, we have that:

$\mathit{First}_k(\alpha\,\sigma) \cap \mathit{First}_k(\beta\,\sigma) \;=\; \emptyset$.

(ii) A context-free grammar G is *strong* $LL(k)$ iff for every pair of distinct productions $A \rightarrow \alpha$ and $A \rightarrow \beta$ with $\alpha \neq \beta$, we have that:

$\mathit{First}_k(\alpha\,\mathit{Follow}_k(A)) \cap \mathit{First}_k(\beta\,\mathit{Follow}_k(A)) \;=\; \emptyset$.

EXAMPLE 1.3.6. The grammar G with axiom S and the following productions:

$S \rightarrow aAaa \mid bAba$
$A \rightarrow b \mid \varepsilon$

is an $LL(2)$ grammar as the above Example 1.3.3 shows, but it is not a strong $LL(2)$ grammar. Indeed, we have that:

$\mathit{Follow}_2(A)) \;=\; \{aa,\, ba\}$ and

$\mathit{First}_2(b\,\mathit{Follow}_2(A)) \cap \mathit{First}_2(\varepsilon\,\mathit{Follow}_2(A)) \;=\; \{ba,\, bb\} \cap \{aa,\, ba\} \;=\; \{ba\}$.

We leave it to the reader to check the following facts which we state without proofs.

FACT 1.3.7. The rules we have given above for constructing the parsing table for $LL(1)$ parsing, can be obtained by instantiating with $k = 1$ the rules for $LL(k)$ parsing for $k \geq 1$.

For any $k \geq 0$, if a language L has an $LL(k)$ grammar then L has a strong $LL(k)$ grammar.

FACT 1.3.8. Let L be a language that can be generated by an $LL(k)$ grammar for some $k \geq 0$. Then: (i) $L-\{\varepsilon\}$ can be generated by an $LL(k+1)$ grammar in Greibach normal form, and (ii) L can be generated by an $LL(k+1)$ grammar G in Greibach normal form and we have to consider also the production $S \rightarrow \varepsilon$ iff $\varepsilon \in L$ [**2**, page 362].

In the following two examples the reader may see in action the techniques for producing from an $LL(1)$ grammar an equivalent $LL(2)$ grammar in Greibach normal form. These techniques constitute the basis for the proof of the above Fact 1.3.8.

EXAMPLE 1.3.9. [**From $LL(1)$ Grammars to $LL(2)$ Grammars in Greibach Normal Form**] Let us consider the $LL(1)$ grammar G with axiom S and the following productions:

$S \rightarrow AB$
$A \rightarrow aA \mid \varepsilon$
$B \rightarrow bA \mid \varepsilon$

We want to derive an equivalent grammar in Greibach normal form. In that equivalent grammar we also allow the production $S \rightarrow \varepsilon$ if $\varepsilon \in L(G)$. In our case, indeed, we have the production $S \rightarrow \varepsilon$ because $\varepsilon \in L(G)$.

We start from the axiom S and we perform *leftmost unfolding* steps (that is, we unfold the leftmost nonterminals), thereby producing new sentential forms from old ones. We stop these unfolding steps when we get either (i) ε, or (ii) a terminal symbol, or (iii) a terminal followed by a non-nullable symbol. By doing so, we generate a tree of sentential forms in which every leaf corresponds to a production of the desired grammar in Greibach normal form (see Figure 1.3.3).

