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

LABORATORIO 2

PARADIGMAS DE PROGRAMACIÓN

Alberto Rodríguez Z.

Profesores:	Roberto González
	Daniel Gacitúa
	Víctor Flores
Fecha de Entrega:	21-05-2018

Santiago de Chile

1- 2018

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCIÓN…….....4

CAPÍTULO 2. MARCO TEÓRICO…….5
2.1. PARADIGMA LÓGICO…...……....5

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA……...6
3.1. ANÁLISIS……..6

CAPÍTULO 4. DESCRIPCIÓN DE LA SOLUCIÓN…….7

CAPÍTULO 5. RESULTADOS……...10

CAPÍTULO 6. CONCLUSIONES……..11

REFERENCIAS.…….12

TABLA DE FIGURAS

Figura 1.- Diagrama funcionamiento paradigma lógico……...5
Figura 2.- Predicado "SemillaDeDecisiones"…….8
Figura 3.- Predicado que comprueba los strings……...9
Figura 4.- Conversación probando resultados…….10
Figura 5.- Prueba del predicado "test"…….10

CAPÍTULO 1. INTRODUCCIÓN

El presente informe tiene como objetivo principal ser una referencia a la línea de pensamiento y al contexto de desarrollo del código fuente que lo acompaña para la presentación del laboratorio 2 del curso de Paradigmas de Programación. En esta ocasión se hará uso del paradigma de programación Lógico con el lenguaje de programación Prolog.

En primer lugar, el problema a resolver es la creación e implementación de un chatbot. De este mismo modo, es relevante mencionar que un chatbot se puede definir como programas computacionales que pueden mantener una conversación con un ser humano, sin embargo, esta interacción queda limitada por un contexto/temática y una serie de parámetros. Asimismo, los chatbot resultan ser de utilidad para manejar algunas fases de la conversación como pueden ser saludos de bienvenida, solicitud de datos y responder a algunas preguntas específicas hechas por el usuario.

Por consiguiente, en este informe la solución implementada para el chatbot trabaja en base a la temática de atención al cliente en un negocio de comida rápida, en donde se crean flujos conversacionales protocolares, estructuras gramaticales simples y vocabulario conocido por el chatbot. De tal forma dando los parámetros y temática se puede mantener una conversación con el chatbot.

Por lo tanto, se establece como objetivo de este laboratorio, la aplicación de las técnicas aprendidas en clases, además ir más allá e investigar cuando corresponda, con el fin de crear un algoritmo eficaz que funcione bajo el paradigma lógico.

CAPÍTULO 2. MARCO TEÓRICO

2.1 PARADIGMA LÓGICO

Este paradigma está basado en la lógica formal, es decir, analiza la validez de los razonamientos teniendo en cuenta únicamente el valor de la verdad (verdadero o falso) de cada enunciado. El paradigma Lógico se orienta en definir las metas de un cómputo en vez desarrollar un algoritmo que pueda alcanzar dichas metas, las unidades de programación de este lenguaje son los hechos y las reglas (también llamadas Cláusulas de Horn).

La Programación Lógica y la Funcional (paradigma utilizado en el laboratorio anterior) forman parte de la conocida Programación Declarativa, es decir, la programación consiste en indicar como resolver un problema mediante sentencias, en la Programación Lógica se trabaja en una forma descriptiva, estableciendo relaciones entre entidades. La idea esencial de este paradigma es que el programador se enfoca solamente en la lógica, el control del algoritmo se delega a la máquina abstracta.

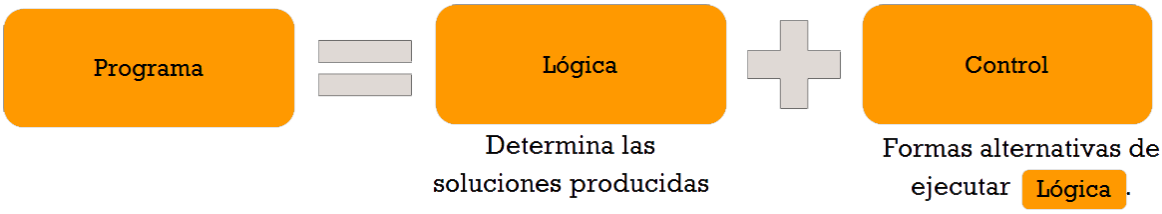

Figura 1.- Diagrama del funcionamiento del paradigma Lógico

Al solo definir las metas y sus condiciones de forma lógica, se deja al interprete lógico (máquina abstracta) busca un conjunto de todas las soluciones posibles para dicho problema, esta técnica se llama resolución.

