

Cómo hacer un *parser* en Go

(y no morir en el intento)



Edgardo Hames - ehames@bitlogic.io



BITLOGIC.io

2 problemas y 1 concepto

Problema 1: Opciones de Programa

```
./myprog -i 12 -m 10
```

```
import "flag"
```

```
func main() {  
    var iterations, maxThreads int  
    flag.IntVar(&iterations, "i", 100, "number of iterations")  
    flag.IntVar(&maxThreads, "m", 4, "max number of threads")  
  
    ...  
    flag.Parse()  
    ...  
}
```

Problema 2: Unmarshaling JSON

```
type Person struct {  
    ID string `json:"id"`  
    Name string `json:"name"`  
}
```



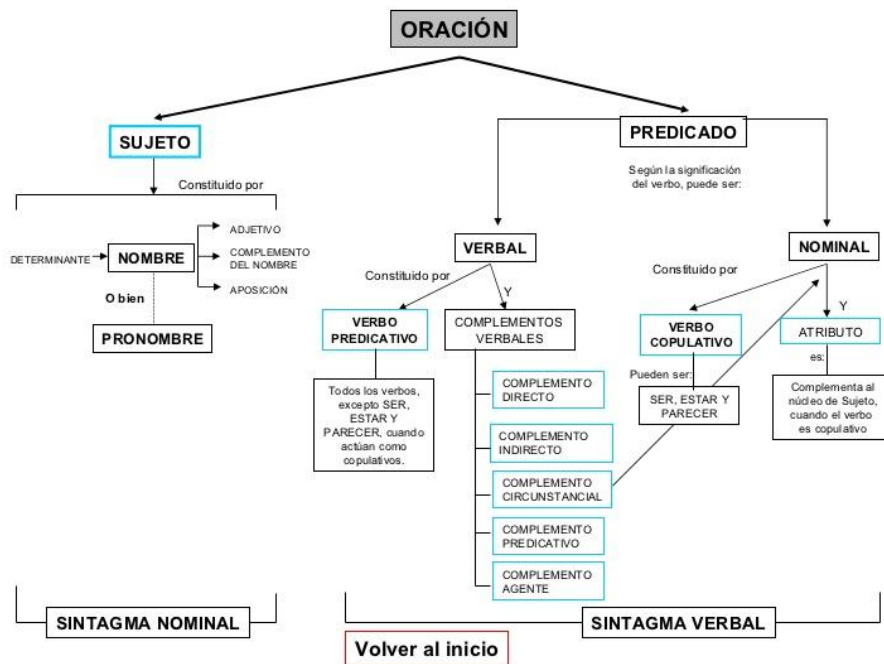
```
p := Person{ID: "123", Name: "Noam"}
```

```
{  
  "id": "123",  
  "name": "Noam"  
}
```

¿Qué es un *parser*?

Análizador sintáctico de una *cadena de símbolos* conforme a un *conjunto de reglas gramaticales*.

Del latín *pars orationis* (categorías gramaticales)



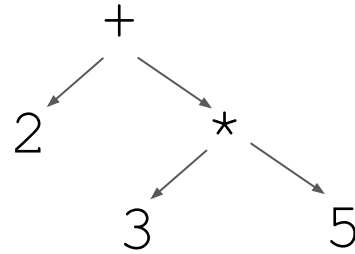
Otros problemas que involucran análisis sintáctico

- Evaluación de expresiones
 - "2 +3 *5"
- Compilación de un programa
- Interpretación de scripts
 - Ruby, Python, Perl
 - Maven, Gradle
 - Gherkin (Godog, Jbehave, etc),

"2 +3 *5"

Los espacios
blancos son
basura

 Parser



Análisis lexicográfico (lexer)

Convierte cadena de entrada en secuencia de *tokens*.

