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

LABORATORIO 1

PARADIGMAS DE PROGRAMACIÓN

Gabriel Gaete L.

Profesores: Roberto González
Daniel Gacitúa
Víctor Flores

Fecha de Entrega: 23 de Abril del 2018

Santiago de Chile

1 - 2018

TABLA DE CONTENIDO

CAPÍTULO 1. INTRODUCCIÓN *4*

CAPÍTULO 2. MARCO TEÓRICO *5*

2.1 PARADIGMA FUNCIONAL **5**

2.2 TIPO DE DATO ABSTRACTO (TDA) **5**

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA *6*

3.1 ANÁLISIS **6**

CAPÍTULO 4. DESCRIPCIÓN DE LA SOLUCIÓN *7*

CAPÍTULO 5. RESULTADOS *9*

CAPÍTULO 6. CONCLUSIONES *11*

CAPÍTULO 7. REFERENCIAS *12*

TABLA DE FIGURAS

Figura 1 Representación de un mensaje 7
Figura 2 Ejemplo de conversación simulada a través de la función test..... 8
Figura 3 Estructura chatbot1 9
Figura 4 Ejemplo de composición de funciones para generar una conversación..... 10

CAPÍTULO 1. INTRODUCCIÓN

Un *chatbot* o un bot conversacional es *"un robot capaz de hablar e interactuar imitando el comportamiento humano, ya sea oral o por escrito, respondiendo a las preguntas y reclamos de los usuarios."* (Herrero, s.f.) En otras palabras, es un programa que permite simular una conversación con una persona, entregando respuestas automáticas a entradas hechas por un usuario. Esta conversación, habitualmente se establece a través de texto, sin embargo, hay modelos que disponen de una interfaz multimedia. Tambíen se han desarrollado chatbots que utilizan conversores de texto a sonido, dotando de mayor realismo a la interacción con el usuario.

El presente informe tiene por objetivo principal ser una referencia a una línea de pensamiento y a un contexto de desarrollo del código fuente que lo acompaña para la presentación del primer laboratorio del curso *"Paradimas de programación"*. En esta oportunidad, se hará uso del paradigma funcional bajo el lenguaje de programación *Scheme*.

El paradigma funcional está basado principalmente en el concepto (matemático) de función. Una de las características principales de este paradigma es la ausencia de lo que se conoce como *'variables'*, permitiendo que no se produzcan *'efectos colaterales',* en contraste con otros paradigmas en los que sí se producen cambios de estado. Además de esto, otra de las características principales es que las funciones son tratadas como ciudadanos de primera clase, es decir, pueden pasarse como parámetros a otras funciones, pueden ser devueltas por otras funciones, pueden combinarse (composición) para formar otras nuevas, tienen un tipo de dato asociado, etc.

El problema a resolver en este laboratorio es el desarrollo de un *chatbot*, con el cual se deberá mantener una conversación básica, protocolar, en la que exista un flujo conversacional coherente. El contexto para este chatbot será una venta de pasajes a capitales regionales dentro de Chile.

La solución implementada trabaja en base a un conjunto de estructuras que permiten simular lo que es el *chatbot* haciendo el uso de listas y de sus respectivas propiedades. Junto con esto, se ha implementado un Tipo de Dato Abstracto (TDA) para los mensajes que se intercambien entre usuario/chatbot; esto quiere decir, que se han implementado funciones de pertenencia, selectoras y modificadoras para trabajar sobre estas estructuras.

CAPÍTULO 2. MARCO TEÓRICO

2.1 PARADIGMA FUNCIONAL

Como se dijo anteriormente, el paradigma funcional tiene sus bases en el concepto matemático de función. Dado que el presente proyecto debe ser abordado bajo este modelo, se deben tener claras las herramientas proporcionadas, como también el lenguaje utilizado *(Scheme)*. El paradigma funcional brinda la posibilidad de utilizar funciones anónimas, las cuales se caracterizan por no estár ligadas a un identificador; son *"creadas"* en tiempo de ejecución. Otra de las características que brinda este paradigma, es el concepto de evaluación perezosa, el cual retrasa la carga de un recurso hasta el momento mismo de su utilización. Por último, otro concepto "fuerte" que debe conocerse para la implementación de la solución a este laboratorio, es el concepto de función como ciudadano de primera clase, el cual permite la existencia de funciones de orden superior. Esto brinda la posibilidad de que una función "retorne" a otra, o la reciba como uno de sus argumentos. Estas herramientas, propias del paradigma funcional, son aplicadas en este laboratorio. Se omite la explicación de las restantes, como el concepto de encapsulación, recursión, etc.

