Compilers

CMPT 432

Lab 9

Goals	Manipulating Grammars
Notes	It's good to have a nice grammar that's easily (!) parsed in a top-down manner because it's LL(1). But not all grammars come to us in that way. Thus, we may have to be manipulative and change them. Let's practice.
Resources	*Crafting a Compiler* • Read chapters 5.5 and 6.1-2 • Do exercises 5.5. If you're into it, take a shot at exercise 6.51. That's quite interesting to think about. *Dragon* • Read chapters 2.4.5, 4.5-6, and 4.8 • Do exercises 4.5.3 and 4.6.5
Submitting	Commit a PDF of your work to your GitHub repository and I'll take a look at it.

LEFT FACTORING

We have seen that left recursion interferes with predictive parsing, and that it can be eliminated. A similar problem occurs when two productions for the same nonterminal start with the same symbols. For example:

$S \rightarrow$ if E then S else S
$S \rightarrow$ if E then S

In such a case, we can *left factor* the grammar – that is, take the allowable endings (*else* S and ϵ) and make a new nonterminal X to stand for them:

$S \rightarrow$ if E then S X
$X \rightarrow$
$X \rightarrow$ else S

The resulting productions will not pose a problem for a predictive parser. Although the grammar is still ambiguous – the parsing table has two entries for the same slot – we can resolve the ambiguity by using the *else* S action.

from *Modern Compiler Implementation in Java* by Andrew Appel