Paradigma Funcional de Programación

Laboratorio N°1 - SCHEME

Kaream S. Badillo
kaream.badillo@usach.cl

Índice

1. Introducción

Hoy en día, en una era tan tecnológica y digitalizada, la interacción entre humanos y máquinas se ha convertido inevitable en nuestra vida cotidiana, por lo tanto, por consecuencias se necesitan más agentes de conversación, que son conocidos como los chatbots, que es un sistema automatizado ampliamente usado para simplificar la comunicación entre personas y maquinas. Por ende, En este informe se hablará de Scheme, un lenguaje de programación funcional en el cual se Eligio para desarrollar un chatbot que será capaz de mantener conversaciones con usuarios de manera personalizada.

Cómo se mencionó, Scheme es un lenguaje de programación funcional en la cual se puede desarrollar aplicaciones de inteligencia artificial[i]. Se ha elegido Scheme puesto que es el principal requisito que se nos otorgó para realizar este proyecto, que nos permitirá aprender y comprender profundamente la programación funcional.

A lo largo de este informe, describiremos paso a paso la creación de nuestro chatbot en Scheme, desde la definición de las funciones hasta la implementación del uso de codigo. Además, se explicarán las estrategias utilizadas durante el trabajo.

Al final de este proyecto, no solo habremos desarrollado un chatbot funcional en Scheme, sino que también habremos adquirido valiosos conocimientos sobre programación funcional y aplicaciones de inteligencia artificial, lo que nos permitirá enfrentar desafíos más complejos en el futuro.

2. Descripción breve del problema

En este informe, el objetivo del proyecto será crear y administrar chatbots. Estos son también identificados como "Respuesta de Interacción a Texto" (ITR) que son altamente estructuradas, y responden de acuerdo con las opciones proporcionadas por el usuario. A diferencia de los chatbots más avanzados que usan inteligencia artificial (AI)[ii], los ITR no realizan un procesamiento complejo ni utilizan inteligencia artificial.

La problemática para desarrollar en este proyecto es llevar a cabo una serie de tareas esenciales relacionadas con estos chatbots, la creación de chatbots individuales, la identificación de chatbots, la adición de preguntas y opciones a los chatbots, la especificación de enlaces para guiar las interacciones entre chatbots y la posibilidad de interactuar con los chatbots, mediante de opciones predefinidas o con palabras clave, dando respuestas coherentes.

También en la descripción de la creación de los chatbots es que estos pueden ser interconectados, a lo q conlleva a una variedad de interacciones entre ellos

En este informe se enfocara en la programación funcional utilizando el lenguaje Scheme. Este paradigma consiste en que los programas se basan en funciones que operan con datos inmutables(no se pueden modificar). Funciones de primer orden(pasar los argumentos como variables), Recursión(en vez de bucles), Funciones de orden superior(funciones q reciben funciones y devuelven resultados).

En resumen, se trabajara con un nuevo paradigma dentro de la programación para adquirir nuevos conocimientos y herramientas para enfrentar los problemas y adversidades del futuro en el ámbito profesional

