Ambiguity

this is a property of a grammar.

Def: A grammar is ambiguous if there is a string w that can be derived using the grammar using two different derivation trees.

E.g. $\Sigma = \{a, b, c, (,), *, +\}$

$L = \{$ algebra expressions w/ a's, b's & c's as variables $\}$

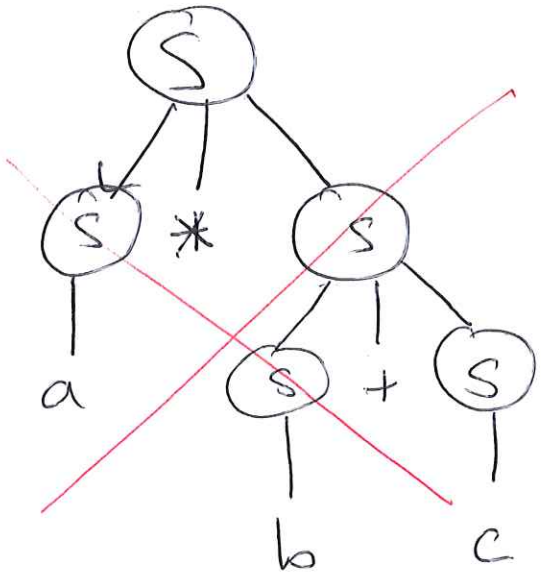
Grammar:

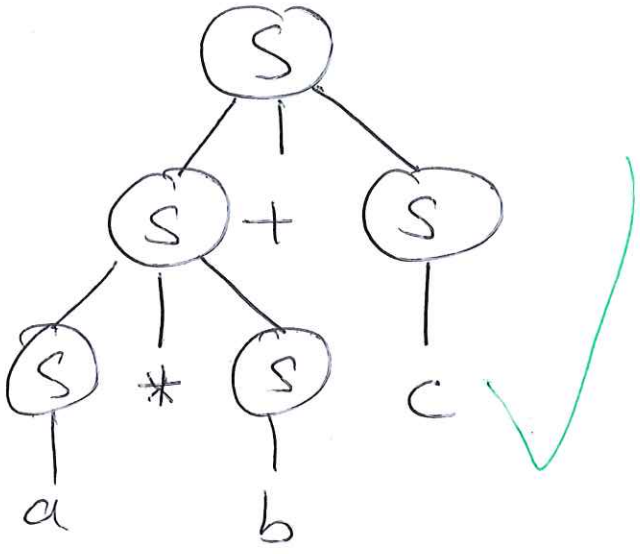
$$S \longrightarrow a \mid b \mid c \mid (S) \mid S * S \mid S + S$$

Consider the string

$a * b + c$

This has 2 derivation trees:





We would like to have a grammar that is unambiguous - has only one parsing (ie. one deriv. tree) (and we'd like the one the forces the parsing that agrees w/ usual semantics)

Variables:

Expression (start variable)

~~Factor~~ ~~Product~~

Term (an expression is a sum of terms - term = summand)

