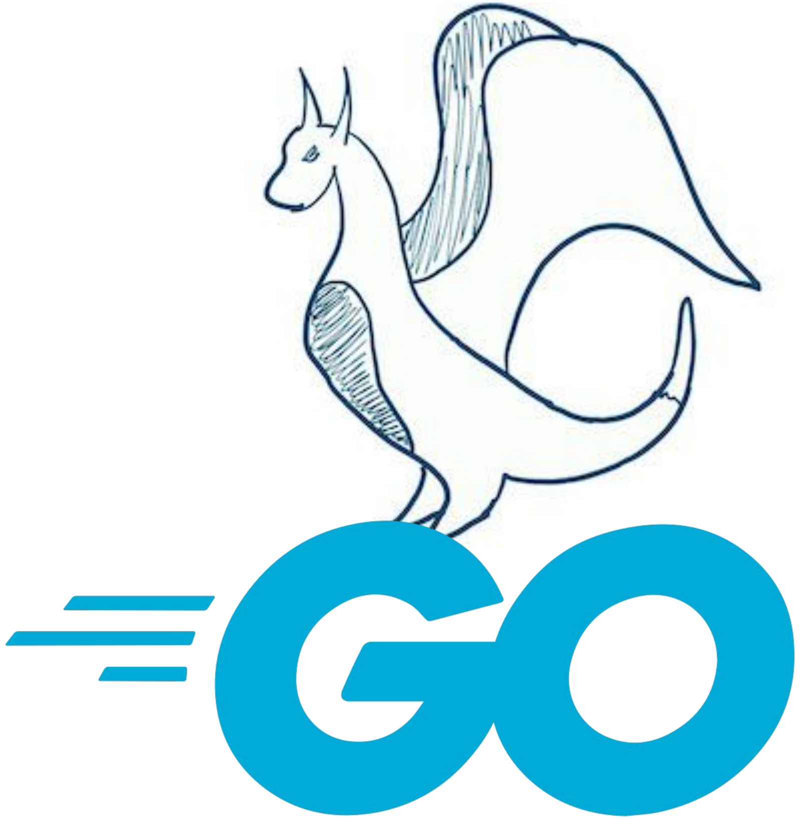
Compilador para Go

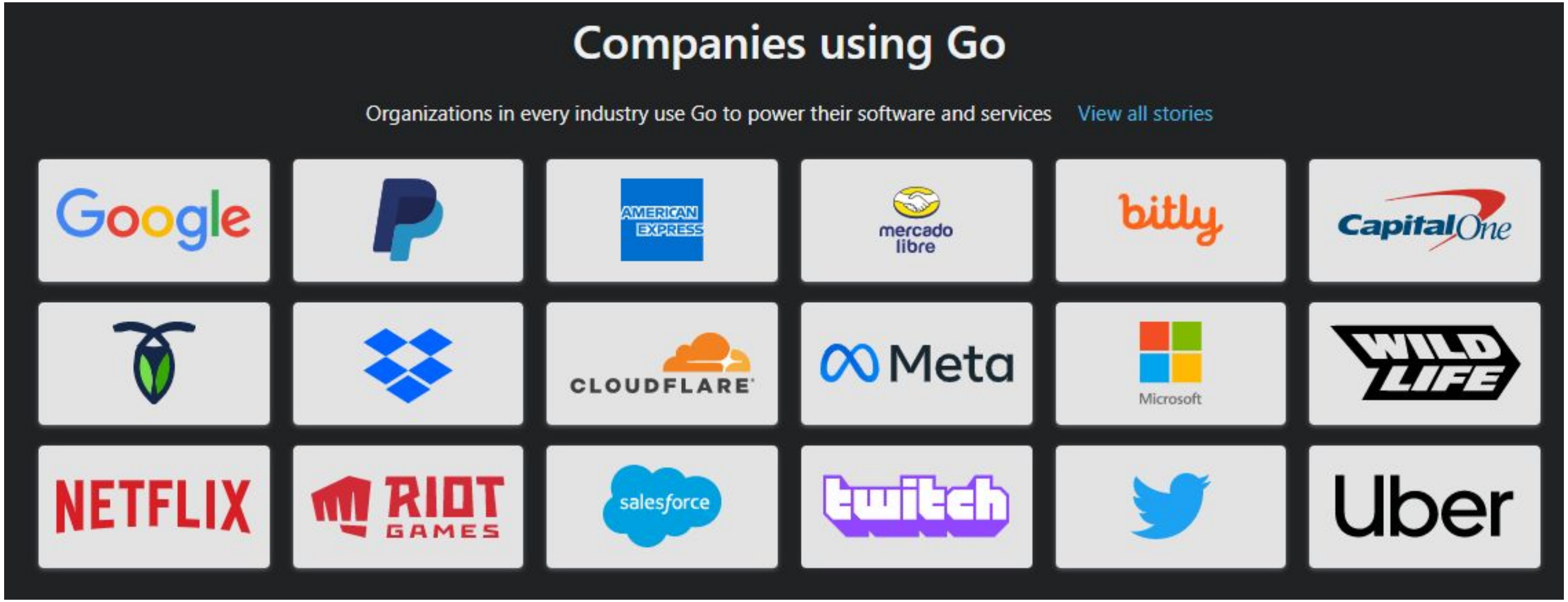
Juan Lara & Santiago Jimenez

Introducción

- Un lenguaje de programación de código abierto apoyado por Google.
- Concurrente, compilado, estructurado, orientado a objetos y con tipado estático.
- Inspirado en C (i386, amd64 y ARM)
- Seguridad de memoria y recolección de basura.
- Creado en **2009** por:

