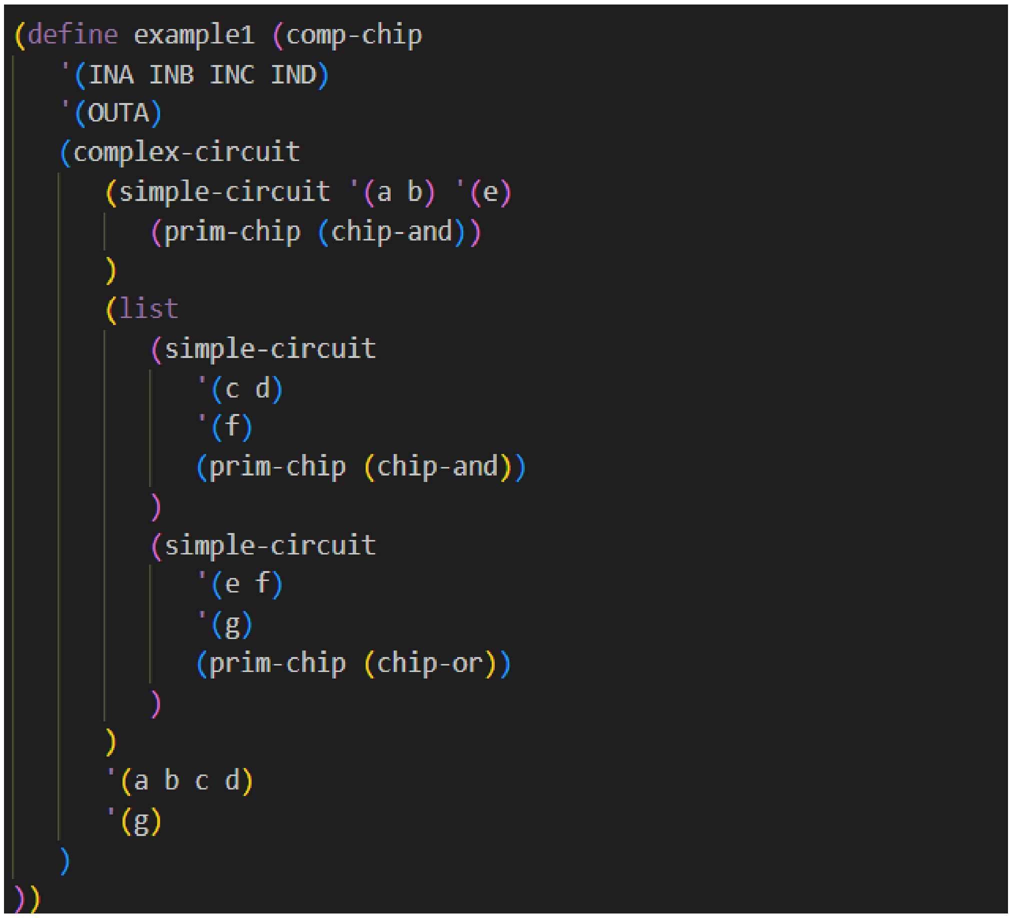
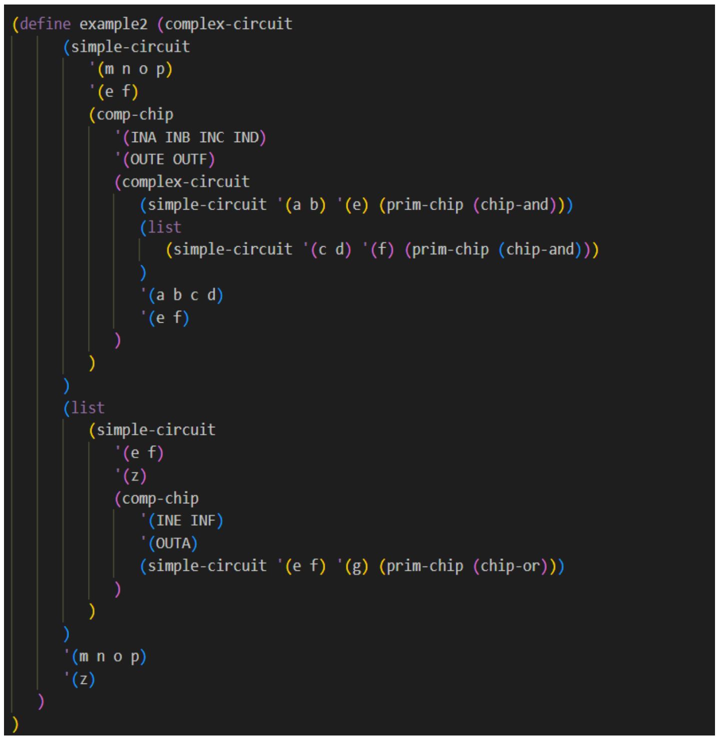
Se realiza el documento con las pruebas de cada archivo correspondiente

Representacion-datatype.rkt

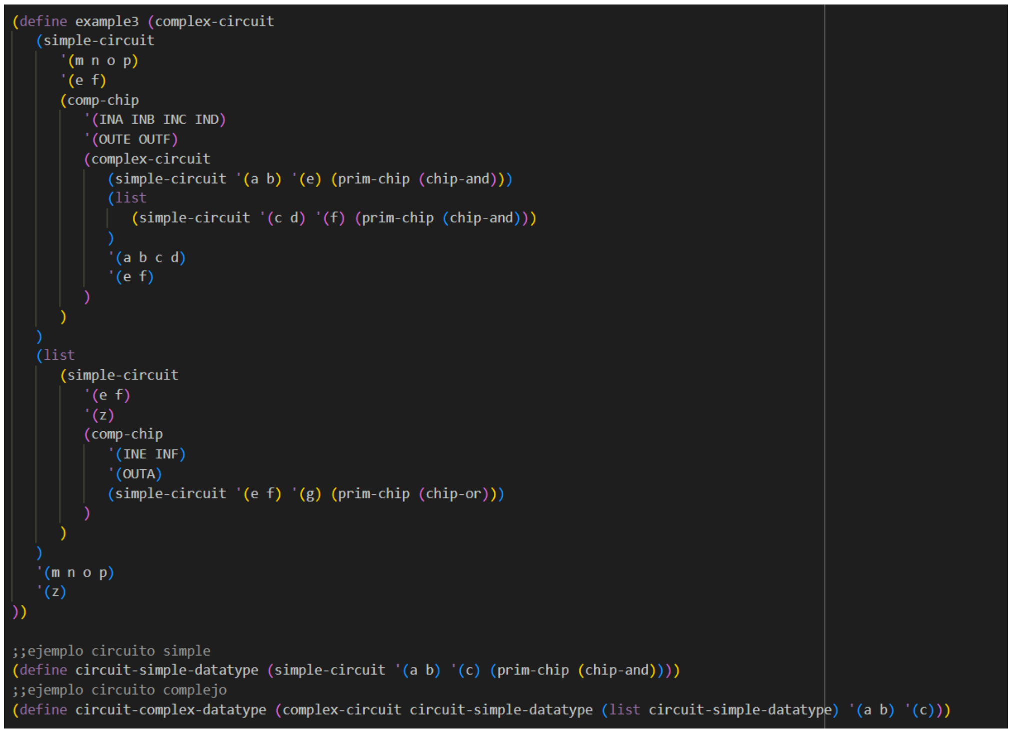
Se definieron algunos ejemplos y se mostrará el resultado de estos



```
(define example1 (comp-chip
    '(INA INB INC IND)
    '(OUTA)
    (complex-circuit
        (simple-circuit '(a b) '(e)
            (prim-chip (chip-and))
        )
        (list
            (simple-circuit
                '(c d)
                '(f)
                (prim-chip (chip-and))
            )
            (simple-circuit
                '(e f)
                '(g)
                (prim-chip (chip-or))
            )
        )
        '(a b c d)
        '(g)
    )
))
```

