CI4721 - Lenguajes de Programación II

Examen 2 — *20 puntos*

1. Toda la pregunta 1 fue desarrollada en el Repositorio de Github :)

   Para la parte B, el tipo de la variable es el que se encuentra resaltado en cada uno de los diagramas.

2. Toda la pregunta 2 fue desarrollada en el Repositorio de Github :)

3. Las secciones a, b y c de la pregunta 3 fueron desarrolladas en el Repositorio de Github :)

   Las traducción a reglas de inferencia (sección d) se muestra a continuación:

   Ya que el ambiente ρ comienza vacío, primero añadiremos las reglas de inferencia para las siguientes producciones de la gramática:

   ```
   E ->  num      { E.type = INT  }
   |     true     { E.type = BOOL }
   |     false    { E.type = BOOL }
   |     null     { E.type = NULL }
   ```

   Que son representadas por las reglas de inferencia

   $$\frac{true}{\rho, < num, INT > \vdash num := INT}$$

   $$\frac{true}{\rho, < true, TRUE > \vdash true := BOOL}$$

   $$\frac{true}{\rho, < false, FALSE > \vdash false := BOOL}$$

   $$\frac{true}{\rho, < null, NULL > \vdash null := NULL}$$

   respectivamente.

   La producción:

   ```
   E -> E1 + E2    {
                     if (E1.type == INT /\ E2.type == INT) {
                       E.type = INT
                     } else {
                       E.type = ERROR
                     }
                   }
   ```

es representada por:

$$\frac{\rho \vdash e_1 : \text{INT} \qquad \rho \vdash e_2 : \text{INT}}{\rho \vdash e_1 + e_2 : \text{INT}}$$

La producción:

```
E -> E1 /\ E2 {
       if (E1.type == BOOL /\ E2.type == BOOL) {
         E.type = BOOL
       } else {
         E.type = ERROR
       }
     }
```

es representada por:

$$\frac{\rho \vdash e_1 : \text{BOOL} \qquad \rho \vdash e_2 : \text{BOOL}}{\rho \vdash e_1 \wedge e_2 : \text{BOOL}}$$

La producción:

```
E -> E1 < E2 {
       if (E1.type == INT /\ E2.type == INT) {
         E.type = BOOL
       } else {
         E.type = ERROR
       }
     }
```

es representada por:

$$\frac{\rho \vdash e_1 : \text{INT} \qquad \rho \vdash e_2 : \text{INT}}{\rho \vdash e_1 < e_2 : \text{BOOL}}$$

La producción:

```
E -> E1 ?! E2 {
       if (E1.type == NULL) {
         E.type = E2.type
       } else {
         E.type = E1.type
       }
     }
```

es representada por:

$$\frac{\rho \vdash e_1 : \text{NULL} \qquad \rho \vdash e_2 : \tau}{\rho \vdash e_1? : e_2 : \tau}$$

y

$$\frac{\neg(\rho \vdash e_1 : \text{NULL})}{\rho \vdash e_1? : e_2 : \rho(e_1)}$$

La producción:

```
E -> E1 !!      {
                  if (E1.type != NULL) {
                    E.type = E1.type
                  } else {
                    E.type = ERROR
                  }
                }
```

es representada por:

$$\frac{\neg(\rho \vdash e_1 : \text{NULL})}{\rho \vdash e_1!! : \rho(e_1)}$$

La producción:

```
E -> ( E1 )     { E.type = E1.type }
```

es representada por:

$$\frac{\rho \vdash e_1 : \tau}{\rho \vdash (e_1) : \tau}$$

La producción:

```
S -> repeatWhen E lt S1 gt S2 {
                  // Si la expresion a comparar con 0 no es un
                  // entero es un error, o alguno de los bloques
                  // es un error, entonces devuelvo un error.
                  if (E.type != INT \/ S1.type != ERROR \/
                      S2.type != ERROR ) {
                    S.type = ERROR
                  }
                  // Si todo esta en orden, devuelvo VOID.
```

```
            else {
              S.type = VOID
            }
        }
```

es representada por:

$$\frac{\rho \vdash e : \text{INT} \qquad \rho \vdash s_1 : \text{VOID} \qquad \rho \vdash s_2 : \text{VOID}}{\rho \vdash \mathit{repeateWhen}\ \ e\ \ \mathit{lt}\ \ s_1\ \ \mathit{gt}\ \ s_2 : \text{VOID}}$$

4. Tipo más general de las expresión:

Expresión	Tipo	Sustitución
f	γ	
x	ρ	
cmap	β	
cmap(f,x)	δ	$\beta = \gamma \times \rho \to \delta$
x	ρ	
null	$\text{list}(\alpha_a) \to \mathit{bool}$	
null(x)	bool	$\rho = \mathit{list}(\alpha_a)$
[]	$\text{list}(\alpha_b)$	
x	$\text{list}(\alpha_a)$	
head	$\text{list}(\alpha_c) \to \alpha_c$	
head(x)	α_a	$\alpha_a = \alpha_c$
f	γ	
f(head(x))	σ	$\gamma = \alpha_a \to \sigma$
f	$\alpha_a \to \sigma$	
x	$\text{list}(\alpha_a)$	
tail	$\text{list}(\alpha_c) \to \mathit{list}(\alpha_c)$	
tail(x)	$\text{list}(\alpha_a)$	$\alpha_a = \alpha_c$
cmap	$(\alpha_a \to \sigma) \times \mathit{list}(\alpha_a) \to \delta$	
cmap(f, tail(x))	δ	
concat	$\text{list}(\alpha_d) \times \mathit{list}(\alpha_d) \to \mathit{list}(\alpha_d)$	
concat(f(...), cmap(...))	$\mathit{list}(\alpha_d)$	$\sigma = \mathit{list}(\alpha_d), \delta = \mathit{list}(\alpha_d)$
if	$\mathit{bool} \times \alpha_e \times \alpha_e \to \alpha_e$	
$if(...)$	$\mathit{list}(\alpha_b)$	$\alpha_e = \mathit{list}(\alpha_b), \alpha_b = \alpha_d$
$match$	$\alpha_f \times \alpha_f \to \alpha_f$	
$match(cmpa(...), if(...))$	$\mathit{list}(\alpha_b)$	$\alpha_f = \mathit{list}(\alpha_b)$

5. Toda la pregunta 5 fue desarrollada en el Repositorio de Github :)

6. Toda la pregunta 6 fue desarrollada en el Repositorio de Github :)