Listado de Criterios y Convenciones
Capítulo 8: El Formalismo Básico

Objetivo:

La intención de este listado de criterios y convenciones es disponer de forma unificada, y conjunta de todos aquellos criterios que se han acordado para el desarrollo de la práctica. Cada uno de los items de este documento explican de forma clara y detallada la convención adoptada por la cátedra respecto de temas como: definición de tipos de funciones, expresiones booleanas aceptas en las guardas, pattern matching de funciones, determinismo de funciones, conjunción y disyunción para el caso de evaluación de expresiones no definidas, etc.

Este material complementa el material teórico práctico del capítulo 8. Cualquier duda al respecto de los ítems, o en el caso de necesitar un explicación más detallada de los mismos, preguntar a los docentes y auxiliares de la cátedra.

---

1. El tipo para las tuplas lo notamos (A,B) en lugar de A x B

2. En una definición por pattern matching los patrones NO se evalúan en orden, como sí ocurre en Haskell. Por lo tanto, si un argumento encaja con más de un patrón, la función que se define debe devolver el mismo valor todas las cláusulas donde el argumento encaja con el patrón (en caso contrario no sería función). Por esto, lo más conveniente es escribir patrones disjuntos.

3. Lo mismo vale para las guardas en una definición por análisis por casos: si los argumentos de una función verifican más de una guarda, la función debe devolver el mismo valor en todas ellas.

4. En una definición por análisis por casos SI se puede tener un único caso.

5. Una definición local, dentro de una definición por análisis por casos, tiene ámbito en todas las guardas y expresiones superiores a donde se escribe.

6. NO pueden usar cuantificaciones en las definiciones de funciones. El formalismo básico, cuando habla sobre la formación de elementos de tipo Bool, solamente menciona las constantes True y False, variables de tipo Bool y los conectores /\, \/, ¬, ≡, ≢, ⇒, ⇐ .

7. div y mod las tomamos como funciones prefijas que toman los argumentos de a uno y también las aceptamos de manera infija, entre medio de los operandos.

8. Currificación NO lo estamos dando. No se da ni en teoría ni se evalúa. Entonces solamente diferenciamos una función:

   f: Int -> Int -> Bool
   f.x.y = x+y

   de una:

   f': (Int, Int) -> Bool
   f'.(x,y) = x+y

   diciendo que la primera toma dos argumentos y la segunda solamente uno.

9. Las expresiones aritméticas las tomamos como del tipo más restrictivo posible, siendo Nat más restrictivo que Int e Int más restrictivo que Real. Buscar lo más restrictivo lo vamos a hacer solamente sobre las expresiones aritméticas. Por ejemplo, si queremos tipar la función g dada por la definición g.(x,y) = x vamos a dar el tipo g : (A,B) -> A, donde vamos a dar justamente el tipo más general para ambas componentes de la tupla que g toma como argumento.

10. Los tipos de las funciones aritméticas son los siguientes:

+: Nat -> Nat -> Nat +: Int -> Int -> Int +: Real -> Real -> Real	-: Nat -> Nat -> Nat -: Int -> Int -> Int -: Real -> Real -> Real
x: Nat -> Nat -> Nat x: Int -> Int -> Int x: Real -> Real -> Real	/: Nat -> Nat -> Real /: Int -> Int -> Real /: Real -> Real -> Real
div: Int -> Int -> Int div: Nat -> Nat -> Nat	mod: Nat -> Nat -> Nat mod: Int -> Int -> Int

Obviamente tenemos más combinaciones posibles para +, -, x. Elegimos las tres

que se muestran (con los argumentos y valor de retorno del mismo tipo), simplemente para facilitar la inferencia de tipo.

Los tipos para la división /, siempre devuelven un Real, ya que dividir dos naturales o dividir dos enteros puede no se exacto y en ese caso retornamos un Real.
Para div y mod adoptamos dos alternativas para cada una, siempre del mismo tipo argumentos y valor del retorno, y siendo posibles Nat e Int.

11. El punto decimal es . (punto) y no , (coma).

12. Asumimos que el /\ y el \/ tienen cortocircuito:

    - evaluando una conjunción de izq. a der., si se encuentra un operando que evalúa a false, toda la expresión evalúa a false.

    - evaluando una disyunción de izq. a der., si se encuentra un operando que evalúa a true, toda la expresión evalúa a true.

    Por lo tanto, si definimos:

    $f.xs \doteq (\neg (xs = [\,]) \wedge hd.xs = 1 \rightarrow E_0$
    $\Box$
    $B_1 \rightarrow E_1$
)

    cuando f toma la lista vacía la guarda equivale a:

    $\neg ([\,] = [\,]) \wedge hd.\,[\,]$

    los cuál es False /\ "Indefinido", pero el segundo operando del /\ no llega a evaluarse.