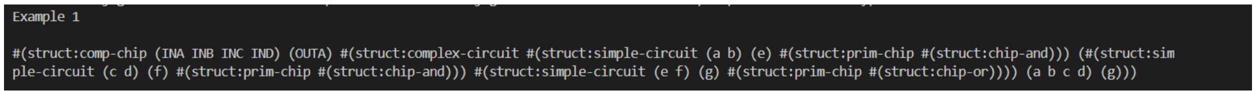


```
(define example2 (complex-circuit
      (simple-circuit
         '(m n o p)
         '(e f)
         (comp-chip
            '(INA INB INC IND)
            '(OUTE OUTF)
            (complex-circuit
               (simple-circuit '(a b) '(e) (prim-chip (chip-and)))
               (list
                  (simple-circuit '(c d) '(f) (prim-chip (chip-and)))
               )
               '(a b c d)
               '(e f)
            )
         )
      )
      (list
         (simple-circuit
            '(e f)
            '(z)
            (comp-chip
               '(INE INF)
               '(OUTA)
               (simple-circuit '(e f) '(g) (prim-chip (chip-or)))
            )
         )
      )
      '(m n o p)
      '(z)
   )
)
```

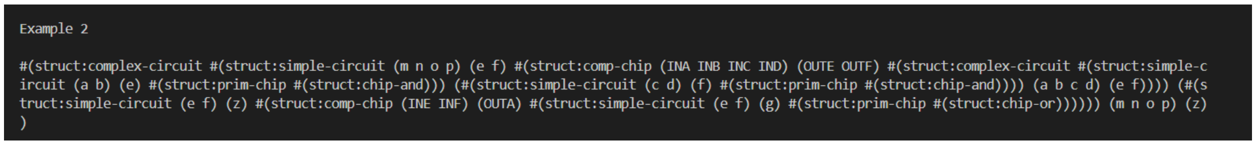


```
(define example3 (complex-circuit
   (simple-circuit
      '(m n o p)
      '(e f)
      (comp-chip
         '(INA INB INC IND)
         '(OUTE OUTF)
         (complex-circuit
            (simple-circuit '(a b) '(e) (prim-chip (chip-and)))
            (list
               (simple-circuit '(c d) '(f) (prim-chip (chip-and)))
            )
            '(a b c d)
            '(e f)
         )
      )
   )
   (list
      (simple-circuit
         '(e f)
         '(z)
         (comp-chip
            '(INE INF)
            '(OUTA)
            (simple-circuit '(e f) '(g) (prim-chip (chip-or)))
         )
      )
   )
   '(m n o p)
   '(z)
))

;;ejemplo circuito simple
(define circuit-simple-datatype (simple-circuit '(a b) '(c) (prim-chip (chip-and))))
;;ejemplo circuito complejo
(define circuit-complex-datatype (complex-circuit circuit-simple-datatype (list circuit-simple-datatype) '(a b) '(c)))
```

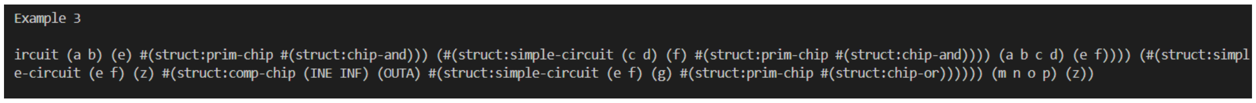
Y los resultados fueron los siguientes:



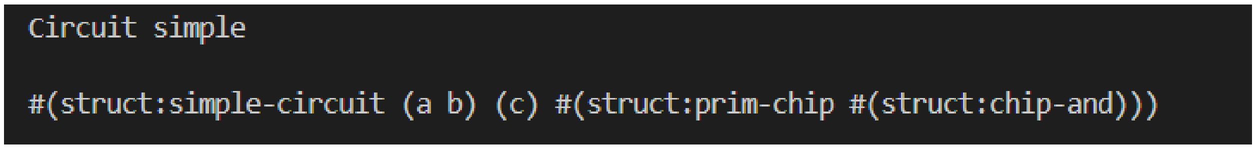
```
Example 1

#(struct:comp-chip (INA INB INC IND) (OUTA) #(struct:complex-circuit #(struct:simple-circuit (a b) (e) #(struct:prim-chip #(struct:chip-and))) (#(struct:sim
ple-circuit (c d) (f) #(struct:prim-chip #(struct:chip-and))) #(struct:simple-circuit (e f) (g) #(struct:prim-chip #(struct:chip-or)))) (a b c d) (g)))
```



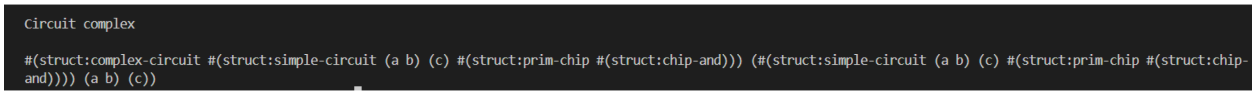
```
Example 2

#(struct:complex-circuit #(struct:simple-circuit (m n o p) (e f) #(struct:comp-chip (INA INB INC IND) (OUTE OUTF) #(struct:complex-circuit #(struct:simple-c
ircuit (a b) (e) #(struct:prim-chip #(struct:chip-and))) (#(struct:simple-circuit (c d) (f) #(struct:prim-chip #(struct:chip-and)))) (a b c d) (e f)))) (#(s
truct:simple-circuit (e f) (z) #(struct:comp-chip (INE INF) (OUTA) #(struct:simple-circuit (e f) (g) #(struct:prim-chip #(struct:chip-or)))))) (m n o p) (z)
)
```



```
Example 3

ircuit (a b) (e) #(struct:prim-chip #(struct:chip-and))) (#(struct:simple-circuit (c d) (f) #(struct:prim-chip #(struct:chip-and)))) (a b c d) (e f)))) (#(struct:simpl
e-circuit (e f) (z) #(struct:comp-chip (INE INF) (OUTA) #(struct:simple-circuit (e f) (g) #(struct:prim-chip #(struct:chip-or)))))) (m n o p) (z))
```



