Shift-reduce example 1

- $L \rightarrow L\ ;\ E \mid E$
  $E \rightarrow id$

  Input: x; y

Action	Stack	Input
Sh		x ; y
R E→id	x	; y
R L→E	E	; y
Sh	L	; y
Sh	L ;	y
R E→id	L ; y	
R L→L;E	L ; E	
Accept	E	

Shift-reduce example 2

- $E \rightarrow E + T \mid T$
  $T \rightarrow T * P \mid P$
  $P \rightarrow id \mid int$

Input: x + 10 * y

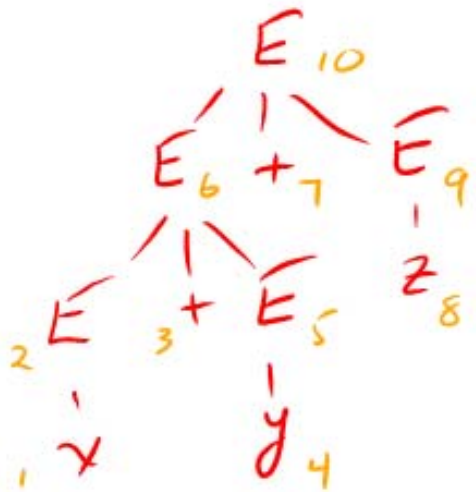
Action	Stack	Input
Sh		x + 10 * y
Red	x	+ 10 * y
Red	P	+ 10 * y
Red	T	+ 10 * y
Sh	E	+ 10 * y
Sh	E +	10 * y
Red	E + 10	* y
Red	E + P	* y
Sh	E + T	* y
Sh	E + T *	y
Red	E + T * y	
Red	E + T * P	
Red	E + T	
Acc	E	

Shift-reduce example 3

- **Grammar:** $E \rightarrow E + E \mid E * E \mid id$

  Input: x + y + z

Show a parse tree, and corresponding s/r parse, that represents left-associativity of addition.

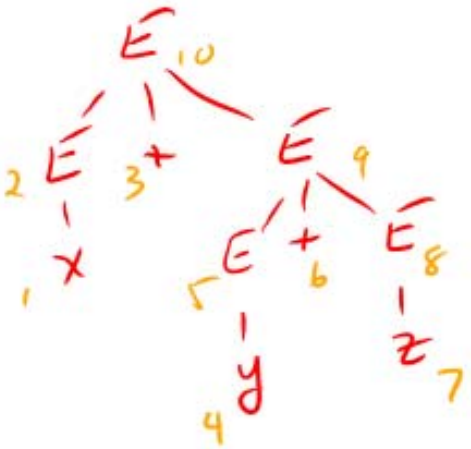


Action	Stack	Input
Sh		x+y+z
Red	x	+y+z
Sh	E	+y+z
Sh	E+	y+z
Red	E+y	+z
Red	E+E	+z
Sh	E	+z
Sh	E+	z
Red	E+z	
Red	E+E	
Acc	E	

Shift-reduce example 3 (cont.)

- **Grammar:** $E \rightarrow E + E \mid E * E \mid id$
  Input: `x + y + z`

Show a parse tree, and corresponding s/r parse, that represents right-associativity of addition.



Action	Stack	Input
Sh		x+y+z
Red	x	+y+z
Sh	E	+y+z
Sh	E+	y+z
Red	E+y	+z
Sh	E+E	+z
Sh	E+E+	z
Red	E+E+z	
Red	E+E+E	
Red	E+E	
Acc	E	

Dealing with ambiguity (cont.)

- For `x*y*z`, consider where the two stack configurations that can occur for the two parse trees differ. What is the correct decision?

Stack Input

E * E * z

shift → E * E *

red → E ⇐ correct - left-assoc. of mult

- Do the same for `x+y*z`:

E + E * z

shift → E + E * ⇐ correct - precedence of mult. over add.

red → E

- and for `x*y+z`:

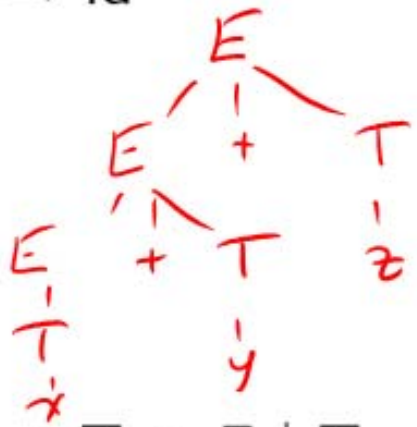
E * E + z

shift → E * E +

red → E ⇐ Correct - precedence of mult. over add.

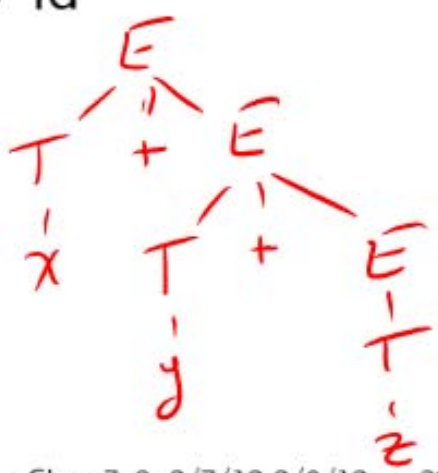
More examples of $SC(G)$

E → E + T | T
T → id



$SC(G) = \{ id, T, E, E+, E+id, E+T \}$

E → T + E | T
T → id



$SC(G) = \{ id, T, T+, T+id, T+T, T+T+, T+T+id, T+T+T, T+T+E, T+E, E \}$

In general,
$SC(G) = (T+)^*(id \mid T \mid E)$