2.2 TIPO DE DATO ABSTRACTO (TDA)

Corresponde a un método de abstracción para tipos de datos de manera tal que se puedan definir operaciones sobre éste sin preocuparse de la implementación. Un TDA está compuesto por funciones constructoras encargadas de definir la estructura del nuevo tipo de dato basado en los tipos de datos atómicos (nativos de cada lenguaje), funciones selectoras que permiten obtener información desde una estructura o "variable" del tipo de dato entre otras más, funciones modificadoras que, como su nombre indica, permiten modificar el valor de alguno de los identificadores del TDA. Cabe destacar, que como en el caso de Scheme, no existen variables como tal, las funciones modificadoras consisten en crear copias del elemento original con la modificación pertinente.

CAPÍTULO 3. DESCRIPCIÓN DEL PROBLEMA

Se solicita crear un algoritmo en el lenguaje de programación *Scheme* que simule un bot conversacional (chatbot). Para este laboratorio, el tema del chatbot será una venta de pasajes a capitales regionales de Chile. Este debe funcionar en base a una estructura chatbot, un *log* o historial de conversaciones, y una semilla (seed), para utilizar funciones pseudo-aleatorias. El algoritmo debe ser capaz de realizar las siguientes funcionalidades:

1. **Implementación de un TDA:** Haciendo uso de la estructura TDA de 6 capas, implementar abstracciones adecuadas al problema.

2. **beginDialog:** Función que retorna un log modificado, agregando una etiqueta que indica el inicio de una convesación con un chatbot, un *ID* o identificador del chatbot, y un primer mensaje de bienvenida al usuario.

3. **sendMessage:** Función que retorna un log modificado, representando el envío de mensajes por parte del usuario hacia el chatbot y su respectiva respuesta.

4. **endDialog**: Función que retorna un log modificado, agregando una despedida por parte del chatbot, además de una etiqueta que indica el fin de una conversación.

5. **rate:** Función que permite evaluar el desempeño del chatbot en la conversación, a partir de dos calificaciones, una entregada por el usuario, y otra calificación dada por el mismo programa. Esta función retorna un chatbot con un registro actualizado de sus calificaciones.

6. **test:** Función que permite simular una conversación entre un usuario y un chatbot. Retorna un log actualizado con el resultado de la conversación simulada.

Estas funciones tienen un formato definido para su implementación y desarrollo, trabajan con estructuras específicas, por lo mismo, se han definido funciones auxiliares para trabajar con los formatos requeridos.

3.1 ANÁLISIS

El principal problema que se presenta en este laboratorio es la de definir funciones y estructuras con las que se pueda trabajar, dado que no se puede modificar la firma de las funciones requeridas, la implementación de cada uno de estos componentes es escencial para garantizar un correcto funcionamiento del chatbot.

CAPÍTULO 4. DESCRIPCIÓN DE LA SOLUCIÓN

Como se mencionó en el capítulo anterior, el problema principal recae en la representación de las estructuras a utilizar, tanto del *log* como de los mensajes en sí, por la complejidad que le agrega o le quita a las funciones con las que trabaja.