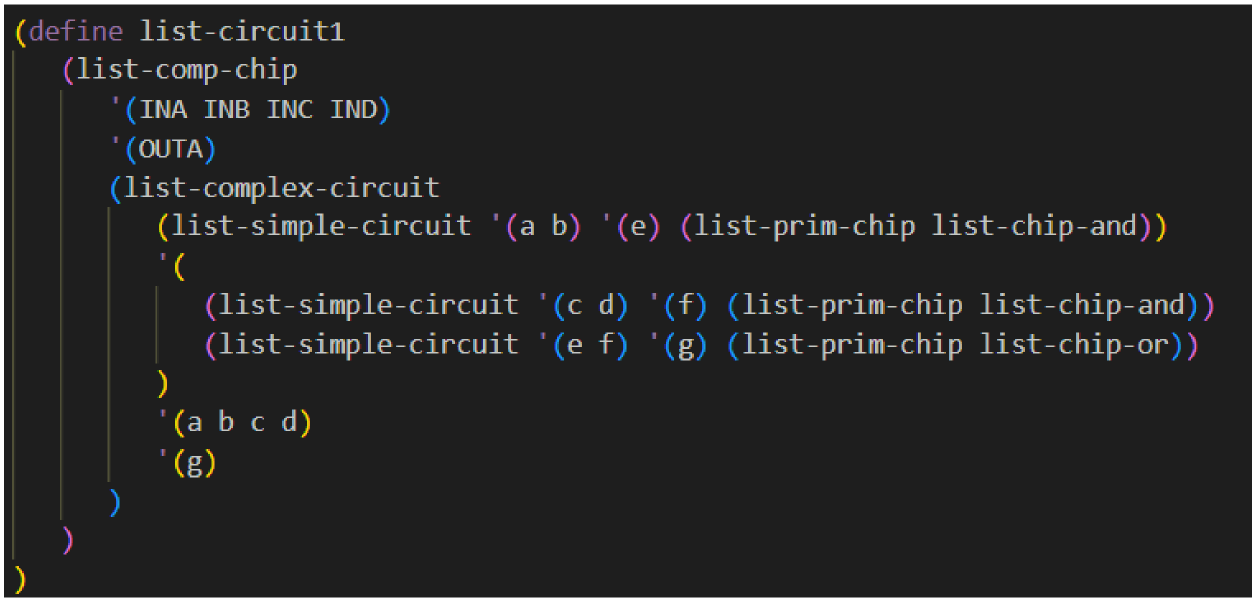
```
Circuit simple

#(struct:simple-circuit (a b) (c) #(struct:prim-chip #(struct:chip-and)))
```



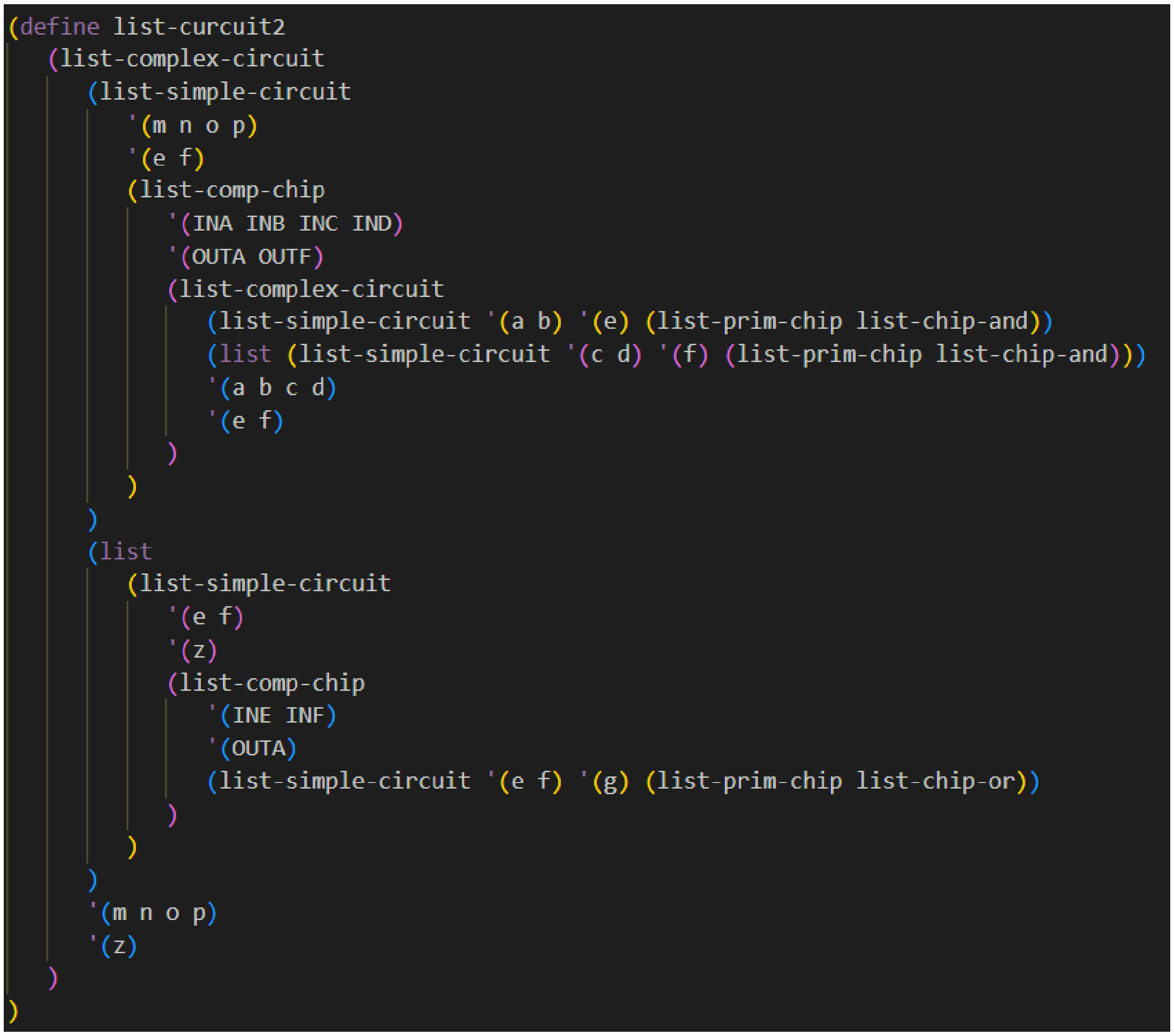
```
Circuit complex

#(struct:complex-circuit #(struct:simple-circuit (a b) (c) #(struct:prim-chip #(struct:chip-and))) (#(struct:simple-circuit (a b) (c) #(struct:prim-chip #(struct:chip-
and)))) (a b) (c))
```

