Lenguajes de Programación
Tarea 7

Karla Ramírez Pulido Alan Alexis Martínez López

Semestre 2023-2 **Fecha de inicio:** 24 de mayo 2023
Facultad de Ciencias UNAM **Fecha de entrega:** 01 de junio 2023

Integrantes:
Dania Paula Gongora Ramírez
Salgado Tirado Diana Laura

Instrucciones

Resolver los siguientes ejercicios de forma clara y ordenada de acuerdo a los lineamientos de entrega de tareas disponibles en la página del curso.

Ejercicios

1. Define los siguientes conceptos con tus propias palabras y en no más de cinco renglones:

a. Variable de tipo.

Se utiliza para representar un tipo abstracto en lugar de un tipo específico, permite reutilizar código de manera más flexible, ya que la variable de tipo puede ser sustituida por un tipo concreto al utilizarla.

b. Polimorfismo explícito.

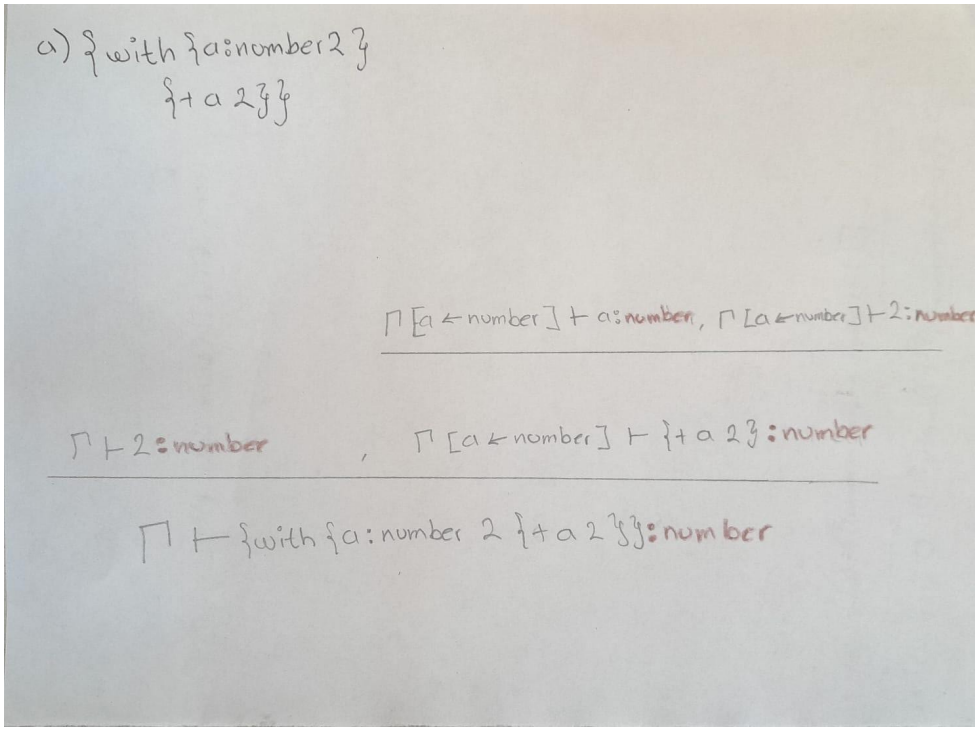
Permite especificar directamente los diferentes tipos de datos que un método puede recibir como parámetros o devolver como resultado, esto permite adaptar el comportamiento de una función a diferentes tipos de datos.

c. Polimorfismo implícito.

