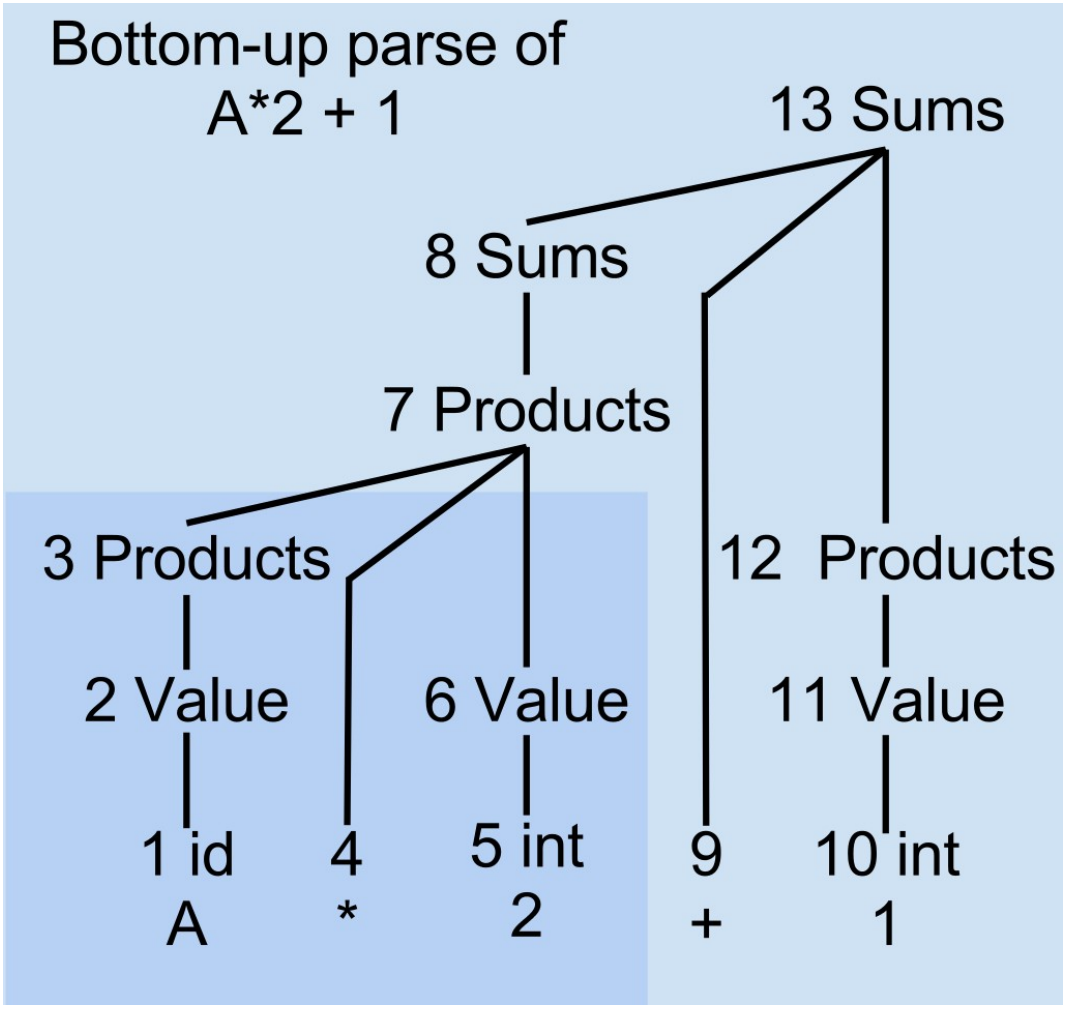
A simple PDF document

In computer science, LR parsers are a type of bottom-up parser that analyse deterministic context-free languages in linear time.[1] There are several variants of LR parsers: SLR parsers, LALR parsers, canonical LR(1) parsers, minimal LR(1) parsers, and generalized LR parsers (GLR parsers). LR parsers can be generated by a parser generator from a formal grammar defining the syntax of the language to be parsed. They are widely used for the processing of computer languages.

An LR parser (left-to-right, rightmost derivation in reverse) reads input text from left to right without backing up (this is true for most parsers), and produces a rightmost derivation in reverse: it does a bottom-up parse – not a top-down LL parse or ad-hoc parse. The name "LR" is often followed by a numeric qualifier, as in "LR(1)" or sometimes "LR(k)". To avoid backtracking or guessing, the LR parser is allowed to peek ahead at k lookahead input symbols before deciding how to parse earlier symbols. Typically k is 1 and is not mentioned. The name "LR" is often preceded by other qualifiers, as in "SLR" and "LALR". The "LR(k)" notation for a grammar was suggested by Knuth to stand for "translatable from left to right with bound k."[1]

LR parsers are deterministic; they produce a single correct parse without guesswork or backtracking, in linear time. This is ideal for computer languages, but LR parsers are not suited for human languages which need more flexible but inevitably slower methods. Some methods which can parse arbitrary context-free languages (e.g., Cocke–Younger–Kasami, Earley, GLR) have worst-case performance of O(n3) time. Other methods which backtrack or yield multiple parses may even take exponential time when they guess badly.[2]

The above properties of L, R, and k are actually shared by all shift-reduce parsers, including precedence parsers. But by convention, the LR name stands for the form of parsing invented by Donald Knuth, and excludes the earlier, less powerful precedence methods (for example Operator-precedence parser).[1] LR parsers can handle a larger range of languages and grammars than precedence parsers or top-down LL parsing.[3] This is because the LR parser waits until it has seen an entire instance of some grammar pattern before committing to what it has found. An LL parser has to decide or guess what it is seeing much sooner, when it has only seen the leftmost input symbol of that pattern.


Bottom-up parse of
A*2 + 1
13 Sums
8 Sums
7 Products
3 Products
12 Products
2 Value
6 Value
11 Value
1 id
A
4
*
5 int
2
9
+
10 int
1