Juan Camilo Garcia Saenz - 2259416
Jean Carlos Lerma Rojas - 2259305

Informe Taller 2

Se realizó modificación en la especificación gramatical para poder hacer uso de cond, length, first, rest, nth y cond. Se hizo esta modificación con el fin de especificar que le entrará al realizar

Las modificaciones en la especificación gramatical tienen como objetivo extender el lenguaje para que este pueda soportar las listas ***"list"***, estructuras de control ***"cond"*** y las operaciones sobre listas como ***length, first, rest y nth.***

1. **Definición de listas:** Para que pudiera soportar listas se añade las producciones ***"cons"*** y ***"empty"***. El propósito de estas es que el constructor cons se usa para crear una lista, tomando dos elementos: la cabeza y el resto, que el resto puede ser otra lista o ***"empty".*** Empty representa una lista vacía. Estas dos producciones permiten que el lenguaje reconozca las listas.

```
;; listas
(expresion ("cons" "(" expresion expresion ")") list-exp)
(expresion ("empty") list-empty-exp)
```

2. **Estructura de control cond:** Para manejar la estructura de control ***"cond"*** se define una estructura condicional con múltiples cláusulas, el fin de esta es permitir que la cláusula ***"cond"*** evalúe condiciones multiples. La expresión puede tener varias condiciones seguidas de una acción correspondiente, y el bloque ***"else"*** se ejecuta si ninguna de las condiciones es verdaderas.

```
;;cond
(expresion ("cond" (arbno expresion "==>" expresion) "else" "==>" expresion "end") cond-exp)
```

3. **Funciones de lista:** se añaden ***length, first, rest y nth*** para trabajar con listas.
    a. **length:** Devuelve la longitud de una lista.
    b. **first**: Devuelve el primer elemento de la lista (la cabeza).
    c. **rest:** Devuelve la lista sin el primer elemento (el resto).
    d. **nth:** Devuelve el elemento en la posición n de la lista.

Juan Camilo Garcia Saenz - 2259416
Jean Carlos Lerma Rojas - 2259305

```
;;Fin listas
(expresion ("length" "(" expresion ")") length-prim)
(expresion ("first" "(" expresion ")") first-prim)
(expresion ("rest" "(" expresion ")") rest-prim)
(expresion ("nth" "(" expresion expresion ")") nth-prim)
```

Ejemplo de lista:

```
------>cons(1 cons(2 cons(3 empty)))
(1 2 3)
```

Se le envía: cons(1 cons(2 cons(3 empty))) y esta retornará la lista (1 2 3)

Ejemplo cond

```
------>cond
-(x,1)==>1
-(x,2)==>2
-(x,4)==>4
else==>9
end
2
```

Ejemplo de las funciones de lista:

- length: Devuelve la longitud de una lista.

  ```
  ------>length(cons (1 cons (2 cons(3 empty))))
  3
  ```

  Se le envía la lista y esta nos devuelve el tamaño de esta.

- first: Devuelve el primer elemento de la lista (la cabeza).

  ```
  ------>first(cons (4 cons (3 cons (2 empty))))
  4
  ```

  Se le manda toda la lista y aplica un car y devuelve el primer valor que encuentra.

- rest: Devuelve la lista sin el primer elemento (el resto).

Juan Camilo Garcia Saenz - 2259416
Jean Carlos Lerma Rojas - 2259305

```
------>rest(cons (4 cons (3 cons (2 empty))))
(3 2)
```

Se le envía la lista y manda el resto de la lista, menos la cabeza

- nth: Devuelve el elemento en la posición n de la lista.

Ahora bien, las modificaciones realizadas para evaluar la expresión fueron para poder soportar las nuevas expresiones ya mencionadas.

- Cambio: Soporte para listas (cons): cons construye una lista a partir de un elemento que será la cabeza de la lista y una lista que será la cola
    - Si el segundo argumento no es una lista válida, se genera un error.
    - Si el segundo argumento es una lista (o empty), se combina con el primer argumento.
    - Cambios que se realizaron en el código.

```
;;listas
(list-exp (rator rands)
          (let ((head (evaluar-expresion rator amb))
                (tail (evaluar-expresion rands amb)))
            (if (list? tail)
                (cons head tail)
                (eopl:error "Error: el segundo argumento de cons no es una lista" tail))))
(list-empty-exp ()
                '()
                )
```

