UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA
Departamento de Ingeniería Informática

Paradigmas de Programación

Laboratorio N°2: Paradigma Lógico - Lenguaje Prolog

Simulador sistema de Chatbots

Realizado por: Javier Salas Mardones
Profesor: Víctor Flores Sánchez
Sección: B-2

Santiago, 13 de noviembre de 2023

Índice de contenido

1 INTRODUCCIÓN

Este informe presenta el trabajo realizado en el laboratorio semestral de la asignatura de Paradigmas de Programación en la Universidad de Santiago de Chile, donde el objetivo central es explorar el paradigma de programación lógico mediante el uso del lenguaje Prolog.

El proyecto consiste en desarrollar un sistema para la creación, despliegue y administración de chatbots simplificados. Este sistema tiene como finalidad no solo poner a prueba nuestras habilidades en programación lógica, sino también comprender cómo las distintas formas de programación pueden ser aplicadas en soluciones tecnológicas prácticas y relevantes en el mundo actual.

En este contexto, el proyecto tiene como reto la construcción de un software que, a través de de las consultas de Prolog permita al usuario interactuar con chatbots. Estos chatbots son de Interacción de Respuesta a Texto (ITR), se caracterizan por su estructura basada en opciones específicas y una interacción limitada en cuanto al procesamiento del lenguaje natural, evitando el uso de inteligencia artificial avanzada.

El informe tiene la siguiente estructura: Primero, se tiene una descripción breve del problema, luego la descripción del paradigma, el análisis del problema, diseño de la solución, aspectos de implementación, instrucciones y ejemplos de uso, resultados y autoevaluación y finalmente una conclusión del proyecto de laboratorio realizado.

2 DESCRIPCIÓN DEL PROBLEMA

El problema central por resolver en este proyecto es el desarrollo de un sistema eficaz y funcional para la creación, despliegue y administración de chatbots simplificados, utilizando el paradigma de programación lógico con el lenguaje Prolog. La dificultad principal radica en simular un sistema interactivo de chatbots que, a pesar de su estructura simplificada, debe ser lo suficientemente versátil para permitir al usuario realizar una variedad de operaciones como crear, vincular e interactuar con estos chatbots.

Un desafío adicional es la necesidad de que estos chatbots funcionen dentro de un paradigma que no se basa en el procesamiento avanzado del lenguaje natural o en la inteligencia artificial, sino que se enfoca en respuestas estructuradas basadas en opciones específicas. Esto implica un enfoque en la eficiencia y precisión en la programación lógica para manejar interacciones basadas en texto y la correcta interpretación de predicados e instrucciones del usuario.

Por lo tanto, el problema a resolver es cómo diseñar e implementar un sistema de chatbots en Prolog que sea intuitivo y fácil de usar para los usuarios finales, al tiempo que mantiene una estructura lógica clara y eficiente en el código.

3 DESCRIPCIÓN DEL PARADIGMA

El paradigma de programación lógico, ejemplificado en este proyecto a través del uso de Prolog, representa una metodología fundamentalmente diferente a los enfoques imperativos o funcionales comúnmente utilizados en la programación. Este paradigma se centra en el uso de la lógica formal y las relaciones para expresar algoritmos y resolver problemas. En lugar de secuencias de comandos o instrucciones, la programación lógica utiliza un conjunto de hechos y reglas que describen las relaciones entre diferentes entidades y permite la deducción de conclusiones.

Prolog, en particular, es un lenguaje de programación que se destaca por su enfoque en la representación de conocimientos y la inferencia lógica. Sus características únicas, como el emparejamiento de patrones, la recursión y la retroalimentación, lo hacen ideal para resolver problemas complejos que involucran estructuras de datos no convencionales y algoritmos que requieren un enfoque declarativo.

En el contexto de nuestro proyecto, Prolog permite modelar y manejar de manera eficiente la lógica detrás de los chatbots estructurados. Su capacidad para gestionar patrones de datos y su naturaleza declarativa facilita la creación de un sistema donde las interacciones del usuario y las respuestas del chatbot se rigen por un conjunto predefinido de reglas y relaciones lógicas, en lugar de flujos de control o estructuras de datos típicos.

Esta aproximación es fundamental para abordar el desafío de implementar un sistema de chatbots simplificado, donde la eficiencia, la precisión y la capacidad de manejar múltiples escenarios de interacción a través de un enfoque basado en reglas es crucial. El paradigma de programación lógico, por lo tanto, no solo representa el medio técnico para desarrollar el proyecto, sino que también plantea un reto intelectual en cuanto a la manera de pensar y estructurar la solución al problema.

