

Parsing		
<ul> <li>find a derivation of a shown</li> </ul>	<+y*z in the gramn	$\begin{array}{c} E \longrightarrow E + E \\ E \longrightarrow E * E \\ E \longrightarrow (E) \\ E \longrightarrow x \\ E \longrightarrow y \\ E \longrightarrow z \end{array}$
$E \Longrightarrow E + E$ $\Longrightarrow E + E * E$ $\Longrightarrow E + y * E$ $\Longrightarrow x + y * E$ $\Longrightarrow x + y * z$	$E \Longrightarrow E + E$ $\Longrightarrow x + E$ $\Longrightarrow x + E * E$ $\Longrightarrow x + y * E$ $\Longrightarrow x + y * z$	$E \Longrightarrow E * E$ $\Longrightarrow E + E * E$ $\Longrightarrow x + E * E$ $\Longrightarrow x + y * E$ $\Longrightarrow x + y * z$
CPSC 229: Foundations of Computation • Spring 2024		30

#### Parsing

- *parsing* refers to the process of finding a derivation for a string *w* using the rules of grammar *G*, or showing that no such derivation exists
  - that  $w \in L(G)$  is important for establishing that w is syntactically correct
  - finding a derivation for *w* is an important first step in semantic analysis

#### Parsing

CPSC 229: Foundations of Computation • Spring 2024

- in order to have an efficient parsing algorithm for a grammar, we need
  - an *unambiguous* grammar only a single derivation is possible
  - a deterministic process only one rule can be applicable in a given step

#### CPSC 229: Foundations of Computation • Spring 2024

### Left Derivations

CPSC 229: Foundations of Computation • Spring 2024

CPSC 229: Foundations of Computation • Spring 2024

- in general, there are many possible derivations for a string in *L*(*G*)
- in a *left derivation*, always replace the leftmost nonterminal in the next step

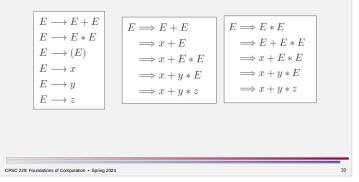
$E \longrightarrow E + E$	$E \Longrightarrow E + E$	$E \Longrightarrow E + E$
$E \longrightarrow E * E$	$\implies E + E * E$	$\Rightarrow x + E$
$E \longrightarrow (E)$	$\implies E + y * E$	$\implies x + E * E$
$ \begin{array}{c} E \longrightarrow x \\ E \longrightarrow y \end{array} $	$\implies x + y * E$	$\implies x + y * E$
$E \longrightarrow g$ $E \longrightarrow z$	$\implies x + y * z$	$\implies x + y * z$

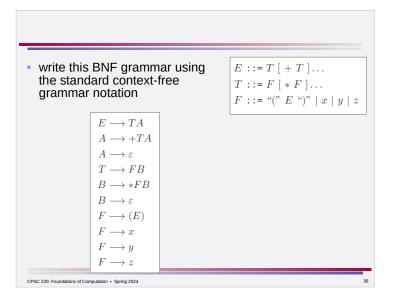
any string that has a derivation has a left derivation
the same rules are applied, it's just the order that is changed

# 1. Show that each of the following grammars is ambiguous by finding a string that has two left derivations according to the grammar: a) $S \rightarrow SS$ b) $S \rightarrow ASb$ $S \rightarrow aSb$ $S \rightarrow \varepsilon$ $S \rightarrow bSa$ $A \rightarrow aA$ $S \rightarrow \varepsilon$ $A \rightarrow a$

#### Ambiguous Grammars

• a context-free grammar *G* is *ambiguous* if there is a string  $w \in L(G)$  such that *w* has more than one left derivation according to *G* 



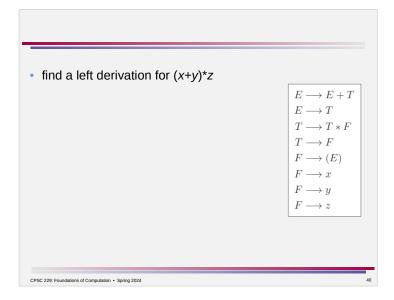


LL(1) Grammars	
<ul> <li>in an unambiguous grammar G, at each</li> </ul>	$E \longrightarrow TA$
step in a left derivation there's only one production that can be applied that will	$A \longrightarrow +TA$
lead to a correct derivation of w	$A \longrightarrow \varepsilon$
	$T \longrightarrow FB$
<ul> <li>G is an LL(1) grammar if that one production can be determined by (only)</li> </ul>	$B \longrightarrow *FB$
looking ahead to the next symbol in w	$B \longrightarrow \varepsilon$
-LL(1) means w is read Left-to-right and a Left	$F \longrightarrow (E)$
derivation is constructed by looking ahead at	$F \longrightarrow x$
most 1 character in w	$F \longrightarrow y$
<ul> <li>LL(1) grammars are nice in practice because it is straightforward to write a computer program to parse them</li> </ul>	$F \longrightarrow z$

CPSC 229: Foundations of Computation • Spring 2024

give a left deriva
E - A - A - A - A - T - B - B - F - F - F - F - F - F - F - F

<b>2.</b> Consider the string $z+(x)$ to each of the grammars		lerivation of this string according given in this section
to cach of the grammars	61, 62, and 63, as	given in this section.
$E \longrightarrow E + E$	$E \longrightarrow TA$	$E \longrightarrow E + T$
$E \longrightarrow E * E$	$A \longrightarrow +TA$	$E \longrightarrow T$
$E \longrightarrow (E)$	$A \longrightarrow \varepsilon$	$T \longrightarrow T * F$
$E \longrightarrow x$	$T \longrightarrow FB$	$T \longrightarrow F$
$E \longrightarrow y$	$B \longrightarrow *FB$	$F \longrightarrow (E)$
$E \longrightarrow z$	$B \longrightarrow \varepsilon$	$F \longrightarrow x$
	$F \longrightarrow (E)$	$F \longrightarrow u$
	$F \longrightarrow x$	$F \longrightarrow z$
	$F \longrightarrow y$	
	$F \longrightarrow z$	



## LR(1) Grammars

CPSC 229: Foundations of Computation • Spring 2024

- not all unambiguous context-free grammars are LL(1)
- *G* is an *LR*(1) grammar if it is always possible to tell which rule to apply at each step of the *right* derivation by (only) looking ahead to the next symbol in *w*