Instituto Tecnológico de Costa Rica

Area académica de Ingeniería en Computadores

II Semestre 2019

Tarea 2. Paradigma lógico. Nombre código: CallCenterLog

Call Center LOG

Equipo de trabajo

- **Jesús Sandoval Morales** [2018203706] - *Desarrollador* - shuzz22260427
- **Alejandro Ibarra Campos** [2018089951] - *Desarrollador* - AlejandroIbarraC
- **Esteban Alvarado Vargas** [2018109336] - *Desarrollador* - estalvgs1999

Responsable Técnico:

- **Marco Rivera Meneses** - *Profesor del Curso*

Curso:

Lenguajes, Compiladores e Intérpretes - CE 3104

Lenguaje utilizado:

Prolog

a. Descripción del Problema

Call Center Log - Introducción

Call Center Log corresponde a la *segunda tarea* del curso de *Lenguajes, Compiladores e Intérpretes (CE3104)*, el cual consiste en la implementación de un Sistema Experto (SE) para **solución de problemas** comunes de un CallCenter de IT. El presente es un sistema de chat que permite al usuario realizar **consultas al Bot** y recibir la mejor solución. Esta aplicación está desarrollada bajo el **paradigma de programación lógico** utilizando el lenguaje **Prolog**, sus funciones deben ser las siguientes:

- **Consultas**

  El usuario debe ser capaz de preguntarle al programa acerca de soluciones sobre problemas relacionados a los dispositivos permitidos.

- **Comunicación:**

  El programa (bot) debe ser capaz de mantener una “conversación” con el usuario de forma tal que emule el servicio dado en un centro de servicio “call center”. Dicha conversación deberá ser estrictamente orientada a la resolución de problemas con los dispositivos.

- **Efectividad:**

El programa debe ser capaz de resolver de forma efectiva el problema planteado por el usuario, dicha condición aplica para problemas que se presenten respetando las reglas establecidas previamente.

Objetivos

Objetivo General

Reafirmar los conocimientos obtenidos en clases acerca del Paradigma lógico por medio del desarrollo de una aplicación.

Objetivos Específicos

- Plantear una solución a un problema usando Prolog
- Aplicar los conceptos de programación lógica.
- Aplicar el concepto de reglas y hechos en el lenguaje lógico y aplicarlo en un programa desarrollado por el equipo de trabajo

b. Estructuras de datos desarrolladas y usadas

En este apartado se define de manera precisa y concisa las estructuradas que fueron utilizadas para el desarrollo y buen funcionamiento del proyecto.

Listas:

La principal estructura usada para el desarrollo del programa requerido fueron las listas. Dicha estructura se usó con el fin de almacenar diversos datos y emular de una forma simple una base de datos en la cual Prolog pudiese buscar a la hora de correr el programa y que el usuario le realizase una pregunta al Bot

La razón principal para el uso de esta estructura es debido a que gracias a esto se puede simular una jerarquía como si fuese un árbol; y debido al método de búsqueda de predicados que maneja Prolog (siendo este back tracking). Gracias a esto, las listas resultan la forma mas óptima de organizar los datos.

Hechos y reglas implementados.

Al ser Prolog un lenguaje de tipo lógico para su funcionamiento se deben definir reglas y hechos, los cuales serán los principales motores sobre los que girará el lenguaje y toda su estructura.

- **HECHOS:**

A. *Nombres:* El archivo “nombres.pl” contiene una amplia lista de nombres que permiten la autenticación del usuario por la aplicación. Cada nombre se define con la etiqueta correspondiente. Ej: “nombre(esteban)”.

B. *Respuestas:* Las respuestas se almacenan en el archivo “db.pl”, como una lista de listas, donde cada elemento es un string con posibles respuestas. La relación/hecho se define como “respuestas(tipo de respuesta, lista de respuestas)”. Se tienen respuestas para los casos en los que la entrada del usuario sea de despedida, saludo, obtener dispositivo, obtener nombre, gracias, fin de conversación, agradecimiento, estado personal del programa, preguntas aleatorias.

Además, aquí se almacenan las oraciones respuesta, determinadas en la tabla “CSR-Prolog.pdf” adjuntado en la carpeta de documentación, así como las causas, preguntas y referencias especificadas en la misma tabla. Esta información es utilizada por las reglas de “chat.pl” para devolver la información correcta al usuario.

Finalmente, en este mismo archivo se declaran las entradas patrón de entradas “saludos” como “hola, saludos”, palabras de agradecimiento, y los dispositivos válidos para los cuales se pueden brindar soluciones a problemas.

