Homework #16: 3.2.2, 3.2.3, 3.2.4b

Today: a different way to represent derivations.

Consider
\[ S \rightarrow AB \]
\[ A \rightarrow aA \quad A \rightarrow e \quad B \rightarrow bB \quad B \rightarrow e \]

and now consider a new representation of a derivation:

Parse Tree:
- root = genly the start symbol
- intermediate nodes = non-terminals
- leaves = terminals

read the terminals left to right.

string is: aaab

Def. Parse tree, root, leaves, yield for an arbitrary \( G = (V, \Sigma, R, S) \).

(1) \( \bullet a \) is a parse tree.
- the single node is both a root and a leaf.
Note: \( a \in \Sigma \).
(2) if $A \rightarrow e$ is a rule in $R$

```
A
```

is a parse tree

the root is labeled $A$

the one leaf is labeled $e$

yield is $e$.

(3) if

```
A1
```

```
An
```

are parse trees and $A \rightarrow A1 \ldots An \in R$

then

```
A
```

```
A1
```

```
An
```

```
T1
```

```
yn
```

```
Tn
```

```
y1
```

```
\ldots
```

is a parse tree.

the root is $A$

the leaves are the leaves of the constituent trees
Def. A path is a sequence of nodes from root to leaf.

The height of the parse tree is the length of the longest path.

Parse trees represent derivations of strings in $L(G)$ so that the superficial differences between derivations, owing to the order of application of rules, are suppressed.

Leftmost and rightmost derivations

A leftmost derivation exists in every parse tree – obtained by repeatedly replacing the leftmost non-terminal.

A rightmost derivation exists in every parse tree – obtained by repeatedly replacing the rightmost non-terminal.

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

(a) $A \Rightarrow^* w$
(b) There is a parse tree with root $A$ and yield $w$.
(c) There is a leftmost derivation $A \Rightarrow_{L*}^* w$.
(d) There is a rightmost derivation $A \Rightarrow_{R*}^* w$.

Again, parse trees represent derivations without superficial differences of order of rule application. However...

Parse trees will be different, of course, for a different choice of rules to apply.

For instance, consider the following grammar for generating boolean expressions without parentheses.

$$
R: 
E \rightarrow E \| E \\
E \rightarrow ! E \\
E \rightarrow N \\
B \rightarrow true \mid false \\
N \rightarrow x1 \mid x2 \mid x3
$$

$$
E \Rightarrow E \| E \Rightarrow ! E \| E \Rightarrow ! E \| N \| E \Rightarrow ! x1 \| E \Rightarrow ! x1 \| N
$$
The first two generate the same parse tree. The third is completely different.

Grammars with strings that have two or more distinct parse trees are called **ambiguous**.

Can sometimes disambiguate a grammar by restructuring it.

There are some CFLs with the property that all CFGs generating them are ambiguous – call these inherently ambiguous.

Ex. \{a^i b^i c^k | i=j or i=k\}

Every string a^n b^n c^n has two distinct derivations.