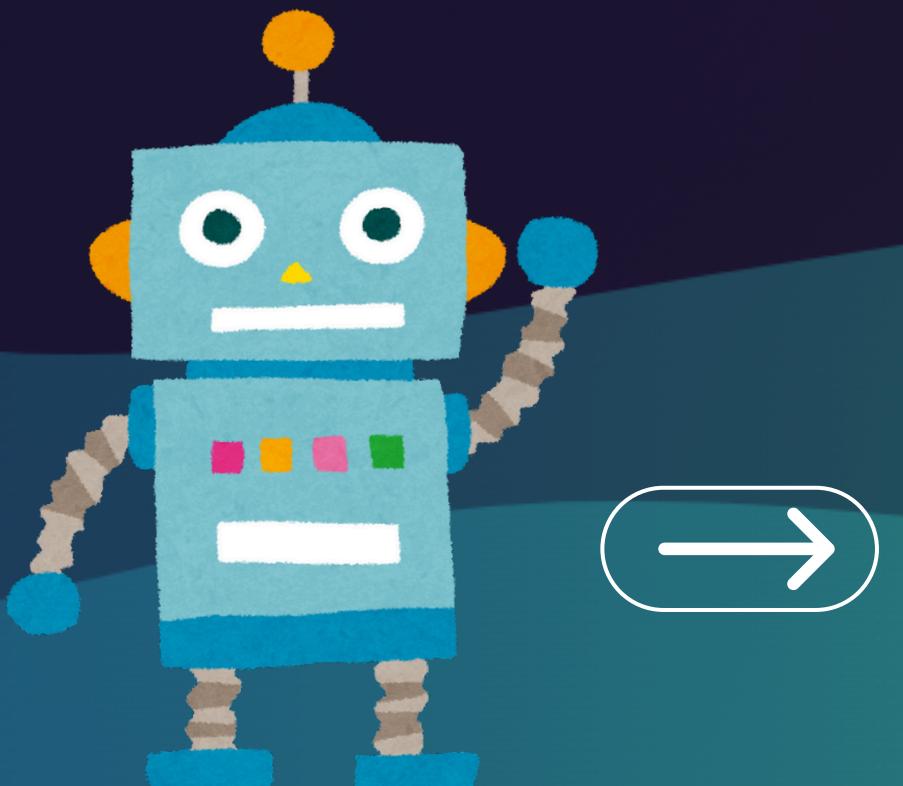

OP. FUNCIONALES

PIONEROS DE LAS OPERACIONES FUNCIONES

Integrantes:

Gastelum Fierro Brian de Jesus (373408)
Soto Ortiz Kevin Humbertio(377689)
Angel Rey Rodriguez(377864)
Adrian Balderas Rosas(373488)
Jorge Abraham Garcia Sojo (373267)



Introducción

En el cálculo lambda, el cual es una base teórica de la programación funcional y la computación, encontramos tres conceptos esenciales para comprender cómo se realizan las reducciones o evaluaciones de expresiones:

- Redex (Reducible Expression): Una expresión que puede ser reducida.
- Regla- β (Beta-Reducción): La regla que define cómo se realiza esa reducción.
- Reductum: El resultado obtenido tras aplicar la reducción.

REDEX

una subexpresión que se puede simplificar mediante la aplicación de una regla de reducción. Es una expresión a la que se le puede aplicar una operación o sustitución, como una llamada a una función con todos sus argumentos listos o una fórmula con valores reemplazables.

Comprender los conceptos fundamentales de la reducción en el cálculo lambda y su papel en la evaluación de expresiones.

¿Qué es un Redex?

Un "redex" es un término en el cálculo lambda que puede ser reducido aplicando una regla de evaluación, específicamente la regla beta.

Forma General: Siempre tiene la forma de una aplicación de función donde el primer término es una abstracción lambda (una función).

- Un redex tiene la forma:
- $(\lambda x.M) N$

Donde:

- $\lambda x.M$ es una abstracción lambda (una función).
- N es el argumento aplicado a la función.

Propósito: Los redexes son los "bloques de evaluación" que permiten simplificar expresiones complejas paso a paso.

Regla B

Es la regla fundamental de evaluación en el cálculo lambda. Permite **aplicar** una función a su argumento, sustituyendo todas las ocurrencias de la variable ligada en el cuerpo de la función por el argumento

¿Qué es la regla B?

La regla beta es la regla fundamental de computación en el cálculo lambda. Es el equivalente a "ejecutar una función" o "llamar una función con un argumento" en los lenguajes de programación tradicionales.