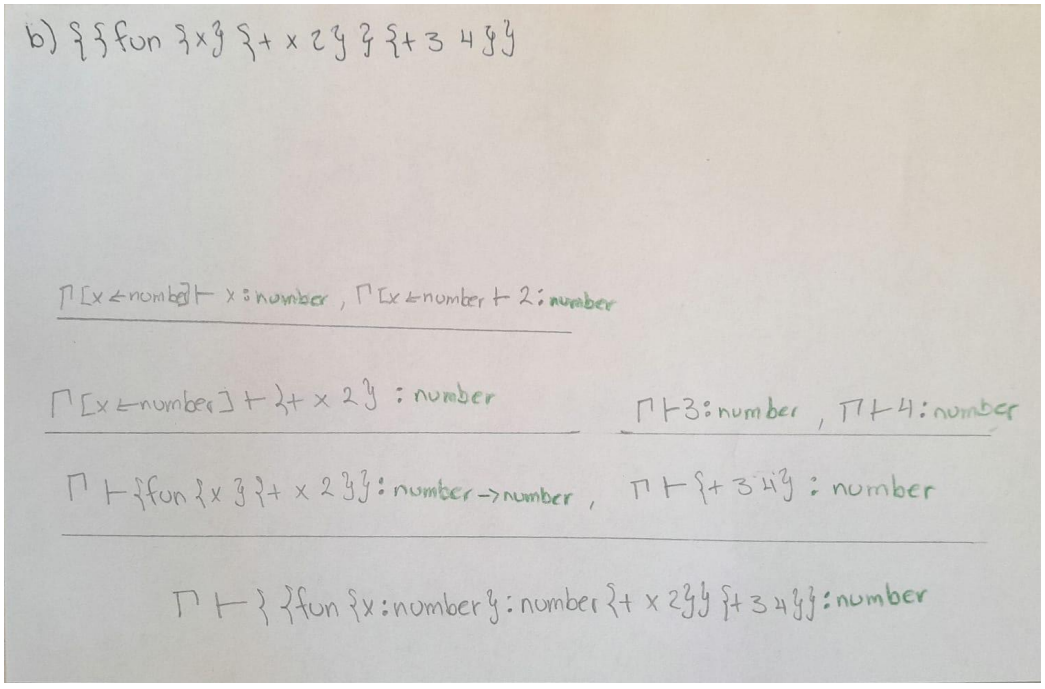
Este funciona sin la necesidad de especificar explícitamente los tipos de datos involucrados, en los casos de polimorfismo implícito el lenguaje de programación deduce automáticamente los tipos,por lo tanto permite utilizar múltiples tipos de datos sin necesidad de declararlos directamente.

