Taller de Álgebra I

Clase 7 - Conjuntos

Primer cuatrimestre 2020

Problema

¿Cómo hacemos una función que calcule el conjunto de partes?

Conjunto de Partes

Sea A un conjunto. El *conjunto de partes* de A, que se nota $\mathcal{P}(A)$, es el conjunto formado por todos los subconjuntos de A, o sea el conjunto cuyos elementos son los subconjuntos de A. Es decir,

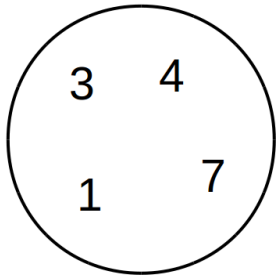
$$\mathcal{P}(A) = \{B : B \subseteq A\}$$

Sea $A = \{1, 2, 3\}$, entonces:

$$\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

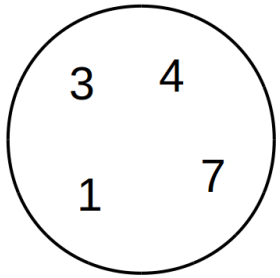
Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.



Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

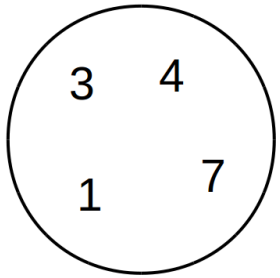


Probemos con las listas de enteros, `[Int]`

- ¿Podríamos representar este conjunto con la lista `[1,3,4,7]`?

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

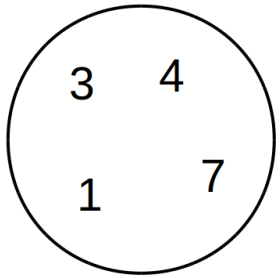


Probemos con las listas de enteros, `[Int]`

- ¿Podríamos representar este conjunto con la lista `[1,3,4,7]`?
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

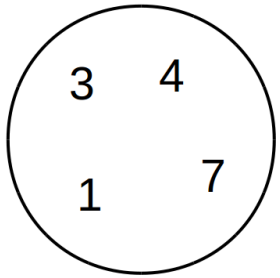


Probemos con las listas de enteros, `[Int]`

- ¿Podríamos representar este conjunto con la lista `[1,3,4,7]`?
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.
- ¿Y la lista `[1,3,4,7,7,7,1,4,7]`? ¿Sirve para representar a nuestro conjunto?

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.

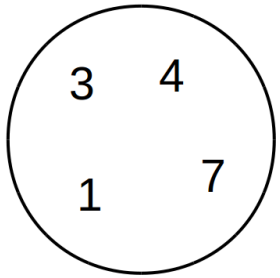


Probemos con las listas de enteros, `[Int]`

- ¿Podríamos representar este conjunto con la lista `[1,3,4,7]`?
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.
- ¿Y la lista `[1,3,4,7,7,7,1,4,7]`? ¿Sirve para representar a nuestro conjunto?
  - Las listas pueden tener elementos repetidos, pero eso no tiene sentido con conjuntos.

Conjuntos

Supongamos que queremos representar un **conjunto** de números enteros.



Probemos con las listas de enteros, `[Int]`

- ¿Podríamos representar este conjunto con la lista `[1,3,4,7]`?
  - También con `[4,1,3,7]`, `[3,7,4,1]`, `[7,3,1,4]`, ...
  - Todas estas listas son **distintas**, pero representan al **mismo** conjunto.
  - El orden de los elementos es relevante para las listas, pero no para conjuntos.
- ¿Y la lista `[1,3,4,7,7,7,1,4,7]`? ¿Sirve para representar a nuestro conjunto?
  - Las listas pueden tener elementos repetidos, pero eso no tiene sentido con conjuntos.

Vamos a usar `[Int]` para representar conjuntos de enteros, pero dejando claro que hablamos de conjuntos (sin orden ni repetidos).

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	`[]`

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	`[]`
$\{x\} \cup A$	

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	`[]`
$\{x\} \cup A$	`(:)`

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	`[]`
$\{x\} \cup A$	`(:)`

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	`[]`
$\{x\} \cup A$	`(:)`
$x \in A$	

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	`[]`
$\{x\} \cup A$	`(:)`
$x \in A$	`pertenece`

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	**Representación Haskell**
$\emptyset$	`[]`
$\{x\} \cup A$	`(:)`
$x \in A$	`pertenece`

Ejercicios

Implementemos estas operaciones:

1. `vacio :: [Int]`
2. `pertenece :: Int -> [Int] -> Bool`
3. `agregar :: Int -> [Int] -> [Int]`

Expresiones y operaciones básicas sobre conjuntos

¿Qué operaciones usamos cuando trabajamos con conjuntos? ¿Cómo se representan en Haskell?

Conjuntos	Representación Haskell
$\emptyset$	`[]`
$\{x\} \cup A$	`(:)`
$x \in A$	`pertenece`

Ejercicios

Implementemos estas operaciones:

1. `vacio :: [Int]`
2. `pertenece :: Int -> [Int] -> Bool`
3. `agregar :: Int -> [Int] -> [Int]`

Pensar

¿Cómo podemos representar la operación # (el cardinal de un conjunto)?

