Instituto tecnológico superior de jerez

Ingeniería en sistemas computacionales

Programación Lógica y Funcional

Modelo de programación funcional

Mapa conceptual y cuestionario

Docente: M.T.I. Salvador Acevedo Sandoval

Alumna: Viviana Michel Chávez Juárez

Numero de control: 14070013

Fecha: 06/03/2020

¿Qué es la EVALUACIÓN PEREZOSA?

La evaluación perezosa o llamada por necesidad 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. Esta compartición del cálculo puede reducir el tiempo de ejecución de ciertas funciones de forma exponencial, comparado con otros tipos de evaluación

Posterga la evaluación de la expresión hasta que su valor es realmente demandado por el programa en ejecución.

- **Codificación**. La evaluación perezosa permite alcanzar un modelo de programación defensivo basado en aserciones [+] que resulta generalmente más sinóptico y sencillo de leer y entender. En lugar de comprobar las precondiciones de una operación a base de un abuso de sentencias y expresiones condicionales se utilizan expresiones lógicas como prefijo de las operaciones a realizar para que operen de guarda que dispara la ejecución de la operación a la derecha sólo si se satisface cierta condición ambiental. En el código (A) del Listado 1 pueden verse algunos ejemplos de este uso canónico de construcciones.
- **Recursión**. De forma similar al caso anterior, los esquemas de diseño funcional recursivo también se prestan mucho a hacer uso de este tipo de construcciones. En el código (B) del Listado 1 aparece un ejemplo de función de búsqueda recursiva que determina si un elemento x se encuentra dentro de un vector xs. Tal función consiste en una sola expresión que indica que el vector contendrá al elemento, si éste se encuentra en la posición de análisis en curso p o está en alguna posición posterior a este índice, comenzando desde el 0. Y todo ello siempre que p no supere la longitud del vector. En este caso, la evaluación perezosa nos servirá para que se pare la recursión cuando esta última condición sobre p deje de ser cierta o el elemento se haya encontrado.
- **Rendimiento**. La evaluación perezosa también puede ser relevante en temas de rendimiento. El código (C) del Listado 1 muestra una función similar a la anterior, pero en este caso aplicando búsqueda binaria. Básicamente, en cada iteración recursiva, el vector se parte - de forma imaginaria - en 2 subvectores iguales con ayuda de dos índices posicionales p y q. Buscar el elemento x consiste en recurrir con la búsqueda en cada subvector. El proceso terminará cuando se llegue a alguna situación unitaria donde py q coincidan y pueda comprobarse que en ese punto reside el elemento buscado. La ventaja de la evaluación perezosa en este escenario consiste en que todo el proceso de encadenamiento compositivo por operaciones de disyunción lógica se para automáticamente en cuanto se encuentra la primera coincidencia ahorrando muchos ciclos de cómputo. Recuerde que el modelo de ejecución es secuencial y que las operación has no se lanzan en paralelo.

¿Qué ventajas tiene la implementación de la evaluación perezosa?

La evaluación perezosa puede también reducir el consumo de memoria de una aplicación, ya que los valores se crean solo cuando se necesitan. Sin embargo, es difícil combinar con las operaciones típicas de programación imperativa, como el manejo de excepciones o las operaciones de entrada / salida, porque el orden de las operaciones puede quedar indeterminado.
Los beneficios de la evaluación perezosa son:

- El incremento en el rendimiento al evitar cálculos innecesarios.
- La capacidad de construir estructuras de datos potencialmente infinitas.
- La evaluación perezosa puede también reducir el consumo de memoria de una aplicación.

¿Qué desventajas tiene la implementación de la evaluación perezosa?

Es difícil de combinar con las operaciones típicas de programación imperativa, como el manejo de excepciones o las operaciones de entrada/salida, porque el orden de las operaciones puede quedar indeterminado. Además, la evaluación perezosa puede conducir a fragmentar la memoria.

Ejemplo de aplicación de programación con evaluación perezosa en Scala

Scala admite la evaluación perezosa para argumentos de funciones usando notación: `def func(arg: => String)` . Dicho argumento de función podría tomar un objeto `String` normal o una función de orden superior con `String` tipo de retorno `String` . En el segundo caso, el argumento de la función sería evaluado en el acceso al valor.

```
def calculateData: String = {
  print("Calculating expensive data! ")
  "some expensive data"
}

def dumbMediator(preconditions: Boolean, data: String): Option[String] = {
  print("Applying mediator")
  preconditions match {
    case true => Some(data)
    case false => None
  }
}

def smartMediator(preconditions: Boolean, data: => String): Option[String] = {
  print("Applying mediator")
  preconditions match {
    case true => Some(data)
    case false => None
  }
}

smartMediator(preconditions = false, calculateData)

dumbMediator(preconditions = false, calculateData)
```

`smartMediator` llamada a `smartMediator` devolverá el valor Ninguno e imprimirá el mensaje `"Applying mediator"` .

`dumbMediator` llamada a `dumbMediator` devolverá el valor Ninguno e imprimirá el mensaje `"Calculating expensive data! Applying mediator"` .

```
function calcular ( func1(), func2(), func3() ) {
   ...
}
```

Un lenguaje sin evaluación perezosa ejecutaría las funciones func1(), func2() y func3() para obtener los parámetros finales y después continuaría con el cuerpo de la función. En cambio un lenguaje con evaluación perezosa empieza con el código de la función sin evaluar previamente las tres funciones. Las irá evaluando cuando aparezcan en el código y sean realmente necesarias. De esta forma, si un argumento no se utiliza, nunca será evaluado (puede ocurrir que alguno de los parámetros esté dentro de un if que no se cumple)

¿Qué son las funciones de alto orden?

