Facultad de Ciencias - UNAM
Lógica Compuacional 2023-2
Práctica 1: Introducción a Haskell

Javier Enríquez Mendoza
Ramón Arenas Ayala
Oscar Fernando Millan Pimentel
Kevin Axel Prestegui Ramos

06 de Febero de 2023
Fecha de entrega: 19 de Febrero de 2023

1 Introducción

"Haskell es un lenguaje de programación estandarizado multi-propósito, ***puramente funcional****, con* ***evaluación perezosa*** *y memorizada, y* ***con tipado estático****. Su nombre se debe al lógico estadounidense Haskell Curry, debido a su aportación al* ***cálculo lambda****, el cual tiene gran influencia en el lenguaje".*

A lo largo del curso de Lógica Computacional, utilizaremos Haskell como herramienta para la implementación de los conceptos de diseño de lenguajes que se estudiaran en la materia. Los conceptos dados en la definición anterior son de suma importancia para poder comprender de mejor manera el funcionamiento del lenguaje.

Puramente Funcional (Purely functional): Todas las funciones en Haskell son funciones en el sentido matemático. No hay sentencias ni instrucciones, solo expresiones que no pueden mutar variables (locales o globales) ni acceder a estados como el tiempo o los números aleatorios.

Evalución Perezosa (Lazy Evaluation): Es una estrategia de evaluación que retrasa el cálculo de una expresión hasta que su valor sea necesario, y que también evita repetir la evaluación en caso de ser necesaria en posteriores ocasiones.

Tipado estático (Statically typed): Cada expresión en Haskell tiene un tipo que se determina en tiempo de compilación. Todos los tipos compuestos por la aplicación de la función tienen que coincidir. Si no lo hacen, el programa será rechazado por el compilador. Los tipos se convierten no solo en una forma de garantía, sino en un lenguaje para expresar la construcción de programas.

2 Objetivo

El objetivo de esta práctica, es poner en práctica los conceptos revisados durante las sesiones de laboratorio sobre la programación en el lenguaje Haskell, así como el que los alumnos refuercen el entendimiento de estos y puedan realizar programas complejos utilizando este lenguaje.

3 Justificación

El motivo de ser de esta práctica, es el hecho de que como se mencionó anteriormente, Haskell será nuestra herramienta de trabajo durante el curso, por lo que es de gran importancia que los alumnos adquieran un nivel de entendimiento sobre los conceptos del lenguaje para el poder desarrollar las prácticas posteriores de manera correcta.

Es importante recalcar, que como se mencionó durante las sesiones de clase, este no es un curso de programación en Haskell, ni nos enfocaremos solo en conceptos de programación de Haskell, este será nuestra herramienta para implementar conceptos de lógica computacional.

4 Desarrollo de la práctica

4.1 Tipos de datos Algebraicos

Se realizarán varios ejercicios básicos donde se pondrán en práctica los conceptos de tipos de datos algebraicos, y la capacidad de abstracción.

1. Crea un tipo de dato llamado `Shape` que represente a las figuras geométricas. Este tipo de dato debe poder representar los *círculos, cuadrados, rectángulos, triángulos y trapecios.* Para esto deben utilizar como constructores `Circle, Square, Rectangle, Triangle, Trapeze.` Posteriormente se debe definir las funciones `area :: Shape -> Float` que calcula el área de una figura, `perimeter :: Shape -> Float` que calcula el perímetro de una figura e `instance Ord Shape` una instancia de la clase Ord para nuestras figuras, donde el criterio de comparación sea el área de estas.

2. Crea un tipo de dato llamado `Point` que represente un punto en un plano cartesiano. Para esto deben utilizar un sinonimo y utilizar el tipo de dato `Float`. Posteriormente define las funciones `distance :: Point -> Point -> Float` para calcular las distancia entre dos puntos, y `fromO :: Point -> Float` para calcular la distancia de un punto al origen.

3. En la isla Funcional, la cual cuenta con un área de terreno infinita, habita la tribu Haskelliums. Esta tribu tiene algunas costumbres parecidas a las nuestras, pero otras que son muy distintas. Por ejemplo:

- Tienen un nombre, un primer apellido y un segundo apellido. Cuando tienen hijos, los padres eligen el nombre, el primer apellido del hijo es el primer apellido del primer padre y el segundo apellido es el primer apellido del segundo padre, además de que este hijo al nacer vivirá en casa de sus padres.
- Ellos viven casas con formas de figuras geométricas. Todas las casas son de un solo piso con paredes de dos metros y medio de alto.
- En el centro de la isla, está la plaza que es donde trabajan todos los Haskelliums. Si ellos viven cerca de la plaza (menos de 300 u) se van en bicicleta, pero si viven lejos utilizan moto. La bicicleta se mueve con una velocidad a 30u/t, mientras que la moto a una velocidad de 70u/t.