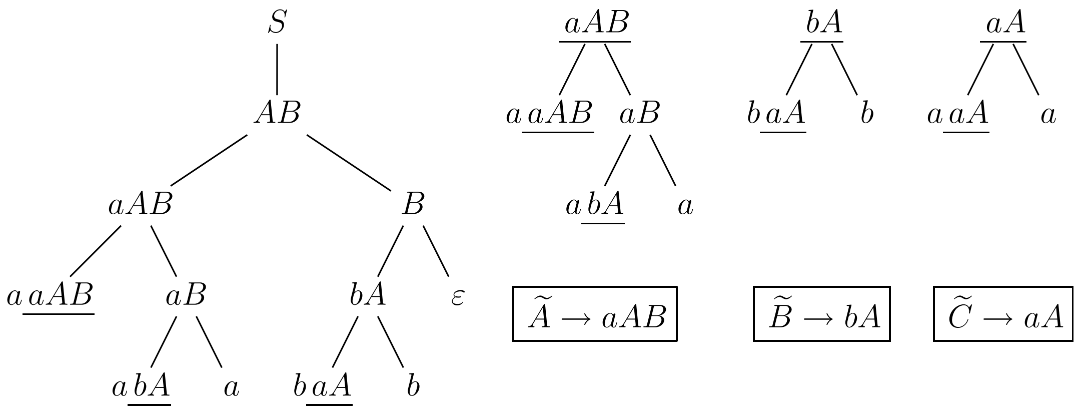

FIGURE 1.3.3. Trees of sentential forms obtained by unfolding the leftmost nonterminals. A and B are nullable nonterminal symbols.

In each leaf the sentential form without its leftmost terminal is used as the root a new tree of sentential forms. The construction of new trees of sentential forms terminates when all leaves of all trees of sentential forms are either: (i) ε, or (ii) a terminal symbol, or (iii) a terminal symbol, say a, followed by a sequence, say σ, of symbols labeling the root of a tree. In Case (iii), if we introduce a new nonterminal, say A, and the production $A \rightarrow \sigma$, and if we perform a folding step at the leaf of a tree, we get at that leaf the terminal a followed by the nonterminal A.

In the case of our grammar above we introduce the productions: (i) $\widetilde{A} \rightarrow aAB$, (ii) $\widetilde{B} \rightarrow bA$, and (iii) $\widetilde{C} \rightarrow aA$, and we perform the folding steps by using these productions, we get from the root-to-leaves paths of the trees of Figure 1.3.3 the following productions in Greibach normal form:

$$\begin{array}{l}
S \rightarrow a\widetilde{A} \mid a\widetilde{B} \mid a \mid b\widetilde{C} \mid b \mid \varepsilon \\
\widetilde{A} \rightarrow a\widetilde{A} \mid a\widetilde{B} \mid a \\
\widetilde{B} \rightarrow b\widetilde{C} \mid b \\
\widetilde{C} \rightarrow a\widetilde{C} \mid a
\end{array}$$

Note that no unit productions are generated. We leave it to the reader to check that these productions belong to an $LL(2)$ grammar (besides the production $S \rightarrow \varepsilon$). □

EXAMPLE 1.3.10. **[From $LL(1)$ Grammars to $LL(2)$ Grammars in Greibach Normal Form]** Let us consider the $LL(1)$ grammar G_α with axiom T and the following productions:

$$\begin{array}{lll}
T \rightarrow F\,T' & & \\
T' \rightarrow \varepsilon & \mid \times F\,T' & \qquad (G_\alpha) \\
F \rightarrow (\,T\,) & \mid a &
\end{array}$$

Note that grammar G_α is an $LL(1)$ grammar which can be derived from the following left recursive grammar G_β with productions:

$$\begin{array}{ll}
T \rightarrow F & \mid T \times F \\
F \rightarrow (\,T\,) & \mid a
\end{array} \qquad (G_\beta)$$

Thus, grammar G_β is *not* an $LL(1)$ grammar. Starting from the grammar (G_α), by performing some unfolding steps by using the productions of the grammar (G_α) according to the rules described in the above Example 1.3.9, we get the trees of sentential forms which we have depicted in Figure 1.3.4.

