UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

INFORME DE LABORATORIO 1: CREACION DE UN SISTEMA PARA IMPLEMENTAR CHATBOT EN SCHEME

Nombre: Alvaro Muñoz Araya

Profesor: Víctor Flores Sánchez

Asignatura: Paradigmas de Programación

Fecha: 9 de octubre de 2023

Indice

1. INTRODUCCIÓN

Durante este informe se detallarán las características del proyecto abordado para la asignatura de Paradigmas de la programación, en concreto a lo que se refiere a la programación funcional. Utilizando Scheme a través del compilador Dr.Racket nos adentraremos a conocer en profundidad las características de este paradigma.
Se detallará la situación planteada para el laboratorio, comenzando por una descripción del problema, también una descripción del propio paradigma utilizado, para luego dar paso a los objetivos y análisis del problema, a lo que se propondrá un diseño de solución. Finalmente se podrá encontrar con la parte de implementación, las instrucciones de uso, resultados, autoevaluación del trabajo realizado y conclusión.

1.1 DESCRIPCIÓN DEL PROBLEMA

Para este laboratorio se requiere la implementación de un Sistema de Chatbots, con esto se quiere decir que, se busca un programa creado a partir del lenguaje Scheme que pueda crear, vincular, interactuar entre otras muchas cosas, con Chatbots.
Se debe especificar que la manera de interactuar con el sistema será en base a línea de comandos, de una manera muy estructurada, lo que nos permite ingresar palabras clave que los chatbots podrán reconocer y manipular, por lo que no se requiere implementar inteligencia artificial.

1.2 DESCRIPCIÓN DEL PARADIGMA

Este primer laboratorio como ya antes se ha mencionado, se trabajará bajo el paradigma de programación funcional. Para entender mejor este concepto, debemos entender de que es una función, lo cual no es mas que un proceso de cambio en donde se recibe uno o más elementos de entrada denominados dominio y entrega uno o más elementos denominados recorrido.
Gracias al paradigma funcional podemos intentar describir de forma computacional la realidad que nos rodea, bajo el pensamiento de que todo se puede representar como una función.
En este contexto haremos uso del lenguaje de programación Scheme, que, aunque no es un lenguaje de programación funcional puro, evitaremos declarar variables globales que nos permitan modificar valores internos del programa, ya que esto es mas bien parte de la programación imperativa.
Algunos conceptos importantes a tener en cuenta de la programación funcional son:

- Recursividad: Nos permite generar ciclos a partir de funciones que se llaman a si misma dentro del dominio de dicha función, los tipos principales de recursión son: recursión de cola, recursión natural y recursión arbórea.
- Función anónima: Es una función a la cual no se le asigna un nombre especifico y está basada en el cálculo lambda.

- Currificación: Consiste en transformar una función que utiliza múltiples argumentos en una secuencia de funciones que utilizan un único argumento (la operación inversa a la composición de funciones en matemáticas).
- Composición: es un acto o mecanismo para combinar funciones simples en complejas de modo que el resultado de cada función es pasado como argumento a la siguiente y el resultado de la última función es el de todas.

1.3 OBJETIVOS

El principal objetivo es aprender el paradigma de la programación funcional, como se implementa en una situación real, para que situaciones nos puede ayudar, además de sus ventajas y desventajas.
Otro objetivo es aplicar los conocimientos aprendidos del lenguaje Scheme, que provee de las herramientas necesarias para construir el sistema de Chatbots.

2. DESARROLLO

2.1 ANALISIS DEL PROBLEMA

Teniendo en cuenta que los chatbots que serán implementados al sistema corresponden al chatbot del tipo ITR (Respuesta de Interacción a Texto), debemos construir un ambiente propicio para que el usuario pueda crear su propio sistema.

Para ello debemos aclarar cada concepto que se verá involucrado en el sistema:

Chatbot: Es el encargado de contener la información necesaria para generar una interacción con el usuario, se asocia a un ID para poder generar conexiones con otros chatbots.

Options: Son las opciones de texto que tiene un chatbot en respuesta a una interacción especifica con el usuario, cada “Option” debe tener un numero asociado que nos permitirá seleccionarlo dentro del las demás opciones que contiene un Chatbot

System: Un sistema contiene a todos los Chatbots en su conjunto, hay una variedad de funciones que se relacionan con el sistema, como por ejemplo crear Chatbots, eliminarlos o modificarlos.

