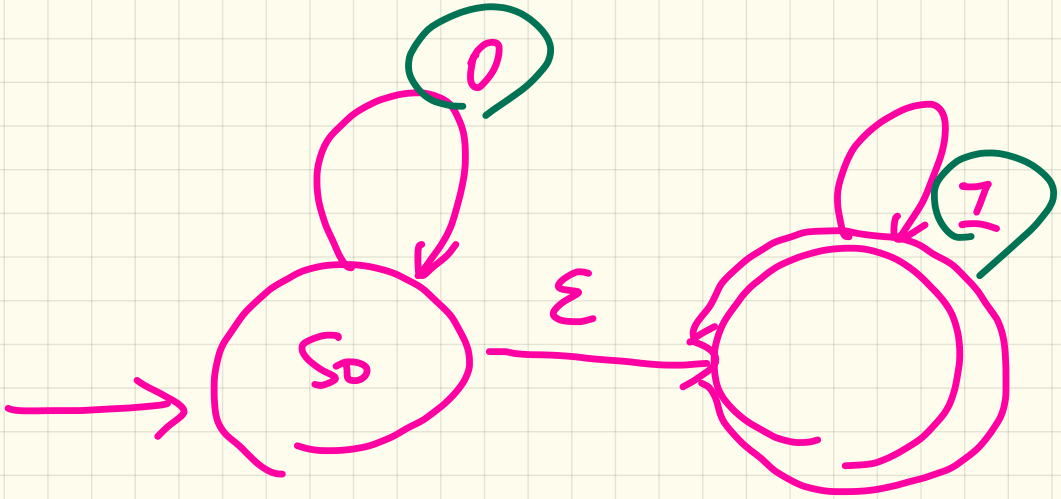
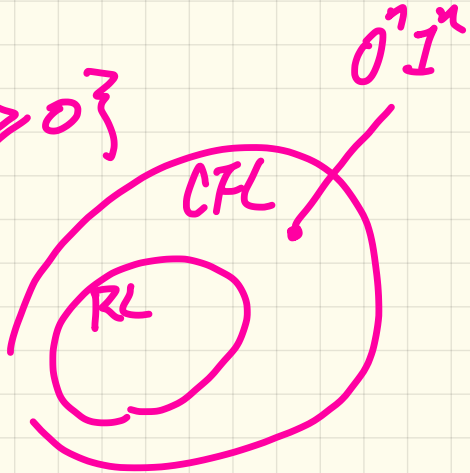
Lecture 6
Wednesday January 22



