Universidad Nacional Autónoma de México
Facultad de Ciencias

Estructuras Discretas

Práctica 7

Javier Enríquez Mendoza **Mauricio E. Hernández Olvera**

28 de Noviembre de 2018
Fecha de entrega: 9 de Diciembre de 2018

Instrucciones generales

La práctica debe resolverse en el archivo `Relacion.hs` y conservando las firmas de las funciones idénticas a las que se muestran en cada ejercicio. Cada función y definición debe estar debidamente comentada con la especificación de ésta.

Se tomará en cuenta la legibilidad y el estilo del código.

Esta práctica será considerada como de recuperación, esto quiere decir que los puntos obtenidos serán considerados como puntos extra sobre la calificación de las prácticas. Pudiendo obtener un máximo de 5.5 puntos.

Relaciones.

Utilizando la siguiente definición para **Relaciones Binarias** en Haskell, resolver los siguientes ejercicios.

```
-- Sinónimo para representar relaciones binarias.
type RelacionB a b = [(a,b)]
```

1 Operaciones Entre Relaciones

Ejercicio 1.1 (0.5 pts.) Definir la función `unionR` que calcula la unión de dos relaciones binarias.

```
unionR :: Eq a => Eq b => RelacionB a b -> RelacionB a b -> RelacionB a b
```

```
> unionR [(1,1),(2,2),(3,3)] [(1,1),(2,2),(3,3)]
[(1,1),(2,2),(3,3)]
> unionR [(1,1),(1,2),(1,3)] [(2,1),(3,1)]
[(1,1),(1,2),(1,3),(2,1),(3,1)]
```

Ejercicio 1.2 (0.5 pts.) Definir la función `interseccion` que regresa la intersección de dos relaciones binarias.

```
interseccion :: Eq a => Eq b =>
                RelacionB a b -> RelacionB a b -> RelacionB a b
```

```
> interseccion [(1,1),(2,2),(3,3)] [(1,1),(2,2),(3,3)]
[ (1,1),(2,2),(3,3)]
> interseccion [(1,1),(1,2),(1,3)] [(2,1),(3,1)]
[]
```

Ejercicio 1.3 (0.5 pts.) Definir la función `diferencia` que calcula la diferencia de dos relaciones binarias.

```
diferencia :: Eq a => Eq b =>
              RelacionB a b -> RelacionB a b -> RelacionB a b
```

```
> diferencia [(1,1),(2,2),(3,3)] [(1,1),(2,2),(3,3)]
[]
> diferencia [(1,1),(1,2),(1,3)] [(2,1),(3,1)]
[(1,1),(1,2),(1,3)]
```

Ejercicio 1.4 (0.5 pts.) Definir la función `productoCartesiano` que calcula el producto cartesiano de dos conjuntos (listas) como una relación binaria.

```
productoCartesiano :: [a] -> [b] -> RelacionB a b
```

```
> productoCartesiano [1,2] ['a','b']
[(1,'a'), (2,'a'),(1,'b'),(2,'b')]
> productoCartesiano [1,2,3] [2,3]
[(1,2),(1,3),(2,2),(2,3),(3,2),(3,3)]
```

2 Propiedades

Ejercicio 2.1 (0.5 pts.) Definir la función `simetrica` que dice si una relación es simétrica.

```
simetrica :: Eq a => RelacionB a a -> Bool
```

```
> simetrica [(1,2),(2,1),(3,3)]
True
> simetrica [(1,2),(2,3),(3,1),(1,1)]
False
```

Ejercicio 2.2 (0.5 pts.) Definir la función `antisimetrica` que dice si una relación es antisimétrica.

```
antisimetrica :: Eq a => RelacionB a a -> Bool
```

```
> antisimetrica [(1,2),(2,1),(3,3)]
False
> antisimetrica [(1,2),(2,3),(3,1),(1,1)]
True
```

Ejercicio 2.3 (0.5 pts.) Definir la función `reflexiva` que dice si una relación es reflexiva sobre el conjunto (como lista) que recibe.

```
reflexiva :: Eq a => [a] -> RelacionB a a -> Bool
```

```
> reflexiva [1,2,3,4] [(1,1),(2,2),(3,3),(1,2)]
False
> reflexiva [1,2,3] [(1,1),(2,2),(3,3),(1,2)]
True
```

Ejercicio 2.4 (0.5 pts.) Definir la función `antireflexiva` que dice si una relación es antireflexiva.

```
antirelfexiva :: Eq a => RelacionB a a -> Bool
```

```
> antireflexiva [(1,1),(2,2),(3,3),(1,2)]
False
> antireflexiva [(1,2),(2,3),(3,1)]
True
```

Ejercicio 2.5 (0.5 pts.) Definir la función `asimetrica` que dice si una relación es asimétrica.

```
asimetrica :: Eq a => RelacionB a a -> Bool
```

```
> asimetrica [(1,1),(2,2),(3,3),(1,2)]
False
> asimetrica [(1,2),(2,3),(3,1)]
True
```

3 Operaciones sobre Relaciones

Ejercicio 3.1 (0.5 pts.) Definir la función `inversa` que calcula la inversa de una relación binaria.

```
inversa :: Eq a => RelacionB a a -> RelacionB a a
```

```
> inversa [(1,2),(2,1),(3,3)]
[(2,1),(1,2),(3,3)]
> inversa [(1,2),(2,3),(3,1),(1,1)]
[(2,1),(3,2),(1,3),(1,1)]
```

Ejercicio 3.2 (0.5 pts.) Definir la función `complemento` que calcula el complemento de una relación binaria sobre el conjunto que recibe (como lista).

```
complemento :: Eq a =>[a] -> RelacionB a a -> RelacionB a a
```

```
> complemento [1,2,3] [(1,2),(2,1),(3,3)]
[(1,1),(1,3),(2,2),(2,3),(3,1),(3,2)]
> complemento [1,2,3,4] [(1,2),(2,3),(3,1),(1,1)]
[(1,3),(1,4),(2,1),(2,2),(2,4),(3,2),(3,3),(3,4),(4,1),(4,2),(4,3),(4,4)]
```

Entrega

- La entrega se realiza mediante correo electrónico a la dirección de los ayudantes de laboratorio (javierem_94@ciencias.unam.mx y mauriciohdez08@ciencias.unam.mx).
- Es **necesario** que el correo se envíe a ambos ayudantes.
- La practica deberá ser entregada en equipos de máximo 3 personas.
- Se debe entregar un directorio `numeroCuenta_P07`, dónde `numeroCuenta` es el número de cuenta de un integrante del equipo. Dentro del directorio se debe incluir:
  - Un archivo readme.txt con los nombres y números de cuenta de los alumnos, comentarios, opiniones, críticas o ideas sobre la práctica.
  - Los archivos requeridos en la práctica. Debe enviarse código lo más limpio posible.
- Los archivos requeridos para esta práctica son: `Relacion.hs`.
- El directorio se deberá comprimir en un archivo con nombre `numeroCuenta_P07.tar.gz`, dónde `numeroCuenta` es el número de cuenta de un integrante del equipo.
- Únicamente **un integrante** del equipo deberá enviar el correo con la práctica.
- El asunto del correo debe ser [ED-20191-P07].
- Se recibirá la práctica hasta las 23:59:59 horas del día fijado como fecha de entrega, cualquier práctica recibida después no será tomada en cuenta.
- **Cualquier práctica total o parcialmente plagiada, será calificada automáticamente con cero y no se aceptarán más prácticas durante el semestre.**