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

LABORATORIO 1
PARADIGMAS DE LA PROGRAMACIÓN

BASTIÁN GONZALO VERA PALACIOS

Profesor: Roberto González

Santiago - Chile
1-2018

Tabla de Contenido

Índice de Figuras

CAPÍTULO 1. INTRODUCCIÓN

Dentro de los principales paradigmas para los distintos tipos de problemas que se puede encontrar, en este informe se hará referencia al pensamiento y la resolución del problema señalado en el laboratorio 1, el cual se debe realizar a través del paradigma Funcional con el lenguaje de programación Scheme.

El problema a resolver es la creación de un chatbot, un mecanismo el cual permite generar una conversación con un usuario con un propósito, el cual puede ser desde ofrecer un servicio hasta resolver consultas sobre algún tema en particular.

Para poder resolver este problema se tendrá que desarrollar un método el cual pueda analizar el mensaje entregado por el usuario, encontrar los principales conceptos dentro de él, y posteriormente entregar la respuesta respectiva a la consulta realizada, todo esto hasta que el usuario decida terminar la conversación y entregar un puntaje respecto a la calidad de ésta.

Dado lo mencionado anteriormente, se plantea como objetivo de este laboratorio, poder aplicar e implementar todas las técnicas y métodos aprendidos en clases, junto a la investigación de posibles recursos que presente el lenguaje y el paradigma para facilitar así el desarrollo y el trabajo en este.

CAPÍTULO 2. MARCO TEÓRICO

2.1 REQUIRE RACKET/DATE

Esta herramienta importada de Racket que fue empleada para obtener valores como la hora y fecha del sistema.

2.2 LAZY EVALUATION

Lazy Evaluation (también conocida como Evaluación Perezosa) es una estrategia de evaluación utilizada para retrasar la ejecución de una orden hasta el momento que esta sea necesaria, en la cual se realizará su llamado. Esta técnica se utiliza con el propósito de incrementar el rendimiento de un algoritmo gracias a evitar realizar cálculos innecesarios.

2.3 ESTRUCTURAS

Las estructuras son una método de almacenar información de distintos tipo con el propósito de facilitarnos el manejo de esta información, en este caso, la creación del Chatbot junto con sus características particulares, tales como sus respuestas preestablecidas para la conversación entre otros.

2.4 TDA

Los tipos de datos abstractos son una herramienta facilita la administración de ciertos valores los cuales podemos trabajar con características propias a nuestra implementación, facilitando el uso de estas durante el desarrollo de el trabajo.

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA

Se solicita la creación de un Chatbot, el cual corresponde a un sistema de respuestas automáticas según la lectura de un mensaje ingresado por el usuario, logrando así una conversación fluida en la cual el usuario pueda lograr desde recibir información hasta solicitar algún servicio, entre otras cosas.
Dentro de los principales requisitos solicitados para la creación y el funcionamiento de este programa se encuentran:

- **Log**: Registro el cual contendrá la conversación que se desarrolle entre el usuario y el chatbot.
- **beginDialog** : Función que indica el comienzo de la conversación y genera un mensaje de bienvenida hacia el usuario.
- **endDialog** : Función que indica el término de la conversación entregando un mensaje de término hacia el usuario.
- **sendMessage**: Función en la cual el usuario entrega un mensaje junto con ciertos valores establecidos.
- **rate**: Sistema de puntuación en donde el usuario como el mismo chatbot evalúa la calidad de la conversación.
- **test**: Función encargada de realizar una prueba respecto al funcionamiento del chatbot y sus respuestas.

Estas funcionalidades son la base del planteamiento del chatbot. Dentro de los otros requisitos para el desarrollo de este chatbot, se encuentran:

- el uso de un método llamado *Lazy Evaluation*, el cual en algún punto de la conversación podrá en espera la respuesta a una de las preguntas que realice el usuario, con el propósito de que le entregue la respuesta cuando éste lo solicite.
- La implementación de una personalidad propia para el chatbot, con distintos tipos de respuestas según se plantee la situación.
- Manejo del historial de la conversación con el fin de utilizar la información en caso de necesitarla.

Toda la ejecución de el programa se realizará desde la consola del programa, por ende no se necesita ningún elemento externo para la ejecución del código.

