CSE322

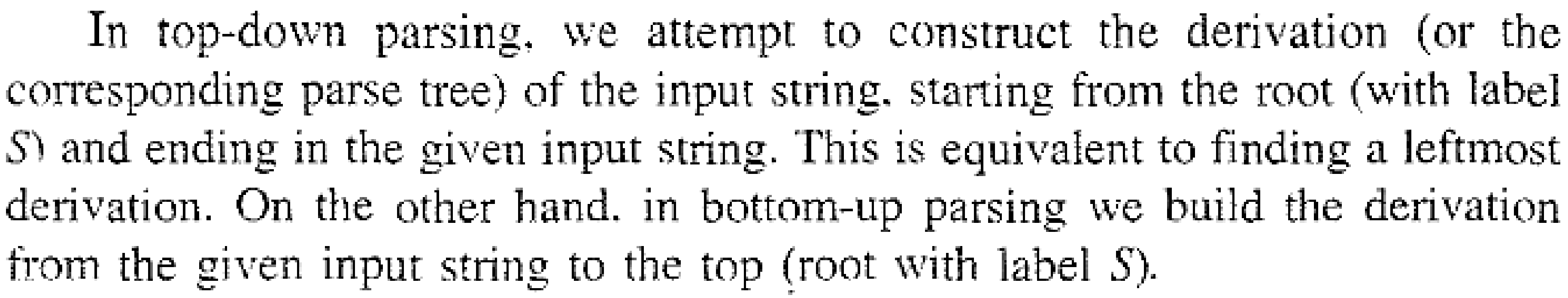
PARSING: Top-Down and Bottom-Up Parsing
LL(K) and LR(K) grammar

Lecture #35

Parsing

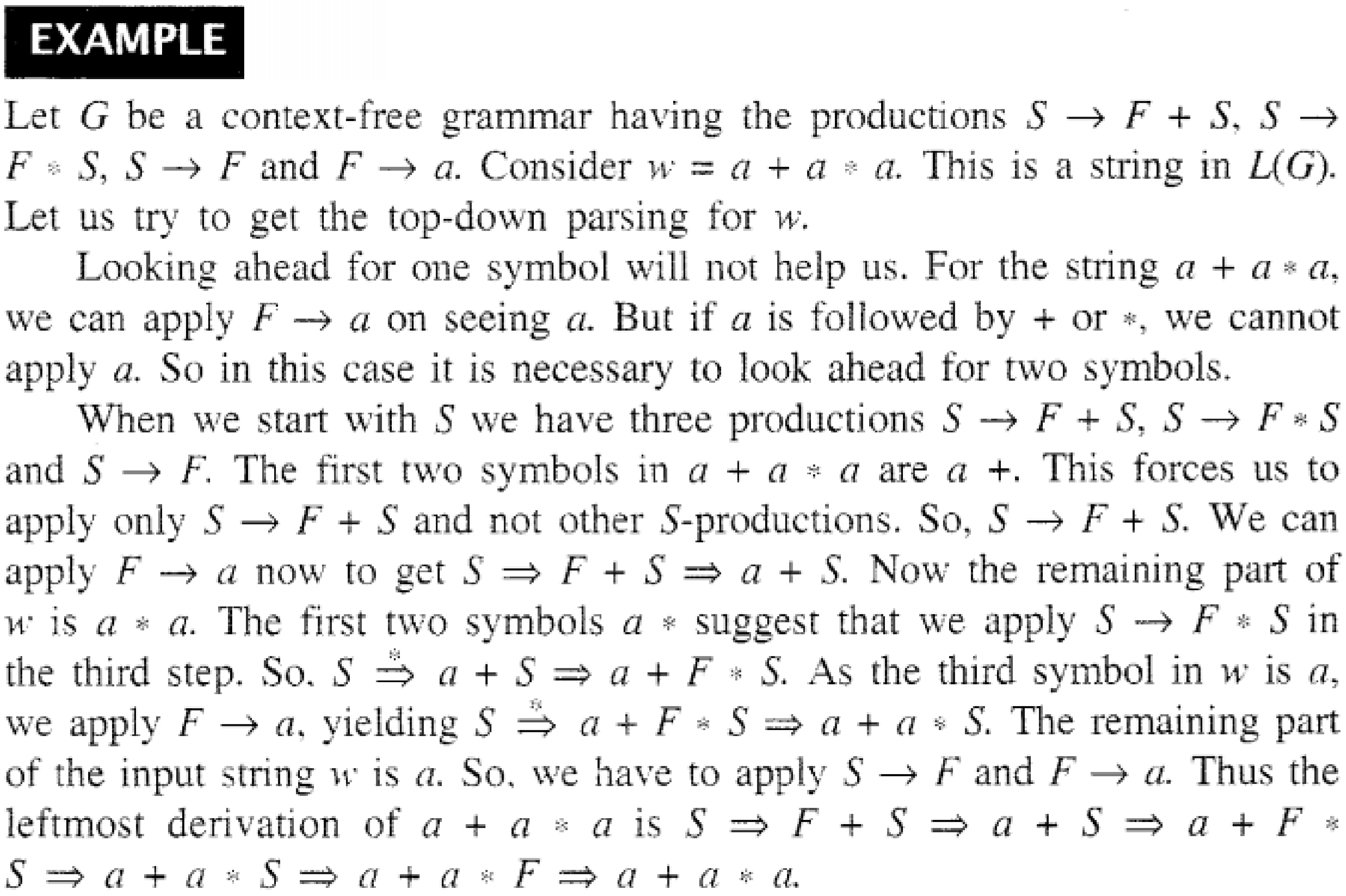
- Parsing is process of splitting a sentence into words.
- Two types of Parsing
  - **Top Down**
  - **Bottom Up**