Representacion-listas.rkt

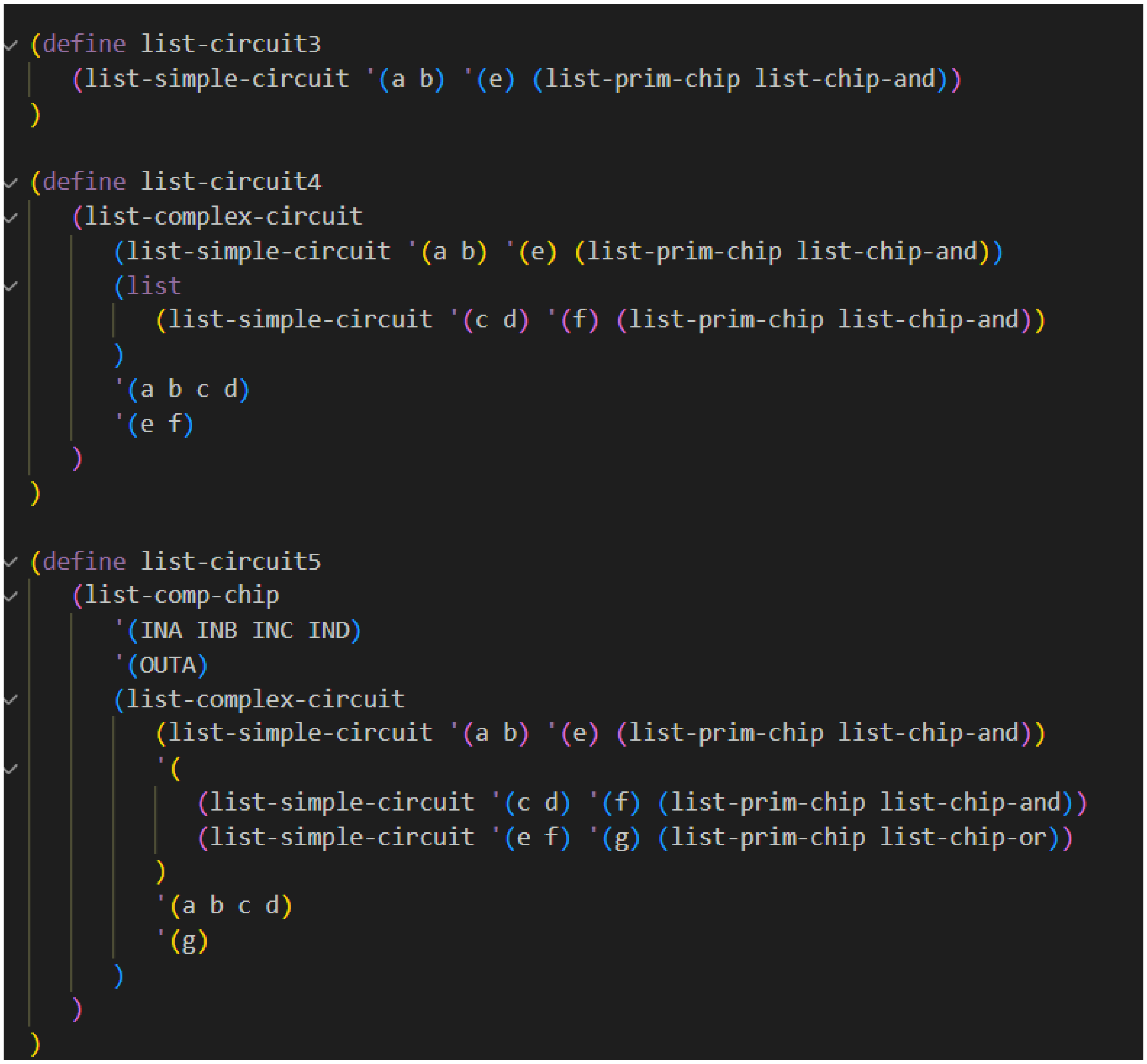
Se definieron algunos ejemplos y luego se mostrarán los resultados de estos:



```
(define list-circuit1
   (list-comp-chip
      '(INA INB INC IND)
      '(OUTA)
      (list-complex-circuit
         (list-simple-circuit '(a b) '(e) (list-prim-chip list-chip-and))
         '(
            (list-simple-circuit '(c d) '(f) (list-prim-chip list-chip-and))
            (list-simple-circuit '(e f) '(g) (list-prim-chip list-chip-or))
         )
         '(a b c d)
         '(g)
      )
   )
)
```

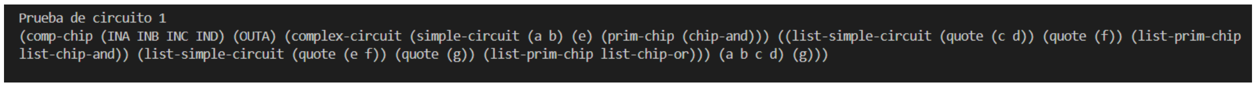


```
(define list-curcuit2
    (list-complex-circuit
        (list-simple-circuit
            '(m n o p)
            '(e f)
            (list-comp-chip
                '(INA INB INC IND)
                '(OUTA OUTF)
                (list-complex-circuit
                    (list-simple-circuit '(a b) '(e) (list-prim-chip list-chip-and))
                    (list (list-simple-circuit '(c d) '(f) (list-prim-chip list-chip-and)))
                    '(a b c d)
                    '(e f)
                )
            )
        )
        (list
            (list-simple-circuit
                '(e f)
                '(z)
                (list-comp-chip
                    '(INE INF)
                    '(OUTA)
                    (list-simple-circuit '(e f) '(g) (list-prim-chip list-chip-or))
                )
            )
        )
        '(m n o p)
        '(z)
    )
)
```