If we introduce the productions: (i) $R \rightarrow)\,T'$, and (ii) $B \rightarrow \times F\,T'$, and then we perform the folding steps by using these productions, we get from the root-to-leaves paths of the trees of Figure 1.3.4, the grammar G_γ which has the following productions in Greibach normal form:

$$\begin{array}{llll}
T \rightarrow (\,T\,R & \mid a\,B & \mid a & \\
R \rightarrow \,) & \mid)\,B & & \qquad (G_\gamma) \\
B \rightarrow \times(\,T\,R & \mid \times\, a\,T & &
\end{array}$$

Note that no unit production is generated. There is no production $T \rightarrow \varepsilon$ because T is not nullable. We leave it to the reader to check that the grammar (G_γ) is an $LL(2)$ grammar. □

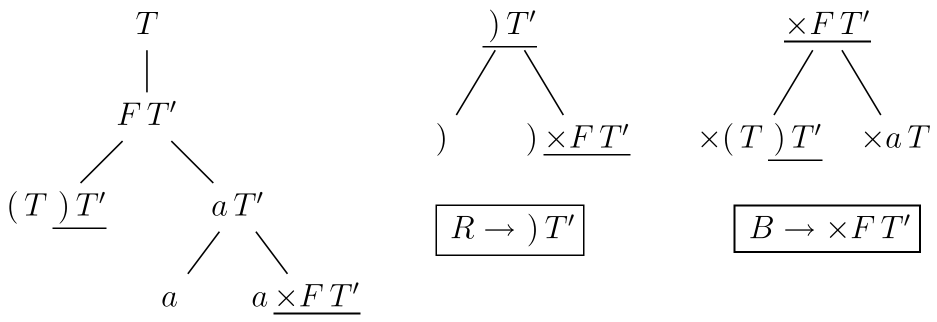
T
$F\,T'$
$(\,T\,\underline{)\,T'}$
$a\,T'$
a
$a\,\underline{\times F\,T'}$
$\underline{)\,T'}$
$)$
$)\,\underline{\times F\,T'}$
$\boxed{R \rightarrow)\,T'}$
$\underline{\times F\,T'}$
$\times(\,T\,\underline{)\,T'}$
$\times a\,T$
$\boxed{B \rightarrow \times F\,T'}$

FIGURE 1.3.4. Trees of sentential forms obtained by unfolding the left-most nonterminals.

FACT 1.3.11. For any $k \geq 0$, there are languages which are generated by an $LL(k{+}1)$ grammars and cannot be generated by any $LL(k)$ grammar.

FACT 1.3.12. For any $k \geq 0$, it is decidable whether or not two $LL(k)$ grammars generate the same language.

FACT 1.3.13. If a grammar G is in Greibach normal form and for each nonterminal A there exists at most one production of the form $A \rightarrow a\,\alpha$ for some terminal symbol a and some $\alpha \in V_N^*$ and the only ε-production allowed is $S \rightarrow \varepsilon$, then G is $LL(1)$.

Recall that in any $LL(1)$ grammar we allow ε-productions for each nonterminal symbol and, thus, the class of such grammars properly includes the class of the grammars in Greibach normal form mentioned in the above Fact 1.3.13.

As already mentioned, we have the following result.

FACT 1.3.14. A language $L \in LL(0)$ iff $L = \emptyset$ or L is a singleton.

FACT 1.3.15. Every languages which is generated by an $LL(k)$ grammar for some $k \geq 0$, is a deterministic context-free language, but there exists a subclass C of the deterministic context-free languages such that for every language L in C there is no $k \geq 0$ such that L can be generated by an $LL(k)$ grammar.

FACT 1.3.16. The language $L = \{a^n b^n \mid n \geq 1\} \cup \{a^n c^n \mid n \geq 1\}$ is a deterministic context-free language and there is no $k \geq 0$ such that $L = LL(k)$ [**3**, page 689].

In the following Section 1.4 we will see that the language $L = \{a^n b^n \mid n \geq 1\} \cup \{a^n c^n \mid n \geq 1\}$ is $LR(1)$.

THEOREM 1.3.17. [**Hierarchy of $LL(k)$ Grammars and Languages**] For every $k \geq 0$, (i) the class of the $LL(k)$ grammars is properly contained in the class of the $LL(k+1)$ grammars, and (ii) the class of the $LL(k)$ languages is properly contained in the class of the $LL(k+1)$ languages.

FACT 1.3.18. [**7**] For any $k \geq 0$, the class of the class of $LL(k)$ languages is properly contained in the class of the $LR(1)$ languages.