lec 03/03

A CFL can be constructed from many diff CFGs.

one CFL ⟵ CFG_1, CFG_2, ⋮

$X \rightarrow \alpha$

X: Non terminal; α: string of terminals + non terminals

Ambiguous / Unambiguous – diff parse trees for same string

↓

can also have 2 derivations for same string even if unambiguous.

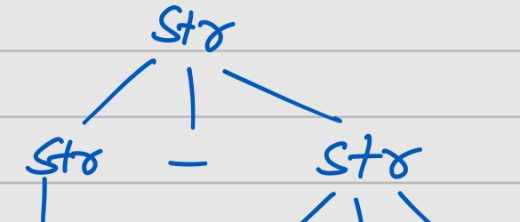
fix derivation order – then unique derivation for string.

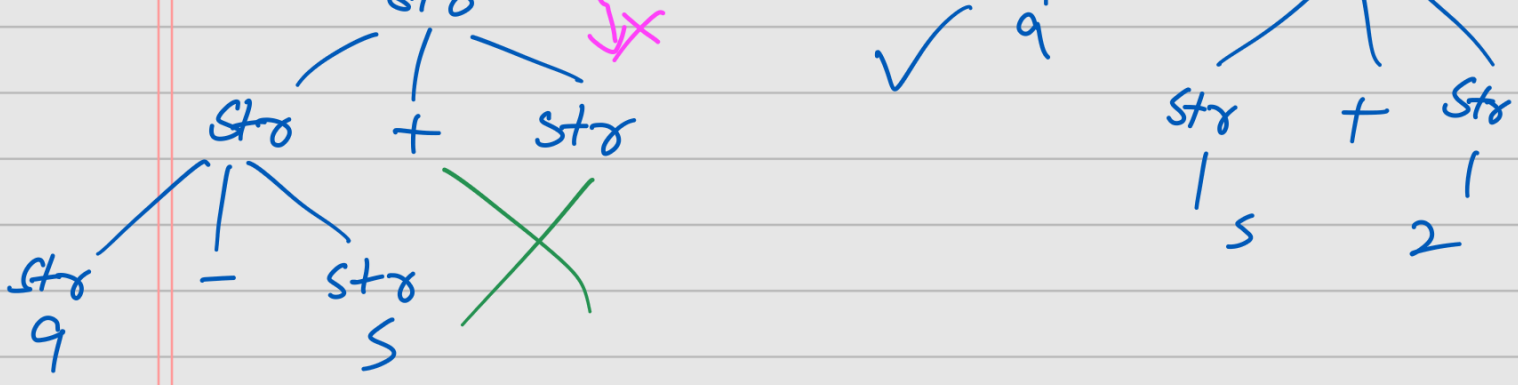
but for ambiguous grammar – we can resolve same non terminal to diff productions and get the same string

associativity – $2+5+9$ (same symbol)
precedence – $2\times5+9$

Precedence & associa → tell us in which order the parse tree works.

e.g. + > −





C → else matches the closest if.

Recursive Descent parser – does not work with left recursive grammar. $S \rightarrow S\alpha$

First set of a non terminal N – set of all terminals that N can generate on the first position in all strings it can generate

$S \rightarrow A \mid B \quad A \rightarrow a \quad B \rightarrow b \quad S = \{a, b\}$
$A = \{a\} \quad B = \{b\}$

String — of only terminals
sentential form — string of non terminals + terminals

first set of sentential form – first of all non terminal or terminal if leftmost

$A \rightarrow aC \mid Cb \quad A = \{a, \text{firstset}(C)\}$

$A \rightarrow XYZ$

first set of $A = fs(X)$

if X is nullable $(X \rightarrow \varepsilon) \Rightarrow$

$fs(A) = fs(X) \cup fs(Y)$

if Y is also nullable

$fs(A) = fs(X) \cup fs(Y) \cup fs(Z)$

instead of blindly picking prodⁿ, we can pick sent. forms / Non terminals that can derive the first terminal. Then go for next termid

pascal declⁿ of array:

array [integer] of integer