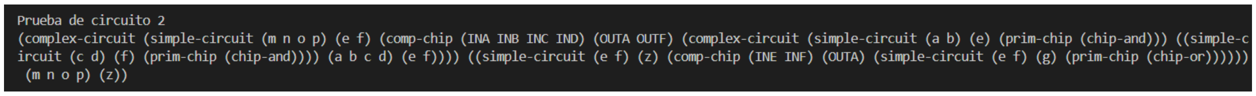


```scheme
(define list-circuit3
    (list-simple-circuit '(a b) '(e) (list-prim-chip list-chip-and))
)

(define list-circuit4
    (list-complex-circuit
        (list-simple-circuit '(a b) '(e) (list-prim-chip list-chip-and))
        (list
            (list-simple-circuit '(c d) '(f) (list-prim-chip list-chip-and))
        )
        '(a b c d)
        '(e f)
    )
)

(define list-circuit5
    (list-comp-chip
        '(INA INB INC IND)
        '(OUTA)
        (list-complex-circuit
            (list-simple-circuit '(a b) '(e) (list-prim-chip list-chip-and))
            '(
                (list-simple-circuit '(c d) '(f) (list-prim-chip list-chip-and))
                (list-simple-circuit '(e f) '(g) (list-prim-chip list-chip-or))
            )
            '(a b c d)
            '(g)
        )
    )
)
```

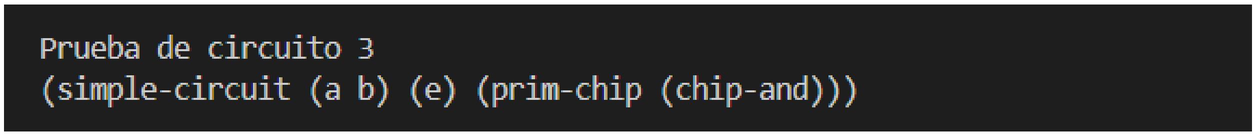
Y los resultados son los siguientes:



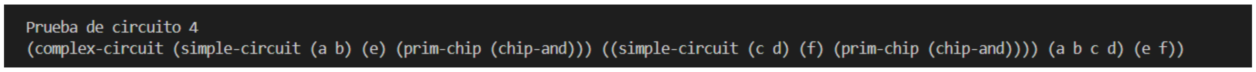
```
Prueba de circuito 1
(comp-chip (INA INB INC IND) (OUTA) (complex-circuit (simple-circuit (a b) (e) (prim-chip (chip-and))) ((list-simple-circuit (quote (c d)) (quote (f)) (list-prim-chip
list-chip-and)) (list-simple-circuit (quote (e f)) (quote (g)) (list-prim-chip list-chip-or))) (a b c d) (g)))
```



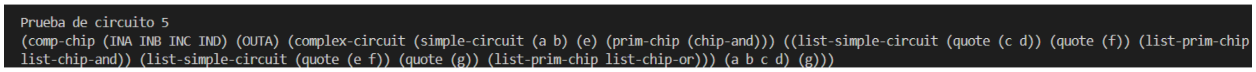
```
Prueba de circuito 2
(complex-circuit (simple-circuit (m n o p) (e f) (comp-chip (INA INB INC IND) (OUTA OUTF) (complex-circuit (simple-circuit (a b) (e) (prim-chip (chip-and))) ((simple-c
ircuit (c d) (f) (prim-chip (chip-and)))) (a b c d) (e f)))) ((simple-circuit (e f) (z) (comp-chip (INE INF) (OUTA) (simple-circuit (e f) (g) (prim-chip (chip-or)))))
 (m n o p) (z))
```



```
Prueba de circuito 3
(simple-circuit (a b) (e) (prim-chip (chip-and)))
```



```
Prueba de circuito 4
(complex-circuit (simple-circuit (a b) (e) (prim-chip (chip-and))) ((simple-circuit (c d) (f) (prim-chip (chip-and)))) (a b c d) (e f))
```

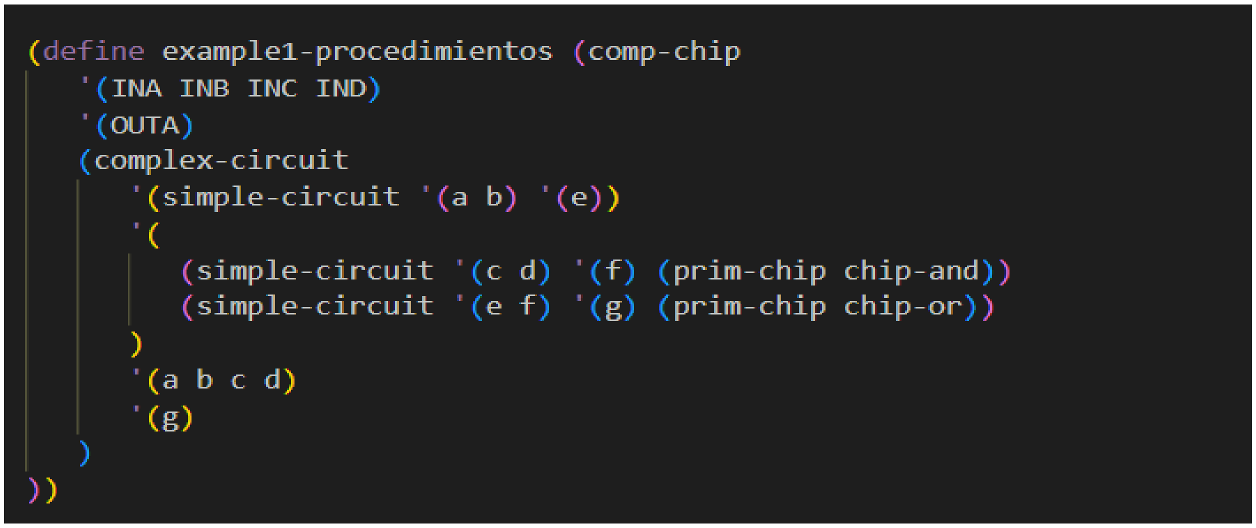


```
Prueba de circuito 5
(comp-chip (INA INB INC IND) (OUTA) (complex-circuit (simple-circuit (a b) (e) (prim-chip (chip-and))) ((list-simple-circuit (quote (c d)) (quote (f)) (list-prim-chip
list-chip-and)) (list-simple-circuit (quote (e f)) (quote (g)) (list-prim-chip list-chip-or))) (a b c d) (g)))
```