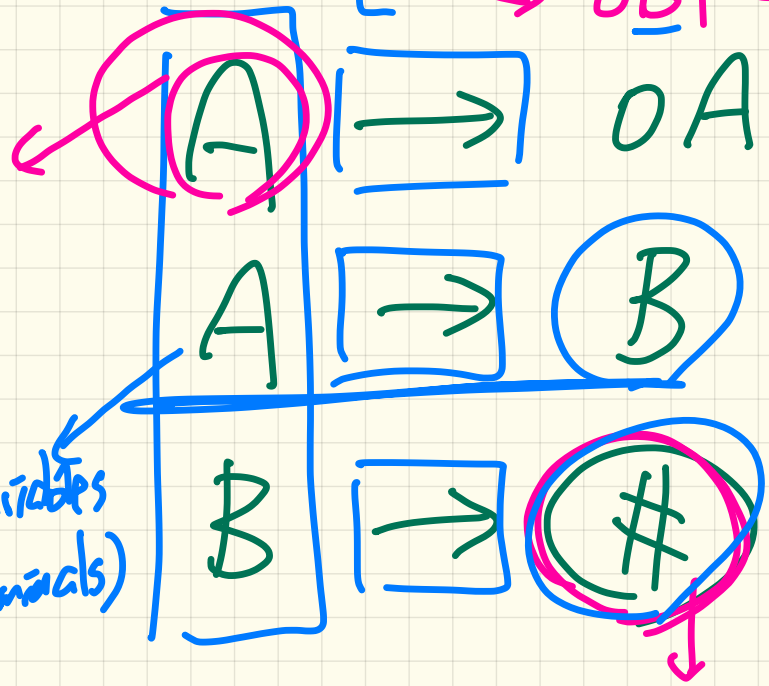
$\{0^n 1^n \mid n \geq 0\}$



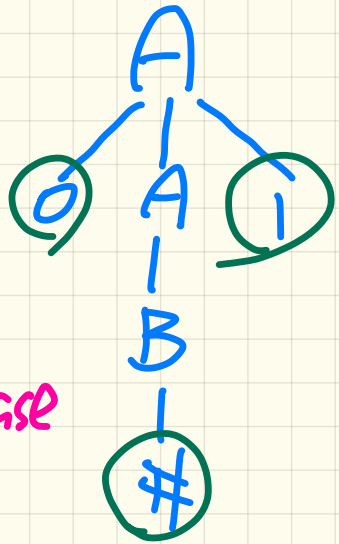
derivation

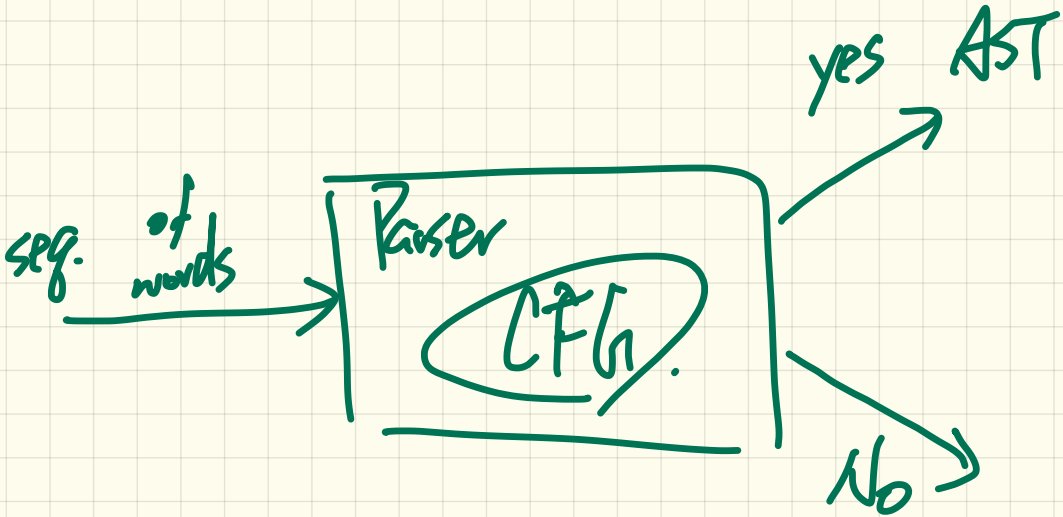
$A \Rightarrow 0A1$
$\Rightarrow 0B1 \Rightarrow 0\#1$

Start



$01\#\in L$





if (true }

L_1

$\{w \mid w \text{ is palindrome}\} =$

$\{ww^R \mid w \in \Sigma^*\}$ L_2

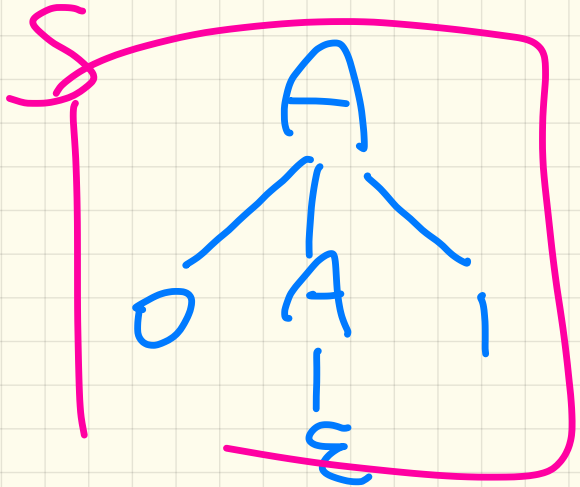
$1 \in L_1$

$1 \notin L_2$

A A A

$A \to 0|$

unnecessary.