Como ya mencionado las unidades de programación de este paradigma son los hechos y las reglas:

Hecho: es una relación entre objetos, un registro de información verdadera.

Regla: Formulación deductiva donde nuevos hechos pueden ser inferidos de hechos existentes, se compone de dos partes, la cabeza y el cuerpo, el cuerpo formado por varios hechos.

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA

Se solicita crear un algoritmo en el lenguaje de programación Prolog que simule la conversación entre el usuario y un chatbot, un chatbot que contiene personalidad y un vocabulario. La conversación debe funcionar en base a una temática manteniendo una conversación fluida con un inicio de la conversación y un final. La conversación se va almacenando en un registro llamado log que se divide en dos, la variable InputLog que corresponde al registro histórico de ha sostenido el chatbot, y la variable OutputLog que corresponde a la salida de la mayoría de los predicados solicitados. También existe la semilla seed que normará el comportamiento de los predicados ante funciones pseudoaleatorias. El algoritmo debe poder efectuar los siguientes predicados:

1.**beginDialog**: Permite iniciar una nueva conversación con el chatbot donde el mensaje inicial es ofrecido por el chatbot.
2.**sendMessage**: Permite realizar envío de los mensajes del usuario al chatbot, donde este debe poder generar una respuesta a este mensaje.
3.**endDialog:** Permite dar cierre a la conversación donde el mensaje final es enviado por el chatbot.
4.**logToStr:** Permite obtener una representación comprensible para su usuario de su TDA log, transforma la representación en un string en el cual se puede usar un display con el resultado de este predicado.
5.**test:** Permite ilustrar el desarrollo de una conversación entre un usuario y un chatbot. Este usuario en una lista de strings que corresponder a los mensajes enviados por el usuario.

Además de estos predicados, se deben implementar abstracciones de una estructura TDA de 6 capas y se debe representar e implementar el TDA en los predicados del código sin perjudicar la eficiencia de este.

Todos estos predicados tienen un dominio y una meta definidas para su desarrollo, aunque se tiene libertad en algunos casos del orden o que predicados a parte de estos se trabajen en ellas.

3.1 ANÁLISIS

El principal problema radica en la capacidad de darle “sentido” a una conversación, ya que cuando el usuario envía un mensaje, el chatbot debe ser capaz de responderle lógicamente en relación con lo preguntado por este. Por lo tanto, debe existir una gran posibilidad de respuestas para que el chatbot tenga la capacidad de seleccionar la indicada.

De otro modo, uno de los puntos complejos de trabajar en Prolog, es que al no contar con las instrucciones While y For, no se puede iterar la cantidad de respuestas que puede existir, la solución se debe desarrollar únicamente con reglas y hechos, además de saber trabajar recursivamente sobre las listas de Prolog.

CAPÍTULO 4. DESCRIPCIÓN DE LA SOLUCIÓN

Puesto que hay predicados y parámetros definidos, el desafío se centra en poder trabajar estos predicados entre sí, ósea si se cumple una regla sirva como hecho de otra regla y así poder obtener las salidas de los predicados. Siguiendo el orden de los parámetros se define primero la estructura chatbot.

El chabot está formado mayormente de hechos, estos contienen todas las posibles respuestas que entrega el chatbot al usuario, entre las respuestas están los saludos de bienvenida, las despedidas y las determinadas respuestas de alguna pregunta echa por el usuario. Además, existen tres personalidades que son amable, animado y agresivo, entonces el estado de ánimo de la respuesta que envía el chatbot es por una de estas personalidades, la personalidad que adoptará el chatbot será escogida por el usuario. El saludo es una regla que devuelve el mensaje contenido en algún hecho de saludo, este saludo dependerá de la hora actual a la que sea ejecutado el programa, es decir si se realiza en la mañana, tarde o noche. El programa se desarrolla en SWI-Prolog por lo que tendemos que usar una función definida en este lenguaje, la función usada es “get_time” con la que obtendremos la hora actual seguido de la función “convert_time” que se obtendrá la fecha y la hora exacta. La despedida también es una regla que devuelve el mensaje contenido en algún hecho de despedida.

