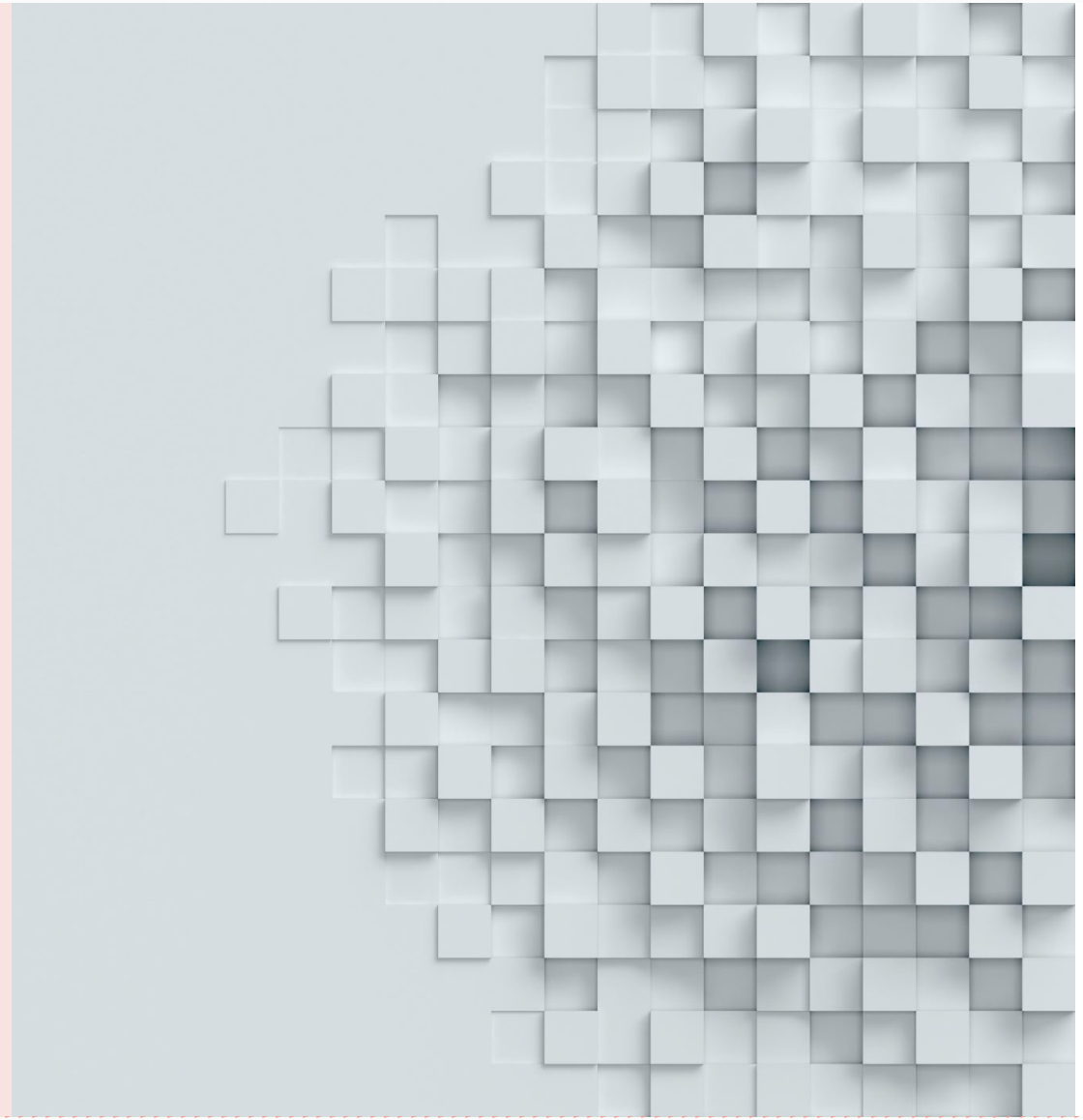


Análisis sintáctico usando PLY

Prof. Maureen Murillo
Teoría de la Computación
Escuela de Computación e Informática
Universidad de Costa Rica

Basado en Ply Documentation – Release 4.0

¿Qué cara tiene el parser de PLY?

- Se especifica como una gramática BNF (Backus-Naur Form). BNF es una manera de describir gramáticas libres de contexto. Usada ampliamente para especificar gramáticas de lenguajes de programación.

Ejemplo de gramática para expresiones aritméticas

```
expression : expression + term
           | expression - term
           | term

term       : term * factor
           | term / factor
           | factor

factor     : NUMBER
           | ( expression )
```

¿Qué más tiene el parser de PLY?

- Se puede especificar el comportamiento semántico mediante la técnica llamada "**traducción basada en sintaxis**" (sintax directed translation).