  In Formal Language we have :-



In top-down parsing, we attempt to construct the derivation (or the corresponding parse tree) of the input string, starting from the root (with label *S*) and ending in the given input string. This is equivalent to finding a leftmost derivation. On the other hand, in bottom-up parsing we build the derivation from the given input string to the top (root with label *S*).

Top Down Parsing



EXAMPLE

Let G be a context-free grammar having the productions $S \to F + S$, $S \to F * S$, $S \to F$ and $F \to a$. Consider $w = a + a * a$. This is a string in $L(G)$. Let us try to get the top-down parsing for w.

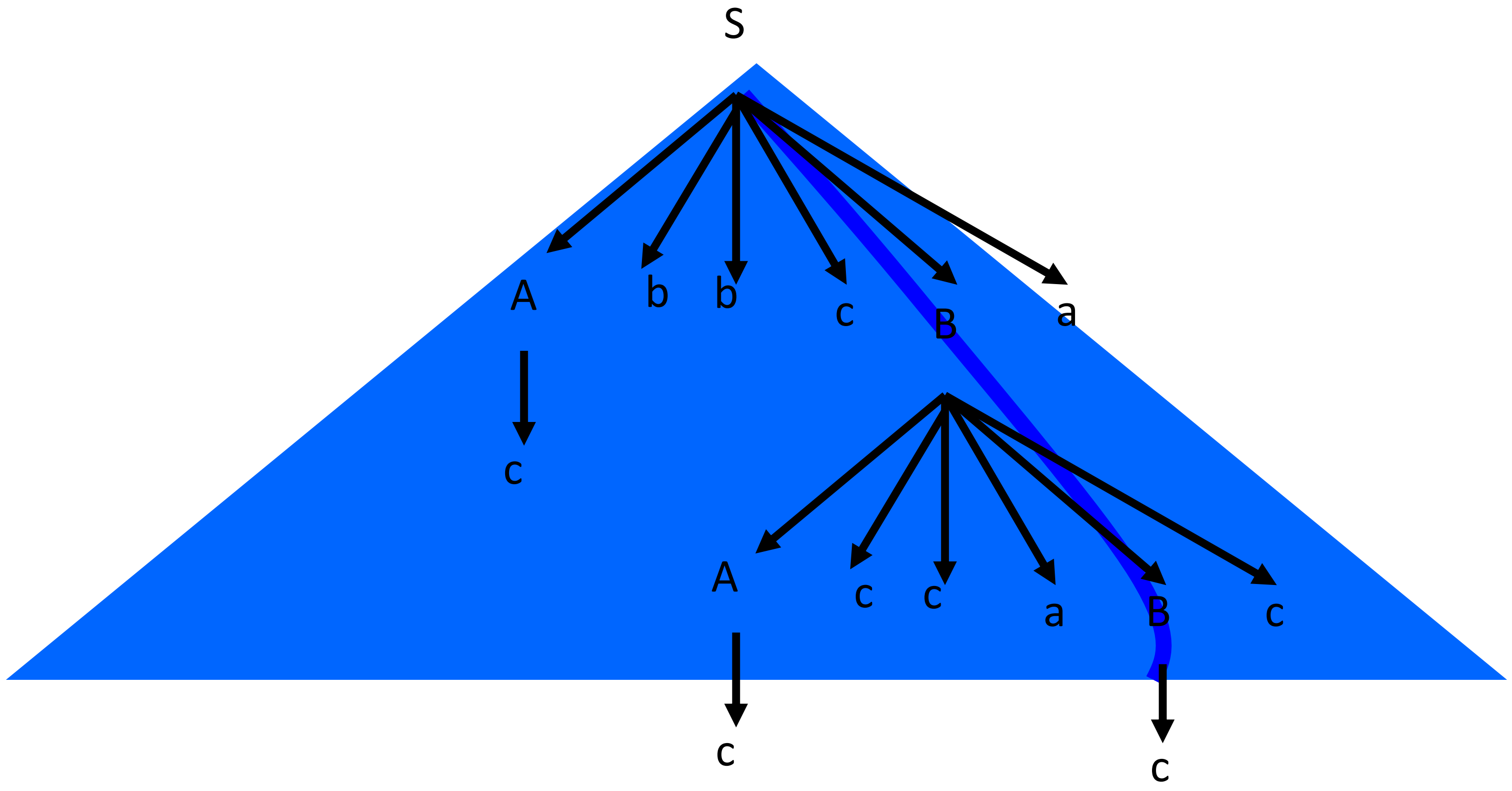
Looking ahead for one symbol will not help us. For the string $a + a * a$, we can apply $F \to a$ on seeing a. But if a is followed by $+$ or $*$, we cannot apply a. So in this case it is necessary to look ahead for two symbols.