Así, definiendo una implementación en *Scheme* para los mensajes, sería una lista de tres elementos, donde el primero será una estructura de tipo fecha, el segundo elemento será un string que representará al emisor del mensaje, mientras que el último elemento de la lista corresponde al contenido del mensaje. A continuación, se presenta un ejemplo de la representación en *Scheme* de un mensaje.

```
> (message (current-date) "Usuario" "¡Hola Mundo!")
(#(struct:date*
   49
   59
   21
   19
   4
   2018
   4
   108
   #t
   -10800
   258412122
   "-03")
 "Usuario"
 "¡Hola Mundo!")
```

Figura 1 Representación de un mensaje

De esta forma, al momento de crear una representación para el log, se puede acceder a toda la información de los mensajes a través de los selectores para este TDA, logrando que las funciones que deben retornar un log actualizado (**beginDialog, sendMessage, endDialog**) tengan una manera directa de trabajar con este. Por otro lado, para lograr interpretar los mensajes que entrega el usuario, con el fin de entregar una respuesta que se acomode al flujo de la conversación, se trata al contenido del mensaje del usuario como una lista de strings; luego estos son intersectados con una series de palabras para interpretar el mensaje. Una vez interpretado el mensaje, se determina la respuesta que debe entregar el chatbot.

Por último, para la función **rate**, específicamente para la autoevaluación, se utiliza un criterio basado en el largo de la última conversación (sólo se puede evaluar la última conversación finalizada). El criterio utilizado para evaluar, se basa en que si la conversación es de un largo mínimo para vender un pasaje, la evaluación es la máxima. Por otro lado, si la conversación es extremadamente larga para vender un pasaje, la calificación es la mínima. Por último, si la conversación es lo suficientemente corta como para lograr vender un pasaje, la evaluación no se puede determinar.

Teniendo todas estas funciones, la función **test** ya puede ser implementada. La siguiente figura, muestra el log entregado por la función test.

```
> (test user1 chatbot1 '() 60)
("[22-4-2018] 19:43:3 ID:0 BeginDialog"
 "[22-4-2018] 19:43:3 Bot: Buenas tardes, mi nombre es Bot, y estoy aquí para
ayudarte con tu próximo viaje. ¿Cuál es tu nombre?"
 "[22-4-2018] 19:43:3 Usuario: Gabriel"
 "[22-4-2018] 19:43:3 Bot: Gabriel cuéntame, ¿a dónde quieres viajar? Recuerda
que por el momento sólo ofrecemos viajes a capitales regionales del país."
 "[22-4-2018] 19:43:3 Usuario: ¿Cuándo habrán pasajes para Brasil?"
 "[22-4-2018] 19:43:3 Bot: Voy a averigüarlo. Te respondo en un momento."
 "[22-4-2018] 19:43:3 Usuario: Me gustaría viajar al sur"
 "[22-4-2018] 19:43:3 Bot: Perdón, pero no he entendido lo que me has dicho...
¿podrías ser un poco más claro?"
 "[22-4-2018] 19:43:3 Usuario: Respóndeme lo que te pregunté"
 "[22-4-2018] 19:43:3 Bot: A la pregunta que me habías hecho, el servicio se
encontrará disponible desde el próximo verano."
 "[22-4-2018] 19:43:3 Usuario: Quiero viajar a Coyhaique"
 "[22-4-2018] 19:43:3 Bot: ¡Excelente elección! Coyhaique es ideal en esta época
del año, no te arrepentirás. Viajar hacia allá cuesta $33.000 pesos ¿Desea
confirmar esa ciudad?"
 "[22-4-2018] 19:43:3 Usuario: Sí"
 "[22-4-2018] 19:43:3 Bot: Bien, ahora para confirmar la cantidad y la fecha de
los pasajes, debe ingresar a nuestro sitio web"
 "[22-4-2018] 19:43:3 Bot: Hasta la próxima, espero haberte ayudado."
 "[22-4-2018] 19:43:3 ID:0 EndDialog")
```

Figura 2 Ejemplo de conversación simulada a través de la función test

En la figura anterior, se tiene un ejemplo de cómo es la representación del log en una conversación con un chatbot. Por otro lado, se puede ver la aplicación del concepto *evaluación perezosa*, al momento de preguntarle por un viaje hacia Brasil.

CAPÍTULO 5. RESULTADOS

A pesar de las llamadas recursivas, y tomando como referencia el IDE DrRacket, el uso de memoria se mantiene en un nivel bajo. Por otro lado, el tiempo de ejecución de las funciones requeridas es aceptable, donde no existe un tiempo de respuesta notable para el usuario, siendo la impresión por pantalla del log lo que más tiempo toma.