  Robert Griesemer, **Rob Pike** y **Ken Thompson**





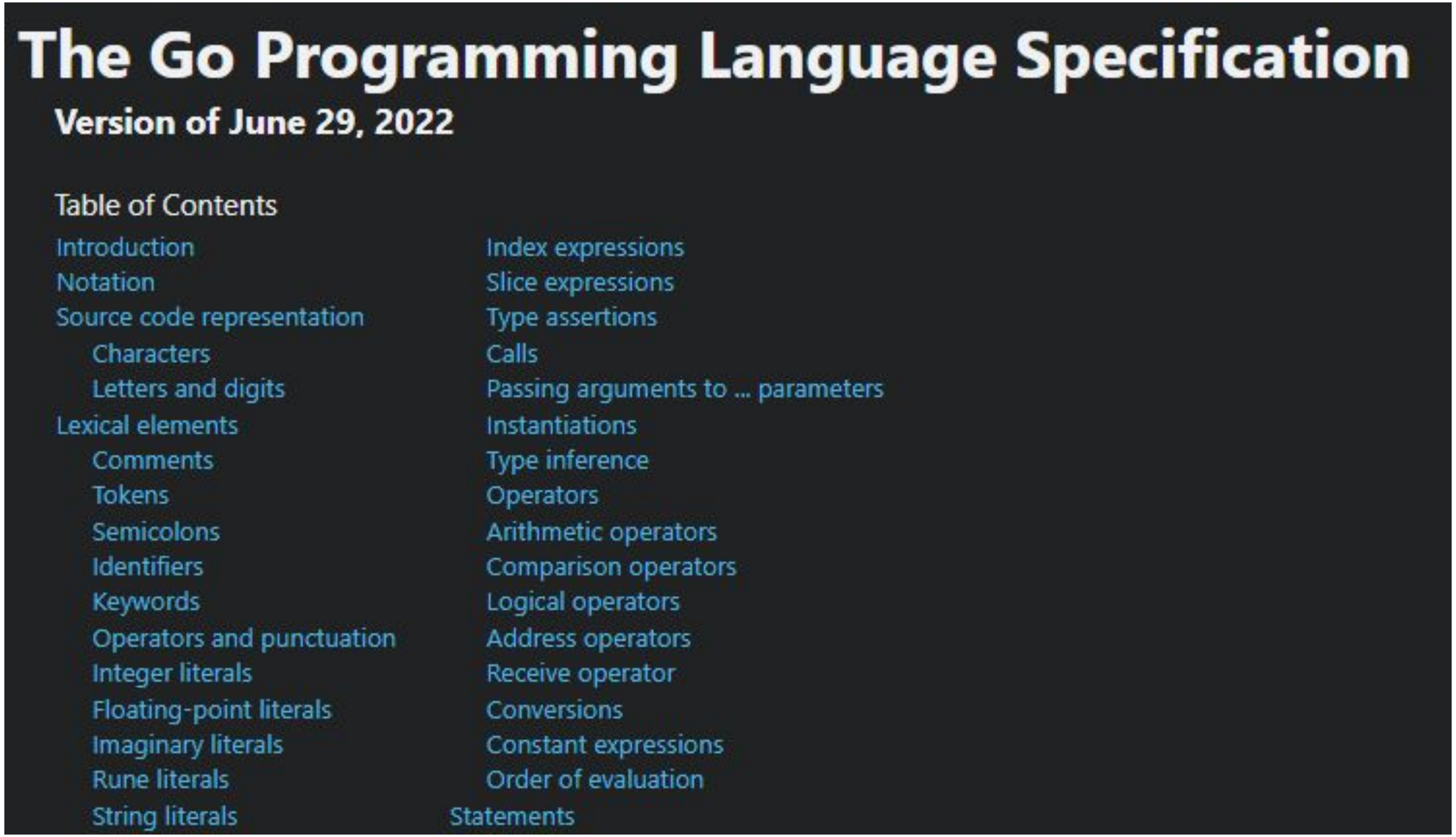
Companies using Go

Organizations in every industry use Go to power their software and services   View all stories

¿Todo