"2 +3 *5"



Token	Categoría
2	Literal entero
+	Operador adición
3	Literal entero
*	Operador multiplicación
5	Literal entero

"2 +3 *5"

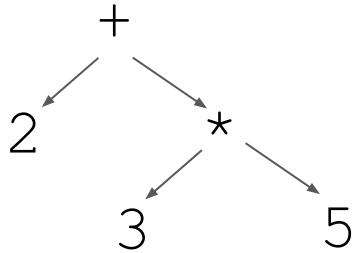


Lexer

["2", "+", "3", "*", "5"]



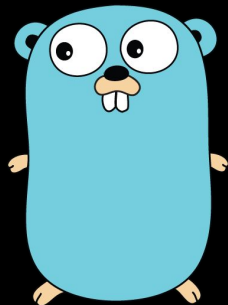
Parser



lex & yacc

Cómo hacer un *lexer* y un *parser* en Go

(y no morir en el intento)



Edgardo Hames - ehames@bitlogic.io



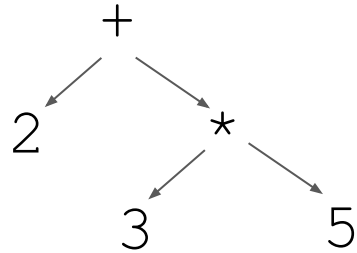
BITLOGIC.io

"2 +3 *5"

↓ Lexer

["2", "+", "3", "*", "5"]

↓ Parser



↓ Evaluador

30

Cómo hacer un *lexer*, un *parser* y un evaluador en Go

(y no morir en el intento)



Edgardo Hames - ehames@bitlogic.io



BITLOGIC.io

CUANDO POR FIN



**PUDISTE PONER EL
TITULO QUE QUERIAS**

Teoría

$$t \rightarrow W^+ b$$

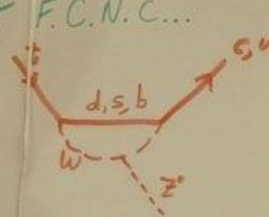
$$BR(t \rightarrow Wb) = \frac{\Gamma(t \rightarrow Wb)}{\Gamma(t \rightarrow Wq)}$$

$$= \frac{|V_{tb}|^2}{|V_{td}|^2 + |V_{ts}|^2 + |V_{tb}|^2}$$

$$\approx \frac{(0.9745)^2}{(0.0094)^2 + (0.040)^2 + (0.9745)^2} = 99.82\%$$

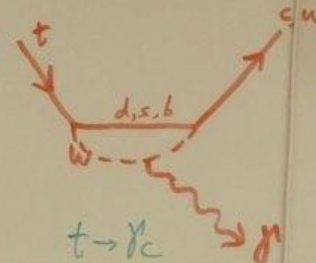


but F.C.N.C...