When we start with S we have three productions $S \to F + S$, $S \to F * S$ and $S \to F$. The first two symbols in $a + a * a$ are a $+$. This forces us to apply only $S \to F + S$ and not other S-productions. So, $S \to F + S$. We can apply $F \to a$ now to get $S \Rightarrow F + S \Rightarrow a + S$. Now the remaining part of w is $a * a$. The first two symbols a $*$ suggest that we apply $S \to F * S$ in the third step. So. $S \overset{*}{\Rightarrow} a + S \Rightarrow a + F * S$. As the third symbol in w is a, we apply $F \to a$, yielding $S \overset{*}{\Rightarrow} a + F * S \Rightarrow a + a * S$. The remaining part of the input string w is a. So, we have to apply $S \to F$ and $F \to a$. Thus the leftmost derivation of $a + a * a$ is $S \Rightarrow F + S \Rightarrow a + S \Rightarrow a + F * S \Rightarrow a + a * S \Rightarrow a + a * F \Rightarrow a + a * a$.

Remember Parse Trees

Parse tree for S ⇒ AbbcBa ⇒* cbbccccaBca

⇒ cbbccccacca

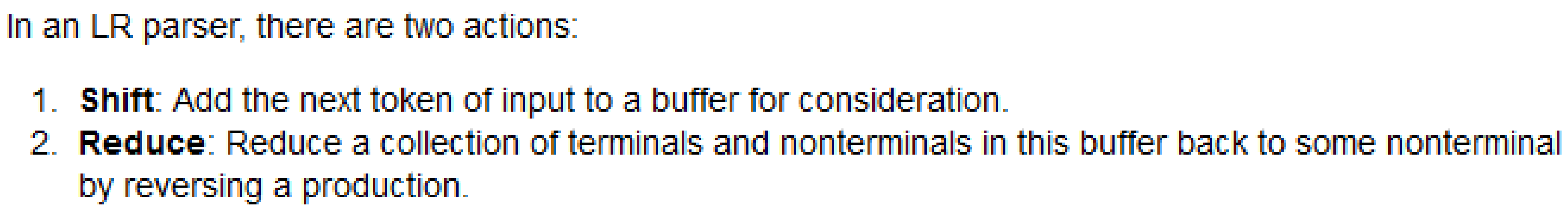


LL(K) Grammar and Properties

- A Grammar G having Property(by looking ahead for K symbols we derive a given input string in L(G) is called an LL(K) Grammar.
- Example:-
- Let G = ({S, A, B}, {a, b} , P , S) where P consists of $S \rightarrow aAB$, $S \rightarrow bBA$, $A \rightarrow bS$, $A \rightarrow a$, $B \rightarrow aS$,
- $B \rightarrow b$, w= abbbab is in L(G)

LR(K) Grammar

- LR(K) stands for left to right scan of the input string producing a rightmost derivation using k symbol look ahead on the input string.