El numero de la semilla seed les da variabilidad a las respuestas del chatbot, por lo que existen tres posibles respuestas para una cierta pregunta del usuario. Para que siempre la semilla de una de estas posibles respuestas se crea el predicado “semillaDeDecisiones” donde su salida puede ser 1,2 o 3 (que equivale al resto de la semilla ingresada), las cuales cada uno representa una respuesta a una determinada pregunta, existe una respuesta del chatbot según su personalidad y según el numero seed, por lo que existen nueve posibles respuestas a una determinada pregunta del usuario (tres para cada personalidad), ejemplo como el chatbot es sobre un negocio de comida rápida hay respuestas que ofrecen ofertas, cosa que si uno elige otro seed talvez no devuelva esta misma respuesta.

```
% ////////////////////////////////SEED/////////////////////////////////

semillaAmable(1).
semillaAnimado(2).
semillaAgresivo(3).

%regla que determina que numero tendra el SEED que puede ser 1,2 o 3
%la entrada son el Seed ingresado por el usuario y un validador
%la meta es obtener el numero 1,2 o 3 como semilla
semillaDeDecisiones(Seed,SEED,VAL):- semillaAgresivo(SEED),VAL = 0,VAL is Seed mod 3;
                                     semillaAmable(SEED),VAL = 0,VAL is Seed mod 2;
                                     semillaAnimado(SEED),VAL = 1,VAL is Seed mod 2.
```

Figura 2.- Predicado que tiene como meta devolver el numero de la semilla

De otro modo, ya que en Prolog no existen las iteraciones se necesita poder enviar información de un predicado a otra, por lo que se trabaja con el TDA de listas en Prolog, ya que tienen la implementación y representación necesaria para poder solucionar el problema. Por otra parte, el log, que es el registro histórico de conversaciones, se trabaja como lista, InputLog corresponde al registro histórico y el OutputLog al nuevo registro con el nuevo mensaje agregado.

Comenzando con el predicado “beginDialog” se ingresa la personalidad del chatbot, el inputLog que inicialmente es una lista vacía, la semilla seed y el outputLog que será la variable de salida con el log modificado, esta es enviado a un constructor que creará el outputLog, primero comprueba si la lista ingresada es una lista vacía y se transforma la respuesta del chatbot como un string y se agrega como primer elemento en el outputLog. Lo mismo ocurre con el predicado “endDialog” que crea el outputLog en el mismo constructor pero la lista inputLog no debe ser vacía, con lo que agregar el mensaje de despedida al a lista OutputLog que contendría el log anterior (InputLog) y el nuevo mensaje.

Para desarrollar el predicado “sendMessage” es parecido al anterior, pero tiene un parámetro adicional que es el mensaje enviado por el usuario (string), este predicado igual es enviado a un constructor que agrega a la lista el mensaje enviado por el usuario y la respuesta del chatbot, cada uno como los dos últimos elementos del OutputLog. Para que el chatbot sepa que tiene que tiene que responder existe un predicado que deba saber que debe responder el chatbot ante tal mensaje.

El predicado antes mencionado se llama “respuestaCbot” tiene como entradas el mensaje, el chatbot y la semilla seed, en la cual se verifica que responder. Primero el mensaje del usuario se transforma en una lista con cada palabra como elemento, esto se realiza gracias a la función ya definida por prolog llamada “split_string”, después se verifica cual es la personalidad del chatbot para saber con qué estado de ánimo responderá el chatbot, para después pasar a una seguidilla de “or” en donde entre cada uno hay una cierta cantidad de strings. Si esos string están contenidos en la lista de palabras devolverá la respuesta determinada que también dependerá de la personalidad y del seed. Si esos string no están contenidos en la lista pasará

al siguiente condicional y así sucesivamente. Si la palabra no cumple con ninguna condición la última condición arrojará como mensaje del chatbot que no entendió el mensaje.