2. Realiza el juicio de tipo para cada una de las siguientes expresiones.

a. *{* with *{* a : number 2 *}*
{ + a 2 *} }*

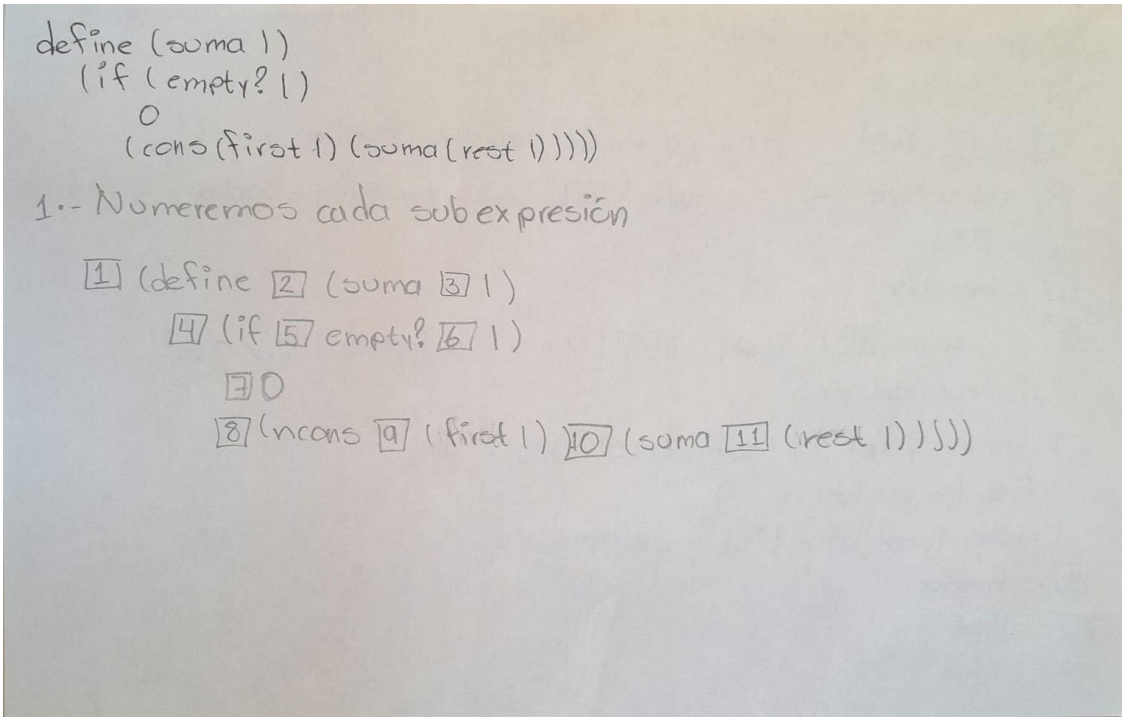


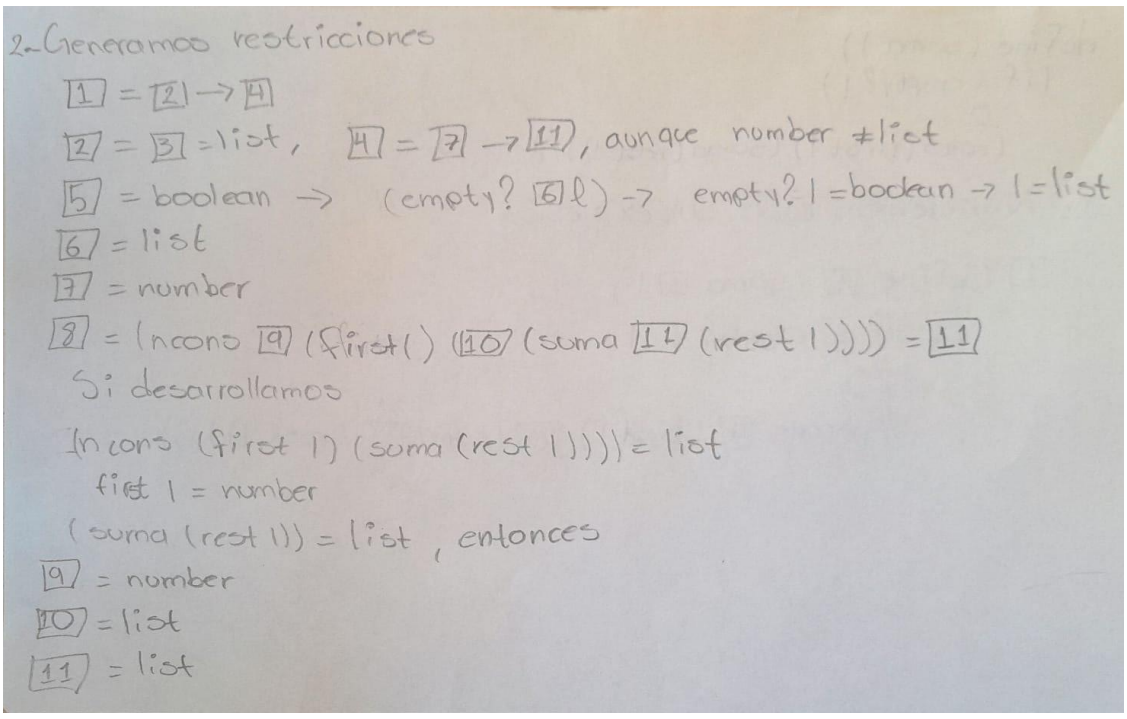
b. *{ {* fun *{* x*} { +* x 2*} } { +* 3 4*} }*