Se ha hecho uso de recursión, tanto NATURAL como de COLA, siendo esta última la más utilizada a lo largo del programa debido al mejor rendimiento de esta. La recursividad NATURAL sólo fue utilizada por cumplir requerimientos exigidos.

Para utilizar cualquier función del programa, se requiere crear una estructura chatbot, la cual, para efectos de evitar su creación en las funciones, se ha asociado una al identificador *chatbot1*.

```
(define chatbot1 (make-chatbot
  '("Buenos días, mi nombre es Bot y estoy aquí para ayudarlo a seleccionar un destino. ¿Me podría decir su nombre?"
    "Hola, mi nombre es Bot, espero ser de ayuda para buscar un viaje que le acomode. ¿Cuál es su nombre?")
  '("Buenas tardes, mi nombre es Bot, y si quieres viajar, conmigo debes hablar. ¿Cómo debo llamarte?"
    "Buenas tardes, mi nombre es Bot, y estoy aquí para ayudarte con tu próximo viaje. ¿Cuál es tu nombre?")
  '("Buenas noches, mi nombre es Bot, y estoy aquí para ayudarte a elegir tu próximo destino. ¿Cómo debería llamarte?"
    "Buenas noches, mi nombre es Bot, y estoy aquí para que conversemos sobre tu viaje, pero antes, ¿Cuál es tu nombre?")
  '(" cuéntame, ¿a dónde quieres viajar? Recuerda que por el momento sólo ofrecemos viajes a capitales regionales del país."
    " ¿a qué capital regional deseas viajar? Puedes hacerlo a cualquier región de Chile. Yo te recomiendo el norte."
    " y bueno, ¿a qué capital regional te gustaría ir? El sur es hermoso en toda época del año.")
  '("Disculpa, no he logrado entenderte del todo... ¿podrías ser un poco más claro?"
    "Perdón, pero no he entendido lo que me has dicho... ¿podrías ser un poco más claro?")
  '("¡Es la mejor elección que pudiste elegir! "
    "¡Excelente elección! ")
  '(" es un lugar precioso! Los pasajes hacia allá cuestan "
    " es ideal en esta época del año, no te arrepentirás. Viajar hacia allá cuesta ")
  (list (list "Arica" "$32.200 pesos") (list "Iquique" "$30.100 pesos") (list "Antofagasta" "$21.600 pesos") (list "Copiapó" "$15.000 pesos")
        (list "La Serena" "$9.100 pesos") (list "Valparaíso" "$6.500 pesos") (list "Rancagua" "$3.000 pesos") (list "Talca" "$6.500 pesos")
        (list "Concepción" "$13.900 pesos") (list "Temuco" "$14.900 pesos") (list "Puerto Montt" "$19.900 pesos")
        (list "Coyhaique" "$33.000 pesos") (list "Punta Arenas" "3000") (list "Valdivia" "$17.900 pesos"))
  '("¡Perfecto! Ahora, para confirmar pasajes, debe ingresar a nuestro sitio web."
    "Bien, ahora para confirmar la cantidad y la fecha de los pasajes, debe ingresar a nuestro sitio web")
  '("¿A qué ciudad entonces te gustaría ir?"
    "No hay problema, puedes elegir un nuevo destino")
  '("Consultaré con mis superiores, te responderé en unos minutos."
    "Voy a averigüarlo. Te respondo en un momento.")
  '("A la pregunta que me habías hecho, el servicio comenzará en un tiempo estimado de 6 meses."
    "A la pregunta que me habías hecho, el servicio se encontrará disponible desde el próximo verano.")
  '("Hasta luego, espero haber sido de ayuda en esta oportunidad."
    "Hasta la próxima, espero haberte ayudado.")
  (list (list 0 0))
  )
  )
```

