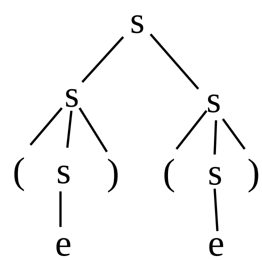
1 Parse Trees

Parse trees are a representation of derivations that is much more compact. Several derivations may correspond to the same parse tree. For example, in the balanced parenthesis grammar, the following parse tree:



corresponds to the derivation

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow (S)(S) \Rightarrow (S)() \Rightarrow ()()$$

as well as this one:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow ()(S) \Rightarrow ()()$$

and some others as well.

- In a parse tree, the points are called *nodes*. Each node has a *label* on it.
- The topmost node is called the *root*. The bottom nodes are called *leaves*.
- In a parse tree for a grammar G, the leaves must be labelled with terminal symbols from G, or with ϵ. The root is often labeled with the start symbol of G, but not always.
- If a node N labeled with A has children $N_1, N_2, \ldots, N_k$ from left to right, labeled with $A_1, A_2, \ldots, A_k$, respectively, then $A \rightarrow A_1 A_2, \ldots A_k$ must be a production in the grammar G.
- The *yield* of a parse tree is the concatenation of the labels of the leaves, from left to right. The yield of the tree above is ()().

1.1 Leftmost and Rightmost Derivations

- In a leftmost derivation, at each step the leftmost nonterminal is replaced. In a rightmost derivation, at each step the rightmost nonterminal is replaced.
- Such replacements are indicated by $\stackrel{L}{\Rightarrow}$ and $\stackrel{R}{\Rightarrow}$, respectively.
- Their transitive closures are $\stackrel{L}{\Rightarrow}^*$ and $\stackrel{R}{\Rightarrow}^*$, respectively.

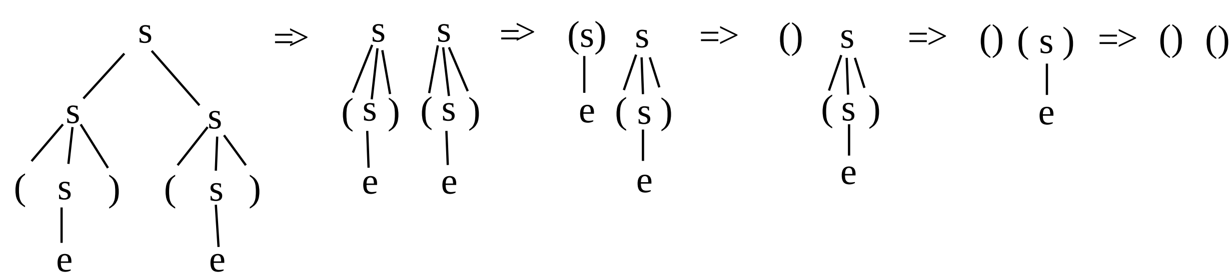
In the balanced parenthesis grammar, this is a leftmost derivation:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

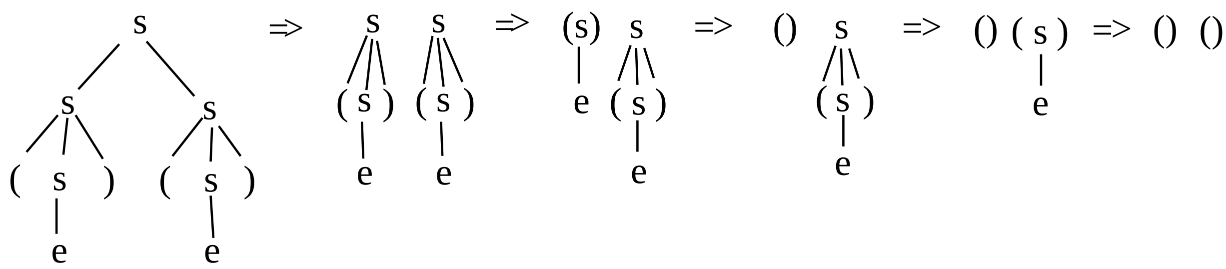
This is a rightmost derivation:

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ()()$$

It is possible to obtain a derivation from a parse tree and vice versa. Here is an example of obtaining a derivation from a parse tree, going from left to right:



- In this case, we obtained a leftmost derivation, but we could also have obtained a rightmost derivation in a similar way.
- Using the same diagram, going from right to left, starting with only an arbitrary derivation, we can obtain a parse tree:



