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

Laboratorio 2 - Paradigma Lógico
Sistema de chatbots simplificado
Prolog

Nombre: Vicente Gerardo Arce Palacios
Curso: Paradigmas de Programación
Sección: A-1
Profesor: Roberto González Ibañez

13 de Noviembre de 2023

Tabla de contenidos

1. Introducción

En el presente informe se abordará el problema propuesto en el enunciado del laboratorio 2 del ramo "Paradigmas de Programación". En este segundo laboratorio se tiene como requisito el uso del Paradigma lógico (el cual fue diseñado en sus inicios para resolver problemas relacionados con la inteligencia artificial), en donde se utilizó "Prolog" como lenguaje de programación. El informe se separara en una breve descripción del problema y del paradigma a utilizar, más adelante en el desarrollo se detallara el análisis del problema, el diseño de la solución, algunos detalles en la implementación junto con las instrucciones de uso, se darán los resultados y una autoevaluación para finalmente dar una conclusión al proyecto.

1.1. Descripción del problema

Se presenta como desafió el desarrollar un sistema contenedor de chatbots en el lenguaje Prolog, usando únicamente el paradigma lógico. El sistema mencionado permite añadir chatbots, registrar usuarios, loguear usuarios y desloguear usuarios. Los usuarios cuando son registrados y logueados correctamente tienen la capacidad de interactuar con un sistema de chatbots previamente creado, esto de puede lograr a través del predicado systemTalkRec. Para lograr todo esto y más se debe pensar en como representar estos datos abstractos, de los cuales se puede deducir que el sistema debe ser un sistema que contenga los chatbots, y los chatbots a su vez contienen flujos, flujos que contienen opciones. Cada uno de los elementos mencionados corresponde a lo que se le conoce como un "TDA" y es de suma importancia el como se representara cada uno de estos para darle solución al problema. Para resolver el problema se concluyo en que los TDA serán listas, el como se resolverán los requisitos pedidos es algo que se vera con más profundidad en las siguientes secciones, pero el problema radica en el como trabajar con listas y como se implementaran los predicados para manipular estas listas.

1.2. Descripción del paradigma

El paradigma lógico es parte de la familia de paradigmas declarativos, este no gira en torno al como, sino al que. El paradigma de la programación lógica empieza a desarrollarse en la primera mitad de la década de los 70 basándose en las clausulas de Horn, en esta época es donde nace el lenguaje de programación Prolog, el cual fue creado en sus inicios como una herramienta para resolver problemas vinculados con la inteligencia artificial.
El paradigma lógico se basa en la definición de un conjunto de **hechos** (verdades) y **reglas** (afirman hechos en base a otros hechos), el conjunto de hechos es lo que se conoce como **base de conocimientos**. Una vez definidos los hechos, se pueden realizar **consultas** a la base de conocimientos. Una de las principales herramientas que se usan al momento de programar en el paradigma lógico es la recursividad, esto debido a la no existencia de condicionales. El paradigma lógico se basa también en: **unificación**, **árboles** y **backtracking**.

2. Desarrollo

En este apartado se abordara el análisis del problema, el diseño de la solución, aspectos en la implementación, las instrucciones de uso junto con los resultados y una autoevaluación.

2.1. Análisis del problema

Dados los requisitos pedidos en el laboratorio, lo primero que se debe hacer es identificar los TDA necesarios para la resolución del problema. Se nota que el TDA principal es el TDA system, el cual guarda los chatbots, usuarios registrados y logueados junto con el chatHistory asociado a cada usuario. El sistema, por los requisitos solicitados, tendrá la capacidad de registrar y actualizar el historial de los usuarios. Otra cosa a tener en cuenta, es que los chatbots que están en el sistema, contienen flujos que a su vez contienen opciones, lo cual puede derivar a pensar en estos como nuevos TDA.
El laboratorio al tener que implementarse en el lenguaje Prolog, se decidió implementar como listas cada uno de los TDA, por ende el TDA system deberá ser una lista con atributos tales como un nombre del sistema, código del chatbot inicial, lista de chatbots, lista de usuarios registrados, lista de usuarios logueados, una lista que contenga el chatHistory para cada usuario registrado, código de chatbot actual, código de flujo actual y un código que indique si una simulación se ha iniciado. El sistema se pensó de esta manera para poder implementar correctamente los predicados solicitados. Algunos de los predicados pedidos fueron el **systemAddUser**, el cual permite registrar usuarios al sistema, **systemLogin**, el cual permite loguear usuarios en el sistema solo si se encuentran registrados y no existe un usuario ya logueado, **systemLogout**, el cual permite desloguear al usuario. También se pidieron predicados un poco más complejos, como el **systemTalkRec**, el cual permite al usuario que se encuentre logueado interactuar con el sistema,el **systemSynthesis**, el cual permite recuperar el historial de un usuario, y el **systemSimulate**, el cual permite hacer una simulación de un usuario interactuando con el sistema. Todas los predicados anteriormente mencionados permiten la interacción o modificación del sistema con algún propósito especifico, pero para lograr implementar estas mismas se debe pensar en como se deben implementar los TDA que componen el sistema, como ya se menciono, un sistema contiene chatbots, los chatbots contienen flujos y los flujos contienen opciones. Lo anteriormente descrito se puede pensar como un árbol, en donde la **raíz** sería el sistema que como hijos están los chatbots, chatbots que como hijos tienen flujos y flujos que como hijos tienen opciones, opciones que serian las **hojas**, cada uno de estos TDA que se podrían pensar como parte de un gran árbol fueron implementados con la finalidad de poder suplir los requisitos pedidos, el como fue esta implementación se abordara en el diseño de la solución.