Figura 3 Estructura chatbot1

Por otro lado, también se necesita un log, el cual sólo es una lista inicialmente vacía (o el retorno de las funciones **beginDialog**, **sendMessage** y **endDialog**). La siguiente figura muestra un ejemplo de llamado a las funciones.

```
> (endDialog chatbot1 (sendMessage "Sí" chatbot1 (sendMessage "me gustaría ir a Valparaíso" chatbot1
(sendMessage "Respóndeme lo que te pregunté" chatbot1 (sendMessage "¿Cuándo habrán pasajes a Quilicura?"
chatbot1 (sendMessage "Gabriel" chatbot1 (beginDialog chatbot1 '() 0) 0) 0) 0) 0) 0) 0)
("[20-4-2018] 16:7:2 ID:0 BeginDialog"
 "[20-4-2018] 16:7:2 Bot: Buenas tardes, mi nombre es Bot, y estoy aquí para ayudarte con tu próximo
viaje. ¿Cuál es tu nombre?"
 "[20-4-2018] 16:7:2 Usuario: Gabriel"
 "[20-4-2018] 16:7:2 Bot: Gabriel cuéntame, ¿a dónde quieres viajar? Recuerda que por el momento sólo
ofrecemos viajes a capitales regionales del país."
 "[20-4-2018] 16:7:2 Usuario: ¿Cuándo habrán pasajes a Quilicura?"
 "[20-4-2018] 16:7:2 Bot: Voy a averigüarlo. Te respondo en un momento."
 "[20-4-2018] 16:7:2 Usuario: Respóndeme lo que te pregunté"
 "[20-4-2018] 16:7:2 Bot: A la pregunta que me habías hecho, el servicio se encontrará disponible desde
el próximo verano."
 "[20-4-2018] 16:7:2 Usuario: me gustaría ir a Valparaíso"
 "[20-4-2018] 16:7:2 Bot: ¡Excelente elección! Valparaíso es ideal en esta época del año, no te
arrepentirás. Viajar hacia allá cuesta $6.500 pesos ¿Desea confirmar esa ciudad?"
 "[20-4-2018] 16:7:2 Usuario: Sí"
 "[20-4-2018] 16:7:2 Bot: Bien, ahora para confirmar la cantidad y la fecha de los pasajes, debe ingresar
a nuestro sitio web"
 "[20-4-2018] 16:7:2 Bot: Hasta la próxima, espero haberte ayudado."
 "[20-4-2018] 16:7:2 ID:0 EndDialog")
```

Figura 4 Ejemplo de composición de funciones para generar una conversación

Tras la finalización de cada función existe memoria utilizada por DrRacket, por lo que se recomienda liberar esta memoria tras cada uso.

Por último, tras evaluar el rendimiento y el funcionamiento del programa, se puede afirmar que Scheme no es un buen lenguaje para la realización de este proyecto, dado que realizar operaciones sobre el log no son sencillas.

CAPÍTULO 6. CONCLUSIONES

El paradigma funcional permite expresar la solución al laboratorio mediante un lenguaje expresivo y matemáticamente elegante, además de evitar el concepto de cambios de estado, por lo que la modificiación de valores no es permitida. Y esto último es una de las armas de doble filo del paradigma funcional. Por un lado, la resolución del laboratorio se hace complicada sin el uso de variables, dado que no se puede “actualizar” el historial de mensajes en sí. Sin embargo, el paradigma funcional sí entrega las herramientas para lograr esto, como puede ser la composición de funciones (estrategia utilizada, ver Figura 4). Esto permite predecir el comportamiento del programa de manera más fácil.

Por otro lado, por la misma razón no existe el concepto de “iteración” como tal. Esto permite que en particular, el paradigma funcional permita comprender de mejor manera el concepto de recursividad. A pesar de esto, para problemas en los que se deban procesar mayores cantidades de información, este paradigma podría no ser el más eficiente, dado que se ocupan grandes cantidades de memoria, sobretodo en recursiones de tipo NATURAL.

Concluyendo, puesto que se ha logrado desarrollar un bot conversacional como se ha requerido (usando el paradigma funcional con el lenguaje de programación Scheme), es posible concluir que se ha logrado el objetivo principal del laboratorio.

CAPÍTULO 7. REFERENCIAS

Herrero, C. (s.f.). No son mis cookies. Recuperado el 21 de Abril de 2018, de No son mis cookies: http://nosinmiscookies.com/que-es-un-chatbot/