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

INFORME LABORATORIO Nº1
PARADIGMA FUNCIONAL
SCHEME/RACKET

Fecha: 09/10/2023
Nombre: Thomas Gustafsson Cortés.
Profesor: Roberto González.
Sección: A-1.

1. Introducción

En el presente informe se explicará la implementación de un sistema simplificado que puede crear, desplegar y administrar un chatbot. Utilizando el paradigma funcional y el ambiente de programación Racket, siendo Scheme el lenguaje de programación utilizado.

2. Descripción del problema

El problema nace desde la necesidad de poder crear un sistema de chatbots ITR (Respuesta de Interacción a Texto) simplificada, girando todo en torno a la utilización del paradigma funcional en todo momento. El sistema en su fase final debe de poder lograr una fluida comunicación/interacción entre el usuario y el chatbot, y de un chatbot con otro.

3. Descripción del paradigma

El paradigma funcional consiste en un tipo de paradigma declarativo donde se basa completamente en el uso de funciones matemáticas, esto causa que la perspectiva del mundo esté orientado a esto mismo, a funciones. También su principal fuerte es que se centra en el “qué” de los problemas, utilizando el cálculo lambda que es el uso de la escritura de la notación prefija y ayuda a prescindir en el nombre de la función en torno al mapeo de esta misma, también se utiliza la composición de funciones, funciones de orden superior, donde esto es cuando una función se le es implementada sobre otra función, y por último pero no menor, se usa la currificación, que es cuando a una función se le deja con un solo argumento, teniendo que implementar esta función mediante una secuencia de varias funciones.

4. Análisis del problema

Los requisitos específicos que se deben de cubrir son el de crear un ambiente contenedor de chatbots, crear chatbots donde se puedan identificar, añadir preguntas, opciones y poder enlazarlos mediante flujos hacia otros chatbots. Interacción con el chatbot mediante opciones y textos considerando palabras claves y ofrecer mediante el chatbot una síntesis de las interacciones hechas.

También hay funciones específicos a cubrir, estos son:

- **option:** Crea opciones a escoger, esta construye una opción para un flujo de un chatbot, siendo estos únicos mediante su propio id. Cada opción se une a un chatbot y un flujo específicos.
- **flow:** Función que construye un flujo de un chatbot, siendo este identificado mediante un id y verifica que las opciones creadas no se repitan.
- **flow-add-option:** Modifica un flujo para poder añadirle una nueva opción, pero primero verifica que no esté repetida mediante su id, si está repetida la opción, no es agregada y se mantiene el flujo inicial.
- **chatbot:** Función que crea un chatbot con un id único, también verifica que no haya flujos repetidos en el nuevo chatbot, esto es verificado por medio del id de los flujos.
- **chatbot-add-flow:** Añade un nuevo flujo a un chatbot, verificando que esté no se repita, esto se valida mediante el id del flujo, si es así, no se añade y se mantiene el chatbot inicial. Añadir que se debe implementar un tipo de recursión, en este caso se usó la recursión natural.
- **system:** Función que construye un sistema de chatbots y deja registro de la fecha y hora de la creación. Este también contiene el historial del chat de cada usuario, donde este mismo lo puede visualizar cuando lo desee.
- **system-add-chatbot:** Función que añade un nuevo chatbot a la lista de chatbots de un sistema en específico, pero primero verifica que este nuevo chatbot, por medio de su id, no se repita. Si está repetido, se mantiene el sistema inicial sin cambios.
- **system-add-user:** Añade un nuevo usuario al sistema, verificando que no se repita su nombre en otros usuarios, ya que si es así, no será agregado a al sistema.
- **system-login:** Hace iniciar al usuario de sesión en el sistema, pero primero se comprueba que no esté "conectado", que no haya nadie con su misma id (su nombre) o que no existe una sesión ya iniciada por otro usuario.
- **system-logout:** Permite cerrar la sesión anteriormente abierta por un mismo usuario.
- **system-talk-rec:** Función que permite al usuario interactuar con el chatbot, revisando previamente que el usuario haya iniciado sesión.
- **system-talk-norec:** Función que permite lo mismo que la función anterior system-talk-rec pero es implementada de forma declarativa.
- **system-synthesis:** Entrega una síntesis del chatbot para un usuario en particular, esto es a partir del historial del chat del usuario que está en el respectivo sistema.
- **system-simulate:** Permite que exista una simulación de interacción entre dos chatbots. De forma pseudoaleatoria, se da la posibilidad de que interactúen de forma “realista”, si no hay más interacciones, la simulación termina, esto es al igual con la cota superior, que es el número máximo de interacciones.

