Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

Informe Laboratorio 1 Paradigmas de Programación Funcional

Nombre: Amaru Monje Rojas

Rut: 21.130.814-1

Profesor: Roberto González Ibañez

Índice

Introducción:

En este informe se detallan el proceso de trabajo y de ideación llevado a cabo en este proyecto.

El propósito de este proyecto es crear un ambiente de chatbots ITR (Respuesta de Interacción a Texto), usando el paradigma funcional a través del lenguaje de programación Scheme/Racket. (Racket, s.f.)

Descripción del Problema:

Se debe crear mediante programación, un ambiente que permita crear sistemas contenedores de chatbots, crear chatbots, interactuar con uno y ofrecer una síntesis de las interacciones realizadas con un sistema. Para esto se debe realizar la abstracción del problema y estructurar una idea de solución.

Descripción del Paradigma:

El paradigma funcional, está basado en el cálculo lambda, definido por Alonzo Church en 1932. Este último es un modelo que permite computar operaciones utilizando notación pre-fija y se basa en el uso de funciones que permiten llevar a cabo las distintas operaciones que se requieran. (Fernández, 2022)

Análisis del Problema:

Para empezar, se deben considerar las partes del problema, estos son, diseñar un ambiente que contenga chatbots, estos se deben poder modificar y manipular, además se debe ser capaz de interactuar con ellos y dejar un registro para la síntesis, es decir, poder identificarse y almacenar o mantener un historial.

Diseño de la Solución:

Se optó por diseñar TDAs (Tipo de dato abstracto), para cada parte del problema, siendo estos; Sistema, Chatbot, Flow, Option, User, ChatHistory. Para esto se escogió una representación por listas, esto quiere decir que cada TDA es una lista que contiene sus componentes, de los cuales se pueden extraer y modificar sus partes.

Para ejemplificar:

-Sistema: contiene un nombre, el código del chatbot actual, una lista de chatbots, un usuario que es la sesión activa, y una lista de usuarios que están registrados en el sistema.

-Chatbot: contiene el código del chatbot, su nombre, un mensaje de bienvenida, el código de su flujo actual y una lista de todos sus flujos.

-Flujo: contiene su código de identificación, un mensaje de flujo, y una lista de las opciones que posee.

-Opción: contiene un código, el mensaje de la opción, el código del chatbot al que se debe cambiar en caso de ser escogida, el código del flujo para el mismo caso, y una lista de palabras claves que sirven para escoger la opción sin ingresar un número.

-Usuario: contiene el nombre de usuario y una lista de historiales.

-Historial de Chat: se optó por definir el Historial de Chat como el registro de una instancia de conversación, por lo que el historial que posee un usuario es una lista de registros de instancias o historiales de chat. Su contenido es; el mensaje que se ingresó, el nombre del chatbot con el que se interactuó, el mensaje del flujo al que se respondió y las respuestas propias de ese flujo.

Estos TDAs se implementarán bajo el paradigma funcional, con sus respectivos constructores, selectores y modificadores.

Aspectos de Implementación:

Solo se utilizaron las librerías nativas de Racket, así como DrRacket 8.10. (Racket, s.f.)

Para las funciones, se utilizó recursividad de cola en algunos casos, pero mayormente se implementaron mediante definición simple o encapsulación de subfunciones que permiten trabajar con más o distintas entradas.

Se separaron los archivos para cada TDA, además de tener un archivo Main para los requerimientos funcionales y un script de pruebas que permite el testeo y análisis de los avances.

Instrucciones de uso:

Mediante el uso de DrRacket o algún tipo de IDE (Entorno de desarrollo integrado) que permita su compilación y testeo, usando las funciones creadas, se pueden definir los distintos TDAs y crear un sistema.

Para poder interactuar con un sistema, primero se debe definir uno, creando un sistema implica añadirle chatbots, flujos y opciones. Además de esto uno debe realizar un "login" con un nombre de usuario para que así se registren las interacciones en el Historial de Chat. Para ejemplo de definición ver Figura 2, se definió el sistema mostrado en la Figura 1.

Al hacer todo lo mencionado anteriormente, se puede interactuar con un sistema libremente para posteriormente realizar una síntesis del historial y ver por pantalla lo conversado, incluyendo los mensajes ingresados. (Ver Figura 3)

Resultados Obtenidos:

Se logró comprobar con el script de pruebas, que cada función realiza su trabajo correctamente, pudiendo llegar a la etapa de síntesis, y mostrando correctamente la interacción registrada, esto se puede ver en la Figura 3.