2.2. Diseño de la solución

Para llegar al diseño de la solución, se realizó un diagrama para representar el problema, detallado en el Anexo 1. A continuación se especificarán los TDA más importantes (estos cuentan con constructores, selectores, modificadores y otros):
TDA System: [name (string), InitialChatbotCodeLink (integer), chatbots (lista de chatbots), [ ] (lista de historial), [ ] (usuarios registrados), [ ] (usuarios logueados), actualChatbotCodeLink (integer), actualFlowCodeLink (integer), PlaceHolderSimulate (integer)].
TDA Chatbot: [chatbotID (integer), name (string), welcomeMessage (string), startFlowID (integer), flows (lista de flujos del chatbot)].
TDA Flow: [id (integer), name-msg(string), Option(lista de opciones)].
TDA Option: [code (integer), message (string), chatbotCodeLink (integer), InitialFlowCodeLink (integer), Keyword (lista de palabras claves)].

Los TDA anteriormente mencionados son los más importantes para llegar a la solución del problema dado, si bien se implementaron los TDA user y chatHistory, su implementación es más básica que la de los demás ya descritos.
Alguno de los predicados implementados son:
systemAddChatbot: Predicado que permite agregar un chatbot a un sistema si y solo si el id del chatbot a agregar no existe en los chatbots del sistema, si lo anterior no se cumple, arrojara false.
systemAddUser: Predicado que permite ingresar un usuario a un sistema y además crearle su chatHistory siempre y cuando este ya no este en la lista de registrados.
systemLogin: Predicado que loguea a un usuario, primeramente comprueba que el usuario que se quiere loguear este previamente registrado (que se encuentre en la lista de usuarios registrados), y que a su vez no exista ningun usuario logueado (lista de usuarios logueados vacia), entonces ahi se procede a introducir al usuario en la lista de usuarios logueados.
systemLogout: Predicado que desloguea al usuario, se vacia la lista de usuario logueado y se reinician los codigos de chatbot actual, flujo actual y placeholdersimulate.
systemTalkRec: Predicado que permite a los usuarios interactuar con el sistema en tres casos. El primero actualiza códigos en la primera interacción yendose por el chatbot incial y flujo inicial, mientras que el segundo y tercero manejan mensajes numéricos y de palabras clave respectivamente, en este se ingresa al chatbot que tenga el actualchatbotcodelink como id y luego al flujo que tenga el actualflowcodelink, se recuperan las opciones del flujo, y si se trata del segundo caso, se busca por el código de la opción, si se trata del tercer caso se busca por keywords de la opción, finalmente se almacena la información en el chatHistory del usuario y se actualizan los códigos de chatbot y flujo actual.
Si bien se implementaron más predicados, los mencionados con anterioridad son los que tienen un grado mayor de dificultad y en otros, como en el predicado systemSimulate se utilizan los predicados ya nombrados.

2.3. Aspectos de implementación

La realización de este proyecto fue implementada en el lenguaje de programación Prolog utilizándose el IDE Swi-Prolog en la versión 9.0.4.
El proyecto cuenta con cada predicado solicitado, además cada uno de los predicados implementados en el laboratorio cuentan con su documentación, la cual incluye su meta primaria, metas secundarias, descripción y dominio.
El proyecto se encuentra dividido en 7 archivos más un script de pruebas/ejemplos, si se quieren ver los existentes, basta tener instalado el IDE Swi-Prolog, clonar el proyecto y verificar que se encuentren todos los archivos en una carpeta, luego se abre Swi-Prolog y se consulta el archivo main, finalmente se copia y pega el código del script de pruebas.
Ver Anexo 2 para ver en que archivos se encuentra cada predicado.