Para poder comparar los string se necesita poder recorrer la lista, para eso la recorreremos recursivamente hasta que encuentre los strings o la lista quede vacía.

```
% aplica un selector del TDA de lista ya que puede dividir a la lista en
% cabeza y la cola y asi poder obtener los elementos de ella
comprobarPalabra([X|_],Str1,Str2,Str3,Str4):-
    (X=Str1;X=Str2;X=Str3;X=Str4).

comprobarFraseCorta([X|Xs],Str1,Str2,Str3,Str4):-
    X = Str1,comprobarFraseCorta2(Xs,Str2,Str3,Str4).

comprobarFraseCorta2([X|_],Str2,Str3,Str4):-
    X=Str2;X=Str3;X=Str4.


comprobarFrase(Mensaje,Str1,Str2,Str3):-
    contiene(Mensaje,Str1),comprobarFrase2(Mensaje,Str2,Str3).
comprobarFrase2(Mensaje,Str2,Str3):-
    contiene(Mensaje,Str2),comprobarFrase3(Mensaje,Str3).
comprobarFrase3(Mensaje,Str3):-
    contiene(Mensaje,Str3).

% aplica un selector del TDA de lista ya que puede dividir a la lista en
% cabeza y la cola y asi poder obtener los elementos de ella
contiene([X|_],X).
contiene([_|Xs],Y):- contiene(Xs,Y).
```

Figura 3.- Como comprueba si los string están contenidos en la lista, existe un predicado para cada tamaño de la lista

Para poder representar esta lista OutputLog en algo comprensible para el usuario existe el predicado “logToStr” que entrega como salida un string que contiene todos los elementos de la lista, y al aplicarle un display a esta salida se logra la representación.

Existe también el predicado “test” que simula la conversación completa de inicio a final, se ingresan los mismo parámetros que “beginDialog” pero además se ingresa como parámetro user una lista que contiene como elementos los mensajes de la conversación (strings) y cada uno es la respuesta siguiente del usuario, a este predicado se le aplica un recursión de cola que va guardando la conversación en el outputLog, el caso base es cuando el acumulador es mayor que el largo de la lista user.

Para finalizar, la descripción de la solución se realiza un predicado extra que es el predicado “possibleResponses” que consiste en las posibles respuestas que tiene un chatbot ante una pregunta y un estado de ánimo determinado. Lo que hace este predicado es obtener la primera puesta del chatbot con el seed=1 y eso lo agrega a la lista responses, y así realiza lo mismo, pero después con seed=2 y seed=3 y agrega las tres respuestas a la lista de salida.

CAPÍTULO 5. RESULTADOS

Tiempo de respuesta aceptable, es decir, el usuario no nota o queda en espera de resultados, ya sea cuando envía el mensaje en el que el chatbot tiene que responder corriendo todas las variadas posibilidades para verificar, a pesar de usar algunas funciones recursivas, ya que podría tardarse más de lo esperado.

Se realizaron todos los predicados obligatorios más uno de los predicados extras, todos funcionaban de manera exitosa en las pruebas que se hicieron. Las pruebas que se hicieron eran ir cambiando los parámetros del chatbot y del seed, viendo si existía cambios en las respuestas del chatbot, además de preguntarle al chatbot siempre relacionado con el contexto.