User: El identificador del usuario que ingresa al sistema, cada sistema creado es único en relación a cada usuario

Login: Nos permitirá acceder al sistema con un determinado usuario

2.2 DISEÑO DE LA SOLUCIÓN

2.2.1 TDA Option:

option: Constructor del TDA. Es una representación de lista con la finalidad de construcción de opciones para el flujo de un Chatbot. El dominio de la función es: code X message X chatbot-code X initial-flow-code X . keywords y el recorrido es: option

2.2.2 TDA Flow:

Flow: Constructor del TDA. Es una representación en base a lista de la construcción de un Flow. El dominio de la función es: id X name X . options, utiliza la función remove-duplicate para eliminar las opciones duplicadas. El recorrido de la función es: Flow

Flow-add-option: Modificador del TDA. Función que nos permite modificar un Flow para añadirle opciones. El dominio de la función es: Flow X option, también utiliza la función remove-duplicate para eliminar las opciones duplicadas. El recorrido de la función es: Flow

remove-flow-by-id: función itera recursivamente sobre la lista de flujos y elimina el flujo que tiene la misma identificación (ID) que la proporcionada, Dominio: Flow X id, Recorrido: Flow

remove-duplicates: función itera recursivamente sobre una lista para eliminar los duplicados, utiliza la función filter, Dominio: list, Recorrido: list

find-flow-by-id: Esta función itera sobre la lista de flujos y compara la identificación (ID) del flujo actual con la ID proporcionada. Dominio: Flow X id,Recorrido: Bool X Flow

2.2.3 TDA Chatbot:

chatbot: Constructor del TDA. Esta funcion construye un chatbot donde se describen sus elementos a traves de una lista, Dominio: chatbotID (int) X name (String) X welcomeMessage (String) X startFlowId(int) X flows*, Recorrido: chatbot

chatbot-add-flow: Modificador del TDA. Se toma como argumentos un chatbot y un flujo. lo añade a la lista de flujos por medio de recursión de cola. Dominio: chatbot X Flow, Recorrido: chatbot

chatbot-exists?: verifica si un chatbot con un código específico existe en la lista de chatbots proporcionada como argumento. Dominio: chatbot X code, Recorrido: Bool

find-chatbot-by-id: itera recursivamente sobre la lista de chatbots y compara la identificación (ID) del chatbot actual con la ID proporcionada. Si encuentra un chatbot con la misma ID, devuelve ese chatbot. Si no encuentra ninguna coincidencia, devuelve #f. Dominio: chatbot X code, Recorrido: chatbot X bool.

2.2.4 TDA System

system: Constructor del TDA. Crea un nuevo sistema de chatbots con el nombre name, un enlace de código inicial initialChatbotCodeLink y una lista opcional de chatbots chatbots.

Dominio: name X initialChatbotCodeLink X . chatbots, Recorrido: System.

system-add-chatbot: Modificador del TDA. Añade un chatbot al sistema, primero verifica si el chatbot existe. Dominio: System X chatbot, Recorrido: System

system-add-user: Modificador del TDA, similar a System-add-chatbot, añade un usuario al sistema, Dominio: System X user, Recorrido: System

system-login: Verifica si existe el usuario, para añadir un parámetro al sistema con el usuario logeado, Dominio: System X user, Recorrido: System

system-logout: Cierra la sesión del usuario en el sistema, quita el parámetro añadido a system, devolviendo un #f en ese caso, Dominio: System X message, Recorrido: System

system-talk-rec: Permite interactuar con el sistema, toma en cuenta el flujo inicial, además de un mensaje inicial dado por el usuario, trabaja de forma recursiva gracias a la función Chathistory, Dominio: System X message, Recorrido: System.

system-talk-norec: Similar a la anterior pero no usa recursión, Dominio: System X message, Recorrido: System

system-synthesis: ofrece una síntesis del sistema, formeateado a una lista de strings, , para poder ser visualizado con Display, Dominio: system X user, Recorrido: Lista

system-simulate: Función para poder simular un dialogo entre dos chatbos, usa una semilla que se genera con la función myrandom, además de un número máximo de interacciones, Dominio: system X maxInteractions X seed, Recorrido: Lista

update-chatHistory: función que permite actualizar el historial del chat, se llama cada vez que se añade un mensaje en el sistema, permite que no se pierdan los mensajes anteriores, Dominio: chatHistory X user X message, Recorrido: Lista

synthesize-chatHistory: función que toma un historial de chat y lo reformatea para presentarlo de manera diferente, Dominio: system maxInteractions seed, Recorrid: Lista

simulate-dialogue: Esta función simula un diálogo entre dos chatbots en el sistema. Utiliza un máximo de interacciones especificado y una semilla para la generación de números aleatorios. La simulación de diálogo entre chatbots implica que los chatbots intercambien mensajes, elijan opciones y naveguen a través de flujos predefinidos, Dominio: system X maxInteractions X seed, Recorrido: system X maxInteractions X seed.

2.3 ASPECTOS DE IMPLEMENTACIÓN

Se ha usado el compilador de Dr.Racket en su versión 8.10.

Utilizando la función de REQUIRE y PROVIDE se ha podido dividir el código en diferentes archivos, cada uno para un TDA distinto y para algunos archivos que contienen scripts de prueba. Los archivos creados son los siguientes: “TDAOption_207242381_MunozAraya.rkt”, “TDAFlow_207242381_MunozAraya.rkt”,

“TDAChatbot_207242381_MunozAraya.rkt”, “TDASystem_207242381_MunozAraya.rkt”, “main_207242381_MunozAraya.rkt”, “pruebas1_207242381_MunozAraya.rkt”, “pruebas2_207242381_MunozAraya.rkt” y “pruebas3_207242381_MunozAraya.rkt”.