5. Diseño de la solución

Primeramente se empezó haciendo los requisitos principales y básicos para el desarrollo de este laboratorio, creando los constructores principales mediante el uso de listas y definiendo usando el cálculo lambda, esto siendo crucial en todo momento. El “paso a paso” crucial para no necesitar volver del avance del código ya hecho, fue el de crear primeramente el constructor del tda,en este caso option y verificar inmediatamente utilizando funciones de pertenencia para eliminar repeticiones de datos del conjunto de cada parte que posee option como tda, después de esto se crearía sus propios selectores y funciones de pertenencias más amplios para su uso posterior. Este proceso se puede decir que se repite varias veces mientras se va avanzando, pero la dificultad ya recae cuando ya son “listas de listas de listas y así…”, pero esto se puede solucionar rápidamente usando sintaxis propiamente tal de Racket, pero para esto se usó funciones con sus respectivos nombres de los tda para no depender del nombre en cuestión de las funciones anteriormente mencionadas.

6. Aspectos de la implementación

Para la implementación en este laboratorio se estructuró principalmente de 6 TDA’s, siendo estos option, flow, chatbot, system, user y chatHistory. Estos TDA’s fueron necesarios para la implementación total del ambiente esperado, ya que estos son los requerimientos principales obligatorios.

Cada uno de estos TDA’s poseen sus respectivas representaciones, constructores, selectores, modificadores, otras operaciones y funciones de pertenencia.

Se utilizó Racket en su versión 8.10 en lenguaje de Scheme, las bibliotecas empleadas fueron solamente racket/base, las que vienen con Racket.

7. Instrucciones de uso

Para poder usar las funciones hasta crear un sistema que es hasta lo que se pudo avanzar, es necesario implementar opciones debidamente creados, teniendo siempre en cuenta la estructura de estos, después la creación de flujos y chatbots de la misma forma que la de option, después de esto se crea el sistema, pero no logra guardar la hora ni la fecha de creación, cabe destacar que en cualquier momento se puede lograr usar un modificador de cada una de estas estructuras para agregar lo deseado, lastimosamente llega hasta aquí el uso del código en su totalidad, esto es a causa de que no se logró lo esperado.

8. Resultados

A causa de no poder lograr lo esperado en este proyecto, que es poder lograr un ambiente interactivo en un sistema con chatbots, solo se logró llegar hasta la implementación de añadir al sistema un usuario, y con solo eso hay un gran

problema ya que no hay posibilidad de interacciones entre el usuario y el chatbot, ni siquiera está la posibilidad de poder entrar en sesión, también se encuentra el problema de que no se muestra la hora y la fecha esperada en la creación de un sistema.

9. Conclusiones

En conclusión me apena decir que no se logró lo esperado, la implementación en su totalidad del sistema de chatbots, pero cabe destacar el uso de este paradigma presente, el paradigma funcional, que su uso y comprensión a medida que se va implementando y practicando es fácil y simple. Pero la diferencia entre un paradigma que se pueda usar las variables de forma libre, es notoria, pero al final estos tienen objetivos distintos y no menos importantes.

Aunque no se pudo lograr el objetivo inicial, se logró implementar eficazmente este paradigma y lograr un buen avance en la creación de un sistema de chatbots.

10. Referencias

11. Anexos