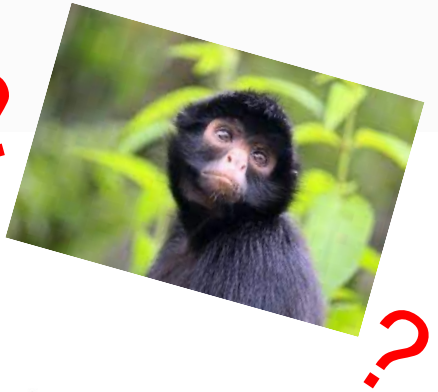
Se pueden ver como objetos con atributos

```
Grammar                                 Action
------------------------------------    ----------------------------------------
expression0 : expression1 + term        expression0.val = expression1.val + term.val
            | expression1 - term        expression0.val = expression1.val - term.val
            | term                      expression0.val = term.val

term0       : term1 * factor            term0.val = term1.val * factor.val
            | term1 / factor            term0.val = term1.val / factor.val
            | factor                    term0.val = factor.val

factor      : NUMBER                    factor.val = int(NUMBER.lexval)
            | ( expression )            factor.val = expression.val
```

¿Cómo hace PLY el *parseo*?



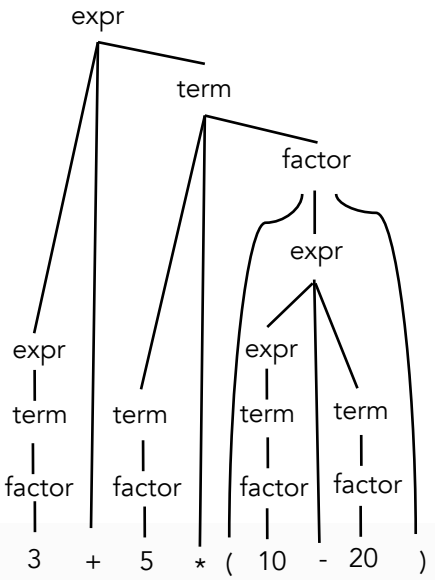
- Utiliza la técnica "LR-parsing" o "shift-reduce parsing".

Step	Symbol Stack	Input Tokens	Action
1		3 + 5 * (10 - 20)$	Shift 3
2	3	+ 5 * (10 - 20)$	Reduce factor : NUMBER
3	factor	+ 5 * (10 - 20)$	Reduce term : factor
4	term	+ 5 * (10 - 20)$	Reduce expr : term
5	expr	+ 5 * (10 - 20)$	Shift +
6	expr +	5 * (10 - 20)$	Shift 5
7	expr + 5	* (10 - 20)$	Reduce factor : NUMBER
8	expr + factor	* (10 - 20)$	Reduce term : factor
9	expr + term	* (10 - 20)$	Shift *
10	expr + term *	(10 - 20)$	Shift (
11	expr + term * (	10 - 20)$	Shift 10
12	expr + term * (10	- 20)$	Reduce factor : NUMBER
13	expr + term * (factor	- 20)$	Reduce term : factor
14	expr + term * (term	- 20)$	Reduce expr : term
15	expr + term * (expr	- 20)$	Shift -
16	expr + term * (expr -	20)$	Shift 20
17	expr + term * (expr - 20	)$	Reduce factor : NUMBER
18	expr + term * (expr - factor	)$	Reduce term : factor
19	expr + term * (expr - term	)$	Reduce expr : expr - term
20	expr + term * (expr	)$	Shift)
21	expr + term * (expr)	$	Reduce factor : (expr)
22	expr + term * factor	$	Reduce term : term * factor
23	expr + term	$	Reduce expr : expr + term
24	expr	$	Reduce expr
25		$	Success!

```
expression : expression + term
           | expression - term
           | term

term       : term * factor
           | term / factor
           | factor

factor     : NUMBER
           | ( expression )
```



Bottom-up

✅

Ejemplo de código en PLY

```
def p_expression_plus(p):
    'expression : expression PLUS term'
    #    ^            ^        ^    ^
    #  p[0]         p[1]     p[2] p[3]

    p[0] = p[1] + p[3]
```

```
def p_expression_plus(p):
    'expression : expression PLUS term'
    p[0] = p[1] + p[3]

def p_expression_minus(p):
    'expression : expression MINUS term'
    p[0] = p[1] - p[3]

def p_expression_term(p):
    'expression : term'
    p[0] = p[1]

def p_term_times(p):
    'term : term TIMES factor'
    p[0] = p[1] * p[3]

def p_term_div(p):
    'term : term DIVIDE factor'
    p[0] = p[1] / p[3]

def p_term_factor(p):
    'term : factor'
    p[0] = p[1]

def p_factor_num(p):
    'factor : NUMBER'
    p[0] = p[1]

def p_factor_expr(p):
    'factor : LPAREN expression RPAREN'
    p[0] = p[2]
```

Podemos caer en gramáticas ambiguas

No ambigua

```
expression : expression + term
           | expression - term
           | term

term       : term * factor
           | term / factor
           | factor

factor     : NUMBER
           | ( expression )
```

Ambigua pero más natural

```
expression : expression PLUS expression
           | expression MINUS expression
           | expression TIMES expression
           | expression DIVIDE expression
           | LPAREN expression RPAREN
           | NUMBER
```

```
precedence = (
    ('left', 'PLUS', 'MINUS'),
    ('left', 'TIMES', 'DIVIDE'),
)
```

¡Solución en PLY!

fin