Facultad de Ciencias - UNAM
Lógica Computacional 2026-2

Práctica 1: Recursión Estructural, Listas, Árboles y Definiciones Inductivas

Marco Vladimir Lemus Yañez
Fernando Cruz Pineda

Fecha de publicación: 17 02 2026
Fecha de entrega: 02 03 2026

1. Introducción

En programación funcional, los tipos algebraicos permiten modelar estructuras matemáticas definidas de manera inductiva. Estas estructuras pueden manipularse mediante recursión estructural, la cual corresponde al principio de inducción estructural en matemáticas.

En esta práctica trabajaremos con:

- Definiciones propias de listas
- Funciones recursivas estructurales
- Uso de fold como esquema general de recursión
- Definición y manipulación de árboles
- Definiciones inductivas de estructuras lógicas

Estas ideas serán fundamentales para trabajar posteriormente con sintaxis y semántica de lógica proposicional.

2. Objetivo

El objetivo de esta práctica es que el alumno:

- Implemente funciones recursivas sobre estructuras inductivas
- Comprenda fold como principio general de recursión

- Implemente funciones sobre árboles binarios
- Defina estructuras inductivas para representar objetos lógicos

3. Justificación

Las estructuras inductivas son fundamentales en ciencias de la computación teórica. Permiten modelar:

- Sintaxis formal
- Estructuras de datos recursivas
- Lenguajes formales
- Sistemas de prueba

La comprensión de recursión estructural es necesaria para el diseño de algoritmos correctos y para la formalización matemática de programas.

4. Desarrollo de la Práctica

4.1. Listas Definidas por el Usuario

Considere la siguiente definición:

```
data List a = Nil | Cons a (List a)
```

1. Implemente la función:

   ```
   reverseL :: List a -> List a
   ```

2. Implemente la función:

   ```
   zip :: (a -> b -> c) -> List a -> List b -> List c
   ```

   Que recibe una función f que opera sobre dos elementos y dos listas l_1, l_2 del mismo tamaño, regresando una lista de $L_3 = [f(l_1[i], l_2[i])]$

4.2. Fold sobre Listas

Considere la función vista en clase:

```
foldrL :: (a -> b -> b) -> b -> List a -> b
```

1. Implemente:

   ```
   lengthL :: List a -> Int
   ```

   usando fold.

2. Implemente:

   ```
   sumL :: List Int -> Int
   ```

   usando fold.

3. Implemente map usando fold.

4.3. Árboles Binarios

Considere:

```
data Tree a =
    Empty
  | Node a (Tree a) (Tree a)
```

1. Implemente:

   ```
   nhojas :: Tree a -> Int
   ```

   que obtiene el número de hojas de un árbol.

2. Implemente recorrido inorder:

   ```
   inorder :: Tree a -> [a]
   ```

3. Implemente búsqueda DFS:

   ```
   dfs :: Eq a => a -> Tree a -> Bool
   ```

4.4. Estructuras Inductivas Lógicas

Defina un tipo de datos para fórmulas proposicionales como el visto en la clase de teoría.

(Punto extra) Definir una función que calcule la profundidad de una fórmula.

5. Especificaciones de Entrega

- **Forma de trabajo:** En equipo de máximo 3 integrantes.
- **Formato de entrega:** Archivo `.hs`
- **Medio de entrega:** TBD.
- **Compilación:** El código debe compilar sin errores en GHC.

6. Sugerencias y Notas

- Probar funciones en GHCi.
- Usar firmas de tipos siempre.