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

LABORATORIO 2

PARADIGMAS DE PROGRAMACIÓN

Catalina Morales Rojas

Profesor: Roberto González
Fecha de Entrega: 21 de mayo de 2018

Santiago de Chile

1 - 2018

TABLA DE CONTENIDOS

TABLA DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

Los chatbots corresponden a programas que permiten mantener una conversación con el usuario, incorporan en su implementación sistemas de inteligencia artificial, lo cual les permite aprender a lo largo de la conversación gustos y preferencias. Las conversaciones que se establecen pueden ser a través de voz o texto. Chatbots conocidos son Siri o cortana, que cuentan principalmente con funciones de búsqueda. El área de servicios es la que cuenta con una mayor implementación de chatbots en la actualidad debido a que ahorran tiempo a los usuarios e incluso adhieren publicidad en su uso.
El objetivo del informe es presentar la manera en que fue planteado, pensado, diseñado y aplicado el laboratorio 2 del curso de Paradigmas de programación, el cual se encuentra implementado en el lenguaje de programación Prolog, bajo el paradigma lógico.

El paradigma funcional utiliza la lógica matemática para realizar relaciones entre sus elementos, su notación es simple y es de tipo declarativo.

Los objetivos del laboratorio son:

- Aplicar los conceptos aprendidos en cátedra y en la investigación personal en la creación del chatbot, haciendo uso de buenas prácticas de programación.
- Realizar una correcta implementación del paradigma lógico, sin faltar a sus estándares.
- Comprender la estructura de los hechos y predicados.
- Apreciar las ventajas y desventajas del paradigma lógico para el desarrollo del laboratorio.

CAPÍTULO 2. MARCO TEÓRICO

2.1. PARADIGMA LÓGICO

Este paradigma basa su funcionamiento en la lógica matemática, generando relaciones entre sus elementos (mediante hechos y reglas). Es un paradigma de tipo declarativo, por lo cual posee un conjunto de condiciones que permiten obtener la solución a un problema. Tal como Scheme, utiliza la recursión como estructura de control básica. Cuenta con cinco partes principales en su código, las cuales son:

1. Dominio: Corresponde a los elementos sobre los cuales opera el código (átomos, listas, entre otros).
2. Hechos: Estructuras o aserciones compuestas por términos.
3. Predicados: En una proposición, corresponde a lo que se afirma de un sujeto.
4. Metas: Corresponde a lo que busca entregar el programa, pueden ser primarias o secundarias.
5. Cláusulas de Horn: sentencias que definen los casos en que el predicado es verdadero.

Además, el paradigma lógico cumple el supuesto del mundo cerrado, en donde cuando el resultado es falso, se debe a la falta de conocimiento con respecto a lo consultado o netamente es falso.

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA

Se solicita crear un chatbot en el lenguaje Prolog, bajo el paradigma lógico, el cual se encontrará orientado a un contexto o temática. El programa debe contar con una serie de parámetros:

1. **Chatbot**: Estructura con diversos parámetros que refleja la personalidad del chatbot y las respuestas que este debe generar.
2. **Log**: Es el registro histórico de las conversaciones que ha tenido el chatbot.
3. **Seed**: Semilla que norma el comportamiento de funciones pseudoaliatorias, para lograr la variabilidad de respuestas.

Además, el programa debe contar con las siguientes funciones: **beginDialog**, **sendMessage**, **endDialog** y **Test**.

3.1 ANÁLISIS

Para implementar la solución, es necesario tener claro cuál es la representación del problema y definir el TDA. Es importante destacar, que para este paradigma solo es posible realizar consultas al programa, por lo cual las cláusulas deben ser específicas, para retornar solo lo solicitado, además se debe tener cuidado con la utilización de cláusulas not, para no salir de la base de conocimiento que posee el chatbot. En este caso, tampoco es posible modificar los hechos, por lo tanto, no se utilizará memoria a largo plazo.

Por lo tanto, lo primero que se debe realizar es generar un hecho chatbot, que contenga las posibles respuestas a los mensajes ingresados por el usuario, además de otro hecho (identificador) que permita relacionar palabras "claves" ingresadas por el usuario con las respuestas del chatbot. El resto del programa se encontrará basado en esta representación.

El programa también debe contar con constructores y funciones de pertenencia, en donde un solo predicado en este paradigma, puede realizar ambas, ya que, si se quiere acceder a un elemento, y este no corresponde a lo esperado, retornará falso. Los selectores permitirán obtener una respuesta del chatbot, por lo cual será necesario contar con una relación entre las palabras claves y las posibles respuestas, que en este caso será la posición en que se encuentra la palabra clave con respecto a la serie de respuestas que se encuentran almacenadas.

Para el caso de los modificadores, corresponden a las funciones que permiten agregar valores al input ingresado para así generar la respuesta al usuario, las cuáles serán útiles a lo largo de todo el programa, ya que es necesario modificar valores en las funciones BeginDialog, sendMessage, endDialog.

También se debe contar con otras funciones, como predicados que permitan entregar la hora y los números "random" que se deben utilizar para generar el identificador de una conversación o para obtener una de las posibles respuestas.