Se probó cada función con el script de pruebas, cada una funciona perfectamente si se respeta el dominio y recorrido de la función. El único error a detallar es que al hacer “logout”, no se regresa el sistema a su estado inicial, por ende, para iniciar una nueva conversación se debe interactuar con el sistema recién creado.

Evaluación Completa:

Se implementaron 14 de 15 requerimientos funcionales, dejando fuera la función system-simulate, esto fue debido a que no se logró idear una forma de usar la seed de manera apropiada.

Las demás funciones están implementadas y 100% funcionales, todo el código está debidamente documentado y categorizado, habiendo trabajado de manera constante usando GitHub.

Conclusiones:

Personalmente creo, que se hizo uso apropiado del paradigma funcional, pudiendo trabajar sin variables y mejorando el entendimiento de una implementación a través de funciones. Además, se pudo completar en gran parte la solución al problema propuesto.

Para el siguiente proyecto, el objetivo será hacer mejor uso del tiempo, empezando antes el trabajo y haciendo mejor uso del aprendizaje del curso.

Anexo:

Figura 1: Diseño Sistema script de pruebas.

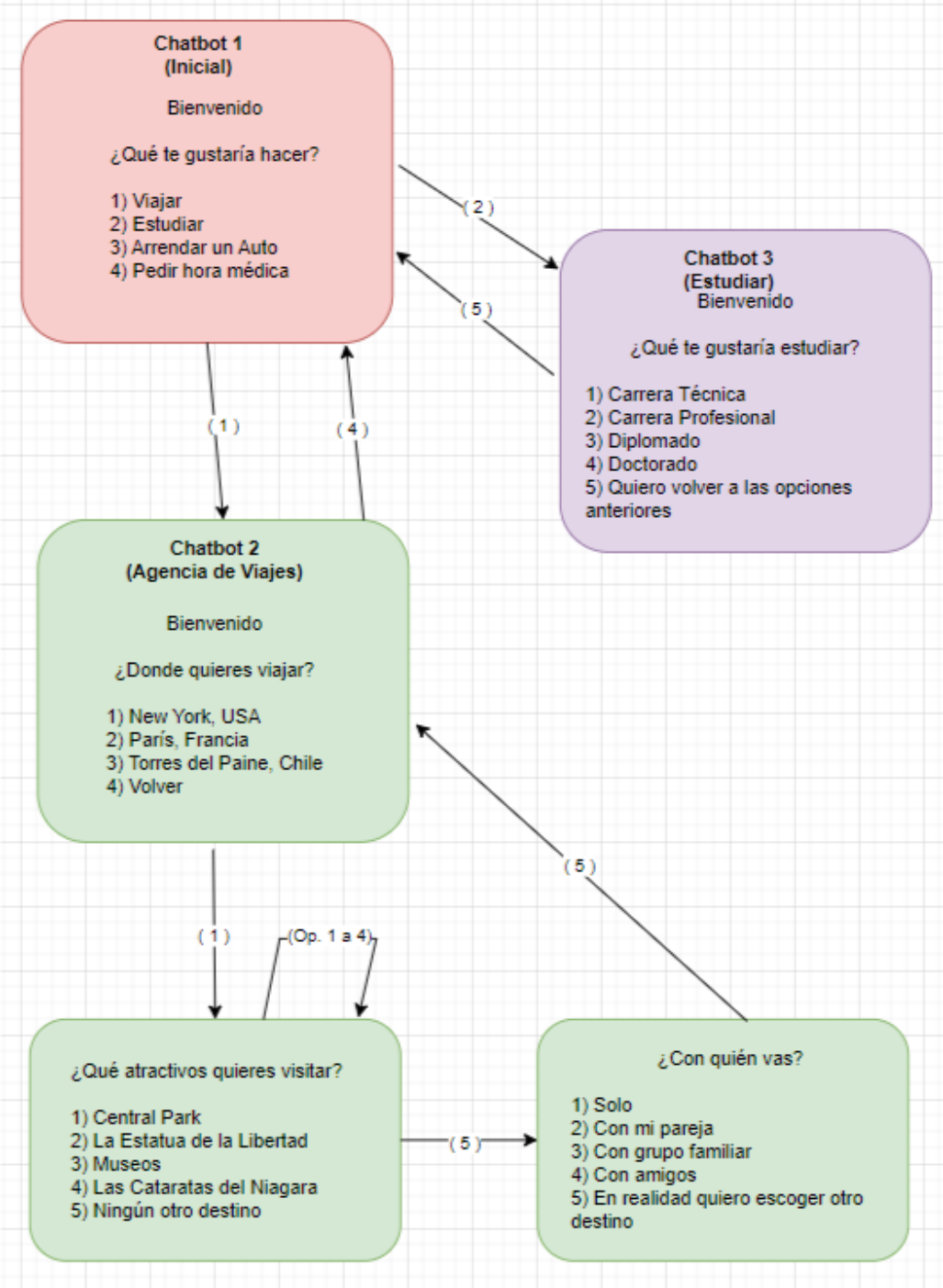

Figura 2: Ejemplo definición de sistema

```
;Chabot0
(define op1 (option 1 "1) Viajar" 1 1 "viajar" "turistear" "conocer"))
(define op2 (option 2 "2) Estudiar" 2 1 "estudiar" "aprender" "perfeccionarme"))
(define f10 (flow 1 "Flujo Principal Chatbot 1\nBienvenido\n¿Qué te gustaría hacer?" op1 op2 op2 op2 op2 op1))
;solo añade una ocurrencia de op2 y op1
(define f11 (flow-add-option f10 op1)) ;se intenta añadir opción duplicada
(define cb0 (chatbot 0 "Inicial" "Bienvenido\n¿Qué te gustaria hacer?" 1 f10 f10 f10 f10))
;solo añade una ocurrencia de f10
;Chatbot1
(define op3 (option 1 "1) New York, USA" 1 2 "USA" "Estados Unidos" "New York"))
(define op4 (option 2 "2) París, Francia" 1 1 "Paris" "Eiffel"))
(define op5 (option 3 "3) Torres del Paine, Chile" 1 1 "Chile" "Torres" "Paine" "Torres Paine" "Torres del Paine"))
(define op6 (option 4 "4) Volver" 0 1 "Regresar" "Salir" "Volver"))
;Opciones segundo flujo Chatbot1