No se han ocupado bibliotecas externas a las básicas incluidas en Scheme.

2.4 INSTRUCCIONES DE USO

Para poder ejecutar el código correctamente debemos tener todos los archivos ubicados en el mismo directorio. Seguido debemos abrir uno de los archivos que contienen los scripts de prueba, ya sea el 1, 2 o 3, podemos copiar estos scripts y pegarlos en el archivo main, o simplemente podemos ejecutarlos desde el mismo archivo dando a run.

2.4.1 EJEMPLOS DE USO

En el siguiente ejemplo se muestra el archivo de pruebas 1, donde se puede ver el funcionamiento de todos los scripts, se puede ver la definición de op1 en este caso como una lista creada para dicha opción:

```
#lang racket
(require "TDAOption_207242381_MunozAraya.rkt")
(require "TDAFlow_207242381_MunozAraya.rkt")
(require "TDASystem_207242381_MunozAraya.rkt")
(require "TDAChatbot_207242381_MunozAraya.rkt")


;scripts de prueba 1
(define op1 (option  1 "1) Viajar" 2 1 "viajar" "turistear" "conocer"
(define op2 (option  2 "2) Estudiar" 3 1 "estudiar" "aprender" "perfe
(define f10 (flow 1 "flujo1" op1 op2 op2 op2 op2 op1))
(define f11 (flow-add-option f10 op1))
(define cb0 (chatbot 0 "Inicial" "Bienvenido\n¿Qué te gustaría hacer?
(define s0 (system "Chatbots Paradigmas" 0 cb0 cb0 cb0))
(define s1 (system-add-chatbot s0 cb0))
(define s2 (system-add-user s1 "user1"))
(define s3 (system-add-user s2 "user2"))
(define s4 (system-add-user s3 "user2"))
(define s5 (system-add-user s4 "user3"))
(define s6 (system-login s5 "user8"))
(define s7 (system-login s6 "user1"))
(define s8 (system-login s7 "user2"))
(define s9 (system-logout s8))
(define s10 (system-login s9 "user2"))


Welcome to DrRacket, version 8.10 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> op1
'(1 "1) Viajar" 2 1 ("viajar" "turistear" "conocer"))
>
```

También podemos ver la definición para f10, cb0 y s10 particularmente.

```
> f10
'(1
  "flujo1"
  ((1 "1) Viajar" 2 1 ("viajar" "turistear" "conocer"))
   (2 "2) Estudiar" 3 1 ("estudiar" "aprender" "perfeccionarme"))))
> cb0
'(0
  "Inicial"
  "Bienvenido\n¿Qué te gustaría hacer?"
  1
  ((1
    "flujo1"
    ((1 "1) Viajar" 2 1 ("viajar" "turistear" "conocer"))
     (2 "2) Estudiar" 3 1 ("estudiar" "aprender" "perfeccionarme"))))))
> s10
'("Chatbots Paradigmas"
  0
  ((0
    "Inicial"
    "Bienvenido\n¿Qué te gustaría hacer?"
    1
    ((1
      "flujo1"
      ((1 "1) Viajar" 2 1 ("viajar" "turistear" "conocer"))
       (2 "2) Estudiar" 3 1 ("estudiar" "aprender" "perfeccionarme")))))))
  (0)
  ("user3" "user2" "user1")
  "user2")
>
```

2.5 RESULTADOS Y AUTOEVALUACIÓN

Cabe mencionar que no todas las funciones fueron implementadas exitosamente, dado los prerrequisitos de implementación, las funciones system-talk-rec, system-talk-norec, system-synthesis, system-simulate no funcionan para el 100% de los casos y necesitan correcciones.
Para el resto de funciones, el sistema no presenta fallos conocidos.

2.5.1 AUTOEVALUACIÓN

Requerimientos Funcionales	Evaluación
TDAs	0.75
option	1
flow	1
flow-add-option	1
chatbot	1
chatbot-add-flow	1
system	1
system-add-chatbot	1
system-add-user	1
system-login	1
system-logout	1
system-talk-rec	0.25
system-talk-norec	0.25
system-synthesis	0.5
system-simulate	0.25

Nota esperada: 4.8

3 CONCLUSIÓN

Para concluir, cabe mencionar que el diseño de solución cumple en ciertos aspectos con el problema planteado inicialmente, Sin embargo, las funcionalidades que permiten interactuar no fueron implementadas de manera exitosa, por lo que es difícil realizar una medición completa del sistema.
Por otra parte, el paradigma de programación funcional fue utilizado durante todo el proceso del proyecto, no se utilizaron otras funcionalidades fuera de lo que es el paradigma y se cumplió con los requerimientos mencionados a la hora de programar bajo este paradigma.

4 BIBLIOGRAFÍA Y REFERENCIAS

Paradigma funcional uvirtual usach:
https://uvirtual.usach.cl/moodle/course/view.php?id=10036&section=11

Programacion en Racket/Scheme:
https://youtube.com/playlist?list=PLTd5ehIj0goMswKV4Glhr4uixrhCmihIt&si=vvHHcdVQh6yOwftQ