Para profundizar en la comprensión del paradigma de programación lógico y su implementación en Prolog, es esencial familiarizarse con algunos términos y conceptos clave:

- **Átomos**: En Prolog, un átomo es un tipo de dato fundamental que representa un literal constante. Puede ser una cadena de caracteres alfanuméricos que comienza con una letra minúscula. Los átomos se utilizan para denotar entidades específicas y son la piedra angular de la representación de hechos en Prolog.

- **Predicados**: Son construcciones que representan relaciones entre diferentes entidades o la propiedad de una entidad. En Prolog, un predicado se define mediante cláusulas y se utiliza para formular consultas y reglas lógicas. Los predicados son el mecanismo principal para expresar la lógica del programa.

- **Consultas**: Una consulta en Prolog es una solicitud para evaluar una expresión lógica basada en los hechos y reglas definidos en el programa. Las consultas permiten interactuar con el programa, solicitando información o verificando si ciertas condiciones se cumplen.

- **Cláusulas de Horn**: Son un tipo de fórmula lógica utilizada en Prolog que representa una regla o un hecho. Una cláusula de Horn se compone de una cabeza y un cuerpo, expresando que si el cuerpo es verdadero, entonces la cabeza también lo es. Estas cláusulas son fundamentales para definir la lógica y el flujo de un programa en Prolog.

- **Backtracking**: Es un mecanismo en Prolog que permite al intérprete buscar a través de diferentes posibilidades para encontrar todas las soluciones a una consulta. Cuando una ruta de ejecución falla, Prolog retrocede automáticamente (backtracks) para explorar otras rutas alternativas.

- **Unificación**: Proceso clave en Prolog que intenta hacer coincidir dos términos (por ejemplo, en una consulta y una regla) ajustando las variables de los términos. La unificación es esencial para el emparejamiento de patrones en Prolog y es lo que permite la flexibilidad y potencia del lenguaje en la consulta de datos y la ejecución de reglas.

Estos conceptos son fundamentales para entender la lógica detrás del desarrollo del sistema de chatbots en nuestro proyecto, y representan las herramientas esenciales con las que hemos construido la solución al problema planteado.

4 ANÁLISIS DEL PROBLEMA

Como se mencionó anteriormente, se necesita realizar un sistema de chatbots simplificado, que permita realizar operaciones para modificar e interactuar con el sistema. Para lograr esto debemos crear distintos tipos de datos que nos sirvan de base para hacer los requerimientos funcionales solicitados.

En primer lugar, debemos considerar que el sistema debe permitir un ambiente contenedor de chatbots, esto implica la capacidad de gestionar múltiples chatbots de manera simultánea y la creación de sus respectivos flujos de interacción. Además de la creación e identificación de los mismos chatbots en donde cada chatbot tiene un propósito específico. También el sistema debe facilitar la inclusión de preguntas y opciones dentro de cada chatbot, las cuales sirven para las posibles interacciones que un usuario pueda tener con el chatbot mediante palabras clave que se vinculan a opciones predefinidas, el sistema está representado de la siguiente forma:

Sistema: Una lista que contiene un nombre, un código inicial que conecta a un chatbot, una lista de chatbots, una lista de usuarios registrados, una lista de historial y un string que representa al usuario que está logueado o no. (String X Int X List X List X List X String).

Luego, algunos elementos importantes que están dentro del sistema:

- **Usuarios**: Corresponden a aquellos que interactúan con el sistema, registrándose para poder realizar las operaciones disponibles.
- **Chatbot**: Permiten simular la conversación humana, actúa como el punto de interacción principal, pueden ser personalizados para diferentes propósitos.
- **Flow**: Corresponde al flujo o secuencia estructurada de interacciones dentro de un chatbot. Cada flujo representa un camino de conversación especifico.
- **Option**: Corresponden a las posibles opciones que puede poseer un flujo que representan las posibles elecciones o respuestas que un usuario puede seleccionar durante la interacción con un chatbot.

Finalmente, algunas de las operaciones que se pueden realizar en el sistema:

- **Creación de Chatbot** (chatbot - constructor): Esta operación permite crear un nuevo chatbot dentro del sistema. Involucra definir atributos clave como el ID del chatbot, nombre, mensaje de bienvenida, ID del flujo inicial, y una lista de flujos asociados. Esta función es esencial para iniciar la estructura de interacción del chatbot con los usuarios.

- **Añadir Flujo a Chatbot** (chatbotAddFlow): Permite expandir la funcionalidad de un chatbot existente añadiendo nuevos flujos. Esta operación es crucial para aumentar la complejidad y la profundidad de las conversaciones que el chatbot puede manejar, permitiendo una mayor variedad en las interacciones con el usuario.

- **Crear Opción en un Flujo** (option - constructor): Esta función se utiliza para crear opciones dentro de un flujo específico. Cada opción incluye un código único, un mensaje asociado,

vínculos a chatbots y flujos específicos, y una lista de palabras clave. Las opciones son fundamentales para guiar la dirección de la conversación en el chatbot.

