Práctica 2

El lenguaje `EAB`. (Semántica)

Favio E. Miranda Perea (favio@ciencias.unam.mx)
Diego Carrillo Verduzco (dixego@ciencias.unam.mx)
Pablo G. González López (pablog@ciencias.unam.mx)

Viernes 7 de septiembre de 2018

Fecha de entrega: Miércoles 19 de septiembre de 2018 a las 23:59:59.

Recordemos la definición del lenguaje de Expresiones Aritmético Booleanas que usamos en el ejercicio semanal anterior.

```
e ::= x | n | true | false
    | e + e | e * e | succ e | pred e
    | not e | and e e | or e e
    | lt e e | gt e e | eq e e
    | if e then e else e end | let x = e in e end
```

Para realizar esta práctica deberán importar alguno de sus módulos que contienen la sintaxis de este lenguaje en `Haskell`.

1 Semántica Dinámica

La semántica dinámica es la que determina cuál es el valor de la evaluación de un programa. Como se vio en clase existen varios estilos para definirla, sin embargo seguiremos utilizando la semántica operacional para definir el comportamiento de los programas a través de un sistema de transiciones.

1. (1 punto) Agrega en el *README* la definición de la semántica dinámica de los operadores booleanos y de relación.

   Sugerencia: Utiliza el comando `\infer` del paquete `proof` para escribir los juicios.

   \infer[Nombre del juicio]{Conclusión}{Premisas}

Implementa las siguientes funciones:

1. (2 puntos) `eval1`. Devuelve la transición tal que `eval1 e = e'` syss $e \to e'$.

   Ejemplo:

```
*Main> eval1 (Add (I 1) (I 2))
I 3
*Main> eval1 (Let "x" (I 1) (Add (V "x") (I 2)))
Add (I 1) (I 2)
```

2. (2 puntos) `evals`. Devuelve la transición tal que `evals e = e'` syss $e \to^* e'$ y e' está bloqueado.

   Ejemplo:

   ```
   *Main> evals (Let "x" (Add (I 1) (I 2)) (Eq (V "x") (I 0)))
   B False
   *Main> evals (Add (Mul (I 2) (I 6)) (B True))
   Add (I 12) (B True)
   ```

3. (2 puntos) `eval`. Devuelve la evaluación de un programa tal que `eval e = e'` syss $e \to^* e'$ y e' es un valor. En caso de que e' no esa un valor deberá mostrar un mensaje de error particular del operador que lo causó.

   Ejemplo:

   ```
   *Main> eval (Add (Mul (I 2) (I 6)) (B True))
   *** Exception: [Add] Expects two Nat.
   *Main> eval (Or (Eq (Add (I 0) (I 0)) (I 0)) (Eq (I 1) (I 10)))
   B True
   ```

2 Semántica Estática

La semántica estática es la que determina cuándo un programa está bien definido mediante criterios sintácticos que son sensibles al contexto, requiriendo que cada variable sea declarada antes de usarse.

La verificación de la correctud estática de un programa se hace a través del sistema de tipos. Un sistema de tipos consiste en una colección de reglas de inferencia que imponen ciertas restricciones en la formación de programas.

Para las expresiones `EAB` definiremos los siguientes tipos:

```
data Type = Nat | Boolean
```

1. (1 punto) Agrega en el *README* la definición de la semántica estática de los operadores booleanos y de relación.

Ahora modelaremos el contexto como una lista de pares que almacenen el nombre de las variables junto con su tipo.

```
type Decl = (Identifier, Type)
type TypCtxt = [Decl]
```

Implementa la siguiente función:

1. (2 puntos) `vt`. Verifica el tipado de un programa tal que `vt Γ e T = True` syss $\Gamma \vdash e : T$.

   Ejemplo:

   ```
   *Main> vt [("x",Boolean)] (If (B True) (B False) (Var "x")) Boolean
   True
   *Main> vt [] (Let "x" (Add (I 1) (I 2))
   (Eq (Mul (Add (V "x") (I 5)) (I 0)) (Add (V "x") (I 2))))
   Boolean
   True
   ```

¡Suerte!