true + false

3 ⇒ 4

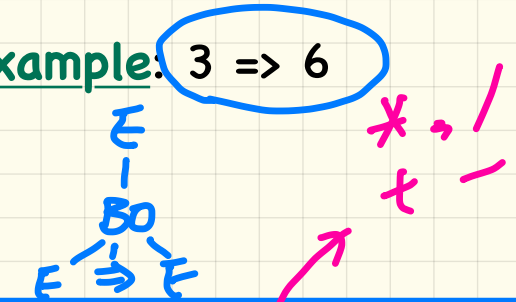
Context-Free Grammar (CFG): Example Version 1

Expression	→	*IntegerConstant*
	|	*BooleanConstant*
	|	*BinaryOp*
	|	*UnaryOp*
	|	(*Expression*)
IntegerConstant	→	*Digit*
	|	*Digit IntegerConstant*
	|	*−IntegerConstant*
Digit	→	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
BooleanConstant	→	`TRUE`
	|	`FALSE`

Example: 3 * 5 + 4 ambiguous

exercise: draw all possible parse trees for it.

Example: 3 => 6



BinaryOp	→	*Expression* + *Expression*
	|	*Expression* − *Expression*
	|	*Expression* * *Expression*
	|	*Expression* / *Expression*
	|	*Expression* && *Expression*
	|	*Expression* || *Expression*
	|	*Expression* => *Expression*
	|	*Expression* == *Expression*
	|	*Expression* /= *Expression*
	|	*Expression* > *Expression*
	|	*Expression* < *Expression*
UnaryOp	→	! *Expression*

Context-Free Grammar (CFG): Example Version 1

Expression	→	*IntegerConstant*
	|	*BooleanConstant*
	|	*BinaryOp*
	|	*UnaryOp*
	|	(*Expression*)
IntegerConstant	→	*Digit*
	|	*Digit IntegerConstant*
	|	– *IntegerConstant*
Digit	→	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
BooleanConstant	→	`TRUE`
	|	`FALSE`

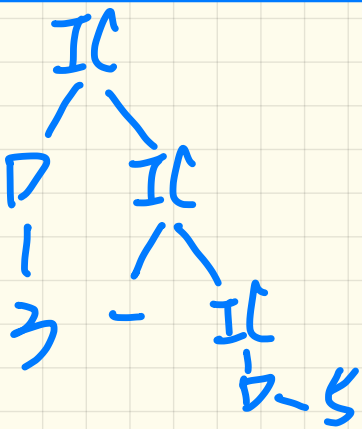
Example: 3 * 5 + 4

BinaryOp	→	*Expression* + *Expression*
	|	*Expression* – *Expression*
	|	*Expression* * *Expression*
	|	*Expression* / *Expression*
	|	*Expression* && *Expression*
	|	*Expression* || *Expression*
	|	*Expression* => *Expression*
	|	*Expression* == *Expression*
	|	*Expression* /= *Expression*
	|	*Expression* > *Expression*
	|	*Expression* < *Expression*
UnaryOp	→	! *Expression*

Context-Free Grammar (CFG): Example Version 2

Expression	→	*ArithmeticOp*
	|	*RelationalOp*
	|	*LogicalOp*
	|	(*Expression*)
IntegerConstant	→	*Digit*
	|	*Digit IntegerConstant*
	|	−*IntegerConstant*
Digit	→	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
BooleanConstant	→	`TRUE`
	|	`FALSE`

Example: (1 + 2) => (5 / 4)



ArithmeticOp	→	*ArithmeticOp* + *ArithmeticOp*
	|	*ArithmeticOp* − *ArithmeticOp*
	|	*ArithmeticOp* ⋆ *ArithmeticOp*
	|	*ArithmeticOp* / *ArithmeticOp*
	|	(*ArithmeticOp*)
	|	*IntegerConstant*
RelationalOp	→	*ArithmeticOp* == *ArithmeticOp*
	|	*ArithmeticOp* /= *ArithmeticOp*
	|	*ArithmeticOp* > *ArithmeticOp*
	|	*ArithmeticOp* < *ArithmeticOp*
LogicalOp	→	*LogicalOp* && *LogicalOp*
	|	*LogicalOp* || *LogicalOp*
	|	*LogicalOp* => *LogicalOp*
	|	! *LogicalOp*
	|	(*LogicalOp*)
	|	*RelationalOp*
	|	*BooleanConstant*

Context-Free Grammar (CFG): Example Version 2

Expression	→	*ArithmeticOp*
	|	*RelationalOp*
	|	*LogicalOp*
	|	(*Expression*)
IntegerConstant	→	*Digit*
	|	*Digit IntegerConstant*
	|	– *IntegerConstant*
Digit	→	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
BooleanConstant	→	`TRUE`
	|	`FALSE`

Example: (1 + 2) / (5 - (2 + 3))

↳ a valid string.