- **Modificar Flujo Añadiendo Opciones** (flowAddOption): Esta operación permite modificar un flujo existente añadiendo nuevas opciones. Es vital para la adaptabilidad y actualización de los chatbots, permitiendo modificar y enriquecer las interacciones disponibles para los usuarios a lo largo del tiempo.

- **Iniciar Sesión en el Sistema** (systemLogin): Permite a un usuario registrado iniciar una sesión en el sistema. Esta función es clave para personalizar la experiencia del usuario, manteniendo un historial de interacciones y preferencias individuales.

- **Cerrar Sesión en el Sistema** (systemLogout): Esta operación finaliza la sesión activa de un usuario, asegurando la privacidad y la seguridad de la información del usuario. Es esencial para gestionar el acceso y mantener la integridad del sistema.

5 DISEÑO DE LA SOLUCIÓN

Luego de analizar y comprender el problema planteado comenzamos con la creación de los diferentes TDA's, los cuales son "system", "chatbot", "flow", "option".

1. **TDA SYSTEM**

   - Representación: Una lista que contiene un nombre, un código inicial que conecta a un chatbot, una lista de chatbots, una lista de usuarios registrados, una lista de historial y un string que representa al usuario que está logueado o no.

2. **TDA CHATBOT**

   - Representación**:** Una lista que contiene un código identificador, un nombre del chatbot, un mensaje de bienvenida, un código identificador de flujo inicial y una lista de flujos.

3. **TDA FLOW**

   - Representación**:** Una lista que contiene un identificador, un nombre del flujo y una lista de opciones.

4. **TDA OPTION**

   - Representación**:** Una lista que contiene un código identificador, un mensaje, un código que conecta con un chatbot, un código de flujo inicial y una lista de palabras clave.

Cabe destacar que no se hizo un TDA para los usuarios ni tampoco para el historial del chat, ya que en mi representación estos son parte del sistema contenedor de chatbots.

6 ASPECTOS DE IMPLEMENTACIÓN

La implementación del proyecto está estructurada en 7 archivos de código: TDA System, TDA Chatbot, TDA Flow, TDA Option, archivo “Main” en donde se encuentran los requerimientos funcionales realizados en este caso solo se realizó hasta el requerimiento 11 (SystemLogout), otro archivo de predicados auxiliares y finalmente un archivo de pruebas para revisar el correcto funcionamiento de los requerimientos funcionales implementados, en dónde en el archivo de pruebas está todo comentado con sus respectivas instrucciones de copiado y pegado para hacer las consultas. En la mayoría de los archivos creados se utiliza “:- consult(‘nombreArchivo.pl’).” Para cargar sus respectivas bases de conocimiento.

Este laboratorio se realizó con el lenguaje de programación Prolog principalmente mediante la versión online SWISH Prolog. Adicionalmente los requerimientos funcionales fueron probados en la versión de escritorio SWI-Prolog 8.4.3 en el sistema operativo Windows 11 (64 bits). Todos los requerimientos fueron implementados mediante predicados estándar del lenguaje Prolog.

7 INSTRUCCIONES DE USO

Lo primero que se debe hacer es ejecutar una instrucción que permite visualizar todas las listas, ya que a veces Prolog acorta las listas y no las extiende, por lo tanto, para evitar esto se debe ingresar el siguiente comando:

```
set_prolog_flag(answer_write_options,[max_depth(0)]).
```

Luego se debe cargar la base de conocimientos, hacer la consulta en el archivo 'pruebas_19080187_SalasMardones.pl'. Si todo carga bien Prolog lo mostrará con mensajes en color verde, lo que significa la base de conocimiento fue cargada con éxito.

Finalmente en el mismo archivo 'pruebas_19080187_SalasMardones.pl' hay una serie de ejemplos para cada requisito funcional implementado, se debe copiar y pegar en la consola de Prolog cada ejemplo (sin seleccionar los comentarios, que pueden ser de una sola línea de código (%) o multilíneas (/* */).