    - Ejemplo:

```
------>cons(1 cons(2 cons(3 empty)))
(1 2 3)
```

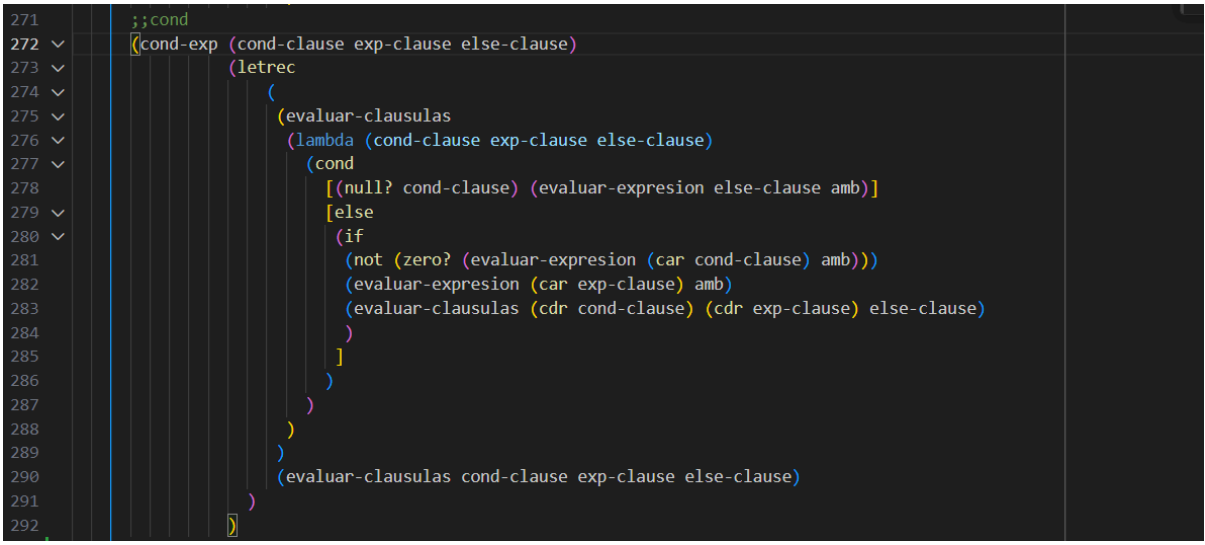
Trabaja de la siguiente manera

1) El primer cons toma 1 como cabeza y (cons 2 (cons 3 empty)) como el resto.
2) El segundo cons toma 2 como cabeza y (cons 3 empty) como resto.
3) El tercer cons toma 3 como cabeza y empty como resto, terminando la lista.

- Cambio: Soporte para conds:
    - (x, 1), (x, 2) y (x, 4) son condiciones que involucran la operación de resta (-).
    - Cada condición evalúa el resultado de restar el valor dado a x. Todo resultado diferente de 0 es verdadero.
    - Si todas las condiciones resultan en 0, se activa la cláusula else.

Juan Camilo Garcia Saenz - 2259416
Jean Carlos Lerma Rojas - 2259305

- Cambios que se realizaron en el código



```
;;cond
(cond-exp (cond-clause exp-clause else-clause)
          (letrec
              (
               (evaluar-clausulas
                (lambda (cond-clause exp-clause else-clause)
                  (cond
                    [(null? cond-clause) (evaluar-expresion else-clause amb)]
                    [else
                     (if
                      (not (zero? (evaluar-expresion (car cond-clause) amb)))
                      (evaluar-expresion (car exp-clause) amb)
                      (evaluar-clausulas (cdr cond-clause) (cdr exp-clause) else-clause)
                      )
                     ]
                    )
                  )
                )
               )
              (evaluar-clausulas cond-clause exp-clause else-clause)
          )
)
```

- Como trabaja:
  - Suponiendo que el valor de x es = 1
  - Primera condición: -(x,1): Evaluamos x - 1 con x = 1. El cálculo es: 1-1 =0 Como el resultado es 0, esta condición es falsa entonces pasamos a la siguiente condición
  - Segunda condición: -(x,2): Evaluamos x - 2 con x = 1. El cálculo es: 1-2=-1 Como el resultado es diferente de 0, esta condición es verdadera entonces el resultado del cond es: 2 y se detienen las evaluaciones posteriores a esta ya que la condición verdadera ya fue satisfecha.

- Cambios para trabajar las funciones de listas.
  - **length**: toma una lista como entrada y devuelve su tamaño, es decir, la cantidad de elementos en la lista.
  - **first**: toma una lista y devuelve su primer elemento (la cabeza de la lista).
  - **rest:** toma una lista y devuelve todo excepto el primer elemento (la cola de la lista).
  - **nth:** toma dos argumentos: una lista y un índice. Devuelve el elemento de la lista en la posición indicada (comenzando desde 0).

Juan Camilo Garcia Saenz - 2259416
Jean Carlos Lerma Rojas - 2259305

```
;;length
(length-prim (exps)
  (length (evaluar-expresion exps amb))
)
;;first
(first-prim (exps)
  (car (evaluar-expresion exps amb))
)
;;rest
(rest-prim (exps)
  (cdr (evaluar-expresion exps amb))
)
;;nth
(nth-prim (list position)
  (let ((pos (evaluar-expresion position amb))
        (lst (evaluar-expresion list amb)))
    (list-ref lst pos)
  )
)
```