ArithmeticOp	→	*ArithmeticOp* + *ArithmeticOp*
	|	*ArithmeticOp* – *ArithmeticOp*
	|	*ArithmeticOp* * *ArithmeticOp*
	|	*ArithmeticOp* / *ArithmeticOp*
	|	(*ArithmeticOp*)
	|	*IntegerConstant*
RelationalOp	→	*ArithmeticOp* == *ArithmeticOp*
	|	*ArithmeticOp* /= *ArithmeticOp*
	|	*ArithmeticOp* > *ArithmeticOp*
	|	*ArithmeticOp* < *ArithmeticOp*
LogicalOp	→	*LogicalOp* && *LogicalOp*
	|	*LogicalOp* || *LogicalOp*
	|	*LogicalOp* => *LogicalOp*
	|	! *LogicalOp*
	|	(*LogicalOp*)
	|	*RelationalOp*
	|	*BooleanConstant*

Choice: report this as compilation error.

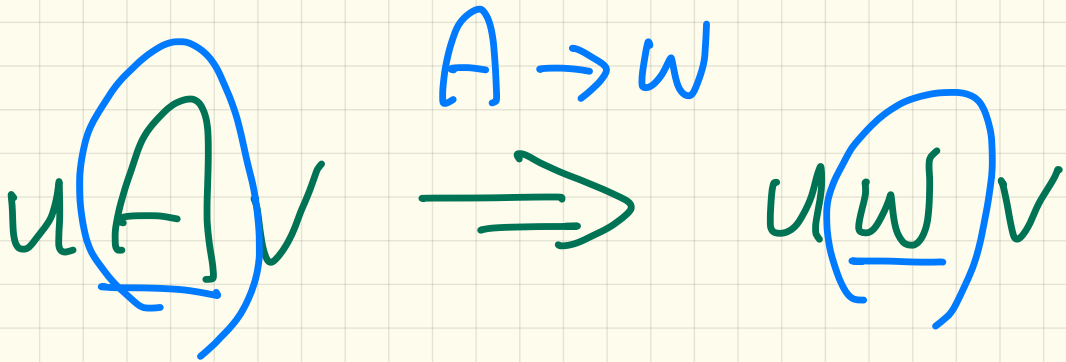
Context-Free Grammar (CFG): Example Version 2

Expression	→	*ArithmeticOp*
	|	*RelationalOp*
	|	*LogicalOp*
	|	(*Expression*)
IntegerConstant	→	*Digit*
	|	*Digit IntegerConstant*
	|	−*IntegerConstant*
Digit	→	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
BooleanConstant	→	`TRUE`
	|	`FALSE`

Example: 3 * 5 + 4

↳ exercise: parse trees

ArithmeticOp	→	*ArithmeticOp* + *ArithmeticOp*
	|	*ArithmeticOp* − *ArithmeticOp*
	|	*ArithmeticOp* ⋆ *ArithmeticOp*
	|	*ArithmeticOp* / *ArithmeticOp*
	|	(*ArithmeticOp*)
	|	*IntegerConstant*
RelationalOp	→	*ArithmeticOp* == *ArithmeticOp*
	|	*ArithmeticOp* /= *ArithmeticOp*
	|	*ArithmeticOp* > *ArithmeticOp*
	|	*ArithmeticOp* < *ArithmeticOp*
LogicalOp	→	*LogicalOp* && *LogicalOp*
	|	*LogicalOp* || *LogicalOp*
	|	*LogicalOp* => *LogicalOp*
	|	! *LogicalOp*
	|	(*LogicalOp*)
	|	*RelationalOp*
	|	*BooleanConstant*



$$S \to (S) \mid SS \mid \varepsilon$$

$$S()$$