$$t \rightarrow Zc$$

$$t \rightarrow Zn$$



$$t \rightarrow Wc$$

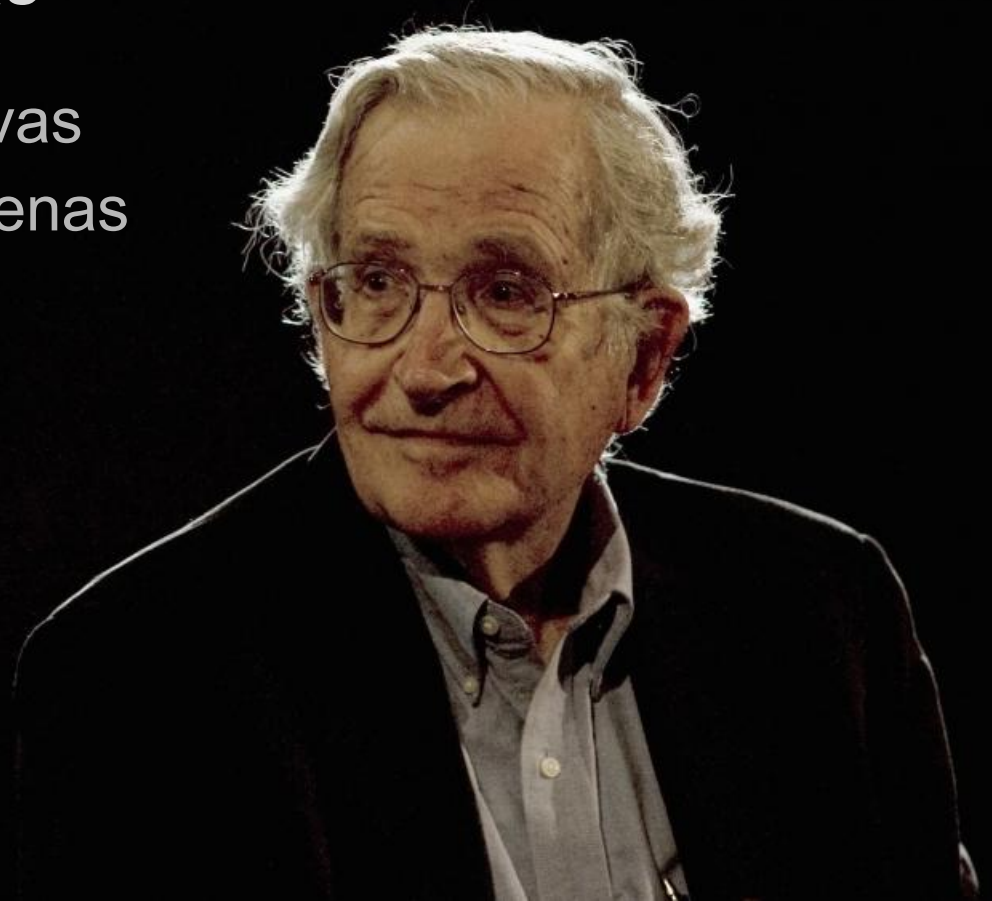
$$t \rightarrow Wn$$

$$\Gamma = \begin{pmatrix} C_{12} C_{13} & \dots \end{pmatrix}$$

Gramáticas Generativas

Conjunto de reglas recursivas
que *generan* todas las cadenas
de un lenguaje.

Chomsky, 1956

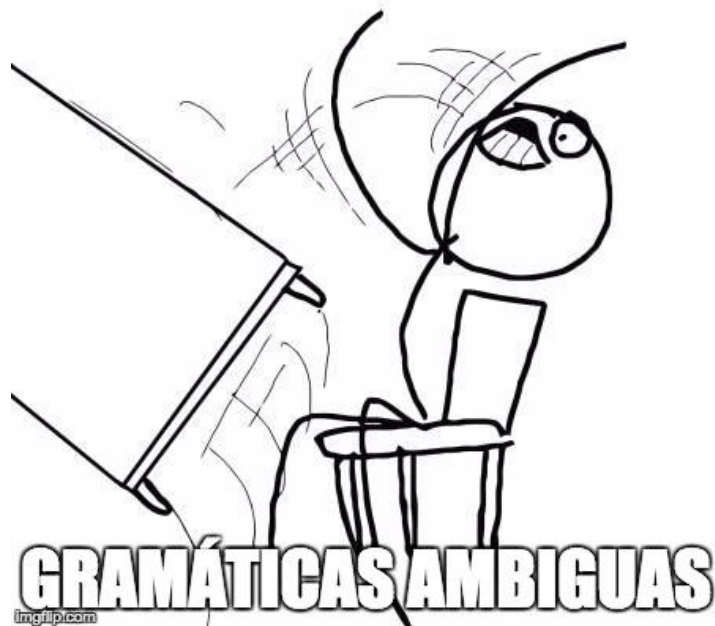


Ejemplo de Gramática Generativa

```
ALT  <- IF cond THEN statement |  
      IF cond THEN statement ELSE statement  
IF   <- "if"  
THEN <- "then"  
ELSE <- "else"  
...
```


¿Es válido el siguiente programa?

```
if  (a > 3) then  
    b := true  
else  
    b := false
```



¿Cómo eliminar la ambigüedad?

