Notas de Clase 1 — Introducción a Haskell

Curso: Lógica Computacional
Semestre: 2026-2
Profesores: Marco Vladimir Lemus Yañez
Fernando Cruz Pineda
Fecha: 9 de febrero de 2026

---

1 Objetivo de la Sesión

Introducir programación funcional en Haskell como herramienta para modelar definiciones matemáticas, en particular definiciones recursivas e inductivas.

2 Herramientas

Durante el curso utilizaremos:

- GHC — Compilador de Haskell
- GHCi — Intérprete interactivo

Sitio oficial:

`https://www.haskell.org/ghcup/`

Comandos básicos en GHCi

```
:t expresion
:load archivo.hs
:reload
:quit
```

Ejemplos:

```
:t True
:t not
:t (+)
```

3 Recursión sobre Números

La recursión permite definir funciones reduciendo el problema a instancias más pequeñas del mismo problema.

Generalmente una función recursiva contiene:

- Caso(s) base
- Caso recursivo

3.1 Paridad

```
par 0 = True
par 1 = False
par n = par (n-2)
```

Ejemplos:

```
par 10
par 7
```

3.2 Potencias de 2

```
pow2 0 = 1
pow2 n = 2 * pow2 (n-1)
```

Ejemplos:

```
pow2 5
pow2 8
```

3.3 División entera entre 2

```
div2 0 = 0
div2 1 = 0
div2 n = 1 + div2 (n-2)
```

Ejemplos:

```
div2 10
div2 7
```

4 Listas

Las listas son estructuras inductivas definidas conceptualmente por:

```
[]
x : xs
```

4.1 Longitud

```
len [] = 0
len (_:xs) = 1 + len xs
```

4.2 Suma de elementos

```
sumL [] = 0
sumL (x:xs) = x + sumL xs
```

4.3 Duplicar elementos

```
dup [] = []
dup (x:xs) = (2*x) : dup xs
```

5 Tipos Algebraicos

En Haskell es posible definir nuevos tipos de datos a partir de constructores.

Este tipo de definiciones corresponden a lo que en matemáticas se conoce como definiciones inductivas.

5.1 Idea General

Un tipo algebraico se define indicando:

- Constructores base
- Constructores recursivos

Los constructores permiten generar todos los elementos del tipo.

Esto es análogo a la forma en que en matemáticas se definen conjuntos inductivamente.

5.2 Ejemplo: Números Naturales

Una definición inductiva típica de los números naturales establece:

- 0 es natural
- Si n es natural, entonces su sucesor también es natural

En Haskell esto se puede expresar como:

```
data Nat = Z | S Nat
```

Donde:

- `Z` representa el cero
- `S n` representa el sucesor de n

5.3 Recursión sobre Tipos Algebraicos

Las funciones sobre tipos definidos inductivamente se construyen usando pattern matching sobre sus constructores.

Por ejemplo, para convertir un número natural a entero:

```
natToInt Z = 0
natToInt (S n) = 1 + natToInt n
```

Observamos que:

- El caso base corresponde al constructor base
- El caso recursivo corresponde al constructor recursivo

5.4 Ejemplo: Suma de Naturales

```
add Z n = n
add (S m) n = S (add m n)
```

5.5 Relación con Inducción Estructural

La forma de definir funciones sobre tipos algebraicos sigue el mismo esquema que la inducción estructural:

- Se define el resultado para los casos base
- Se define el resultado suponiendo resuelto el caso recursivo

Esto permite construir funciones correctas directamente a partir de la estructura del dato.

5.6 Idea Conceptual

La programación funcional permite definir estructuras matemáticas y operar sobre ellas utilizando el mismo principio lógico que se utiliza en demostraciones por inducción.