Finalmente, el programa debe ser capaz de generar un mensaje de bienvenida al usuario, establecer una conversación respondiendo a lo solicitado por quien lo usa o indicando que no entiende lo que el usuario busca, generar un mensaje de despedida, poder realizar una conversación de manera automática y mostrar la conversación de forma entendible al usuario.

CAPÍTULO 4. DISEÑO DE LA SOLUCIÓN

El chatbot implementado es un guía para un estudiante nuevo en la escuela ficticia Hogwarts de magia y hechicería, al cual se le podrán realizar una serie de preguntas.
La representación del TDA chatbot en este caso corresponde a dos hechos, el primero contiene una lista con todas las posibles respuestas a las frases ingresadas por el usuario:

chatbot (chatbot, [" frase1", "frase2", "…", "fraseN"]).

El segundo hecho corresponde a una lista con las "palabras claves", tal como se muestra a continuación:

idChatbot (["Plabra1", "Palabra2", "…","" PalabraN"]).

Cada una de las palabras claves consta con tres posibles respuestas.
El log (tanto inputLog como outputLog) en este caso será manejado como un string, por lo que, al momento de generar la primera consulta, este corresponderá a un string vacío ("").

El principal predicado que permite el funcionamiento del programa corresponde al predicado selector getRespuesta (el cual trabaja junto a idRespuesta), en donde se compara una palabra con la lista de palabras claves, cuando idRespuesta encuentra el valor del índice en que se encuentra la respuesta, este se multiplica por tres (debido a que cada palabra contiene tres posibles respuestas) y finalmente se suma a lo entregado por la cláusula semilla, quien genera un número "seudo aleatorio" entre 0 y 2, para seleccionar finalmente la respuesta que se entregará al usuario.

Los predicados BeginDialog y endDialog tienen un comportamiento bastante similar, ya que ambos mediante la función selectora buscan con una palabra clave ("Saludo y fin" respectivamente), un mensaje de bienvenida y despedida.

Para el caso de sendMessage, se genera una lista de string con el mensaje ingresado por el usuario en la cláusula getRespuesta2, el cual se ingresa palabra a palabra a getRespuesta, donde busca una coincidencia hasta encontrar un resultado o retornar un mensaje al usuario indicando que no entendió su pregunta.

Una herramienta utilizada en la mayor parte de los predicados corresponde a la recursividad, ya que como se trabaja con listas, estas se deben recorrer para encontrar un elemento, por lo cual es necesario recorrer toda la lista comparando lo buscado con el primer elemento, y de no ser encontrado, llamando nuevamente al predicado con la cola de esta. El tipo de recursión que se utiliza es de cola durante todo el programa, lo cual no implica grandes costos espaciales y temporales debido a que cuando encuentra el valor, retorna de manera inmediata el resultado, sin generar estados pendientes.

Finalmente, en la mayor parte de los predicados solicitados se realizó una descomposición de estos para lograr retornar el resultado esperado, el principal ejemplo es la función sendMessage, que hace uso de los predicados getRespuesta y getRespuesta2 que se encarga de obtener la frase del chatbot, el predicado semilla que permite generar el número "random" del elemento que es

tomado de chatbot y el predicado time, que entrega la hora en que se desarrolla la conversación. Por lo cual un gran problema es abordado usando el método de división y conquista y se convierte en pequeños problemas que presentan una menor dificultad.

CAPÍTULO 5. IMPLEMENTACIÓN

5.1 PROGRAMA UTILIZADO

Para ejecutar el código se utilizó el programa SWI-prolog, ya que prolog es un lenguaje interpretado.

5.2 RECURSOS UTILIZADOS

Para el funcionamiento del prograga se debe contar con el archivo "chatbot.pl", en donde se encuentran los hechos chatbot que contiene las posibles respuestas del programa, idChatbot con las palabras claves utilizadas a lo largo del programa y user que contiene a los tres usuarios con los que el programa puede generar una conversación de manera automática.

CAPÍTULO 5. RESULTADOS

El resultado del laboratorio corresponde a un chatbot que puede responder consultas del usuario, generar una conversación de manera automática y mostrar el resultado de manera clara. Debido a que este paradigma solo permite realizar consultas, el retorno que se genera es un resultado booleano, pero en este caso, se retorna el valor faltante del predicado ingresado, que permite que el predicado sea verdadero. Por lo tanto, si los parámetros ingresados en la consulta son incorrectos, el chatbot mostrará el mensaje **falso**.

Este chatbot no permite almacenar información, debido a que los hechos no se pueden modificar, por lo que es imposible realizar uso de memoria a largo plazo.

A pesar del uso de recursión en la mayor parte del laboratorio, no se presenta un retardo en el desarrollo, esto es debido a que no se maneja una gran cantidad de información, además de que la recursión utilizada es de cola, la cual entrega el resultado inmediatamente cuando lo encuentra.