Thus from a parse tree, we can obtain a leftmost or a rightmost derivation, and from an arbitrary derivation, we can obtain a parse tree. This gives us theorem 3.2.1 in the text:

Theorem 1.1 (3.2.1) *Let $G = (V, \Sigma, R, S)$ be a context-free grammar, $A \in V - \Sigma$, and $w \in \Sigma^*$. Then the following are equivalent (TFAE):*

1. *$A \Rightarrow^* w$*
2. *There is a parse tree with root labeled A and yield w*
3. *There is a leftmost derivation $A \overset{L}{\Rightarrow}{}^* w$*
4. *There is a rightmost derivation $A \overset{R}{\Rightarrow}{}^* w$*

Proof:

- We showed above how from a derivation one can construct a parse tree. This shows (1) implies (2).
- Also, we showed above how from a parse tree one can construct a leftmost or a rightmost derivation. This shows that (2) implies (3) and (2) implies (4).
- Finally, leftmost and rightmost derivations are derivations, which shows that (3) implies (1) and (4) implies (1).
- Thus all four conditions are equivalent.

1.2 Ambiguity

Some sentences in English are ambiguous:

Fighting tigers can be dangerous.
Time flies like an arrow.

Humor is also often based on ambiguity. Example jokes:

How do you stop an elephant from charging?
Why did the student eat his homework?
What ended in 1896?

There is also a technical concept of ambiguity for context-free grammars.

> A context-free grammar $G = (V, \Sigma, R, S)$ is *ambiguous* if there is some string $w \in \Sigma^*$ such that there are two distinct parse trees T_1 and T_2 having S at the root and having yield w.

Equivalently, w has two or more leftmost derivations, or two or more rightmost derivations.

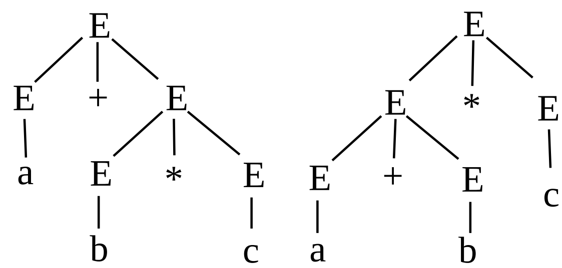
Note that languages are not ambiguous; grammars are. Also, it has to be the *same* string w with two *different* (leftmost or rightmost) derivations for a grammar to be ambiguouos.

Here is an example of an ambiguous grammar:

$$\begin{array}{ll} E \to E + E & E \to a \\ E \to E * E & E \to b \\ E \to (E) & E \to c \end{array}$$

In this grammar, the string $a + b * c$ can be parsed in two different ways, corresponding to doing the addition before or after the multiplication. This is very bad for a compiler, because the compiler uses the parse tree to generate code, meaning that this string could have two very different semantics.

Here are two parse trees for the string $a + b * c$ in this grammar:



Ambiguity actually happened with the original Algol 60 syntax, which was ambiguous for this string:

if x then if y then z else w;

How is this string ambiguous? Which values of x, y, or z lead to the ambiguity?

There is a notion of *inherent ambiguity* for context-free languages; a context-free language L is inherently ambiguous if every context-free grammar G for L is ambiguous. As an example, the language

$$\{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n : n \geq 1, m \geq 1\}$$

is inherently ambiguous. In any context-free grammar for L, some strings of the form $a^n b^n c^n d^n$ will have two distinct parse trees.

Unfortunately, the problem of whether a context-free grammar is ambiguous, is undecidable. However, there are some patterns in a context-free grammar that frequently indicate ambiguity:

$$
\begin{aligned}
&S \to SS\\
&S \to a
\end{aligned}
$$

$$
\begin{aligned}
&S \to A\\
&A \to AA\\
&A \to a
\end{aligned}
$$

$$
\begin{aligned}
&S \to AA\\
&A \to S\\
&A \to a
\end{aligned}
$$

$$
\begin{aligned}
&S \to SbS\\
&S \to a
\end{aligned}
$$

$$
\begin{aligned}
&S \to AbA\\
&A \to S\\
&A \to a
\end{aligned}
$$

The following is not ambiguous:

$$
\begin{aligned}
&S \to aS\\
&S \to bS\\
&S \to \epsilon
\end{aligned}
$$

In general, a production $A \to AA$ causes ambiguity if it is reachable from the start symbol and some terminal string is derivable from A.