Un ejemplo de lo anterior puede ser:

```
option(1, "1- alfajor de manjar", 1, 2, ["alfajor", "manjar", "alfajor de manjar"], O7),

option(2, "2- torta pompadour", 1, 2, ["torta", "pompadour", "torta pompadour"], O8),

option(3, "3- ninguna otra receta", 1, 2, ["torta pompadour"], O9),

option(1, "1- maniana", 1, 3, ["maniana"], O11),

option(2, "2- tarde", 1, 3, ["tarde"], O12),

option(3, "3- noche", 1, 3, ["noche"], O13),

option(4, "4- en realidad no quiero comer", 1, 3, ["volver"], O15),

option(4, "4- cambiar receta", 1, 2, ["cambiar", "volver", "salir"], O10),

flow(2, "flujo 2: Chatbot1\n¿Que dulces te gustan?", [O7, O8, O9, O10], F3),

flow(3, "flujo 3: chatbot1\n¿En que momento del dia?", [O11, O12, O13, O15], F4),

chatbot(1, "Chatbot Chef", "Bienvenido, Que te gustaria cocinar?", 1, [], CB1),

chatbotAddFlow(CB1, F3, CB2),

chatbotAddFlow(CB2, F4, CB3),
```

```
system("system0", 1, [], S0),

option(1, "1- cocinar", 1, 1, ["cocinar", "comida", "comer"], O1),

option(2, "2- ejercitar", 2, 1, ["ejercitar", "ejercicio", "gimnasia"], O2),

option(3, "3- descansar", 3, 1, ["descansar", "flojear", "dormir"], O19),

flow(1, "Flujo 1: Flujo principal Chatbot 1\nBienvenido\n¿Qué te gustaria hacer?", [O1, O2, O19], F1),

chatbot(0, "Inicial", "Bienvenido\n¿Qué quieres hacer hoy?", 1, [F1], CB0),

systemAddChatbot(S0, CB0, S1),

systemAddChatbot(S1, CB3, S2),

systemAddUser(S2, "Priscilla", S3),

systemAddUser(S3, "Javier", S4),

systemAddUser(S4, "Consuelo", S5),

systemLogin(S5, "Javier", S6),

systemLogout(S6, S7),

systemLogin(S7, "Consuelo", S8),

systemLogout(S8, S9).
```

8 RESULTADOS Y AUTOEVALUACIÓN

Se espera crear una simulación de un sistema de chatbots, en donde el programa sea funcional, los posibles errores pueden suceder si se utiliza mal el orden de los predicados de las operaciones, o también si no se usan bien las variables cuando se ejecuta el código, además de que no están implementados todos los requerimientos funcionales por ende si se intenta ejecutar o se hace consulta de uno de estos, el programa fallará.

En el apartado de anexos (**Tabla 1**) se puede observar la autoevaluación del laboratorio. Los requerimientos funcionales implementados fueron probados con diferentes ejemplos para verificar su correcto funcionamiento, luego según los resultados obtenidos son evaluados con la escala de autoevaluación proporcionada por la asignatura que va desde 0 hasta 1.

9 Conclusiones

Se considera que el laboratorio logró un alcance medianamente aceptable, ya que se logró realizar más de la mitad de los requerimientos funcionales solicitados, la implementación fue exitosa hasta “SystemLogout” lo que demuestra la eficacia de Prolog en la gestión de estructuras de datos lógicas y el procesamiento de consultas complejas.

Uno de los principales desafíos fue adaptarse al enfoque declarativo de Prolog, especialmente tras haber trabajado previamente con el paradigma funcional. La naturaleza de Prolog, que prioriza la descripción de “qué” debe hacerse en lugar del “cómo”, requirió un cambio significativo en el enfoque de resolución de problemas, además ambos paradigmas tienen un fuerte enfoque recursivo, por lo que se pudo utilizar algunos predicados con recursión para facilitar la implementación de los requerimientos funcionales.

10 Referencias

1. Flores, V. (2023). “*Proyecto semestral de laboratorio*”. Paradigmas de programación. https://docs.google.com/document/d/1QCk3k-HCShNSByEZHoEIFbrxoOJSnPblyjiJDj3Q9Jk/edit

2. Autores Desconocidos (2023). “SWI-Prolog Documentation”. Documentación online. https://www.swi-prolog.org/pldoc/index.html

ANEXOS

TABLA DE RESULTADOS Y AUTOEVALUACIÓN:

Tabla 1: "Tabla de resultados y autoevaluación".

Requerimiento	Alcance	Prueba	Fallos	Razón Fallo	Puntaje
option	Realizado	Se crean distintas opciones	0	NA	1
flow	Realizado	Se crean distintos flujos	0	NA	1
flowAddOption	Realizado	Se añaden distintas opciones a un flujo	0	NA	1
chatbot	Realizado	Se crean distintos chatbots	0	NA	1
chatbotAddFlow	Realizado	Se añaden distintos flujos a un chatbot	0	NA	1
system	Realizado	Se crean distintos sistemas	0	NA	1
systemAddChatbot	Realizado	Se añaden distintos chatbot a un sistema	0	NA	1
systemAddUser	Realizado	Se añaden usuarios al sistema	0	NA	1
systemLogin	Realizado	Un usuario inicia sesión en el sistema	0	NA	1
systemLogout	Realizado	Un usuario cierra sesión en el sistema	0	NA	1
systemTalkRec	No Realizado	NA	NA	NA	0
systemSynthesis	No Realizado	NA	NA	NA	0
systemSimulate	No Realizado	NA	NA	NA	0