

Paradigmas de la Programación

Informe Laboratorio nro. 2

Paradigma Lógico

*Profesor: Gonzalo Martínez.*

*Alumna: Karina Bustamante H.*

14 de noviembre de 2023

Tabla de contenido

[Introducción 2](#_Toc134312314)

[Descripción del paradigma 2](#_Toc134312315)

[Descripción del problema 4](#_Toc134312316)

[Análisis del problema 4](#_Toc134312317)

[Diseño de la solución 4](#_Toc134312318)

[Consideraciones de implementación 5](#_Toc134312319)

[Instrucciones de uso 6](#_Toc134312320)

[Resultados y evaluación 7](#_Toc134312321)

[Conclusiones 8](#_Toc134312322)

[Referencias 8](#_Toc134312323)

# Introducción

La asignatura Paradigmas de la Programación busca enseñar los diferentes paradigmas que existen frente a la forma de programar. El presente informe, correspondiente al laboratorio Nº 2 de la asignatura, en donde se solicita realizar una implementación basada en los principios del paradigma lógico, para resolver el problema que corresponde al siguiente enunciado:

***“Sistema para la creación, despliegue y administración de chatbots simplificado.***

Para el desarrollo de esta implementación usaremos el lenguaje de programación, Prolog, con su editor virtual, (https://swish.swi-prolog.org/) y a través de la creación de una base de conocimientos que crearemos con predicados, reglas y clausulas, se buscará dar solución a los requerimientos funcionales solicitados, de manera tal, que la solución propuesta permita al usuario aplicar sobre este sistema simplificado de Chatbots acciones como crear, vincular e interactuar, entre otras cosas.

# Descripción del paradigma

El paradigma lógico corresponde al paradigma de programación declarativo, puede usarse como la especificación de un problema en lugar de establecer los pasos necesarios para llegar a una solución. Se basa en la lógica booleana y no existen funciones ni retornos normales. En este paradigma, se describe el problema en términos de relaciones lógicas entre objetos y se utiliza la inferencia lógica para resolver el problema. Se declara una base de conocimiento que es un conjunto de hechos y reglas que describen las relaciones entre los objetos en un dominio de problema. El programador describe el objetivo o la pregunta que se quiere responder en términos de estas relaciones lógicas. Luego, el sistema de programación utiliza la lógica para deducir la respuesta al objetivo a partir de los hechos y reglas definidos en la base de conocimiento. Los hechos, son un tipo de cláusula que describe una relación entre uno o más término, estos hechos se asumen siempre verdaderos.

Los mecanismos básicos de este paradigma son:

* ***Unificación***: se refiere a la técnica de encontrar una asignación de valores a las variables en una expresión lógica que hace que la expresión sea verdadera. Si se encuentra una asignación de valores que haga que las expresiones sean iguales, la consulta se considera resuelta y se devuelve el resultado
* ***Backtraking automático***: Sucede cuando no se logra la unificación, el proceso da un paso atrás para probar otros caminos para lograr la unificación, si se encuentra una asignación de valores que haga que las expresiones sean iguales, la consulta se considera resuelta y se devuelve el resultado, True.
* ***Inferencia***: corresponde a la aplicación de reglas lógicas para derivar nuevas proposiciones a partir de proposiciones previas.

Si comparamos los paradigmas que ya conocemos, Funcional v/s Lógico, tenemos que el paradigma funcional se basa en la evaluación de expresiones matemáticas y la teoría de funciones puras y se debe lograr la inmutabilidad evitando así los cambios de estados, estos últimos, solo se consiguen generando nuevas estructuras de datos en lugar de modificar las existentes. Por otro lado, el paradigma de programación lógico que se basa en la lógica matemática y la teoría de conjuntos, el cual se compone de un conjunto de hechos y reglas lógicas los que se usan para responder consultas mediante la inferencia y lo que se conoce como proceso de unificación. A través de ello, se buscan todas las respuestas posibles mediante la lógica de inferencia utilizando una programación declarativa.

De lo anterior podemos decir que, la programación lógica se basa en la lógica y la inferencia, mientras que la programación funcional se basa en la evaluación de expresiones matemáticas y la composición de funciones.