In an LR parser, there are two actions:

1. **Shift**: Add the next token of input to a buffer for consideration.
2. **Reduce**: Reduce a collection of terminals and nonterminals in this buffer back to some nonterminal by reversing a production.

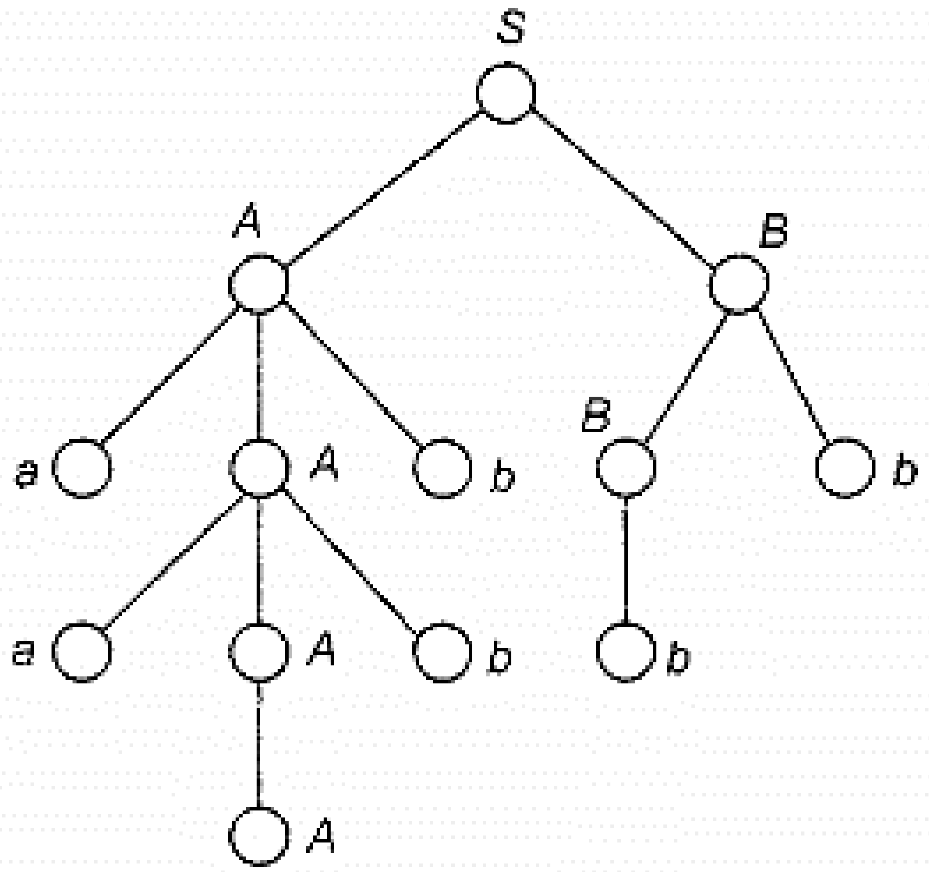
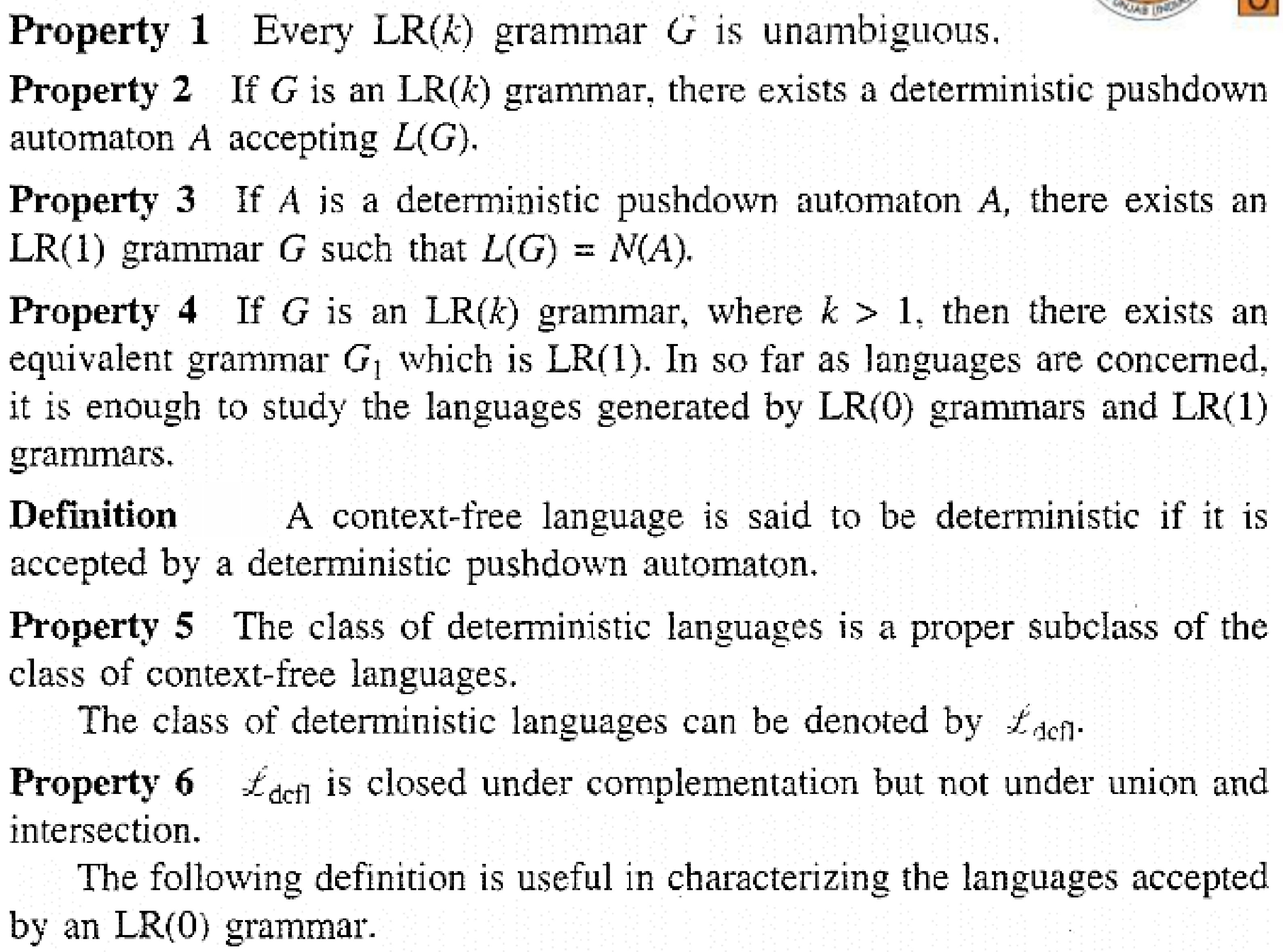