2.4. Instrucciones de uso

Primeramente, se debe verificar que se tengan todos los archivos en una misma carpeta, una vez realizado esto, en el archivo pruebas, existen varios ejemplos para cada una de las operaciones, dicho script debe ser copiado, para luego en Swi-Prolog consultar el archivo main y copiar y pegar el script de pruebas, luego si es necesario se presiona la tecla "Enter" y se procederá a ejecutar la consulta, mostrándose las opciones, flujos, chatbots, sistemas y los strings formateados mostrados con el predicado write/1, cabe destacar que estos mensajes mostrados con write/1, al menos en la versión 9.0.4, se muestran al inicio de la consulta (Algunos de los ejemplos se pueden visualizar en el Anexo 3).
En el caso de querer crear un nuevo sistema, dejando de lado el de prueba, se deben seguir los siguientes pasos:
1) Se deben crear las opciones, esto a través del predicado constructor "option".
2) Añadir a cada flujo las opciones correspondientes, esto se hace con el predicado "flow".
3) Se encapsulan los flujos en los chatbots, esto con el constructor de chatbots "chatbot".
4) Se incluyen los chatbots a un nuevo sistema con el constructor de sistemas "system".
5) Se registran los usuarios con el predicado "systemAddUser", se loguean con el predicado "systemLogin". Una vez logueado el usuario, con el predicado "systemTalkRec" se puede interactuar con el sistema de chatbots. Para ver el historial de un usuario en particular, se puede realizar con el predicado "systemSynthesis". Finalmente con el predicado "systemLogout" se puede desloguear al usuario.
6) Otra alternativa al paso anterior es usar el predicado "systemSimulate", el cual realizara un simulación de una interacción con un sistema de chatbots ya creado.

Un fallo que se puede llegar a generar es cuando se interactúa con el sistema (systemTalkRec), ya que si bien se soporta el case sensitive, si se coloca una opción o keyword que no esta en el chatbot y flujo especifico, el sistema se caerá. Al hacer pruebas del programa no se encontraron más errores, funcionando correctamente todos los predicados.

2.5. Resultados y autoevaluación

El grado de alcance de los requisitos funcionales fue logrado en su totalidad, contándose con todos los requerimientos funcionales y no funcionales. Se lograron los resultados previstos.
La autoevaluación puede ser encontrada en el Anexo 4, con un puntaje siguiendo el siguiente formato: 0: No realizado/ 0.25: Funciona el 25 % de las veces/ 0.5: Funciona el 50 % de las veces/ 0.75: Funciona el 75 % de las veces/ 1: Funciona el 100 % de las veces.

3. Conclusión

Luego de la realización del proyecto, se puede concluir que se cumplió el objetivo de implementar un sistema de chatbots ocupando el lenguaje Prolog, siguiendo únicamente el paradigma lógico, de este se adquirió el aprendizaje necesario sobre el paradigma, el cual en un inicio fue desafiante, pero a medida del avance en el proyecto, fue bajando la dificultad, debido a la comprensión que se tuvo del paradigma.
Haciendo la comparación con el laboratorio del paradigma funcional, este en concreto fue mucho más sencillo, esto se puede deber a que es la segunda vez que se trabaja en un paradigma declarativo o también al parecido que tiene con el lenguaje SQL, y es que si bien en un principio la sintaxis del lenguaje fue un poco chocante, se necesito un poco de practica para acostumbrarse. A comparación del primer laboratorio, el predicado systemTalkRec, si bien tiene una extensión considerable, el código es mucho más legible y entendible, esto puede ser debido a la diferencia notoria entre los paradigmas y como es sabido, a veces es bueno cambiar de paradigma para poder enfrentar alguna problemática.
Para sintetizar, no hubo mayor dificultad en la realización de este proyecto. Con este proyecto de laboratorio se puedo profundizar sobre las herramientas Git, GitHub y el lenguaje Prolog. Lo más importante de este laboratorio es que se aprendió a pensar de una manera lógica y se vio la diferencia que existe entre el paradigma funcional y el paradigma lógico.