Conociendo toda esta información acerca de los Haskelliums, debemos crear un tipo de dato llamado `Haskellium` que represente a un Haskellium, almacenando su ***nombre, primer apellido, segundo apellido, ubicación de su casa y las características de su casa***. Estos se deben almacenar bajo el nombre de `name,lastName1, lastName2, location y houseShape` Para esto debemos usar los tipos de dato creados en los ejercicios anteriores (las figuras y los puntos). Posteriormente crearemos las siguientes funciones:

- `son :: Haskellium -> Haskellium -> String -> Haskellium`
  Dados dos Haskelliums y un String de nombre, regresa un Haskellium que sería hijo de los dos Haskelliums con el nombre dado.
- `houseCost :: Haskellium -> Float`
  Dado un Haskellium, calcula las unidades necesarias para construir su casa, contemplando las paredes y el techo. Para esto se desestimará el grosor de los muros, ya que solo nos interesa el área de las paredes. Notar que se dijo la áltura de las paredes en la descripción de los Haskelliums.
- `timeToWork :: Haskellium -> Float`
  Dado un Haskellium, se calcula el tiempo en unidades t, que le cuesta llegar a su trabajo. Contemplando si este va en bicicleta o en moto.

NOTA: para este ejercicio es importante utilizar las funciones definidas en ejercicios pasados de ser el caso.

4.2 Listas y Funciones

Ahora que se ha comprendido mas a fondo el funcionamineto de los tipos de datos algebraicos, se realizarán ejercicios para poner a prueba el razonamiento lógico y el entendimiento del paradigma funcional.

1. `isPal :: String -> Bool`
   Implementar la función `isPal`, la cual dada una cadena de texto nos dice si es o no un palindromo.
2. `concat' :: [[a]] -> [a]`
   Implementar recursivamente la función `concat'`, la cual dadas una lista de listas, nos regresa la concatenación de todas las listas contenidas por esta.

3. `pascalN :: Int -> [Int]`
   Implementar la funcion `pascalN`, tal que pascalN n, nos regresa la n-ésima fila del triángulo de Pascal.

4. `reversaFr :: [a] ->[a]`
   Implementar usando foldr la función `reversaFr`, la cuál dada una lista nos regresa la lista con los mismos elementos pero en orden opuesto.

5 Especificaciones de entrega

- **GitHub Classroom:** Para realizar la entrega de la práctica se utilizará la plataforma de GitHub Classroom. El link para el assigment es: `https://classroom.github.com/a/MglxHx-8`
- **Equipos:** La práctica puede realizarse en equipos de hasta tres personas. Esto siempre y cuando se respete la política de trabajo colaborativo y podamos ver que ambos miembros del equipo contribuyen en la elaboración de la práctica. En caso contrario las prácticas serán exclusivamente individuales.
- **Fecha de entrega:** La fecha de entrega será la indicada al principio de este documento. No se recibirá ninguna práctica en fechas posteriores a la fecha indicada al menos que el profesor indique una prórroga.
- **Flujo de trabajo:** Se recomienda llevar un flujo de trabajo adecuado, en el cual se realicen commits constantemente y no hacer todo al final y en un solo commit. Esto ya que de esta manera podemos comprobar la colaboración de todos los miembros del equipo, dar feedback a lo largo del tiempo de trabajo para mejorar la entrega de las prácticas, facilitar la resolución de dudas, etc.
- **Evaluación:** Para la evaluación la práctica se sometera a un conjunto de pruebas, y además se realizara una revisión del código. Comprobandose que se cumplan las condiciones indicadas para cada ejercicio.
- **Compilado:** Culaquier prática que al ser descargada no compile, **será evaluada con una calificación de 0**.
- **Datos personales:** Se debe agregar un documento `README.md` en el directorio raíz de sus repositorios, con sus datos personales. En caso de no ser indicados, la calificación no podrá ser asignada.
- **Limpieza y estructura:** en caso de no tener una estructura clara y limpieza dentro de sus repositorios, la calificación se verá afectada negativamente.

6 Sugerencias y Notas

Dudas: Pueden preguntar sus dudas a través de telegram, o correo por el canal que ustedes deseen. Pero les recomendamos preguntar a través del grupo de telegram para que sus demás compañeros también puedan aclarar dudas similares a la suya.

Limitaciones: Pueden utilizar funciones predefinidas siempre y cuando su utilización no derive en que se pierda el objetivo académico del ejercicio que se está realizando. Si se tiene duda acerca del poder utilizar algo pueden preguntarlo.

Buena suerte a todos! ☺☺☺