Universidad del Valle de Guatemala

Facultad de Ingeniería

Ciencias de la Computación y Tecnologías de la Información

Laboratorio 12

Teoría de la Computación

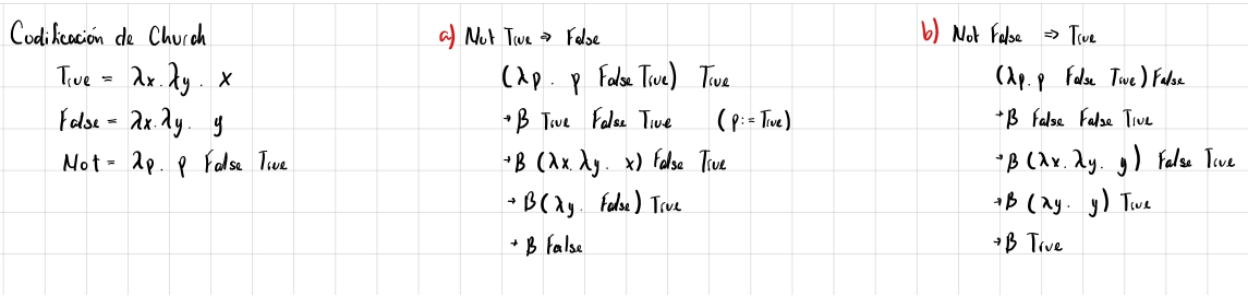
José Luis Gramajo Moraga, Carné 22907

05 de noviembre de 2025, Guatemala, Guatemala

Problema 1: 25%

- Escriba la reducción-β de la operación lógica NOT
- Escriba y explique como se vería la recursión y los ciclos.
- Explique cuando es prudente usar este tipo de programación y cuando no. De un ejemplo para cada caso.

- **Reducción -B de la operación lógica NOT**



- **¿Cómo se ven recursión y ciclos en cálculo-λ / programación funcional?**

Todo se expresa con funciones. La recursión se obtiene con un combinador de punto fijo.

Recursión con el combinador Y

- Y ≡ λf. (λx. f (x x)) (λx. f (x x)) (fija: Y F = F (Y F))

Patrón general para definir una función recursiva g:

1. Escribo un funcorpo (no recursivo) G que recibe como parámetro la "versión recursiva":
    - G ≡ λrec. λn. … (usa rec en llamadas recursivas) …
2. La función recursiva final es g ≡ Y G

"Ciclos" (while/for) como recursión de cola

Un while cond do step puede representarse de forma puramente funcional:

- WHILE ≡ λcond. λstep. λstate. IF (cond state) (WHILE cond step (step state)) state

- **¿Cuándo conviene este estilo y cuándo no?**

Cuando sí:

- Transformaciones de datos inmutables, composición de funciones, razonamiento matemático claro.
- Paralelismo y concurrencia: funciones puras no comparten estado.
- Verificación/razonamiento: el estilo funcional se aproxima al cálculo-λ y facilita pruebas.

Cuando no:

- Algoritmos que dependen fuertemente de mutación in-place por rendimiento/memoria, p. ej. DP en matrices muy grandes, simulaciones numéricas de alto desempeño o estructuras que necesitan actualización destructiva.
- I/O altamente interactivo/estado compartido sin abstracciones funcionales (se complica, aunque existen técnicas como monadas, *effects*, etc.).