Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–4721 – Lenguajes de Programación II
Enero – Marzo 2025

Tarea III

(10 puntos)

A continuación encontrará 3 preguntas, cada una compuesta de diferentes sub–preguntas. El valor de cada pregunta (y sub–pregunta) estará expresado entre paréntesis al inicio de las mismas.

En aquellas preguntas donde se le pida ejecutar un algoritmo o procesar una entrada, incluya los pasos relevantes de la ejecución del mismo con los cuales usted pudo alcanzar su conclusión. Sea lo más detallado y preciso posible en sus razonamientos y procedimientos.

La entrega se realizará por correo electrónico a rmonascal@gmail.com hasta las 11:59pm. VET del Miércoles 19 de Marzo de 2025.

1. (2 puntos) Considere las siguientes declaraciones de tipo de datos:

```
type pokemon = struct
   pi : float
   ka : pokebola
   chu : pokebola
end
```

```
type pokebola = *pokemon
```

   a) (1 punto) Construya el grafo de tipos que se genera para representar al tipo `pokemon`, considerando equivalencia por nombre estricta.

   b) (1 punto) Considere el siguiente conjunto de declaraciones:

```
var a : pokemon
var b = &a : Tb
var c = (*b).chu : Tc
```

```
var d = (*a.chu).pi
var e = *(*(*c).ka).chu
```

   Para las variables `a`, `b`, `c`, `d` y `e`, establezca su tipo (note que para la variable `a` ya está colocado) y marque, en el grafo de tipos de la pregunta anterior, a qué nodo corresponde el tipo de cada una de las variables mencionadas.

2. (3 puntos) Considere los siguiente símbolos con su tipos (potencialmente polimórficos):

`cmap`	:	β	`head`	:	$\forall\alpha.list(\alpha) \to \alpha$
`f`	:	γ	`tail`	:	$\forall\alpha.list(\alpha) \to list(\alpha)$
`x`	:	ρ	`if`	:	$\forall\alpha.bool \times \alpha \times \alpha \to \alpha$
`[]`	:	$\forall\alpha.list(\alpha)$	`concat`	:	$\forall\alpha.list(\alpha) \to list(\alpha) \to list(\alpha)$
`null`	:	$\forall\alpha.list(\alpha) \to bool$	`match`	:	$\forall\alpha.\alpha \times \alpha \to \alpha$

   Considere también la siguiente expresión:

```
match(cmap(f,x), if(null(x), [], concat(f(head(x)), cmap(f, tail(x)))))
```

   Utilice el esquema de verificación visto en clase para determinar el tipo más general de `cmap` y muestre las unificaciones realizadas en cada paso.

3. (5 puntos) Traducción dirigida por sintaxis para verificación de tipos.

   a) (2 puntos) Considere la siguiente gramática de expresiones:

$$\begin{array}{rcl} E & \rightarrow & E + E \\ & | & E \wedge E \\ & | & E < E \\ & | & E \ ?: E \\ & | & E \ !! \\ & | & (E) \\ & | & num \\ & | & true \\ & | & false \\ & | & null \end{array}$$

   Las primeras tres reglas de la producción corresponden a la suma de enteros, la conjunción booleana y el operador relacional "menor que", respectivamente.

   La producción $E \rightarrow E \ ?: E$ representa evaluaciones con valores por defecto en caso de nulidad. Esto es, si se tiene $a \ ?: b$, la expresión evalúa a b si $a = null$ y evalúa a a de lo contrario.

   La producción $E \rightarrow E \ !!$ representa evaluaciones forzadas sin seguridad por nulidad. Esto es, si se tiene $a \ !!$, la expresión evalúa a a si $a \neq null$ y es un error de tipo de lo contrario.

   Construya la regla asociada a esta producción que calcula `E.type`, el tipo de la expresión. El atributo debe ser sintetizado.

   Nota: Puede suponer que el tipo de num es INT, el tipo de $true$ y $false$ es $BOOL$ y el tipo de $null$ es $NULL$.

   b) (1 punto) Utilice el esquema de traducción dirigido por sintaxis definido en la pregunta anterior para construir el tipo de la siguiente frase:

$$(((null \ ?: 42) + (69 \ !!) < (7 \ ? : null)) \wedge null) \ ?: true$$

   Muestre cada uno de los pasos en el árbol de derivación.

   c) (1 punto) Considere la gramática de instrucciones vista en clase y auméntela con una producción de la forma:

$$S \rightarrow repeatWhen \ E \ lt \ S \ gt \ S$$

   Esta producción representa repeticiones indeterminadas, con cuerpos que dependen del signo de la una expresión aritmética. Esto es, si se tiene `repeatWhen a lt b gt c`, la expresión ejecuta `b` siempre que `a <0` y ejecuta `c` siempre que `a >0`. En el caso que `a = 0` entonces sale de la repetición.

   Construya la regla asociada a esta producción que calcula `S.type`, el tipo de la instrucción (recuerde que las instrucciones son siempre de tipo `void` a menos que haya ocurrido algún error). El atributo debe ser sintetizado.

   d) (1 punto) Replantee las reglas definidas para las gramáticas anteriores en términos de reglas de inferencia.