(define op7 (option 1 "1) Central Park" 1 2 "Central" "Park" "Central Park"))
(define op8 (option 2 "2) Museos" 1 2 "Museo"))
(define op9 (option 3 "3) Ningún otro atractivo" 1 3 "Museo"))
(define op10 (option 4 "4) Cambiar destino" 1 1 "Cambiar" "Volver" "Salir"))
(define op11 (option 1 "1) Solo" 1 3 "Solo"))
(define op12 (option 2 "2) En pareja" 1 3 "Pareja"))
(define op13 (option 3 "3) En familia" 1 3 "Familia"))
(define op14 (option 4 "4) Agregar más atractivos" 1 2 "Volver" "Atractivos"))
(define op15 (option 5 "5) En realidad quiero otro destino" 1 1 "Cambiar destino"))
(define f20 (flow 1 "Flujo 1 Chatbot1\n¿Dónde te Gustaría ir?" op3 op4 op5 op6))
(define f21 (flow 2 "Flujo 2 Chatbot1\n¿Qué atractivos te gustaría visitar?" op7 op8 op9 op10))
(define f22 (flow 3 "Flujo 3 Chatbot1\n¿Vas solo o acompañado?" op11 op12 op13 op14 op15))
(define cb1 (chatbot 1 "Agencia Viajes" "Bienvenido\n¿Dónde quieres viajar?" 1 f20 f21 f22))
;Chatbot2
(define op16 (option 1 "1) Carrera Técnica" 2 1 "Técnica"))
(define op17 (option 2 "2) Postgrado" 2 1 "Doctorado" "Magister" "Postgrado"))
(define op18 (option 3 "3) Volver" 0 1 "Volver" "Salir" "Regresar"))
(define f30 (flow 1 "Flujo 1 Chatbot2\n¿Qué te gustaría estudiar?" op16 op17 op18))
(define cb2 (chatbot 2 "Orientador Académico" "Bienvenido\n¿Qué te gustaría estudiar?" 1 f30))
;Sistema
(define s0 (system "Chatbots Paradigmas" 0 cb0 cb0 cb0 cb1 cb2))
```

Figura 3: Extracto de síntesis de un historial

```
user2: hola
Inicial: Flujo Principal Chatbot 1
Bienvenido
¿Qué te gustaría hacer?
1) Viajar
2) Estudiar


user2: 1
Agencia Viajes: Flujo 1 Chatbot1
¿Dónde te Gustaría ir?
1) New York, USA
2) París, Francia
3) Torres del Paine, Chile
4) Volver


user2: 1
Agencia Viajes: Flujo 2 Chatbot1
¿Qué atractivos te gustaría visitar?
1) Central Park
2) Museos
3) Ningún otro atractivo
4) Cambiar destino
```

Referencias

Fernández, P. (09 de Septiembre de 2022). *Que es la programación funcional*. Obtenido de https://openwebinars.net/blog/que-es-la-programacion-funcional-y-sus-caracteristicas/

Racket. (s.f.). *Racket, the Programming Language*. Obtenido de https://racket-lang.org/