CAPÍTULO 4. DESCRIPCIÓN DE LA SOLUCIÓN

La primera etapa para el desarrollo del chatbot, fue establecer las características propias, tales como sus respuestas y sus valores iniciales de rate, y definirlas dentro de la estructura con el mismo nombre.

```
(define-struct chatbot (
                  mensajeBienvenida  ; listado de Mensajes de Bienvenida que presenta el chatbot
                  mensajePregunta    ; listado de preguntas al usuario que presenta el chatbot
                  mensajeFinal       ; listado de frases de despedida del chatbot
                  mensajeLazy        ; mensaje entregado a la hora de realizar una evaluación Perezosa
                  sintomas           ; listado de sintomas que puede presentar el paciente
                  diagnosticos       ; listado de diagnosticos a entregar al paciente segun su sintoma
                  id                 ; identificador de cada chatbot
                  rateUser           ; puntuacion entregada por el usuario
                  rate               ; puntuacion generada por un algoritmo
                  )
  )
```

Figura 1:Estructura del Chatbot

Posterior a la estructura, se genera un TDA para poder facilitar la implementación y el uso de ciertos datos. En este caso, se plantea como TDA un modelo que consiste en un String de dos elementos, los cuales representan el que interpreta el mensaje y el mensaje, respectivamente. Junto con el Constructor, se generan el resto de funciones necesarias para una correcta implementación de los TDA, tales como la función de pertenencia, selectores y modificadores.

```
(define (datos usuario mensaje)
  (if (and (string? usuario) (string? mensaje))
      (string-append usuario " : " mensaje)
      ""
      )
  )
```

Figura 2: Constructor TDA

Una vez creadas y definidas las estructuras sobre las cuales se va a desarrollar el chatbot, se implementan las funciones *beginDialog, endDialog y sendMessage* para empezar a generar las instancias de diálogo.

BeginDialog consiste en una función que recibe el log, correspondiente a una lista, y sobre esta genera un mensaje que contiene una etiqueta numérica que indica el inicio de la conversación, los valores actuales de tiempo, el emisor del mensaje y una bienvenida hacia el usuario, los cuales son ingresados al log y retornados al final de la función.

De manera semejante, la función *endDialog* recibe el log posteriormente modificado y le agrega un mensaje de despedida al usuario, junto con una etiqueta numérica al final, que indica el fin de la conversación.

La función *sendMessage* es la que realiza la mayor cantidad de funciones extra, debido a que recibe el mensaje de el usuario, realiza una primera evaluación para así poder obtener el nombre del usuario y realizar así respuestas más personalizadas.

Posterior a esto recurre a otra función llamada *IntersectarListas*, la que gracias a la técnica de recursión, realiza una comparación entre el mensaje de el usuario convertido en una lista con cada una de las palabras y la lista de posibles respuestas en la información que posee el chatbot con el fin de poder señalar si se puede realizar la respuesta o no. De no poder responder ya que el chatbot no posea la información, este genera una respuesta donde señala que no entendió el mensaje. En caso de que si halle una respuesta válida para el usuario, se llama a la función recursiva *entregarDiagnostico*, la cual vincula una palabra clave con una respuesta valida sobre está.

Durante toda esta selección de qué mensaje se le entrega al usuario, aparece el uso del valor *seed,* que posterior a ser entregado por el usuario, se utiliza dentro de la función myRandom para así, con un cálculo matemático, entregar alguna de las posibles soluciones planteadas.

SendMessage puede ser llamada cuantas veces el usuario quiera realizar consultas respecto a la temática del chatbot.

Una vez que el usuario decida terminar con la conversación realizando un *endDialog*, se habilita la función *Rate*, la cual recibiendo el chatbot y un puntaje entregado por el usuario, recurre a una función anidada que genera gracias a *make-struct,* una estructura nueva la cual posee un nuevo valor de id, el rate entregado por el usuario y un rate propio generado por la función *calculoRate*, que recibe el log y dependiendo del largo de la conversación, entrega un puntaje.

Para finalizar se emplea la función *test,* la cual consiste en una función recursiva y anidada que se utiliza como herramienta para realizar una prueba del funcionamiento e implementación de todas las funciones anteriores. Para su funcionamiento se entrega como parámetro un usuario de prueba (user1, user2, …) los cuales simulan respuestas de usuarios, evaluadas con el chatbot.

```
(define (test user chatbot log seed)
  (define (test2 user chatbot log seed)
    (if (null? user)
        (endDialog bot log seed)
        (test2 (cdr user) chatbot (sendMessage (car user) bot log seed) seed)
        )
    )
  (test2 user chatbot (beginDialog bot '() seed) seed)
  )
```