C. *Elementos léxicos:* En la definición de las gramáticas libres de contexto, en el archivo “idioma.pl”, se hacen las relaciones entre los diferentes elementos del español. Se incluyen adjetivos, adverbios, determinantes, preguntas, pronombres, sustantivos, y verbos.

La estructura de dichos elementos en este programa se tomaron directamente del capítulo 6 del libro “El lenguaje de programación Prolog” de M. Teresa Escrig, por lo que se insta a investigar un poco en esta fuente externa, si se desea saber más acerca de esta parte.

- **REGLAS:**

A. *Generar respuesta:* El archivo de control de para todo el programa es chat.pl. Aquí se encuentra la regla generar_respuesta, definida varias veces (o lógico), de acuerdo a lo que se pretenda buscar. Las respuestas se generan para los siguientes casos.

- Si en la misma frase el usuario ya indicó el dispositivo.
- Si no se introduce un dispositivo, generar una pregunta para el mismo.
- Si en la misma frase, el usuario ya indicó el dispositov y el problema.
- Si en la misma frase, el usuario ya indicó el dispositivo y el problema.
- Si el usuario brinda de el dispositivo y pide soluciones a su problema.
- Si el usuario indica el problema, el dispositivo, y dice la causa, retorna una solución.
- Si se introduce la frase de despedida “adios”.
- Si se introduce un saludo, agradecimiento, pregunta por el nombre, pregunta sobre lo que estudia, pregunta por el estado del programa.
- Una respuesta aleatoria, si no se comprende la pregunta.
- Si se introduce una incoherencia.

B. *Obtener información:* Los datos introducidos se almacenan en tres variables dinámicas, denominadas “nombre_usuario”, para guardar el nombre del usuario, “dispositivo” para el elemento con el que se tenga un problema, y “solicitud”, una variable temporal que referencia al problema y referencia correspondiente.

C. *Patrones:* El sistema experto utiliza, además de gramáticas libres de contexto de tipo BNF, patrones para detectar palabras y frases clave en las entradas del usuario. Estos siguen una oración como una lista donde cada elemento es una palabra, separada por comas. (Estándar de representación de una oración en Prolog, según el libro “El lenguaje de Programación Prolog” de M. Teresa Escrig. Se recomienda la lectura de dicho material si se desea una mayor comprensión). Los patrones contemplados por CallCenterLog son:

- Pregunta por estudios
- Pregunta por nombre
- Pregunta por estado emocional
- Realización de una consulta
- Solicitud de una referencia
- Indicar un problema
- Ofrecer una causa para un problema

D. *Gramáticas libres de contexto BNF:* Se implementaron gramáticas libres de contexto, basadas en el libro “El lenguaje de programación Prolog” de M. Teresa Escrig, y expandidas para contemplar varios casos de entradas de usuario. Estas son utilizadas para verificar si las frases son válidas oraciones en el español.

El capítulo 6 del libro mencionado anteriormente explica de una excelente manera el funcionamiento de este tema, por lo que se recomienda leer el mismo si se desea tener una comprensión de su funcionamiento. Este programa utiliza exactamente la estructura de oraciones ahí definida.

Descripción de los algoritmos utilizados.

Algoritmos Desarrollados

Para el proyecto se desarrollaron algoritmos de tipo lógicos, todos ellos con el fin de poder manejar el funcionamiento y la lógica de comunicación para el Bot.

La explicación de este apartado se adjunta en el documento llamado *“Algoritmos de Interpretación de Consultas de Lenguaje Natural en el Bot CallCenterLog.pdf”*, con el objetivo de simplificar la explicación de una manera más gráfica.

Problemas Conocidos

En esta sección se detallan los ***problemas que se presentan en el desarrollo y no han sido solucionados.*** Sin embargo, actualmente y hasta el instante de la redacción de este documento no ha existido permanencia de ninguno de los problemas encontrados durante el transcurso del presente proyecto, esto; sin embargo, no implica que no se hayan encontrado problemas de manera temporal, dichos problemas serán discutidos en el apartado "Problemas encontrados".

Problemas encontrados

Problemas a la hora de escribir la letra ñ

Durante el desarrollo del algoritmo de read in se encontraron graves problemas a la hora de escribir palabras con dicha letra.

- **Motivos**

  Posteriormente se descubrió que la razón por lo que sucedía esto era debido a la estructura funcional del sistema ASCII en el cual se basa el algoritmo.

