Práctico 0 - Evaluador de Expresiones de Conjuntos Finitos de Enteros en Haskell

Teoría de la Computación
Universidad ORT Uruguay

Marzo 2025

El objetivo de este práctico es codificar[1] en Haskell un *lenguaje* de *Conjuntos finitos de enteros*:

- Sintaxis abstracta.
- Semántica operacional.

tal como han sido descriptas en el documento de especificación.

Se pide, concretamente:

1. Expresiones:
   1.1. Declarar un tipo inductivo (`data`) apropiado para representar las *expresiones de conjuntos finitos de enteros* (sintaxis abstracta).
2. Valores:
   2.1. Declarar un tipo inductivo (`data`) apropiado para representar los *valores* de esas expresiones.
3. Memoria:
   3.1. Definir un tipo (`type`) apropiado para representar a la *memoria*.
   3.2. Definir la búsqueda de una variable en la memoria (lkup: $x \overset{M}{\mapsto} v$).
   3.3. Definir la actualización de la memoria (upd: $M \prec + (x, v)$).
4. Reglas de evaluación:
   4.1. Programar las funciones auxiliares definidas en la especificación del lenguaje:

[1] Otro término técnico utilizado es *embeber*. En inglés se usan *to encode* y *to embed*.

- `belongs :: Int -> [Int] -> Bool`
- `union :: [Int] -> [Int] -> [Int]`
- `intersection :: [Int] -> [Int] -> [Int]`
- `difference :: [Int] -> [Int] -> [Int]`
- `included :: [Int] -> [Int] -> Bool`

4.2. Definir la función (parcial[2]) de *evaluación* o *función semántica*, que recibe una *memoria* y una *expresión*, y retorna la *memoria* actualizada y el *valor* asociado a la *expresión* al evaluarse sobre esa *memoria*.

5. Codificar en el lenguaje de las *expresiones de conjuntos* embebido en Haskell los siguientes conjutnos:

   - `conj1` :: E, que represente al conjunto {1,2,3}.
   - `conj2` :: E, que represente al conjunto {2,3,4}.
   - `conj3` :: E, que represente a la unión de `conj1` y `conj2` ($\{1,2,3\} \cup \{2,3,4\}$).
   - `conj4` :: E, que represente a la intersección de `conj1` y `conj2` ($\{1,2,3\} \cap \{2,3,4\}$).
   - `pert1` :: E, que represente a la expresión que dice si el entero 2 pertenece a `conj1` ($2 \in \{1,2,3\}$).
   - `pert2` :: E, que represente a la expresión que dice si el entero 3 pertenece a `conj4` ($3 \in (\{1,2,3\} \cap \{2,3,4\})$).
   - `incl1` :: E, que represente a la expresión que dice si `conj1` está incluido en `conj2` ($\{1,2,3\} \subseteq \{2,3,4\}$).
   - `incl2` :: E, que represente a la expresión que dice si `conj4` está incluido en `conj2` ($(\{1,2,3\} \cap \{2,3,4\}) \subseteq \{2,3,4\}$).
   - `incl3` :: E, que represente a la expresión que dice si `conj1` está incluido en `conj3` ($\{1,2,3\} \subseteq (\{1,2,3\} \cup \{2,3,4\})$).
   - `ass1` :: E, que represente a la expresión donde se asigna a una variable `w` el conjunto `conj1` ($w := \{1,2,3\}$).
   - `ass2` :: E, que represente a la expresión donde se asigna a una variable `x` el conjunto `conj4` ($x := \{1,2,3\} \cap \{2,3,4\}$).
   - `ass3` :: E, que represente a la expresión donde se asigna a una variable `y` el resultado de `pert2` ($y := 3 \in (\{1,2,3\} \cap \{2,3,4\})$).
   - `ass4` :: E, que represente a la expresión donde se asigna a una variable `z` el resultado de `incl2` ($z := (\{1,2,3\} \cap \{2,3,4\}) \subseteq \{2,3,4\}$).

6. Realizar todos los cambios necesarios para que el lenguaje desarrollado en las partes anteriores permita hallar el conjunto potencia (o partes) de un conjunto dado; y dados dos conjuntos determinar si son iguales.

[2]Cuando indicamos parcial, nos referímos a que puede fallar y no devolver un resultado.