Figura 3: Función Test

CAPÍTULO 5. RESULTADOS

Como resultado de la implementación mencionada anteriormente, se obtuvo un chatbot capaz de responder correctamente a cada una de las inquietudes solicitadas por el usuario y a su vez, como se puede apreciar en la *Figura 4* y en la *Figura 5,* dependiendo de los parámetros que se ingresen en la función *test,* el chatbot genera distintos caminos de conversaciones.

```
> (test user1 bot '() 34)
(1
 "23-4-2018 2:30:41 -> chatbot : Estas hablando con su asistente de salud virtual ¿Cual es tu nombre?"
 "23-4-2018 2:30:41 -> usuario : Maria"
 "23-4-2018 2:30:41 -> chatbot : perfecto, Maria ¿presenta usted alguna molestia o malestar?"
 "23-4-2018 2:30:41 -> usuario : me duele demasiado"
 "23-4-2018 2:30:41 -> chatbot : disculpa pero no entiendo a lo que te refieres, ¿me lo puedes repetir porfavor?"
 "23-4-2018 2:30:41 -> usuario : disculpe, me duele demasiado el estomago"
 "23-4-2018 2:30:41 -> chatbot : es probable que tengas gastroenteritis, asi que tendras que cuidar tu alimentacion"
 "23-4-2018 2:30:41 -> chatbot : tiene alguna otra consulta?"
 "23-4-2018 2:30:41 -> usuario : por lo mismo me cuesta respirar"
 "23-4-2018 2:30:41 -> chatbot : por lo que me cuentas puede que estes con bronquitis, asi que te recomiendo que tomes jarabe para la tos"
 "23-4-2018 2:30:41 -> chatbot : tiene alguna otra consulta?"
 "23-4-2018 2:30:41 -> usuario : y tambien siento calor en demasía"
 "23-4-2018 2:30:41 -> chatbot : es probable que tengas fiebre, te recomiendo usar paños frios para reducir la temperatura"
 "23-4-2018 2:30:41 -> chatbot : tiene alguna otra consulta?"
 "23-4-2018 2:30:41 -> usuario : no"
 "23-4-2018 2:30:41 -> chatbot : Me alegra haber resuelto tus dudas :D"
 "23-4-2018 2:30:41 -> chatbot : "
 "23-4-2018 2:30:41 -> chatbot : Hasta luego, Cuidese y ante cualquier duda puede volver a consultar :)"
 1)
```

Figura 4: Demostracion Función Test con user1

```
> (test user2 bot '() 31)
(1
 "23-4-2018 2:47:31 -> chatbot : Bienvenido! me llaman asistente de salud virtual y tu? ¿como te llamas? "
 "23-4-2018 2:47:31 -> usuario : Rodrigo"
 "23-4-2018 2:47:31 -> chatbot : hola Rodrigo ¿con que puedo ayudarle esta vez respecto a su salud? "
 "23-4-2018 2:47:31 -> usuario : en las mañanas me cuesta mucho respirar"
 "23-4-2018 2:47:31 -> chatbot : por lo que me cuentas puede que estes con bronquitis, asi que te recomiendo que tomes jarabe para la tos"
 "23-4-2018 2:47:31 -> chatbot : tiene alguna otra consulta?"
 "23-4-2018 2:47:31 -> usuario : aveces en la noche tengo muchos mocos"
 "23-4-2018 2:47:31 -> chatbot : por lo que me cuentas puedo decir que te encuentras con un resfriado comun, por lo que necesitas descansar, hidratarte y un tapsin en la noche"
 "23-4-2018 2:47:31 -> chatbot : tiene alguna otra consulta?"
 "23-4-2018 2:47:31 -> usuario : no"
 "23-4-2018 2:47:31 -> chatbot : Me alegra haber resuelto tus dudas :D"
 "23-4-2018 2:47:31 -> chatbot : "
 "23-4-2018 2:47:31 -> chatbot : Gracias por consultar y espero que te mejores :), hasta pronto"
 1)
```

Figura 5: Demostracion Función Test con user2

Debido a inconvenientes con el tiempo durante el desarrollo del laboratorio, no se logró la implementación del concepto *lazy evaluation* en las respuestas del chatbot ni funciones extras tales como *analyse, chat->xml,* entre otros.

CAPÍTULO 6. CONCLUSIONES

El paradigma funcional tiene una poderosa ventaja en comparación con los otros paradigmas respecto a su simplicidad de manejar las funciones con recursión y anidación de estas, logrando así codigos de una extensión menor que la que se presentaría en otros lenguajes.

Respecto a la resolución del laboratorio, se torno interesante el hecho de tener que administrar cada una de las acciones anidando funciones entre sí, ya que no se está acostumbrado a realizar ese ejercicio, debido a que en la mayoría de los casos para acceder a ciertos datos que puntualmente en este laboratorio se consideran como “memoria”, se logra utilizando un elemento externo al código como un archivo o simplemente almacenarlos en alguna variable, recursos los cuales no pertenecen del todo al paradigma. Este laboratorio fomentó el análisis de cuales funciones o tipos de problemas se plantean de mejor manera con este paradigma y cuáles se vuelven más tediosos de implementar gracias a éste.

Finalmente, como se ha podido realizar de manera satisfactoria la base del chatbot y lograr su funcionamiento de manera correcta, es posible concluir que se logró cumplir en gran parte el objetivo principal del laboratorio.