In this chapter we consider another approach to parsing, called an LR(1)-parsing algorithm, as well as some simplified versions of it. We start by describing a general scheme of left-to-right parsing (section 16.1). Then we consider a class of grammars for which this scheme can be implemented easily (LR(0)-grammars, section 16.2). Some extensions of this class are discussed in sections 16.3 (SLR(1)-grammars) and 16.4 (LR(1)- and LALR(1)-grammars). We conclude the chapter with some general remarks about parsing (section 16.5).

16.1 LR-processes

There are two main differences between LR(1)-parsing and LL(1)-parsing. First, we seek a rightmost derivation, not a leftmost one. Second, we construct the derivation from the bottom (beginning with the input string) to the top (the axiom) and not vice-versa (as in LL(1)-parsing).

A rightmost derivation is a derivation where the rightmost nonterminal is replaced at each step.

16.1.1. Prove that any derivable string of terminals has a rightmost derivation. □

It is convenient to look at the rightmost derivation backwards, starting from the input string. Let us define the notion of an LR-process on the input string $A$. This process involves the string $A$ and another string $S$ that contains both terminals and nonterminals. Initially, the string $S$ is empty. The LR-process includes two types of actions:

(1) the first character of $A$ (called the next input symbol and denoted by $\text{Next}$) may be moved to the end of the string $S$ (and deleted from $A$); this action is called a shift action;

(2) if the right-hand side of some production rule is a suffix of $S$, then it can be replaced by the nonterminal that is on the left-hand side of that rule; the string $A$ remains unchanged. This action is called a reduce action.
Let us mention that the LR-process is not deterministic; there are situations where many different actions are possible.

We say that the LR-process on a string $A$ is successful if the string $A$ becomes empty and the string $S$ contains only one nonterminal, and this nonterminal is the initial nonterminal (the axiom).

**16.1.2.** Prove that for any string $A$ (of terminals) a successful LR-process exists if and only if $A$ is derivable in the grammar. Find a one-to-one correspondence between rightmost derivations and successful LR-processes.

*Solution.* The shift action does not change the string $SA$. The reduce action changes $SA$ and this change is a reversed step of a derivation. This derivation is a rightmost one because the reduction is done at the end of $S$ and all symbols of $A$ are terminals. Therefore, each LR-process corresponds to a rightmost derivation.

Conversely, assume that a rightmost derivation is given. Imagine a separator placed after the last nonterminal in the string. When a production rule is applied to that nonterminal, we may need to move the separator to the left (if the right-hand side of the rule applied ends with a terminal). Splitting this move into steps (one symbol per step) we get a process that is exactly an inverted LR-process. □

All changes in the string $S$ during an LR-process are made near its right end. This is why the string $S$ is called the stack of the LR-process.

So the problem of finding the rightmost derivation of a given string is the problem of constructing a successful LR-process on this string. At each step we have to decide whether we want to apply a shift or reduce action, and choose a production rule if several reductions are possible. In the LR(1)-algorithm, the decision is made based on $S$ and the first symbol of $A$. If only information about $S$ is used, it is an LR(0)-algorithm. (The exact definitions are given below.)

Assume that a grammar is fixed. In the sequel, we assume that for each nonterminal there exists a string of terminals derivable from it.

Let $K \rightarrow U$ be one of the grammar’s rules ($K$ is a nonterminal, $U$ is a string of terminals and nonterminals). We consider a set of strings (composed of both terminals and nonterminals) called the left context of the rule $K \rightarrow U$. (Notation: LeftCont($K \rightarrow U$).) By definition, this set contains all the strings that may appear as a stack content immediately before the reduction of $U$ to $K$ in a successful LR-process.

**16.1.3.** Reformulate this definition in terms of rightmost derivations.

*Solution.* Consider all rightmost derivations of the form

$$(\text{axiom}) \rightsquigarrow XKA \rightarrow XUA,$$

where $A$ is a string of terminals, $X$ is a string of terminals and nonterminals, and $K \rightarrow U$ is a production rule. All strings $XU$ that appear in those derivations form the left context of the rule $K \rightarrow U$. Indeed, recall that we assume that for any nonterminal there exists a string of terminals derivable from it; therefore, the rightmost derivation of the string $XUA$ may be continued until a right derivation of some string of terminals is obtained.