Fig. Derivation tree for a^2b^4.

Properties of LR(K)



Property 1 Every LR(k) grammar G is unambiguous.

Property 2 If G is an LR(k) grammar, there exists a deterministic pushdown automaton A accepting $L(G)$.

Property 3 If A is a deterministic pushdown automaton A, there exists an LR(1) grammar G such that $L(G) = N(A)$.

Property 4 If G is an LR(k) grammar, where $k > 1$, then there exists an equivalent grammar G_1 which is LR(1). In so far as languages are concerned, it is enough to study the languages generated by LR(0) grammars and LR(1) grammars.

Definition A context-free language is said to be deterministic if it is accepted by a deterministic pushdown automaton.

Property 5 The class of deterministic languages is a proper subclass of the class of context-free languages.

The class of deterministic languages can be denoted by $\mathcal{L}_{\text{dcfl}}$.

Property 6 $\mathcal{L}_{\text{dcfl}}$ is closed under complementation but not under union and intersection.

The following definition is useful in characterizing the languages accepted by an LR(0) grammar.

Definition 8.3 A context-free language has prefix property if no proper prefix of strings of *L* belongs to *L*.

Property 7 A context-free language is generated by an LR(0) grammar if and only if it is accepted by a deterministic pushdown automaton and has prefix property.

Property 8 There is an algorithm to decide whether a given context-free grammar is LR(*k*) for a given natural number *k*.

Parsing Ambiguity

- If for a input string two or more tree can be drawn then the parsing is ambiguous.