eso?

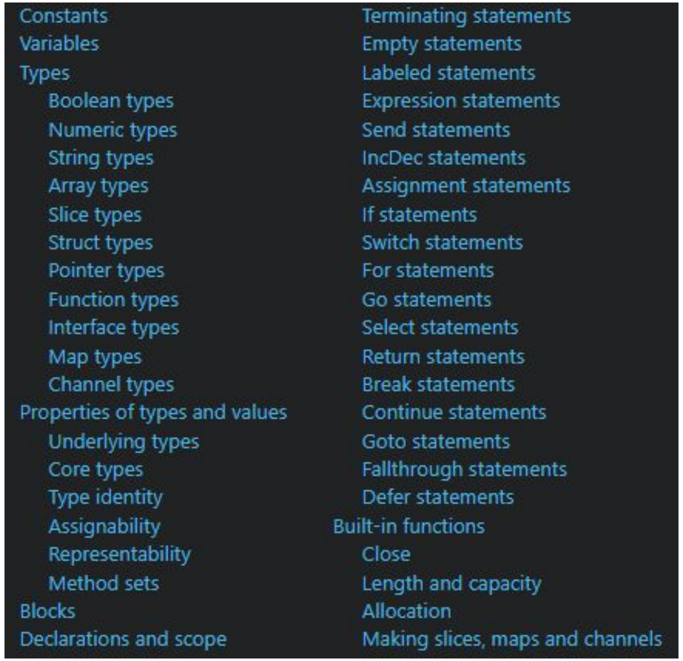
No



The Go Programming Language Specification

Version of June 29, 2022

Table of Contents



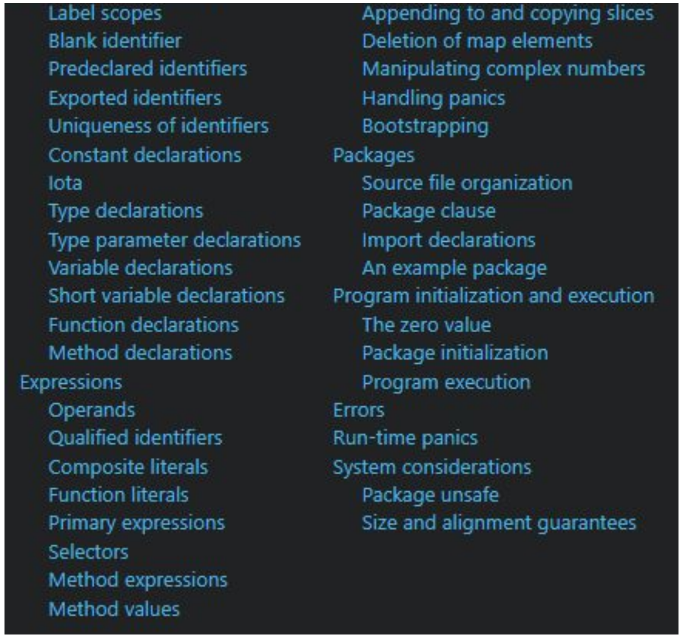
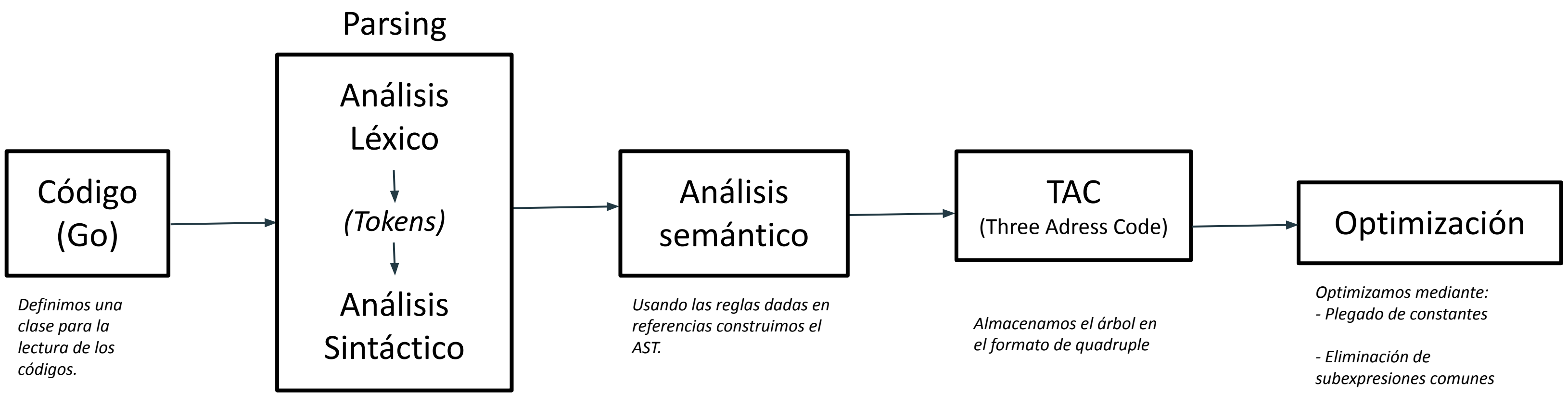


