Parseo y Generación de Código – 2^{do} semestre 2017
Licenciatura en Informática con Orientación en Desarrollo de Software
Universidad Nacional de Quilmes

Segundo parcial

Nota: este parcial es a libro abierto. Se permite tener cualquier material manuscrito o impreso, pero no se permite el uso de dispositivos electrónicos. El parcial se califica con una nota numérica de 1 a 10. Se requiere ≥ 4 en ambos parciales para aprobar la materia. Para promocionar se requiere nota ≥ 6 en ambos parciales y promedio ≥ 7.

Ejercicio 1. Dado el siguiente lenguaje:

```
data Program = While Id Program     -- repetición : while (x != 0) { C }
             | Seq Program Program  -- secuencia  : C1; C2
             | Inc Id               -- incremento : x := x + 1
             | Dec Id               -- decremento : x := x - 1
             | Return Id            -- retornar   : return x
```

Escribir una función `exec :: Program -> Int` para ejecutar un programa y obtener un resultado numérico. Las variables empiezan inicializadas en 0. Si durante la ejecución del programa nunca se ejecuta un `return`, el resultado debe ser 0. Si se ejecuta un comando `return x`, la ejecución del programa finaliza y el resultado final es el valor de la variable `x`. Se pueden usar entornos (`Env Int`) para darle valores a las variables, suponiendo que la operación `lookupEnv` devuelve 0 en caso de que la variable no tenga un valor asociado.

Ejercicio 2. Dado el siguiente programa:

```
r := 1
for i = 1 to n {
    for j = 1 to r {
        r := r + i * j
    }
}
```

a. Generar código de tres direcciones. Para facilitar la lectura, suponer que cada una de las variables (`n`, `i`, `j`, `r`) tiene asociado un registro con el mismo nombre. Además, se pueden utilizar otros registros (`t1`, `t2`, ...) si es necesario.

b. Construir el grafo de flujo de control.

Ejercicio 3. Suponiendo que $\oplus$ es un operador binario:

a. Calcular el número de Ershov de $(a \oplus b) \oplus (c \oplus d)$.

b. Calcular el número de Ershov de $a \oplus (b \oplus (c \oplus d))$.

c. Proponer una optimización que podría hacer el compilador en este contexto, suponiendo que la operación $\oplus$ es asociativa, es decir que $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.

Ejercicio 4. Considerar el lenguaje con los siguientes términos y tipos:

$$t ::= \star \mid <t,t> \mid \texttt{left}(t) \mid \texttt{right}(t) \mid \texttt{const}(t) \mid \texttt{case}(t,t) \mid \texttt{uncurry}(t) \mid t \,\texttt{@}\, t \qquad\qquad A ::= 1 \mid A \times A \mid A + A \mid A \to A$$

Los juicios de tipado son de la forma $t : A$ y están dados por las siguientes reglas:

$$\frac{}{\star : 1}\ \text{T-UNIT} \qquad \frac{t : A}{\texttt{const}(t) : 1 \to A}\ \text{T-CONST} \qquad \frac{t : A \quad s : B}{<t,s> : A \times B}\ \text{T-PAIR} \qquad \frac{t : A \to (B \to C)}{\texttt{uncurry}(t) : (A \times B) \to C}\ \text{T-UNCURRY}$$

$$\frac{t : A}{\texttt{left}(t) : A + B}\ \text{T-SUM}_{\text{L}} \qquad \frac{t : B}{\texttt{right}(t) : A + B}\ \text{T-SUM}_{\text{R}} \qquad \frac{t : A \to C \quad s : B \to C}{\texttt{case}(t,s) : (A + B) \to C}\ \text{T-CASE} \qquad \frac{t : A \to B \quad s : A}{t \,\texttt{@}\, s : B}\ \text{T-APP}$$

a. Dar una derivación del juicio $\texttt{uncurry}(\texttt{const}(\texttt{const}(\star))) \,\texttt{@}\, <\star,\star> : 1$.

b. Dar una derivación del juicio $\texttt{case}(\texttt{const}(\star), \texttt{const}(\star)) \,\texttt{@}\, \texttt{left}(\star) : 1$.

c. Proponer un término t apropiado para que se pueda derivar el siguiente juicio y exhibir una derivación:

$$t : ((1 + 1) \times (1 + 1)) \to 1$$

Justificar todas las respuestas.