Universidad Nacional Autónoma de México

Facultad de Ciencias

Tarea Examen III

Lenguajes de Programación

Dafne Bonilla Reyes
José Camilo García Ponce
Rodrigo Aldair Ortega Venegas

Profesor: Javier Enríquez Mendoza

Ayudante: Andrea Regina García Correa
Ayudante: Karla Denia Salas Jiménez
Ayudante Laboratorio: Ramón Arenas Ayala
Ayudante Laboratorio: Oscar Fernando Millán Pimentel

Noviembre 2023

Lenguajes de Programación

Tarea Examen 3

El constructor alternativo **if** se puede generalizar mediante un operador case definido como sigue:

$$e ::= \ldots \mid \mathsf{case}\ g\ \mathsf{end}$$

$$g ::= e_1 \Rightarrow e_2 \mid g\ ;\ g$$

Una expresión de la forma $e_1 \Rightarrow e_2$ se conoce como expresión resguardada, siendo la expresión e_1 una expresión booleana llamada guardia. Un constructor **case** se evalúa como sigue: recorrer las expresiones resguardadas $e_i \Rightarrow e_j$ en orden de izquierda a derecha (respectivamente de arriba a abajo), hasta hallar la primera expresión resguardada, digamos $e_k \Rightarrow e_l$ tal que $e_k \Rightarrow^*$ **true**, en cuyo caso se procede a evaluar e_l, cuyo valor final es también el resultado de la evaluación de la expresión **case**. Por ejemplo, considérese el siguiente programa:

```
case x = 0 => x ;
        x>2 => x^2 ;
        x>0 => x-2 ;
        x<0 => x+2
end
```

1. Define la sintaxis abstracta del operador case.

$$\frac{\text{g asa}}{\text{Case (g) asa}}$$

$$\frac{g_1 \text{ asa} \quad g_2 \text{ asa}}{g_1; g_2 \text{ asa}}$$

$$\frac{e_1 \text{ asa} \quad e_2 \text{ asa}}{e_1 \Rightarrow e_2 \text{ asa}}$$

2. Define las reglas de transición para modelar la semántica operacional del nuevo operador case.

$$\frac{g_1 \to g_2}{\text{Case } (g_1) \to \text{Case } (g_2)}$$

$$\frac{e_1 \to e_3}{\text{Case } (e_1 \Rightarrow e_2) \to \text{Case } (e_3 \Rightarrow e_2)}$$

$$\frac{}{\text{Case } (True \Rightarrow e_2) \to e_2}$$

$$\frac{g_1 \to g_3}{\text{Case } (g_1; g_2) \to \text{Case } (g_3; g2)}$$

$$\frac{e_1 \to e_3}{\text{Case } (e_1 \Rightarrow e_2; g_2) \to \text{Case } (e_3 \Rightarrow e_2; g_2)}$$

$$\frac{}{\text{Case } (True \Rightarrow e_2; g_2) \to e_2}$$

$$\frac{}{\text{Case } (False \Rightarrow e_2; g_2) \to \text{Case } (g_2)}$$

3. Define las reglas de tipado para la semántica estática del operador case.

$$\frac{\Gamma \vdash \text{g} : \text{T}}{\Gamma \vdash \text{Case (g)} : \text{T}}$$

$$\frac{\Gamma \vdash g_1 : \text{T} \quad \Gamma \vdash g_2 : \text{T}}{\Gamma \vdash g_1; g_2 : \text{T}}$$

$$\frac{\Gamma \vdash e_1 : \text{Bool} \quad \Gamma \vdash e_2 : \text{T}}{\Gamma \vdash e_1 \Rightarrow e_2 : \text{T}}$$

4. Extiende el algoritmo de inferencia de tipos de la nota 8, agregando las reglas de generación de restricciones para el operador case.

$$\frac{\text{g} \mapsto R_1}{\text{Case (g)} \mapsto R_1, [\![\text{Case (g)}]\!] = [\![\text{g}]\!]}$$

$$\frac{g_1 \mapsto R_1 \quad g_2 \mapsto R_2}{g_1; g_2 \mapsto R_1, R_2, [\![g_1]\!] = [\![g_2]\!], [\![g_1; g_2]\!] = [\![g_1]\!], [\![g_1; g_2]\!] = [\![g_2]\!]}$$

$$\frac{e_1 \mapsto R_1 \quad e_2 \mapsto R_2}{e_1 \Rightarrow e_2 \mapsto R_1, R_2, [\![e_1]\!] = \text{Bool}, [\![e_1 \Rightarrow e_2]\!] = [\![e_2]\!]}$$

5. Explica por qué el operador case es azúcar sintáctica en el lenguaje.

   Case es azúcar sintáctica, ya que solo es una forma más legible y sencilla de escribir `if` anidados. Esto es, si no existiera en el lenguaje, no afectaría las capacidades de este, pues en `case` vamos revisando expresiones resguardas y viendo si nos dan `True` o `False` la guardia. Si obtenemos `True` terminamos, pero si obtenemos `False` nos vamos a la siguiente expresión resguardada como si fuera el `else` de `if`.