- Meta-reglas
 - “Si se puede interpretar como una sentencia o una definición, preferir la definición” (C++)
 - “Preferir la coincidencia más larga” (Haskell)

A scene from the movie Toy Story featuring Woody and Buzz Lightyear. Woody, on the left, is a cowboy doll with a yellow and black plaid shirt, a cow-print vest, and a large gold belt buckle. He has a worried expression. Buzz Lightyear, on the right, is a space ranger doll in his iconic green, white, and purple suit. He is gesturing with his right hand, which has purple finger-like appendages. A speech bubble originates from his hand, containing Spanish text. The background is a simple room with a wooden floor and a white door.

El tiempo de ejecución
puede ser exponencial en
el peor caso.



Gramáticas Analíticas

Conjunto de reglas que *deciden* si una cadena pertenece a un lenguaje.

Parsing Expression Grammars

- Son gramáticas analíticas
- Tienen notación similar a EBNF
- Introducen el operador de elección con prioridad /
- Resuelven el problema de *lexing* y *parsing*

Bryan Ford, 2004



Reglas

- ϵ \Rightarrow Reconoce la cadena vacía.
- a \Rightarrow Reconoce el símbolo terminal a .
- A \Rightarrow Reconoce el símbolo no terminal A .
- $e_1 e_2$ \Rightarrow Reconoce e_1 seguida inmediatamente de e_2 .
- e_1 / e_2 \Rightarrow Reconoce e_1 . Si falla, intenta con e_2 .
- e^* \Rightarrow Reconoce cero o más repeticiones de e .
- $!e$ \Rightarrow Reconoce que no sigue la expresión e (no mueve el punto de lectura).

Hay varias reglas más que son *syntactic sugar* de éstas. Por ej. e^+

PEG al rescate

```
alt  <- IF cond THEN statement ELSE statement /  
      IF cond THEN statement  
IF    <- "if"  
THEN  <- "then"  
ELSE  <- "else"  
...  
```

PEG => Packrat Parser

Parser en tiempo lineal
que usa memoization







Iniciar Sesión

Registro Gratuito

Todas las Capacitaciones

Campus Virtual

Menú de Secciones

Bienvenidos al (G-SE) Grupo Sobre Entrenamiento

Líder Mundial en Información y
Capacitación a Distancia en Ciencias
del Ejercicio y Salud.



Próximas Capacitaciones

Cursos, webinars, talleres, posgrados y másteres online en ciencias del ejercicio y medicina del deporte.

Búsqueda de documentos (símil Google)

$w_1 w_2 \dots w_n \Rightarrow$ encuentra los textos que contengan **alguna** palabra

$"w_1 w_2 \dots" \Rightarrow$ encuentra los textos que contengan **todas** las palabras

$-w_1 \Rightarrow$ encuentra los textos que **no contengan** w_1

Implementación de PEG en Go

```
$ go get github.com/pointlander/peg
```

```
$ peg -inline -switch expression/expression.peg
```

A group of five people are gathered around a computer monitor in a dimly lit room. A man with long hair and glasses is seated at the desk, looking at the screen. Four other people (three men and one woman) stand behind him, all looking intently at the monitor. The woman on the far left has her arms crossed. The man next to her is holding his head. The man in the center background is holding a bottle. The man in the foreground right is wearing a striped shirt. The man on the far right is wearing a plaid shirt. The background features a stone fireplace and shelves with books and decorative items.

Show me the Code

Conclusiones

- El problema del análisis sintáctico no tiene solución trivial
- Las PEGs proveen una solución eficiente
- Hay implementaciones de PEG en la mayoría de los lenguajes

¿Preguntas?



Agradecimientos

- Gustavo Burgi (G-SE)
- Federico Aguirre (Bitlogic)