Como muestra la figura 4, ilustra una conversación uniendo todos los predicados antes mencionados (excepto el test y el possibleResponses) en los que va variando el seed ingresado por el usuario, la personalidad del chatbot se mantiene constante.

```
?- beginDialog(amable,[],4,OutputLog),sendMessage("hola, me gustaria una hamburguesa",amable,OutputLog,6,OutputLog2),sendMessage("yap, quiero una hamburguesa vegetariana",amable,OutputLog
2,10,OutputLog3),endDialog(amable,OutputLog3,2,OutputLog4),logToStr(OutputLog4,StrRep),write(StrRep).
BeginDialog el 20/5/2018 a las 20:10:19 -Chatbot: Buenas tardes bienvenido a Mr. Ham que desea ordenar?
20/5/2018 a las 20:10:19 -Usuario: hola, me gustaria una hamburguesa
20/5/2018 a las 20:10:19 -Chatbot: Pero que tipo quieres? especifica
20/5/2018 a las 20:10:19 -Usuario: yap, quiero una hamburguesa vegetariana
20/5/2018 a las 20:10:19 -Chatbot: Bueno Anotado,quiere algo mas?
EndDialog el 20/5/2018 a las 20:10:19 -Chatbot: Adios que le vaya bien
```

Figura 4.- Conversación probando resultados.

Así se obtienen los datos de una conversación determinada, también se puede ver en la figura 5 una prueba de la función test.

```
?- test(user2,agresivo,[],5,OutputLog),logToStr(OutputLog,StrRep),write(StrRep).
BeginDialog el 20/5/2018 a las 20:17:55 -Chatbot: buenas tardes, que quiere?
20/5/2018 a las 20:17:55 -Usuario: hola como estas
20/5/2018 a las 20:17:55 -Chatbot: Ando Pesado asi que no pregunti mejor
20/5/2018 a las 20:17:55 -Usuario: que bueno, me gustaria una hamburguesa
20/5/2018 a las 20:17:55 -Chatbot: Tenemos muchos tipos cual desea?
20/5/2018 a las 20:17:55 -Usuario: que pesado
20/5/2018 a las 20:17:55 -Chatbot: Si que tanto problema?, ya mejor pide algo
20/5/2018 a las 20:17:55 -Usuario: ok..., quiero una bebida pepsi
20/5/2018 a las 20:17:55 -Chatbot: no quedan, pide otra
20/5/2018 a las 20:17:55 -Usuario: no
20/5/2018 a las 20:17:55 -Chatbot: Ya porfin dejaste de pedir
EndDialog el 20/5/2018 a las 20:17:55 -Chatbot: Chao
```

Figura 5.- Prueba del predicado test, representado gracias al predicado logToStr.

CAPÍTULO 6. CONCLUSIONES

Realizando este laboratorio se pudo concluir que el paradigma de programación Lógico tiene las ventajas de ser un paradigma simple de desarrollar, generación rápida de prototipos e ideas complejas, sencillez en la implementación de estructuras complejas y lo mejor es cómo puede mejorarse la eficiencia solo modificando el componente de control sin tener que modificar la lógica del algoritmo.

En comparación al paradigma funcional es bastante parecido, ya que los predicados de prolog son muy similares a las funciones de Scheme, pero la desventaja que tiene el paradigma lógico es que si el programa no contiene suficiente información para contestar una consulta responderá false, en cambio el paradigma funcional te entrega algún resultado así se hace más fácil verificar cual podría ser el error.

En segundo lugar, respecto al problema en si del laboratorio, ha sido interesante plantear el problema de cómo poder desarrollar una conversación con un chatbot, a través de los distintos predicados y técnicas que tiene el paradigma lógico, junto con las implementaciones que tiene el SWI-Prolog. De modo que, se puede afirmar que se realizó un gran trabajo con Prolog al poder trabajar directamente con el TDA de las listas y aplicando recursión. Una de las limitaciones sí que tiene este lenguaje es que, al solo trabajar con hechos y reglas, como los hechos son las respuestas del chatbot existen una gran cantidad, entonces se deben definir demasiados hechos para poder cumplir todas las reglas de respuesta.

Finalmente, puesto que se ha logrado desarrollar un chatbot con cierta temática, con el cual uno puede mantener una conversación, es posible concluir que se ha logrado cumplir con el objetivo principal del laboratorio.

REFERENCIAS

- Kevin Castro, Juan David Rojas. *Programación Lógica.* (España) Recuperado de: https://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/lang.html

- Curso de Paradigmas de programación. *Presentaciones paradigmas catedra(2018-1).* (Chile) Recuperado de https://drive.google.com/drive/folders/1HbFEr1f5MCntqecuqNRPfEsoVowyYssy