Fórmula:

- $(\lambda x.M) N \rightarrow M [x := N]$
- Donde $M [x := N]$ significa "sustituye todas las ocurrencias libres de x en M por N ".

Propósito: Es el mecanismo principal para ejecutar programas en el cálculo lambda, simulando la aplicación de funciones.

Reductum

Es el resultado de aplicar una regla de reducción (como la beta) a un redex. Es la expresión simplificada que se obtiene después del paso de reducción.

¿Qué es reductum?

El reductum es el resultado o la expresión transformada que obtenemos después de aplicar una regla de reducción a un redex. Es la "versión simplificada" después de un paso de cálculo.

Relación:

- Redex + Regla-Beta → Reductum

Propósito: Representa el progreso hacia una forma irreducible (forma normal) de la expresión.

Ejemplo 1: Sustitución Simple

- Expresión: $(\lambda x. x) 5$

- **Identificación del Redex:**

La expresión completa $(\lambda x. x) 5$ es un Redex.

Función: $(\lambda x. x)$ (La función identidad)

Argumento: 5

- **Aplicación de la Regla- β :**

Sustituimos x en el cuerpo x por el argumento 5.

$x[x := 5]$

- **Reductum:**

5

Ejemplo 2: Una Expresión Aritmética

- Expresión: $(\lambda x. x + 2) 3$

- **Identificación del Redex:**

La expresión completa $(\lambda x. x + 2) 3$ es un Redex.

Función: $(\lambda x. x + 2)$

Argumento: 3

- **Aplicación de la Regla- β :**

Sustituimos x en el cuerpo $x + 2$ por 3.

$(x + 2)[x := 3]$

- **Reductum:**

3 + 2, que se simplifica a 5.

Ejemplo 3: Variables Múltiples y No Sustitución

Expresión: $(\lambda x. x * y) 4$

Identificación del Redex:

La expresión completa $(\lambda x. x * y) 4$ es un Redex.

Función: $(\lambda x. x * y)$

Argumento: 4

Aplicación de la Regla- β :

Sustituimos solo la variable x en el cuerpo $x * y$. La variable y se conoce como "variable libre" en este contexto y no se ve afectada.

$(x * y)[x := 4]$

Reductum:

$4 * y$

Ejemplo 4: Aplicación de una Función a otra Función

En el Cálculo Lambda, las funciones son "ciudadanos de primera clase", lo que significa que pueden ser pasadas como argumentos.

Expresión: $(\lambda f. f 3) (\lambda x. x + 1)$

Identificación del Redex:

Toda la expresión es un Redex.

Función: $(\lambda f. f 3)$

Argumento: $(\lambda x. x + 1)$ (¡Otra función!)

Aplicación de la Regla- β :

Sustituimos f en el cuerpo $f 3$ por el argumento $(\lambda x. x + 1)$.

$(f 3)[f := (\lambda x. x + 1)]$

Reductum (Primer Paso):

$(\lambda x. x + 1) 3$

El resultado (el reductum) es a su vez un nuevo Redex. Podemos seguir reduciendo.

Segunda Reducción:

Redex: $(\lambda x. x + 1) 3$

Regla- β : $(x + 1)[x := 3]$

Reductum Final: $3 + 1$, que es 4.

Ejemplo 5: Redex Anidado (Orden de Reducción)

A veces hay más de un Redex en una expresión.

Expresión: $(\lambda x. x^* 2) ((\lambda y. y + 1) 5)$

Aquí tenemos dos Redexes:

El Redex interno: $(\lambda y. y + 1) 5$

El Redex externo: La expresión completa, donde x sería el resultado del Redex interno.

Paso 1: Reducir el Redex interno primero.

Redex: $(\lambda y. y + 1) 5$

Regla- β : $(y + 1)[y := 5]$

Reductum: $5 + 1 = 6$

Paso 2: Sustituir el resultado en la expresión original.

La expresión se convierte en: $(\lambda x. x^* 2) 6$

Paso 3: Reducir el nuevo Redex.

Redex: $(\lambda x. x^* 2) 6$

Regla- β : $(x^* 2)[x := 6]$

Reductum Final: $6^* 2 = 12$

Conclusion

Redex es la parte de la expresión que se puede ejecutar, con la forma $(\lambda\text{-abstracción})(argumento)$.

La Regla Beta es la acción de ejecutarla: sustituir el parámetro por el argumento en el cuerpo de la función.

Reductum es el resultado de esa ejecución.

Este ciclo de Identificar Redex -> Aplicar Regla- β -> Obtener Reductum es el proceso fundamental de cálculo que permite que la programación funcional funcione.