Una función es *de orden superior* si toma una función como argumento o devuelve una función como resultado.

- Usos: Definición de patrones de programación.
  - Aplicación de una función a todos los elementos de una lista.
  - Filtrado de listas por propiedades.
  - Patrones de recursión sobre listas.
- Diseño de lenguajes de dominio específico:
  - Lenguajes para procesamiento de mensajes.
  - Analizadores sintácticos.
  - Procedimientos de entrada/salida.
- Uso de las propiedades algebraicas de las funciones de orden superior para razonar sobre programas.

¿Qué son las MONADAS en programación funcional?

Una mónada es un patrón de diseño funcional que tiene su origen en la teoría de las categorías, aunque no hace falta ser ningún experto en matemáticas para utilizarlas. Una mónada siempre encapsula un tipo de datos que nosotros elegimos para crear instancias de otro tipo nuevo asociado a una computación especial. Es común que dicha computación maneje un caso especial de dicho tipo. Una mónada siempre define un tipo de datos y cómo podemos combinar los valores de dicho tipo.
¿Qué quiere decir todo esto?

Vemos un par de ejemplos que nos pueden ayudar a familiarizarnos con el concepto.

¿Qué son los MONOIDES en programación funcional?

Los monoides son sistemas algebraicos compuestos por (A,◎,e) que cumplen con la estructura de Semigrupo y añaden una nueva restricción: el elemento neutro.

Referencias

Praena, J. A. (14 de 01 de 2018). *Thoughts on Software*. Obtenido de Thoughts on Software: http://joragupra.com/2018/01/que-es-una-monada.html

RIP Tutorial. (s.f.). *RIP Tutorial*. Obtenido de RIP Tutorial: https://riptutorial.com/es/scala/example/28352/argumentos-perezosos-de-evaluacion

https://es.quora.com/Cu%C3%A1l-es-la-ventaja-y-desventaja-de-un-lenguaje-de-programaci%C3%B3n-con-evaluaci%C3%B3n-perezosa

https://javiervelezreyes.com/las-3-evaluaciones-de-la-programacion-funcional/

http://evaluacionperezosa.blogspot.com/2016/03/que-es-la-evaluacion-perezosa.html

Martínez. (06 de 04 de 2016) https://www.goconqr.com/en/p/2613378-evaluacion-perezosa-mind_maps

MODELO DE PROGRAMACIÓN FUNCIONAL

EVALUACIÓN PEREZOSA

- Informacion
  - Es una estrategia de evaluación que retrasa el cálculo de una expresion hasta que su valor sea necesario, y que tambien evita repetir la evaluación en caso de ser necesariaen posteriores ocasiones.
    - Codificación
      - Permite alcanzar un modelo de programación defensivo basado en aserciones que resulta más sinóptico y sencillo de leer y entender.
    - Recusión
      - Permite que se pare la recursion cuando una condición establecida deje de ser cierta o el elemento se haya encontrado.
    - Rendimiento
      - Es importante en temas de rendimiento.
  - Ventajas / Desventajas
    - Incremento en el rendimiento al evitar calculos innecesarios.
      - La capacidad de construir estructuras de datos potencialmente infintas.
        - Reducir el consumo de memoria de una aplicación.
    - Puede conducir a fragmentar la memoria.
      - Al combinar operaciones tipicas de programación imperativa puede haber dificultad porque el oreden se puede indeterminar.
  - Ejemplo

Scala admite la evaluación perezosa para argumentos de funciones usando notación `def func(arg: => String)`. Dicho argumento de función podría tomar un objeto `String` normal o una función de orden superior con `String` tipo de retorno `String`. En el segundo caso, el argumento de la función sería evaluado en el acceso al valor.

```
def calculateData: String = {
  print("Calculating expensive data! ")
  "some expensive data"
}

def dumbMediator(preconditions: Boolean, data: String): Option[String] = {
  print("Applying mediator")
  preconditions match {
    case true => Some(data)
    case false => None
  }
}

def smartMediator(preconditions: Boolean, data: => String): Option[String] = {
  print("Applying mediator")
  preconditions match {
    case true => Some(data)
    case false => None
  }
}

smartMediator(preconditions = false, calculateData)

dumbMediator(preconditions = false, calculateData)
```

`smartMediator` llamada a `smartMediator` devolverá el valor Ninguno e imprimirá el mensaje `"Applying mediator"`.

`dumbMediator` llamada a `dumbMediator` devolverá el valor Ninguno e imprimirá el mensaje `"Calculating expensive data! Applying mediator"`.

Elemento flotante

FUNCIÓNES DE ALTO ORDEN

Una funcion es conocida asi si toma una función como argumento o devuelve una función como resultado

- Usos
  - Definición de patrones de programación.
    - Patrones de recursión sobre listas.
    - Filtrado de listas por propiedades.
    - Aplicación de una función a todos los elementos de una lista.
  - Uso de las propiedades algebraicas de las funciones de orden superior para razonar sobre programas.
    - Analizadores sintácticos.
    - Procedimientos de entrada/salida.
    - Lenguajes para procesamiento de mensajes.
  - Diseño de lenguajes de dominio específico

MONADAS

- ¿Qué son?
  - Monadas
    - Patron de diseño funcional que tiene su origen en la teoria de las categorias.
    - Siempre define un tipo de datos y cómo podemos combinar valores de dicho tipo.
    - Encapsula un tipo de datos elegidos para crear instancias de otro tipo nuevo asociado a una computacion especial.
  - Monoides
    - Son sistemas algebraicos compuestos que cumplen con una estructura de semigrupo y añaden una nueva restriccion el elemento neutro.