Renombres en Haskell

Definición de tipo usando `type`

Definamos un renombre de tipos para conjuntos: `type Set a = [a]`

- Otra forma de escribir lo mismo, pero más descriptivo.
- `type` es la palabra reservada del lenguaje, `Set` es el nombre que le pusimos nosotros.
- Si bien internamente es una lista, la idea es tratar a `Set a` como si fuera conjunto (es un contrato entre programadores).
- Si nuestra función recibe un conjunto, **vamos a suponer** que no contiene elementos repetidos. (Haskell no hace nada para verificarlo.)
- Si nuestra función devuelve un conjunto, **debemos asegurar** que no contiene elementos repetidos. (Haskell tampoco hace nada automático.)
- Además, no hace falta preocuparse por el orden de los elementos. (Haskell no lo sabe.)

Renombres en Haskell

Definición de tipo usando `type`

Definamos un renombre de tipos para conjuntos: `type Set a = [a]`

- Otra forma de escribir lo mismo, pero más descriptivo.
- `type` es la palabra reservada del lenguaje, `Set` es el nombre que le pusimos nosotros.
- Si bien internamente es una lista, la idea es tratar a `Set a` como si fuera conjunto (es un contrato entre programadores).
- Si nuestra función recibe un conjunto, **vamos a suponer** que no contiene elementos repetidos. (Haskell no hace nada para verificarlo.)
- Si nuestra función devuelve un conjunto, **debemos asegurar** que no contiene elementos repetidos. (Haskell tampoco hace nada automático.)
- Además, no hace falta preocuparse por el orden de los elementos. (Haskell no lo sabe.)

Renombres en Haskell

Definición de tipo usando `type`

Definamos un renombre de tipos para conjuntos: `type Set a = [a]`

- Otra forma de escribir lo mismo, pero más descriptivo.
- `type` es la palabra reservada del lenguaje, `Set` es el nombre que le pusimos nosotros.
- Si bien internamente es una lista, la idea es tratar a `Set a` como si fuera conjunto (es un contrato entre programadores).
- Si nuestra función recibe un conjunto, **vamos a suponer** que no contiene elementos repetidos. (Haskell no hace nada para verificarlo.)
- Si nuestra función devuelve un conjunto, **debemos asegurar** que no contiene elementos repetidos. (Haskell tampoco hace nada automático.)
- Además, no hace falta preocuparse por el orden de los elementos. (Haskell no lo sabe.)

Observación

```
1 vacio :: Set Int
2 pertenece :: Int -> Set Int -> Bool
3 agregar :: Int -> Set Int -> Set Int
```

Más operaciones sobre Conjuntos

Ejercicios

1. Implementar la función `incluido :: Set Int -> Set Int -> Bool` que determina si el primer conjunto está incluido en el segundo.
2. Implementar la función `iguales :: Set Int -> Set Int -> Bool` que determina si dos conjuntos son iguales.

Tarea: Parte I

Más ejercicios

Implementar las siguientes funciones

1. `union:: Set Int -> Set Int -> Set Int` que dado dos conjuntos, devuelve la unión entre ellos.
2. `intereseccion :: Set Int -> Set Int -> Set Int` que dado dos conjuntos, devuelve la interesección entre ellos.
3. `diferencia :: Set Int -> Set Int -> Set Int` que dado los conjuntos A y B, devuelve $A \setminus B$.
4. `diferenciaSimetrica :: Set Int -> Set Int -> Set Int` que dado los conjuntos A y B, devuelve la diferencia simétrica, es decir, $A \triangle B$.

Partes de un conjunto

Ejercicios

1. Implementar una función
   `partes :: Set Int -> Set (Set Int)` que genere el conjunto de partes de un conjunto dado.

Partes de un conjunto

Ejercicios

1. Implementar una función
`partes :: Set Int -> Set (Set Int)` que genere el conjunto de partes de un conjunto dado. Por ejemplo, todos los subconjuntos del conjunto $\{1, 2, 3\}$.

```
Ejemplo> partes [1,2,3]
[[], [1], [2], [3], [1, 2], [1,3], [2,3],[1,2,3]]
```

Tarea: Parte II

Más ejercicios

Implementar las siguientes funciones

1. `partesN :: Int -> Set (Set Int)` que genere los subconjuntos del conjunto $\{1, 2, 3, \ldots, n\}$.

```
Ejemplo> partesN 2
[[], [1], [2], [1, 2]]
```

2. `productoCartesiano :: Set Int -> Set Int -> Set (Int, Int)` que dados dos conjuntos genere todos los pares posibles (como pares de dos elementos) tomando el primer elemento del primer conjunto y el segundo elemento del segundo conjunto.

```
Ejemplo> productoCartesiano [1, 2, 3] [3, 4]
[(1, 3), (2, 3), (3, 3), (1, 4), (2, 4), (3, 4)]
```