- **Solución**

  El problema fue solucionado usando el comando “get-code” en consola y obteniendo el valor que daba al digitar la ñ.

Referencia

Warren, D. (1980). *Logic programming and compiler writing. Software: Practice And Experience*, *10*(2), 97-125. doi: 10.1002/spe.4380100203

Problemas con la definición de la lógica verbal.

Durante el desarrollo de la lógica libre de contexto se encontraron varios problemas a la hora de definir las oraciones y su estructura

- **Motivos**

  Debido a la complejidad del idioma español el definir todas sus reglas gramaticales y sintácticas resulta sumamente complejo.

- **Solución**

  El método que se usó para solucionarlo fue restringir las reglas y los hechos para que se acoplaran a la conversación esperada por parte del usuario

Referencias

Escrig, M., & Pacheco, J. *El Lenguaje de Programación Prolog* [Ebook]

Problemas con el algoritmo del Read In

El mayor problema en la parte lógica fue a la hora de crear el algoritmo que interpretara lo que el usuario escribiese y lo acoplara al resto del lógico gramatical para poder formar la respuesta correcta a la pregunta

- **Motivos**

  El principal motivo de esto fue la forma en la cual funciona el paradigma lógico, ósea su función de reglas y hechos, debido a esto se debe ser sumamente especifico a la hora de definir casos.

- **Solución**

  La solución óptima encontrada fue el desarrollo de un algoritmo de reconocimiento que tuviese como base de función el sistema ASCII.

Referencias

Warren, D., & Pereira, F. (1980). *An Efficient Easily Adaptable System for Interpreting Natural Language Queries* (1).

Organización del Proyecto

Plan de actividades

El proyecto se ha desarrollado bajo el planeamiento mostrado en el archivo “ITProLogBot-project-managment.pdf”, adjuntado con esta documentación.

Conclusiones y recomendaciones

Conclusiones

A lo largo de todo el proyecto se obtuvieron múltiples conocimientos los cuales serán, sin duda alguna, provechosos en futuros proyectos, en este apartado se pretende resumirlos de manera concisa y directa.

1. La importancia de trabajar con hechos y reglas bien definidos y precisos.
2. La importancia del paradigma lógico a la hora se simular pseudo tipos de inteligencia artificial o programas de “Machine Learning”.
3. Como grupo se pudo concluir la importancia que tiene los sistemas expertos a la hora de realizar o desarrollar un software que maneje información de un área específica.
4. La importancia del “back tracking” a la hora de realizar búsquedas dentro de un sistema experto con el fin de obtener el resultado óptimo.
5. El paradigma lógico fomenta el trabajo predefinido y la resolución de problemas mediante el uso de incógnitas, predicados y preguntas, así mismo la importancia de esquematizar dichos procesos como arboles ordenados.

Recomendaciones

1. Tratar de comprender el problema planteado antes de empezar a programar, esto con el fin de poder definir los hechos y las reglas de la forma más eficiente.
2. Esquematizar lo mejor posible los problemas y los datos que se van a manejar a la hora de desarrollar el programa.
3. Entender el modo de búsqueda y funcionamiento de back tracking aplicado en Prolog a la hora de hacer una búsqueda.

Bibliografía

Escrig, M., & Pacheco, J. *El Lenguaje de Programación Prolog* [Ebook]

Warren, D. (1980). *Logic programming and compiler writing. Software: Practice And Experience*, *10*(2), 97-125. doi: 10.1002/spe.4380100203

Warren, D., & Pereira, F. (1980). *An Efficient Easily Adaptable System for Interpreting Natural Language Queries* (1).

Bitácora

24 De agosto de 2019.

El día de hoy se empieza la planificación del proyecto, se realiza una reunión presencial a las 12 horas con 11 minutos, justo después de almorzar, la reunión se realizo en un Subway cercano y tuvo como principal objetivo charlar acerca del problema planteado por medio de la tarea y como se podría dividir y realizar de la mejor manera.

Se llega al acuerdo de reunirse al día siguiente por medio de la aplicación “Discord” con el fin de empezar el plan de desarrollo de la primera versión y la división de tareas pertinentes bajo las cuales se regirá el proyecto

25 de agosto del 2019

El día de hoy, usando como medio de comunicación el grupo “Por la teja en lenguajes” se programa la reunión por medio de “Discord” aplicación anteriormente mencionada. La reunión se programa para la noche, específicamente a las 19 horas con 30 minutos, esto con el fin de darle a todos los miembros un horario de disponibilidad abierto.