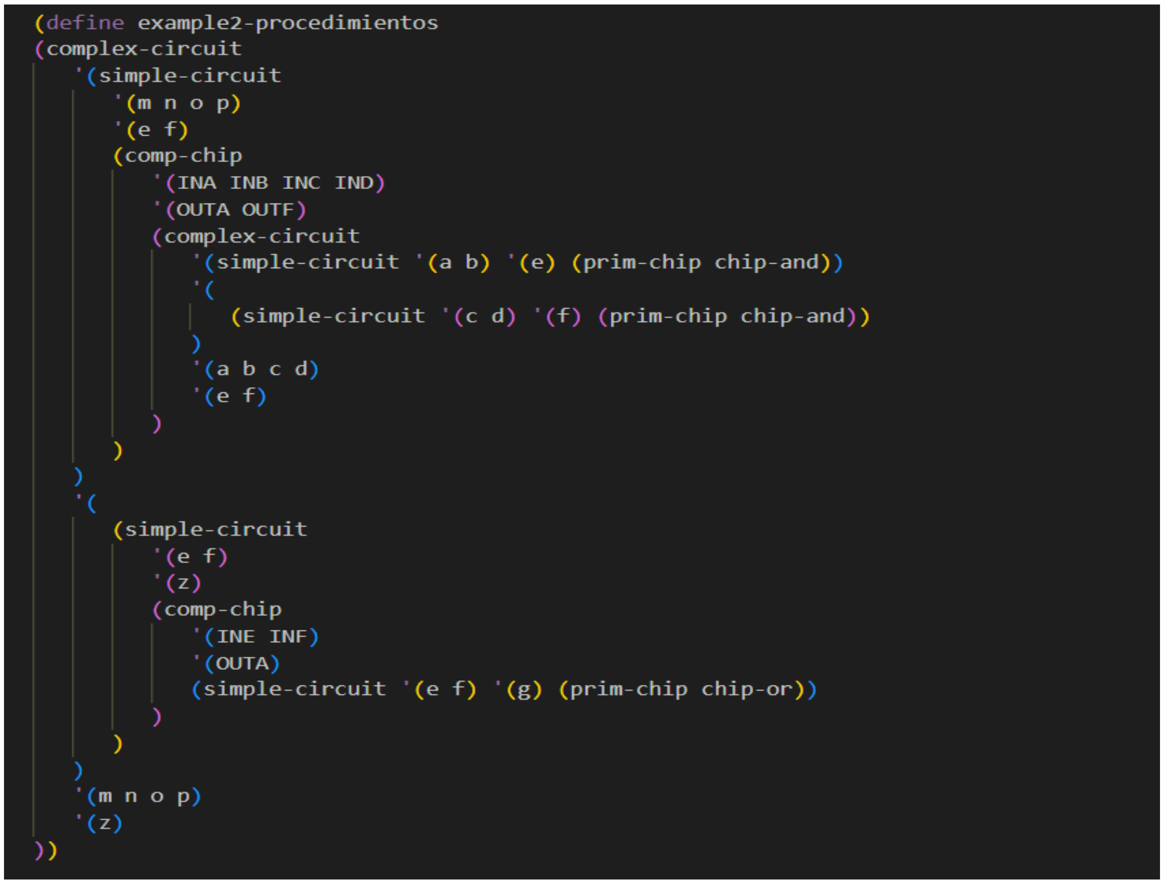
Representacion-procedimientos.rkt

En estos ejemplos mostraremos el uso de los extractores en diferentes tipos de posición dependiendo de la definición de cada circuitos, como mostramos en las siguientes imágenes:

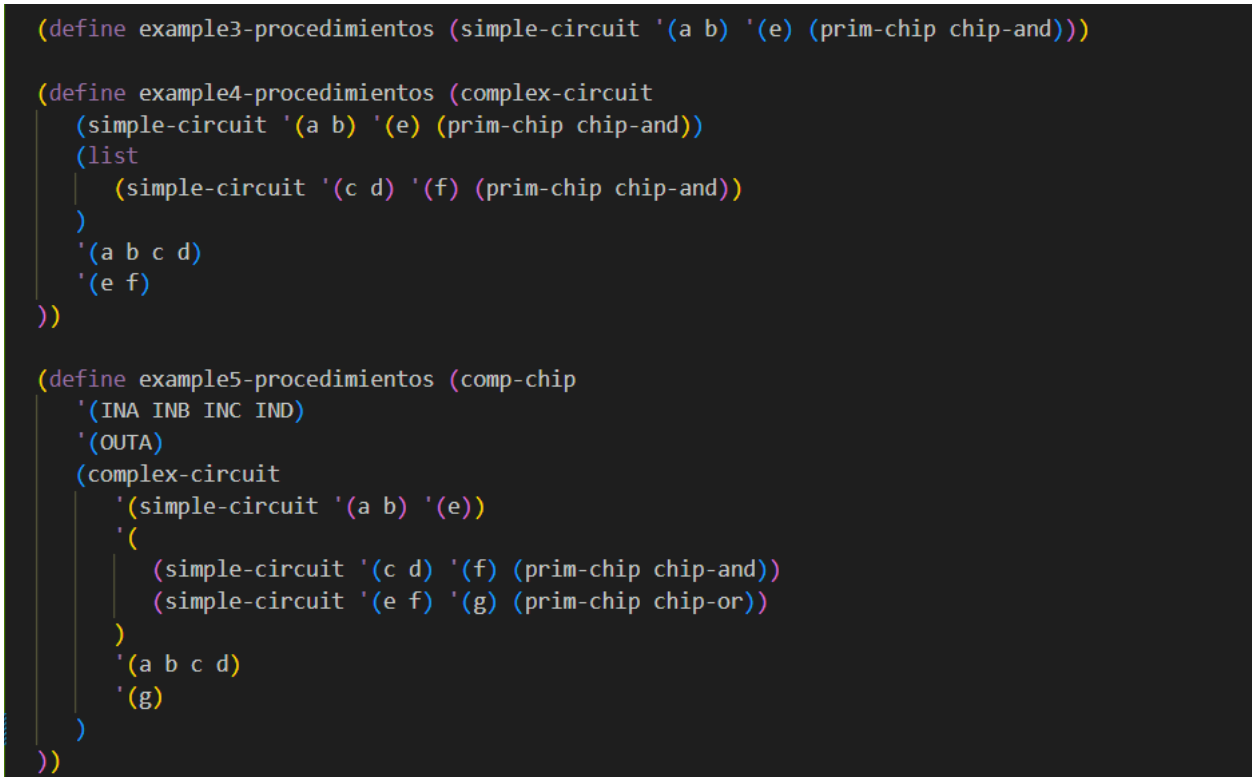
Primero definiremos los ejemplos y luego imprimimos.