Diagrama de solución



Definimos una clase para la lectura de los códigos.

Obtenemos los tokens del análisis léxico y luego el resultado lo corremos para realizar el análisis sintáctico usando la gramática dada.

Usando las reglas dadas en referencias construimos el AST.

Almacenamos el árbol en el formato de quadruple

Optimizamos mediante:
- Plegado de constantes

- Eliminación de subexpresiones comunes

- Empaquetado temporal (para reducir la longitud y complejidad del Código del árbol)

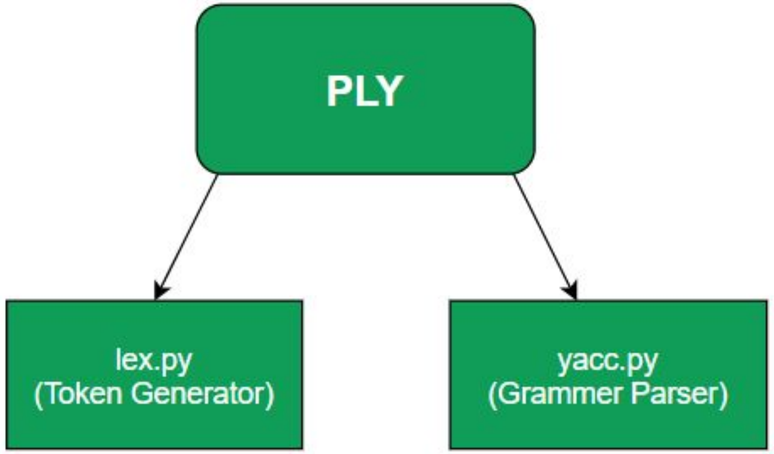
Basado en referencias en las cuales se define la gramática.

Para ello usamos

- Referencias:

  Go (CFG, reglas que definen el lenguaje).

- PLY (Python Lex-Yacc)



