Universdad Nacional Autónoma de México

**Lenguajes de Programacion**
**Examen Parcial V**


- Edgar Montiel Ledesma
  317317794
- Carlos Daniel Cortes Jimenez
  420004846


---

1. Sean A, B, C, D, E con la siguiente relacion de subtipado:

   $E \leq B$

   $D \leq B$

   $B \leq A$

   $C \leq A$

   Para los siguientes subtipados diga si son validos o no. En caso de ser validos, muestre la derivacion formal, en caso de ser invalidos, añada una sola regla que la haga valida y haga la derivacion formal.

   a) $((A + B) \to \{y : E, x : D\}) + ((A\ \ A) \to E) \leq ((A + B) \to \{x : C\}) + ((A \times B) \to C)$

   - Esta expresión es válida, y la derivación formal es la siguiente:
     $((A+B)\to\{y:E,x:D\})+((A\times A)\to E)$
     $\leq ((A+B)\to\{x:C\})+((A\times B)\to C)$
   - Por regla de subtipado:
     $(A+B)\to\{y:E,x:D\}\leq(A+B)\to\{x:C\}$
   - Por regla de subtipado:
     $(A\times A)\to E\leq(A\times B)\to C$
   - Por reglas de subtipado anteriores:
     $A\leq B$(de $B\leq A$)
   - Por regla de subtipado:
     $A\times A\leq A\times B$
   - Por reglas de subtipado anteriores:
     $E\leq B$(de $B\leq A$)
   - Por regla de subtipado:
     $(A\times A)\to E\leq(A\times B)\to C$
   - Por regla de subtipado:
     $(A+B)\to\{y:E,x:D\}+(A\times A)\to E$
     $\leq(A+B)\to\{x:C\}+(A\times B)\to C$

b) $(E \times D \times A \to \{x : A\}) \to C \leq (B \times A \to \{x : A, y : D\}) \to A$
Esta expresión no es inválida. Para hacerla válida, agregaremos la regla de subtipado A≤C.

- La derivación formal es la siguiente:
  $(E \times D \times A \to \{x:A\}) \to C$
  $\leq (B \times A \to \{x:A, y:D\}) \to A$
- Por regla de subtipado:
  $E \times D \times A \to \{x:A\} \leq B \times A \to \{x:A, y:D\}$
- Por reglas de subtipado anteriores:
  A≤C(nueva regla agregada)
- Por regla de subtipado:
  $E \times D \times A \to \{x:A\} \leq B \times A \to \{x:A, y:D\}$
- Por regla de subtipado:
  $(E \times D \times A \to \{x:A\}) \to C \leq (B \times A \to \{x:A, y:D\}) \to A$

Ahora, con la adición de la regla A≤C, la expresión se vuelve válida.

2. Considera los tipos Rectangulo parametrizado por la altura y el ancho y Cuadrado parametrizado por el tamaño de uno de los lados.

   a) Define los tipos anteriores mediante tipos suma y producto.

   b) ¿Que relacion de subtipado existe entre estos tipos? es decir ¿cual de estos es el subtipo y cual el supertipo? justifica tu respuesta, una forma de justificar la respuesta es explicar como sería el proceso de coercion entre el subtipo y el supertipo.

   c) De acuerdo a la relacion de subtipado del inciso anterior, ordena los siguientes tipos funcion:

   Rectangulo → Rectangulo
   Rectangulo → Cuadrado
   Cuadrado → Rectangulo
   Cuadrado → Cuadrado

3. El objetivo de este ejercicio es definir el siguiente mini lenguaje de expresiones aritmeticas y booleanas MinEAB en JPP:

$$e ::= n \ — \ true \ — \ f\,alse \ — \ e + e \ — \ e < e$$

Se busca que se sigan los siguientes puntos.

   a) Defina una clase Expr que incluya los siguientes metodos:

   - isAtom que devuelve true si la expresion no tiene subexpresiones propias.
   - lsub que devuelve la subexpresion izquierda de una expresion no atomica.
   - rsub que devuelve la subexpresion derecha de una expresion no atomica.
   - eval que devuelve el valor de la expresion.

   Esta clase debe ser abstracta en el sentido de que no tiene atributos y por lo tanto sus metodos no hacen nada pero deben ser definidos. Es decir, se considera a Expr como una interfaz.

b) Defina una clase NumExpr que extienda a Expr y de forma que sus instancias correspondan a numeros.

c) Defina una clase BoolExpr que extienda a Expr y de forma que sus instancias correspondan a booleanos.

d) Defina una clase SumExpr que extienda a Expr y de forma que sus instancias correspondan a sumas.

e) Defina una clase LTExpr que extienda a Expr y de forma que sus instancias correspondan a comparaciones de orden.

f) De ejemplos de instancias de cada una de las clases.
Puede suponer definida una clase Value cuyas instancias sean los valores del lenguaje, ya sea naturales o booleanos. Es decir, Value es esencialmente el tipo Nat + Bool. Ademas de otras clases primitivas con los metodos que requiera, especificandolos previamente. Tambien puede utilizar convenientemente la constante de error en cualquier metodo.