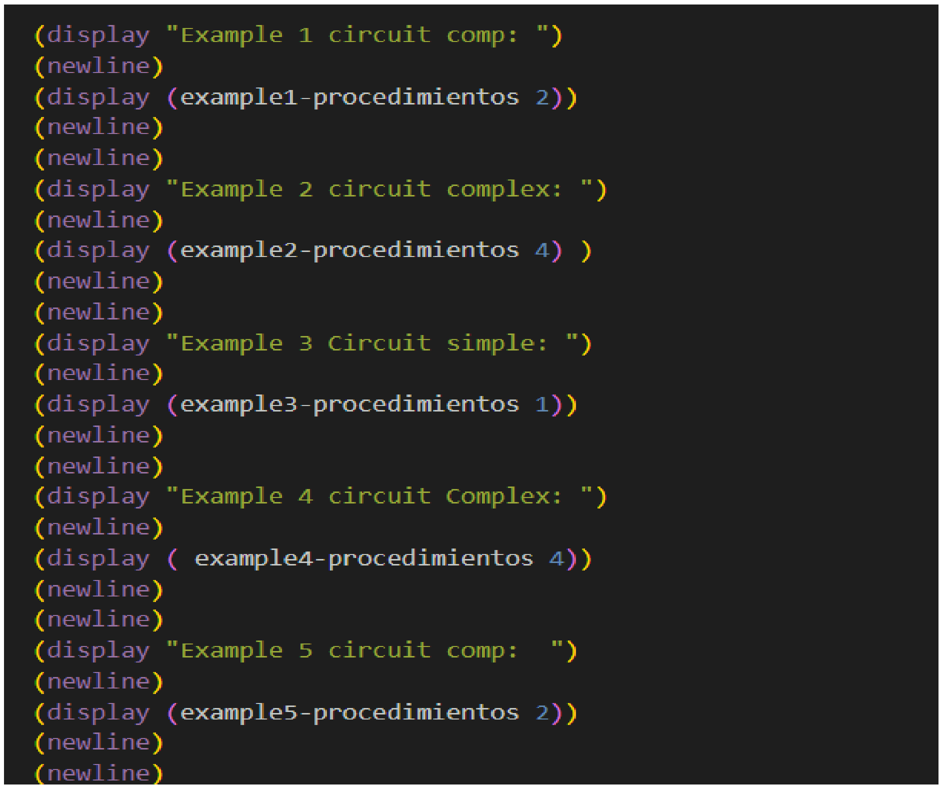
```
(define example1-procedimientos (comp-chip
   '(INA INB INC IND)
   '(OUTA)
   (complex-circuit
      '(simple-circuit '(a b) '(e))
      '(
         (simple-circuit '(c d) '(f) (prim-chip chip-and))
         (simple-circuit '(e f) '(g) (prim-chip chip-or))
      )
      '(a b c d)
      '(g)
   )
))
```



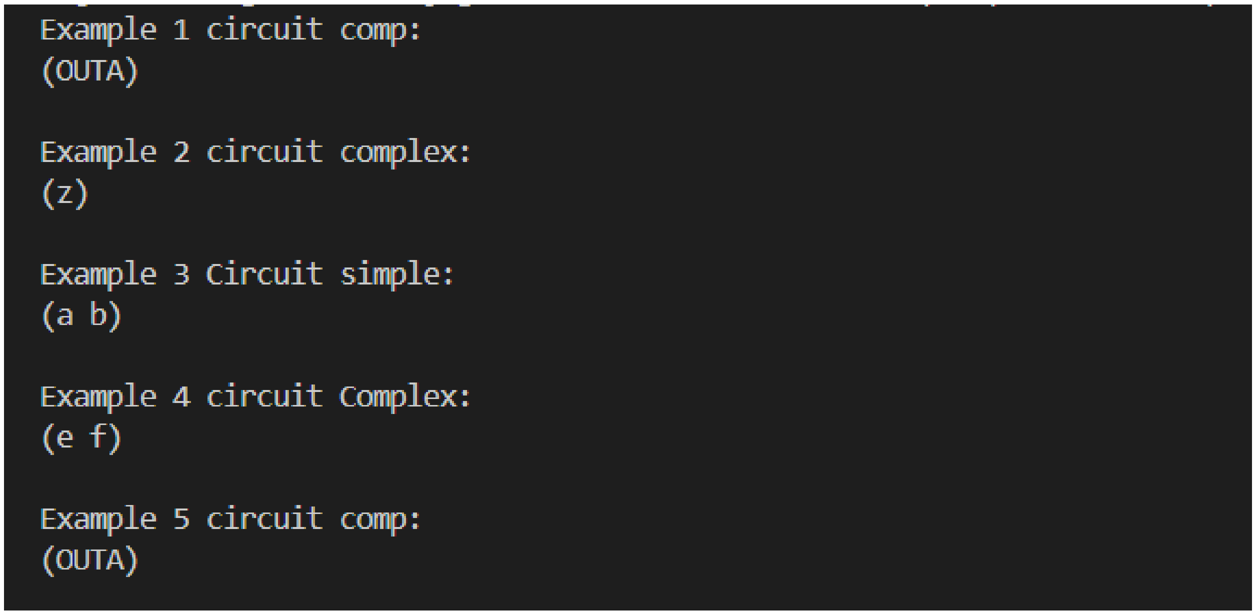
```
(define example2-procedimientos
(complex-circuit
   '(simple-circuit
      '(m n o p)
      '(e f)
      (comp-chip
         '(INA INB INC IND)
         '(OUTA OUTF)
         (complex-circuit
            '(simple-circuit '(a b) '(e) (prim-chip chip-and))
            '(
               (simple-circuit '(c d) '(f) (prim-chip chip-and))
            )
            '(a b c d)
            '(e f)
         )
      )
   )
   '(
      (simple-circuit
         '(e f)
         '(z)
         (comp-chip
            '(INE INF)
            '(OUTA)
            (simple-circuit '(e f) '(g) (prim-chip chip-or))
         )
      )
   )
   '(m n o p)
   '(z)
))
```



```
(define example3-procedimientos (simple-circuit '(a b) '(e) (prim-chip chip-and)))

(define example4-procedimientos (complex-circuit
   (simple-circuit '(a b) '(e) (prim-chip chip-and))
   (list
      (simple-circuit '(c d) '(f) (prim-chip chip-and))
   )
   '(a b c d)
   '(e f)
))

(define example5-procedimientos (comp-chip
   '(INA INB INC IND)
   '(OUTA)
   (complex-circuit
      '(simple-circuit '(a b) '(e))
      '(
         (simple-circuit '(c d) '(f) (prim-chip chip-and))
         (simple-circuit '(e f) '(g) (prim-chip chip-or))
      )
      '(a b c d)
      '(g)
   )
))
```