4. Bibliografía

Chauhan, A. (2023, September 23). Abstract data types. GeeksforGeeks. https://www.geeksforgeeks.org/abstract-data-types/

Multiples Autores. (Septiembre 2013). "El lenguaje de programación PROLOG". Libro online. Recuperado de: https://drive.google.com/file/d/1DgxI64oAB5do7AajCXCTphnGWHTCEmk/view?usp=sharing

Autores Desconocidos. (2021). "Swi-Prolog Documentation". Swi-Prolog. Documentación Online. Recuperado de: https://www.swiprolog.org/pldoc/index.html

5. Anexos

5.1. Anexo 1: Diagrama del sistema

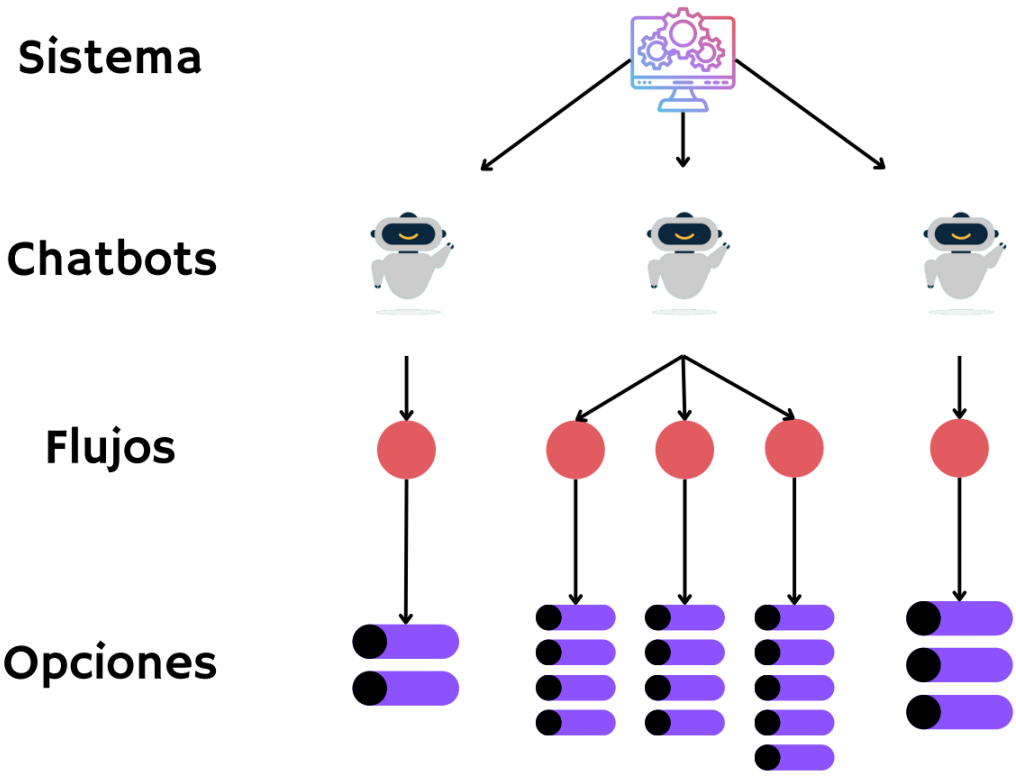

Figura 1: Diagrama de abstracción del sistema.

5.2. Anexo 2: Ubicación de los predicados

Nombre de archivo	TDA_Option	TDA_Flow	TDA_chatbot	TDA_system	TDA_chatHistory	TDA_user	Main
Constructores	option	flow	chatbot	system	chatHistory	user	
		flow2	chatbot2	system2			
Selectores	getCodeOption	getIdFlow	getChatbotIdChatbot	getNameSystem	getUserCH		
	getMessageOption	getNameMessageFlow	getNameChatbot	getInitialChatbotCodeLinkSystem	getHistorialCH		
	getChatbotCodeLinkOption	getOptionsFlow	getWelcomeMessageChatbot	getChatbotsSystem			
	getInitialFlowCodeLinkOption		getStartFlowIdChatbot	getChatHistorySystem			
	getKeywordOption		getFlowsChatbot	getRegisterUsersSystem			
				getLogUsersSystem			
				getActualChatbotCodeLinkSystem			
				getActualFlowCodeLinkSystem			
				getPlaceHolderSimulateSystem			
Modificadores		flowAddOption	chatbotAddFlow	setPlaceHolderSimulateSystem	get-user-chatHistory		systemTalkRec
				systemAddChatbot	get-Historial-chatHistory		systemSynthesis
				systemAddUser			systemSimulate
				systemLogin			
				systemLogout			
Otros	downcaseKeywords	getCodesOptions	getIdsFlows	len	agregarMensajeUsuario		getNumberOptions
		addOption	addFlow	getChatbotIdsChatbot			setRandomChoose
				addChatbot			myRandom
							getPrimerElemento
							recuperarElementoPorString
							recuperarElemento
							getMessagesOptions
Requisitos Funcionales	option	flow	chatbot	system			systemTalkRec
		flowAddOption	chatbotAddFlow	systemAddChatbot			systemSynthesis
				systemAddUser			systemSimulate
				systemLogin			
				systemLogout			

Figura 2: Ubicación de los predicados creados.

