Discussion: Compare Two CFGs

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`

v1

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*

v2

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 1

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

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`

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

Example: 3 * 5 + 4

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`

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

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

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

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`

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*

Q: No semantic analysis at all for Version 2 grammar?

Context-Free Grammar (CFG): Example Version 2

Example: 3 * 5 + 4

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

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 3

Expr	→	*Expr* + *Term*
	|	*Term*
Term	→	*Term* * *Factor*
	|	*Factor*
Factor	→	(*Expr*)
	|	a

Example: a * a + a

Context-Free Grammar (CFG): from RE (1)

RE	CFG
$L(\epsilon)$	
$L(a)$	
$L(E + F)$	
$L(EF)$	
$L(E^*)$	
$L((E))$	

Context-Free Grammar (CFG): from RE (2)

$(0 + 1)^*1(0+1)$

$(00 + 1)^* + (11 + 0)^*$

Context-Free Grammar (CFG): from DFA