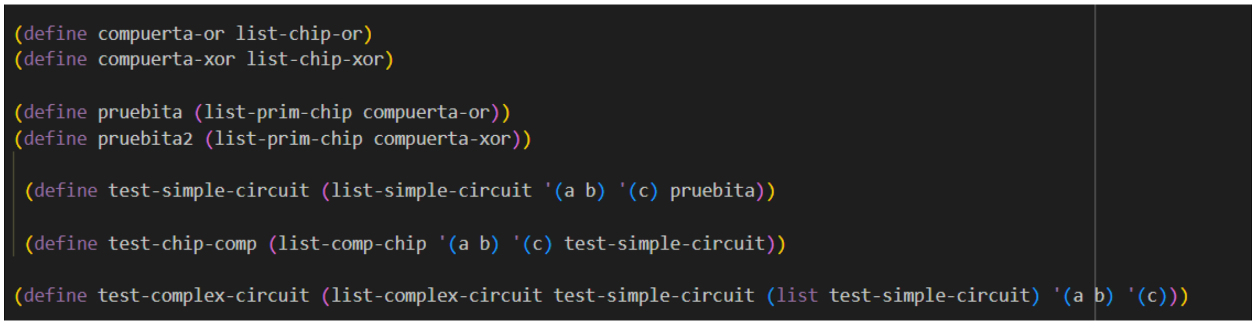
```
(display "Example 1 circuit comp: ")
(newline)
(display (example1-procedimientos 2))
(newline)
(newline)
(display "Example 2 circuit complex: ")
(newline)
(display (example2-procedimientos 4) )
(newline)
(newline)
(display "Example 3 Circuit simple: ")
(newline)
(display (example3-procedimientos 1))
(newline)
(newline)
(display "Example 4 circuit Complex: ")
(newline)
(display ( example4-procedimientos 4))
(newline)
(newline)
(display "Example 5 circuit comp:  ")
(newline)
(display (example5-procedimientos 2))
(newline)
(newline)
```



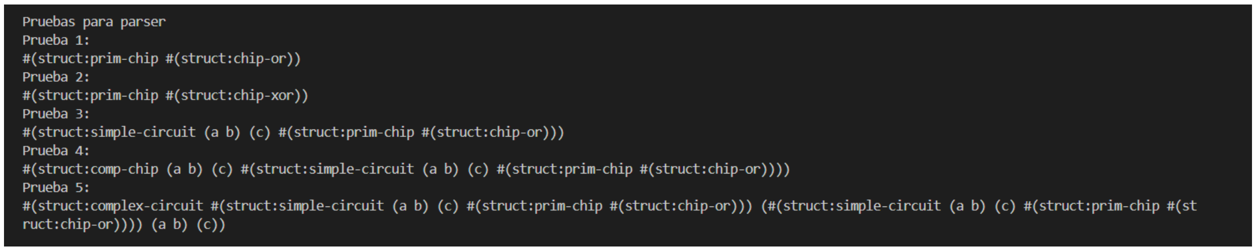
```
Example 1 circuit comp:
(OUTA)

Example 2 circuit complex:
(z)

Example 3 Circuit simple:
(a b)

Example 4 circuit Complex:
(e f)

Example 5 circuit comp:
(OUTA)
```

parser-unparser.rkt

Pruebas parser

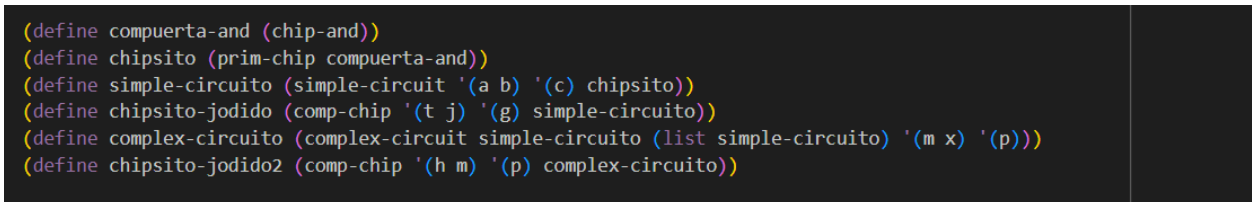


```
(define compuerta-or list-chip-or)
(define compuerta-xor list-chip-xor)

(define pruebita (list-prim-chip compuerta-or))
(define pruebita2 (list-prim-chip compuerta-xor))

 (define test-simple-circuit (list-simple-circuit '(a b) '(c) pruebita))

 (define test-chip-comp (list-comp-chip '(a b) '(c) test-simple-circuit))

(define test-complex-circuit (list-complex-circuit test-simple-circuit (list test-simple-circuit) '(a b) '(c)))
```

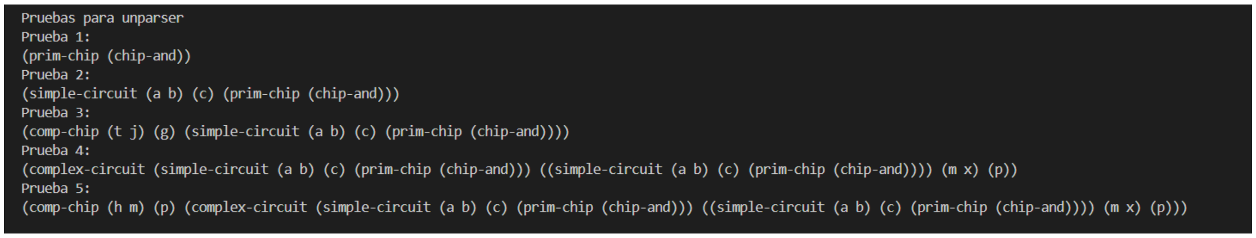


```
Pruebas para parser
Prueba 1:
#(struct:prim-chip #(struct:chip-or))
Prueba 2:
#(struct:prim-chip #(struct:chip-xor))
Prueba 3:
#(struct:simple-circuit (a b) (c) #(struct:prim-chip #(struct:chip-or)))
Prueba 4:
#(struct:comp-chip (a b) (c) #(struct:simple-circuit (a b) (c) #(struct:prim-chip #(struct:chip-or))))
Prueba 5:
#(struct:complex-circuit #(struct:simple-circuit (a b) (c) #(struct:prim-chip #(struct:chip-or))) (#(struct:simple-circuit (a b) (c) #(struct:prim-chip #(st
ruct:chip-or)))) (a b) (c))
```

Pruebas unparser



```
(define compuerta-and (chip-and))
(define chipsito (prim-chip compuerta-and))
(define simple-circuito (simple-circuit '(a b) '(c) chipsito))
(define chipsito-jodido (comp-chip '(t j) '(g) simple-circuito))
(define complex-circuito (complex-circuit simple-circuito (list simple-circuito) '(m x) '(p)))
(define chipsito-jodido2 (comp-chip '(h m) '(p) complex-circuito))
```



```
Pruebas para unparser
Prueba 1:
(prim-chip (chip-and))
Prueba 2:
(simple-circuit (a b) (c) (prim-chip (chip-and)))
Prueba 3:
(comp-chip (t j) (g) (simple-circuit (a b) (c) (prim-chip (chip-and))))
Prueba 4:
(complex-circuit (simple-circuit (a b) (c) (prim-chip (chip-and))) ((simple-circuit (a b) (c) (prim-chip (chip-and)))) (m x) (p))
Prueba 5:
(comp-chip (h m) (p) (complex-circuit (simple-circuit (a b) (c) (prim-chip (chip-and))) ((simple-circuit (a b) (c) (prim-chip (chip-and)))) (m x) (p)))
```