5.3. Anexo 3: Ejemplos

```
-----------------user2-----------------
1699353342.04655 - user2: Hola
1699353342.04655 - Chatbot Inicial: Flujo Principal Chatbot 0
Bienvenido
¿De qué deseas hablar?
1) Peliculas
2) Musica

1699353342.059227 - user2: 1
1699353342.059227 - Chatbot Peliculas: Flujo 1 Chatbot 1
¿Qué tipo de películas te gustan?
1) Accion
2) Terror
3) Ciencia ficcion
4) Animación
5) Volver

1699353342.059276 - user2: Accion
1699353342.059276 - Chatbot Peliculas: Flujo 2 Chatbot 1
¿Cuáles son tus películas de Acción favoritas?
1) The Matrix
2) Jhon Wick
3) The Dark Night
4) ninguna pelicula más, volver
```

Figura 3: Ejemplo interacción con un sistema.

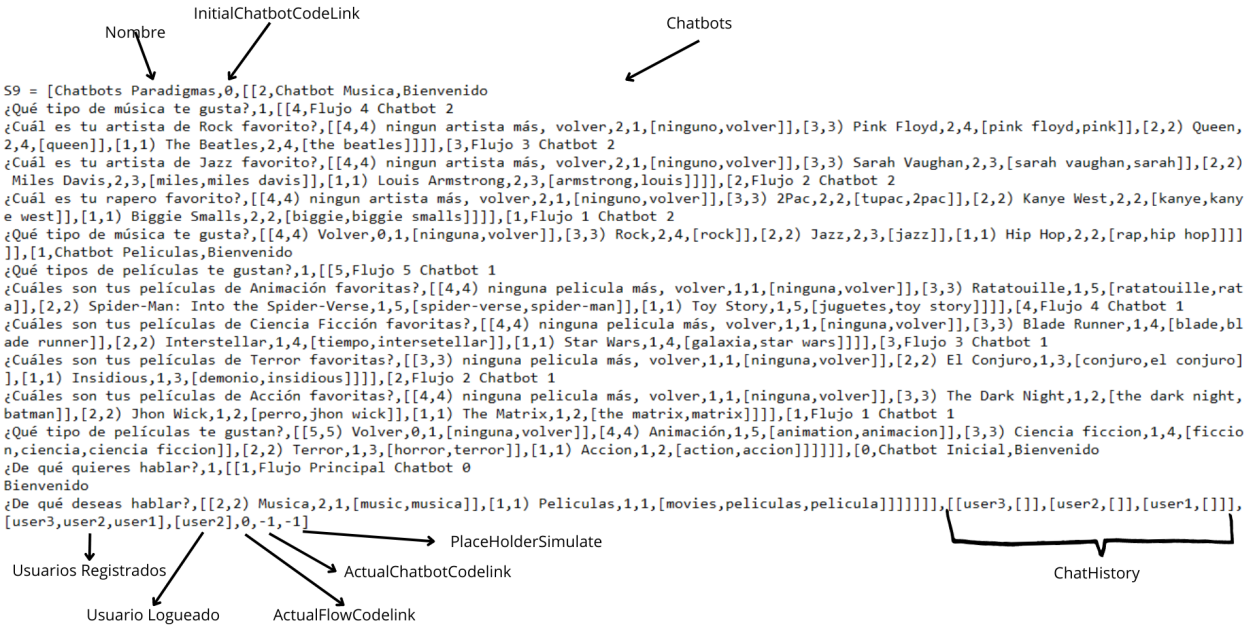

Figura 4: Estructura del sistema s9

5.4. Anexo 4: Autoevaluación

Nombre-archivo	Predicados	Autoevaluación
----------	TDAs	1
tda_option	option	1
tda_fLow	flow	1
tda_flow	flowAddOption	1
tda_chatbot	chatbot	1
tda_chatbot	chatbotAddFlow	1
tda_system	system	1
tda_system	systemAddChatbot	1
tda_system	systemAddUser	1
tda_system	systemLogin	1
tda_system	systemLogout	1
main	systemTalkRec	1
main	systemSynthesis	1
main	systemSimulate	1

Figura 5: Autoevaluación Requisitos funcionales.