Parseo y Generación de Código – 2do cuatrimestre 2023
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. La siguiente gramática $G = (\{S\}, \{:=, +, \mathbf{reg}\}, \mathcal{P}, B)$ describe la sintaxis de los bloques básicos de una máquina de registros con una única operación para sumar:

$$B \ ::= \ \epsilon \mid \mathbf{reg} := \mathbf{reg} + \mathbf{reg}\, B$$

Suponiendo que el símbolo terminal **reg** tiene un atributo **reg**.`nombre` :: `String` que indica el nombre del registro sobre el que se está operando, **extender la gramática con atributos sintetizados y/o heredados** para calcular el conjunto de registros vivos en cada punto del bloque.

Ejercicio 2. La siguiente máquina de registros tiene tres instrucciones que sirven para cargar una variable en un registro, sumar y multiplicar.

```
data Instruccion = ILoad Reg Variable   -- r := x
                 | IAdd Reg Reg Reg     -- r1 := r2 + r3
                 | IMul Reg Reg Reg     -- r1 := r2 * r3
```

Las variables se identifican por nombre y los registros por número, así el tipo `Reg` es un renombre de `Int`, mientras que `Variable` es un renombre de `String`.

Por otro lado, decimos que un *polinomio* es una suma de varios términos y un *término* es un producto de varias variables:

```
data Polinomio = TerminoSolo Termino | Suma Polinomio Termino
data Termino   = VariableSola Variable | Producto Termino Variable
```

Se pide **definir una función** `compilar :: Polinomio -> [Instruccion]` que genere código para calcular el resultado de evaluar el polinomio en la máquina de registros de arriba, **usando únicamente tres registros**.

Ejercicio 3. El siguiente tipo de datos modela las expresiones de un lenguaje con constructores y *pattern matching*:

```
type Id = String          data Patron = PVar Id | PCons String [Patron]
data Expr = EVar Id | ECons String [Expr] | ECase Expr Patron Expr Expr
```

Una expresión puede ser una variable, un constructor aplicado a una lista de expresiones, o una construcción `case`. La construcción `case e1 of { p -> e2 ; _ -> e3 }` recibe una expresión `e1` y la evalúa hasta que sea un valor. Si el valor tiene la forma indicada por el patrón `p`, se procede a evaluar la expresión `e2` ligando las variables correspondientes. En caso contrario, se procede a evaluar la expresión `e3`.

Por ejemplo `(ECons "Succ" [ECons "Succ" [ECons "Zero" []]])` representa el número 2 en notación de Peano, mientras que `(ECase e (PCons "Succ" (PVar "x")) (EVar "x") (ECons "Falla" []))` calcula el predecesor de `e`. Más precisamente, si el valor de `e` es de la forma `Succ(x)` devuelve `x`, y en caso contrario devuelve `Falla()`.

Se pide **implementar un intérprete** `eval :: Expr -> Env -> Valor`, donde `Env` es el tipo de los entornos que asocian variables a valores. Los valores están dados por el tipo de datos `data Valor = VCons String [Valor]`.

Nota. Se puede asumir que los patrones no contienen variables repetidas.

Ejercicio 4. Se extiende el sistema de tipos del cálculo-λ de la siguiente manera:

$$\textbf{Tipos} \quad A \quad ::= \quad \alpha \mid A \to A \mid \bullet A \qquad \textbf{Términos} \quad M \quad ::= \quad x \mid \lambda x.\, M \mid M\, M \mid \bullet M \mid \mathsf{wait}\, x = M \,\mathsf{in}\, M$$

con las siguientes reglas de tipado:

$$\frac{}{\Gamma, x : A \vdash x : A}\ \text{VAR} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B}\ \text{LAM} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : \bullet B}\ \text{APP}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \bullet M : \bullet A}\ \text{DELAY} \qquad \frac{\Gamma \vdash M : \bullet A \qquad \Gamma, x : A \vdash N : B}{\Gamma \vdash \mathsf{wait}\, x = M \,\mathsf{in}\, N : \bullet B}\ \text{WAIT}$$

a. Suponiendo que el tipo B está fijo, exhibir un tipo A tal que el juicio $z : B \vdash \mathsf{wait}\, y = (\lambda x.\, x)\, (\lambda x.\, x) \,\mathsf{in}\, y\, z : A$ sea derivable, y dar una derivación.

b. Exhibir un término M tal que el juicio $\vdash M : (A \to B \to C) \to (A \to B) \to A \to \bullet \bullet \bullet C$ sea válido en el contexto vacío, y dar una derivación.

Justificar todas las respuestas.