Durante la llamada se desarrolla el documento de planeamiento (el cual se adjuntará como parte de la documentación pertinente del proyecto). En dicho documento se dividieron las tareas de la forma mas equitativa posible, tomando en cuando las fortalezas de cada miembro del equipo, así como su mejor área de desempeño.

Después de esto se termina de retocar el tema de la organización y se termina la llamada exactamente a las 23 horas con 09 minutos.

26 de agosto del 2019

El día de hoy se empezó el desarrollo del proyecto tomando como base la división de trabajo asignada el día anterior.

Los resultados de este día fueron:

Alejandro: Empieza a montar los hechos y las reglas sobre los cuales se va a basar el funcionamiento lógico, así mismo diagrama el funcionamiento de forma parcial para cada dispositivo por el cual el usuario pueda preguntar

Esteban: Pide hacer la lógica del algoritmo de read in, por medio del cual el usuario podrá ingresar sus preguntas y problemas, debido a esto se empieza a leer papers e investigar por la solución óptima.

Jesús: Se le asigno la gramática libre de contexto, debido a eso empieza a investigar sobre la mejor forma de diagramar la gramática del español dentro de las definiciones de Prolog.

28 de agosto de 2019

El día de hoy el integrante del grupo, Esteban, tuvo una enorme cantidad de trabajo proveniente del curso de electrónica “Circuitos en Corriente Continua”, debido a esto se decidió empezar a trabajar por medio de Discord hasta horas de la tarde.

La llamada se realizó a la hora propuesta, la cual era a las 18 horas y los resultados obtenidos el día de hoy fueron:

Alejandro: Discute por medio del grupo posibles formas de definir las reglas y los hechos, es asistido por Esteban en varios casos por medio de la llamada.

Esteban: Informa al resto del grupo que se decidió por una implementación de código lógico basado en el sistema ASCII, esto debido a que concluyó que era el método mas óptimo para interpretar lo que el usuario quisiese escribir, además de esto asiste a Alejandro con su parte de hechos y reglas.

Jesús: Empieza a implementar la gramática libre de contexto con un sistema similar al planteado en el libro, siendo así la definición primaria de los sintagmas y datos necesarios.

Se termina la llamada a las 00 horas con 4 minutos.

1 de septiembre de 2019

Para el día de hoy se habían logrado inmensos avances por parte de todos los miembros del grupo, además de esto Alejandro cambio de interprete para el lenguaje a Xcode, el cual es nativo de macOS y le ayudó a optimizar la velocidad con la que programaba:

Los aportes de hoy fueron:

Esteban: Termina el algoritmo de Read in usando el sistema ASCII. Durante su desarrollo se encontró el problema de como añadir y detectar la letra ñ, problema el cual solucionó de manera ingeniosa obteniendo el código de la letra por medio de su terminal y adaptando el código con base a ello.

Alejandro: Termina la definición de todos los hechos y reglas, además de esto crea patrones específicos para todos los dispositivos con ayuda de todo el equipo

Jesús: Termina la gramática y la base de datos en la cual se va a basar, además de esto se termina la estructura de las oraciones tomando en cuenta todos los posibles casos a desarrollarse.

02 de agosto de 2019

El día de hoy fue el examen de Ecuaciones Diferenciales, el cual debían rendir todos los miembros del equipo, debido a esto, el ánimo de trabajo no estaba en su epitome. Aun con esto se terminaron de retocar los puntos finales al proyecto, dándolo básicamente por terminado.

Se empieza la documentación por parte de todos los miembros del equipo como un todo, así como el desarrollo del manual de usuario.

Los aportes de hoy fueron:

Jesús: Realiza la documentación inicial por medio de Word, retoca la gramática para agregar un caso el cual Esteban descubrió no se había tomado en cuenta.

Esteban: Empieza el manual de usuario, corrige a Jesus en la gramática lógica, apoya a Alejandro con el diagrama del proyecto.

Alejandro: Empieza el manual de usuario, realiza un gran diagrama para explicar el funcionamiento del proyecto junto a Esteban.

03 de agosto de 2019

Se agregan los toques finales al manual de usuario, se agregan las referencias de los papers consultados para el desarrollo de los algoritmos y se termina la documentación y la bitácora de forma colectiva.

04 de agosto de 2019

Se da la revisión final al proyecto y se formatea toda la documentación pertinente de forma colectiva.