Dentro de las limitaciones del laboratorio, no fue posible abordar una gran cantidad de respuestas a las preguntas del usuario, esto se debe a que se utiliza solo un identificador para relacionar la frase que ingresa el usuario con la respuesta. Por lo cual, si se busca algo en un tiempo verbal distintos al que conoce el chatbot u otra conjugación, a pesar de poseer la información el chatbot no logrará responder. El resultado de avance de las funciones obligatorias fue el requerido en el enunciado en las funciones beginDialog, sendMessage, endDialog, logToStr y test. Y en el caso de las funciones extras, solo fue posible implementar possiblesResponses.

```
?- logToStr('*********BEGIN DIALOG********** << 4935 >> \n <<[0:19]
21/5/2018>> HOGWARTSBOT: Bienvenido a Hogwarts, hoy sere tu guia en el castillo,
En que puedo ayudarte? \n <<[0:20] 21/5/2018>> USUARIO: quiero saber de Ravenclaw
\n <<[0:20] 21/5/2018>> HOGWARTSBOT: La casa Ravenclaw premia el aprendizaje, la
sabiduria, el ingenio, y el intelecto de sus miembros. Que mas quieres saber? \n
**********END DIALOG********** << 4935 >> \n <<[0:20] 21/5/2018>>
HOGWARTSBOT: Adios, Cuando necesites ayuda no dudes en buscarme ' ,StrRep).
*********BEGIN DIALOG********** << 4935 >>
<<[0:19] 21/5/2018>> HOGWARTSBOT: Bienvenido a Hogwarts, hoy sere tu guia en el
castillo, En que puedo ayudarte?
<<[0:20] 21/5/2018>> USUARIO: quiero saber de Ravenclaw
<<[0:20] 21/5/2018>> HOGWARTSBOT: La casa Ravenclaw premia el aprendizaje, la
sabiduria, el ingenio, y el intelecto de sus miembros. Que mas quieres saber?
**********END DIALOG********** << 4935 >>
<<[0:20] 21/5/2018>> HOGWARTSBOT: Adios, Cuando necesites ayuda no dudes en
buscarme
StrRep = '*********BEGIN DIALOG********** << 4935 >> \n <<[0:19] 21/5/2018>>
HOGWARTSBOT: Bienvenido a Hogwarts, hoy sere tu guia en el castillo, En que puedo
ayudarte? \n <<[0:20] 21/5/2018>> USUARIO: quiero saber de Ravenclaw \n <<[0:20]
21/5/2018>> HOGWARTSBOT: La casa Ravenclaw premia el aprendizaje, la sabiduria, el
ingenio, y el intelecto de sus miembros. Que mas quieres saber? \n **********END
DIALOG********** << 4935 >> \n <<[0:20] 21/5/2018>> HOGWARTSBOT: Adios,
Cuando necesites ayuda no dudes en buscarme '.
```

Figura 1.- Ejemplo conversación con el chatbot.

CAPÍTULO 6. CONCLUSIONES

Debido a que el paradigma lógico se encuentra basado en la generación de relaciones entre los elementos, es bastante útil para realizar programas en donde es necesario establecer vínculos entre sus componentes, como lo fue en el caso del chatbot presentado a lo largo del informe, en donde la principal relación corresponde a las palabras claves y las respuestas. Además, si las cláusulas cuentan con un buen criterio de implementación (principalmente para los casos base de funciones recursivas), serán pocas las líneas que conformarán el código, lo cual hace sencilla su implementación.

Entre las dificultades que presenta el paradigma, específicamente en este laboratorio se encuentra el no uso de ciclos, lo que hace imposible entregar consultas de predicados que requieren recorrer elementos sin usar la recursión, además de no poder modificar los elementos que se encuentran contenidos en los hechos, lo que limita la memoria que puede tener el chatbot. Por otro lado, debido a que se basa en el supuesto del mundo cerrado, hay que tener sumo cuidado con la definición de relaciones, ya que, si falta alguna, el programa puede entregar como verdadero algo que no lo es.

El uso de recursión de cola fue una buena técnica, ya que permite entregar el resultado de manera inmediata cuando se llega a la condición base. Además, es un paradigma que puede ser denominado como “simple”, ya que las funciones son pequeñas, claras y concisas.

Al ser comparado con el paradigma funcional utilizado en el laboratorio anterior, el paradigma lógico fue más fácil de implementar debido a que fue desarrollado en menos líneas de código y es el lenguaje quien realiza todo el proceso, dejando al programador poco trabajo. Ambos paradigmas presentan también similitudes, ya que ambos son de tipo declarativo y usan como principal herramienta la recursión.

Con respecto a los objetivos planteados al inicio, se logró cumplir con la aplicación de los conceptos aprendidos en cátedra y en la investigación personal para la creación del chatbot, haciendo uso de buenas prácticas de programación, comprender la estructura del paradigma lógico y apreciar las ventajas y desventajas del paradigma para el desarrollo del laboratorio.

BIBLIOGRAFÍA

- Anónimo, 2009 “Paradigma lógico” (Recuperado 19/05/2018) https://kevinldp.wordpress.com/4-paradigma-logico/