3. **Análisis del problema,**
   -Análisis del problema respecto de los requisitos específicos que deben cubrir (max 1 página) (10%)
   En acontinuacion, se realizara eun análisis del problema por cada requisito específico:
   Requisito 1 - TDA Option: Se encarga de crear opciones mediante argumentos necesarios para flujos de chatbots.
   Requisito 2 - TDA Flow: Crea flujos para chatbots, recibe un ID (la unicidad de los flujos), un mensaje y una lista de opciones.
   Requisito 3 - TDA flow-add-option: La función "flow-add-option" permite agregar opciones a un flujo existente sin usar recursión. Al igual que la función constructora de flujos, evita duplicados de opciones.
   Requisito 4 - TDA Chatbot: Crea chatbots con un identificador, asocia flujos y se encarga de evitar la duplicación de chatbots.
   Requisito 5 - TDA chatbot-add-flow: Es un modificador que permite agregar flujos a un chatbot mediante la recursión.
   Requisito 6 - TDA System: Es un constructor que crea sistemas de chatbots y enlaza chatbots, incluye historial.
   Requisito 7 - system-add-chatbot: Permite agregar chatbots al sistema y verifica la duplicidad por medio del ID
   Requisito 8 - TDA system-add-user: Es un modificador que se encarga de añadir usuarios al sistema.
   Requisito 9 - TDA system-login : Es una función para dar inicio de sesión y permite que los usuarios registrados iniciar sesión. No permite iniciar sesión si ya existe una sesión activa con otro usuario.
   Requisito 10 – TDA system-logout: Es una función que da cierre de sesión a una sesión abierta dentro del sistema.
   Requisito 11 - system-talk-rec: Es una función que interactúa con el chatbot utilizando recursividad y da la posibilidad que solo usuarios registrados puedan interactuar.
   Requisito 12 - system-talk-norec: Es una función que interactúa con el chatbot de una forma declarativa (sin usar recursividad).
   Requisito 13 - system-synthesis: Proporciona una síntesis del chatbot basándose en el historial para un usuario específico.
   Requisito 14 - system-simulate: La función simula un diálogo entre dos chatbots, registrando el historial (utiliza una semilla para generar números aleatorios y con un máximo de iteraciones)

   Estos requisitos son esenciales para el desarrollo de un simulador de chatbot, y con las Abstracciones Apropiadas incluyendo los tipos de datos abstractos (TDAs) adecuados, como chatbots, flujos de interacción y opciones de respuesta la cual promueve la estructura del proyecto.

4. Diseño de la solución

Hasta el momento se ha desarrollado solo un conjunto de funciones en el lenguaje de programación funcional Scheme para crear y manipular objetos relacionados con un sistema de chatbots. El cual se logró dividiendo el problema en varias funciones que se encargan en tareas específicas como crear opciones, flujos y chatbots, y agregarlos a sistemas, utilizando principalmente estructuras de datos como listas y funciones recursivas para realizar tareas como la verificación de duplicados y la agregación de elementos a listas. Los recursos empleados han sido utilizar las capacidades de Scheme para la creación de las funciones y TDAs, y dentro de lo posible se han usados buenas prácticas de programación funcional.

5. Consideraciones de implementación

Se ha elegido el lenguaje de programación Scheme debido a su idoneidad para la programación funcional, y también se usa Git y GitHub para el control de versiones, lo que simplifica la colaboración y el seguimiento de cambios. También Enel código en cada TDAs se escribieron comentarios explicativos.

Con estas consideraciones se garantizan una implementación segura del proyecto en Scheme de su resguardo y las mejoras en el futuro, siguiendo cada vez mejor las prácticas de desarrollo y programación funcional.

6. Instrucciones de uso

A continuación, se presentan algunos ejemplos de uso:

*Para crear Opciones “*Options”*

(define op1 (option 1 "1) Viajar" 2 1 "viajar" "turistear" "conocer"))

(define op2 (option 2 "2) Estudiar" 3 1 "estudiar" "aprender" "perfeccionarme"))

Resultado: *2 opciones, "op1" y "op2," con sus respectivos argumentos.*

*Para crear Flujos “*Flow”*

(define f10 (flow 1 "Flujo1: mensaje de prueba" op1 op2))

Resultado: Un flujo "f10" con dos opciones vinculadas, "op1" y "op2."

*Para crear chatbot “*chatbot*”

(define cb0 (chatbot 0 "Inicial" "Bienvenido\n¿Qué te gustaría hacer?" 1 f10))

Resultado: Crea un chatbot "cb0" con el mensaje de bienvenida y el flujo de inicio especificados.

*Crea un sistema de chatbot “*system*”

(define s0 (system "Chatbots Paradigmas" 0 cb0))

Resultado: Crea un sistema "s0" con el nombre especificado y el chatbot "cb0."

7. Resultados obtenidos

Requisito	Grado de Alcance
Abstracciones Apropiadas (TDAs)	si
Funciones como Ciudadanos de Primera Clase	Si
Inmutabilidad	Si
Recursión	si
Funciones de Orden Superior	Si
Estructura Organizada	Parcialmente
Historial de Trabajo en GitHub	Si
Script de Pruebas	Si
Prerrequisitos	Parcialmente

Los tipos de prueba a realizar fue mediante un script de pruebas realizados por un archivo .rkt, se logró hacer los 7 primeros requisitos funcionales y todos los requisitos no funerales, no se complementaron los requerimientos funcionales del numero 8 en adelante hasta el número 15, no se completaron por un mal manejo de tiempo y organización.