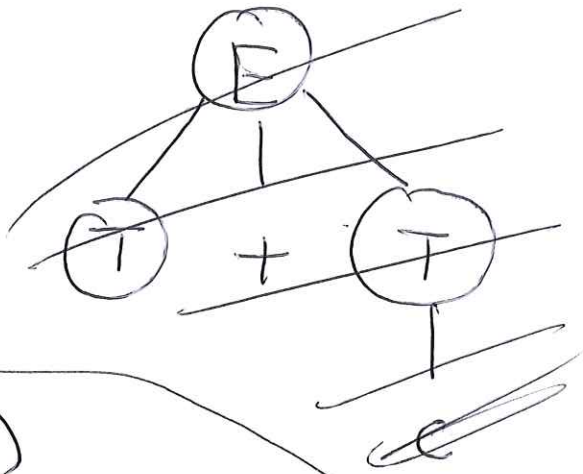
Factor

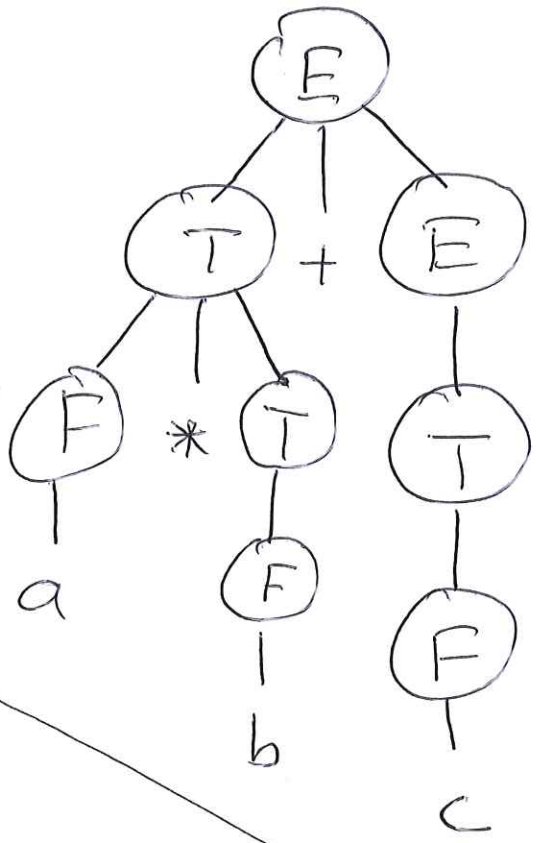
$E \rightarrow T \mid T + E$

$T \rightarrow F \mid F * T$

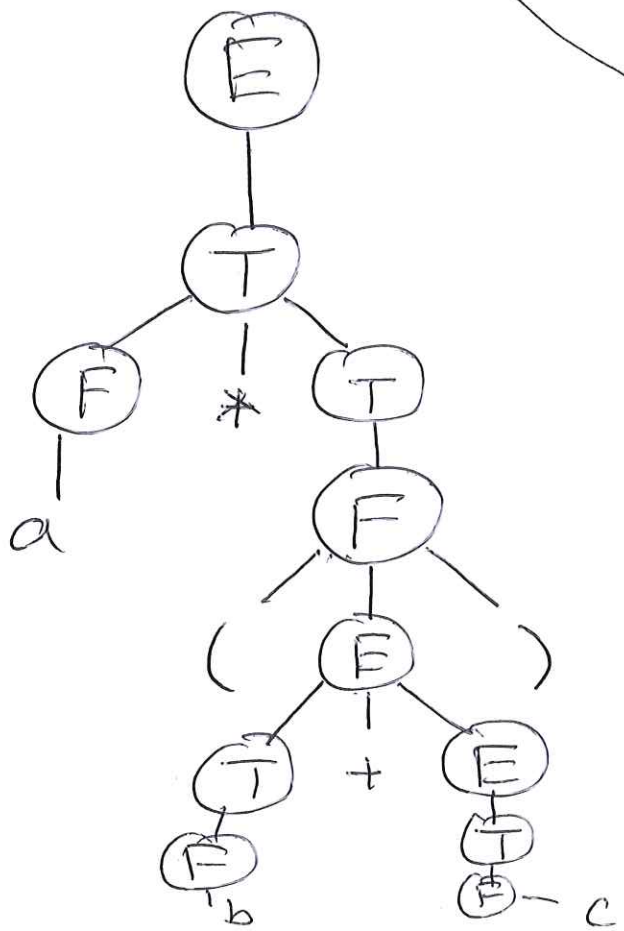
$F \rightarrow a \mid b \mid c \mid (E)$

Parse $a * b + c$





$a * (b + c)$



Unfortunately, there are some context-free languages that have no unambiguous grammars - every grammar for it will be ambiguous. These languages are said to be **inherently ambiguous**.

↑ property of a language, not of a grammar.

(Note - it is possible to unnecessarily add ambiguity to a grammar - so it makes no sense to talk about languages where every grammar is unambiguous.)

E.g.: $\{a^m b^m c^k\} \cup \{a^k b^m c^m\}$

(either $\#a's = \#b's$ or $\#b's = \#c's$)

Grammar:

$S \to XC \mid AY$

$X \to aXb \mid ab$

$Y \to bYc \mid bc$

$C \to cC \mid c$

$A \to aA \mid a$

Any string with $\#a's = \#b's = \#c's$ has 2 derivations

Modifying grammars (in ways that don't change the language)

Goal: We want ~~algor.~~ to study some algorithms that tell us things about a context-free language. To make it easier to describe the algs, we want to have only certain kinds of productions allowed. We want modifications to grammars to let us ~~get them~~ replace unallowed kinds of productions w/ allowed productions, (w/o changing what the grammar produces)

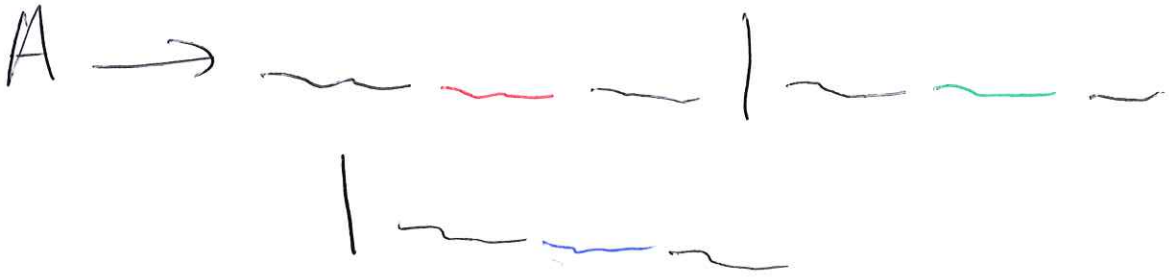
General idea: ~~we can replace a~~ if we have productions

$A \longrightarrow \_\_B\_\_$

and $B \longrightarrow \_\_ \mid \_\_ \mid \_\_$ ← all the prods for B

We can replace

w/



Removing useless productions/variables.

E.g. $S \to aA \mid baB \mid abc$

$A \to bB \mid aS$

$B \to caB \mid baA \mid bS \mid cb$

~~$C \to abA \mid bC \mid aD$~~

~~$D \to SacB$~~

Anything you can't reach from the start state is useless.

(We can use ~~DF~~ depth-first or breadth-first search to mark all the useful vars starting at S - the rest are useless.

Another kind of uselessness:

$S \rightarrow AbB \mid bA \mid BCa \mid ab$

$A \rightarrow aS \mid bB \mid aAB \mid bc$

$B \rightarrow aB \mid bC \mid aBC$

$C \rightarrow bB \mid aC \mid BCAa$

It's not possible to make B's or C's all go away