Análisis léxico

Análisis sintácticos

Acerca del parser

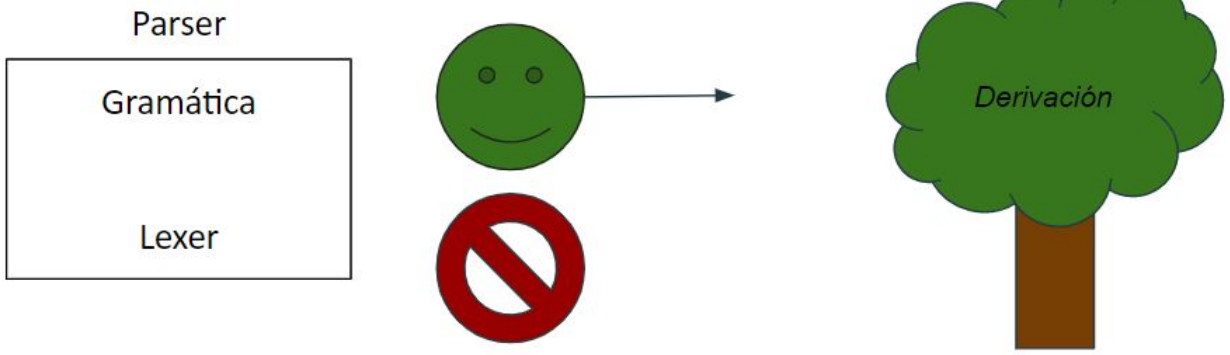


Tabla de Parser

LALR(1)

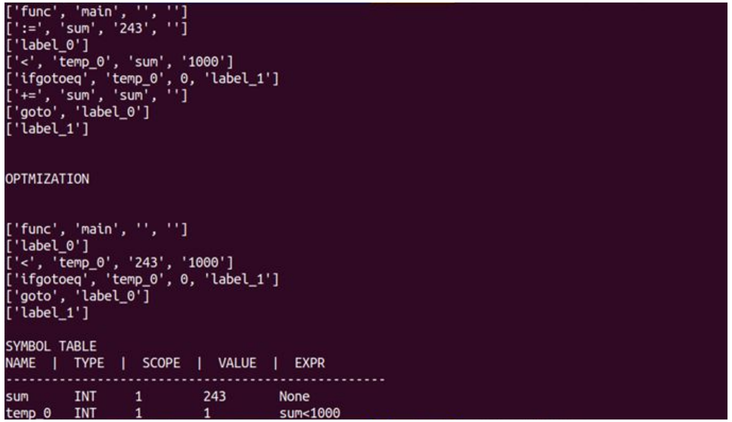
Ejemplo

.Go

```go
package main

import "fmt"

func main() {
     sum := 243
     for sum < 1000 {
          sum += sum
     }
}
```

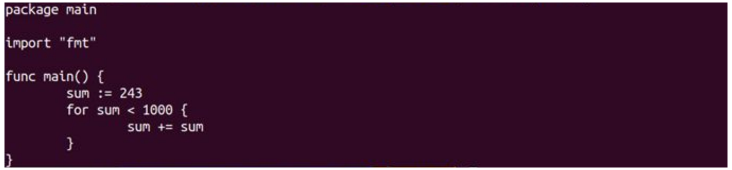


```
['func', 'main', '', '']
[':=', 'sum', '243', '']
['label_0']
['<', 'temp_0', 'sum', '1000']
['ifgotoeq', 'temp_0', 0, 'label_1']
['+=', 'sum', 'sum', '']
['goto', 'label_0']
['label_1']


OPTMIZATION


['func', 'main', '', '']
['label_0']
['<', 'temp_0', '243', '1000']
['ifgotoeq', 'temp_0', 0, 'label_1']
['goto', 'label_0']
['label_1']

SYMBOL TABLE
NAME  |  TYPE  |  SCOPE  |  VALUE  |  EXPR
--------------------------------------------------
sum      INT      1         243       None
temp_0   INT      1         1         sum<1000
```



```
package main

import "fmt"

func main() {
        sum := 243
        for sum < 1000 {
                sum += sum
        }
}
```

Referencias

1. Aho, A. V., & Aho, A. V. (Eds.). (2007). Compilers: Principles, techniques, & tools (2nd ed). Pearson/Addison Wesley.
2. Aho, A. V., Sethi, R., & Ullman, J. D. (1986). Compilers, principles, techniques, and tools. Addison-Wesley Pub. Co.
3. Codewalk: First-class functions in go - the go programming language. (n.d.). Retrieved 3 December 2022, from https://go.dev/doc/codewalk/functions/
4. Go—Operators precedence. (n.d.). Retrieved 3 December 2022, from https://www.tutorialspoint.com/go/go_operators_precedence.htm
5. PLY (Python lex-yacc)—Ply 4.0 documentation. (n.d.). Retrieved 3 December 2022, from https://ply.readthedocs.io/en/latest/
6. The go programming language specification—The go programming language. (n.d.). Retrieved 3 December 2022, from https://go.dev/ref/spec
7. Three address code in Compiler. (2018, May 21). GeeksforGeeks. https://www.geeksforgeeks.org/three-address-code-compiler/