8. Evaluación completa y conclusiones.

En este proyecto, hemos abordado la creación de un sistema de chatbots utilizando el paradigma de programación funcional en Scheme. Hemos logrado avances en la definición de objetos y estructuras de datos, así como en la aplicación de conceptos clave de programación funcional.

Hemos tenido éxito en la creación de opciones, flujos y chatbots, siguiendo principios como la inmutabilidad de datos y la recursión. Sin embargo, el proyecto aún presenta desafíos significativos. No hemos un script de pruebas y también falta abordar el manejo de excepciones y errores.

El paradigma funcional ha demostrado ser efectivo en la creación de estructuras de datos y funciones reutilizables, pero su aplicación a la interacción usuario-chatbot puede resultar menos intuitiva. En futuras iteraciones, se deben superar estos desafíos para lograr un sistema de chatbots completo y funcional.

9. Anexos

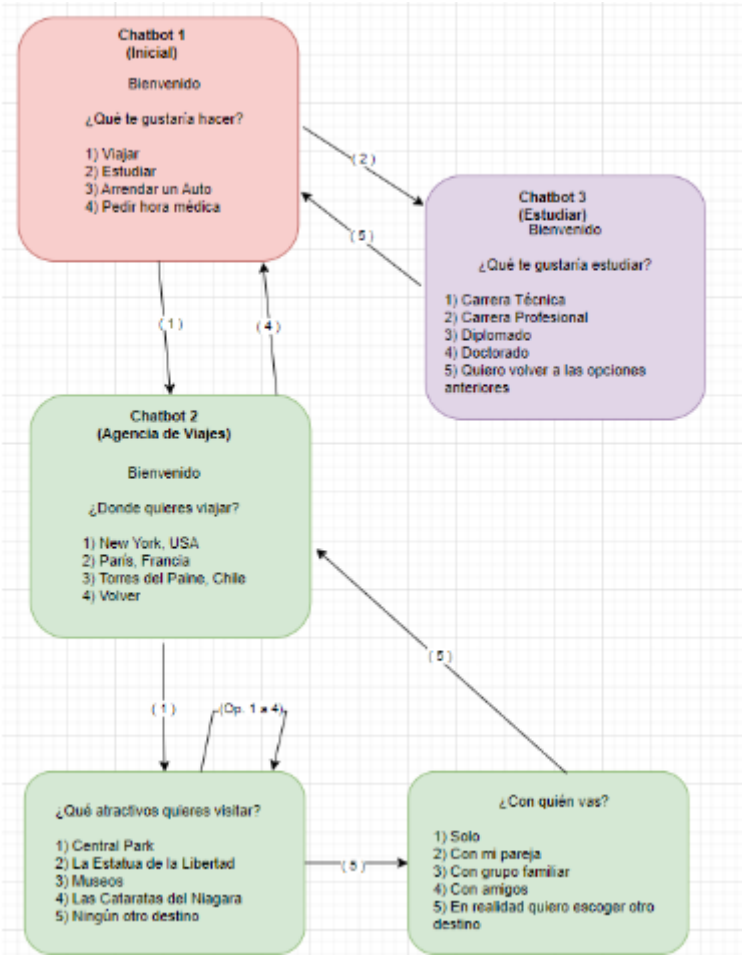

Diagrama: Ejemplo de un chatbot[iii]

10. Referencias

[i] Programación funcional: Ideal para algoritmos. (2020, febrero 11). IONOS Digital Guide. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/programacion-funcional/

[ii] Chatbot con Inteligencia Artificial | Aunoa. (s. f.). Recuperado 10 de octubre de 2023, de https://aunoa.ai/chatbots-inteligencia-artificial/

[iii] Gonzalez Ibanez, R (2023), Instrucciones generales para la entrega de TODOS los laboratorios, USACH