3. Dada la siguiente expresión, realiza su inferencia de tipos generando las restricciones de tipos correspondientes.

```
define (suma l)
  (if (empty? l)
     0
     (cons (first l) (suma (rest l )))))
```





4. Usando el algoritmo de unificación, muestra la inferencia de tipos de la siguiente expresión: ((λ (x) (* x 2)) (+ 2 3))

- Nombramos las sub-expresiones

  ((λ (x) (* x 2)) (+ 2 3))

  1 ((λ (x) (* x 2)) (+ 2 3))
  2 (λ (x) (* x 2))
  3 (* x 2)
  4 (+ 2 3)
  5 2
  6 3
  7 x
  8 2

- Las restricciones son:

  [[2]] = [[4]] → [[1]]
  [[2]] = [[x]] → [[3]]
  [[3]] = number
  [[7]] = number
  [[8]] = number
  [[4]] = number
  [[5]] = number
  [[6]] = number

  Introducimos en el stack las restricciones generadas:

Acción	**Stack**	**Sustitución**
Inicialización	[[2]] = [[4]] → [[1]] [[2]] = [[x]] → [[3]] [[3]] = number [[7]] = number [[8]] = number [[4]] = number [[5]] = number [[6]] = number	empty

Paso 2	[[4]] → [[1]] = [[x]]→ [[3]] [[3]] = number [[7]] = number [[8]] = number [[4]] = number [[5]] = number [[6]] = number	[[2]] → [[4]] → [[1]]
Paso 4	[[4]] = [[x]] [[1]] = [[3]] [[3]] = number [[7]] = number [[8]] = number [[4]] = number [[5]] = number [[6]] = number	[[2]] → [[4]] → [[1]]
Paso 2	[[1]] = [[3]] [[3]] = number [[7]] = number [[8]] = number [[x]] = number [[5]] = number [[6]] = number	[[2]] → [[x]] → [[1]] [[4]] → [[x]]
Paso 2	[[3]] = number [[7]] = number [[8]] = number [[x]] = number [[5]] = number [[6]] = number	[[2]] → [[x]] → [[3]] [[4]] → [[x]] [[1]] → [[3]]
Paso 2	[[7]] = number [[8]] = number [[x]] = number [[5]] = number [[6]] = number	[[2]] → [[x]] → number [[4]] → [[x]] [[1]] → number [[3]] → number

Paso 2	[[8]] = number [[x]] = number [[5]] = number [[6]] = number	[[2]] → [[x]] → number [[4]] → [[x]] [[1]] → number [[3]] → number [[7]] → number
Paso 2	[[x]] = number [[5]] = number [[6]] = number	[[2]] → [[x]] → number [[4]] → [[x]] [[1]] → number [[3]] → number [[7]] → number [[8]] → number
Paso 2	[[5]] = number [[6]] = number	[[2]] → number → number [[4]] → number [[1]] → number [[3]] → number [[7]] → number [[8]] → number [[x]] → number
Paso 2	empty	[[2]] → number → number [[4]] → number [[1]] → number [[3]] → number [[7]] → number [[8]] → number [[x]] → number [[5]] → number [[6]] → number

Algoritmo de Unificación

1. Si X e Y son constante idénticas, no se hace nada.

2. Si X e Y son identificadores idénticos, no se hace nada.

3. Si X es un identificador, reemplaza todas las ocurrencias de X por Y tanto en el stack como en la sustitución, y añade $X \rightarrow Y$ en la sustitución.

4. Si Y es un identificador, reemplaza todas las ocurrencias de Y por X tanto en el stack como en la sustitución, y añade $Y \rightarrow X$ en la sustitución.

5. Si X es de la forma $C(X_1, X_2, ..., X_n)$ para algún constructor C, e Y es de la forma $C(Y_1, Y_2, ..., Y_n)$ (i.e. tienen el mismo constructor), entonces agrega $X_i = Y_i$ para toda 1 i n en el stack.

6. En cualquier otro caso, X e Y no se unifican y se reporta un error.