# Descripción del problema

En esta simulación, nos enfrentamos al reto de desarrollar chatbots utilizando el lenguaje de programación Prolog, específicamente en la categoría ITR (Interacción a Través del Lenguaje Natural), lo que significa que estará diseñado para ofrecer respuestas estructuradas a través de acciones específicas. Las opciones de interacción pueden variar desde el uso de palabras clave con sinónimos hasta una alternativa más simplificada mediante números o letras. Esto permitirá a los usuarios seleccionar sus preguntas de una lista predefinida.

# Análisis del problema

El problema que se intenta resolver corresponde a la creación de un sistema de chatbot simplificado del tipo ITR, que ofrezca respuestas estructuradas mediante acciones específicas. El desafío radica en implementar reglas y lógica en Prolog que permitan a los chatbots interpretar consultas, responder preguntas y llevar a cabo interacciones básicas. Esta solución se limitará a una lista de preguntas que se podrá generar a través del mismo sistema.

# Diseño de la solución

Para abordar el diseño de la solución en Prolog, se debe identificar los objetivos del chatbot, que tipo de preguntas responderá y cuáles son las acciones específicas que debe realizar. Dentro de las acciones clave estará:

* ***Implementación de Reglas Lógicas***: para implementar la lógica del chatbot. Se definirán reglas en Prolog que representen el conocimiento y la lógica detrás de las interacciones de los chatbots. Estas reglas abordaran las preguntas y proporcionaran las respuestas adecuadas en función de la entrada del usuario.
* ***Manejo de Preguntas y Respuestas***: Se deberá desarrollar un mecanismo para que el chatbot entienda y procese las preguntas, creando estructuras de datos para mapear preguntas a respuestas o acciones específicas.
* ***Gestión palabras claves***: Deberá realizar el reconocimiento exacto de palabras claves o números de opciones para realizar operaciones.

En cuanto al registro de la hora y fecha de creación, se ha utilizado el Unix Timestamp, que cuenta los segundos que han transcurridos desde el 1 de enero de 1970 desde las 00.00 hrs. que corresponde al nacimiento de Unix, este será el formato que utilizaremos para representar la fecha (Timestamp) en la solución.

# Consideraciones de implementación

Para llevar a cabo la implementación de nuestro chatbot en Prolog, es esencial tener en cuenta algunos aspectos clave que aseguren su funcionalidad y eficacia. Aquí se detallan las consideraciones principales:

* ***Implementación de Reglas Lógicas*:** Utilizaremos reglas lógicas en Prolog para dar forma a la inteligencia del chatbot. Estas reglas representarán el conocimiento y la lógica subyacente en las interacciones del chatbot. Definiremos reglas que aborden las preguntas frecuentes, permitiendo al chatbot proporcionar respuestas adecuadas en función de la entrada del usuario.
* ***Manejo de Preguntas y Respuestas***: Desarrollaremos un mecanismo que permita al chatbot entender y procesar preguntas de manera efectiva. Esto implicará la creación de estructuras de datos que mapeen preguntas a respuestas o acciones específicas. El objetivo es garantizar una interpretación precisa de las consultas de los usuarios para ofrecer respuestas pertinentes.
* ***Gestión de Palabras Clave***: Implementaremos un sistema de reconocimiento exacto de palabras clave y números de opciones para realizar operaciones específicas. Este enfoque asegurará que el chatbot pueda identificar con precisión las consultas y realizar las acciones correspondientes de manera eficiente.
* ***Registro de Fecha y Hora***: Para el registro de la hora y fecha de creación, emplearemos el formato Unix Timestamp. Este sistema cuenta los segundos transcurridos desde el 1 de enero de 1970 a las 00:00 hrs., que marca el inicio de la era Unix. El uso de Unix Timestamp facilitará la representación uniforme de fechas en nuestra solución, brindando consistencia en el registro temporal.

Estas consideraciones forman la base de nuestra estrategia de implementación en Prolog, asegurando un chatbot capaz de entender, procesar y responder de manera efectiva a las consultas de los usuarios, además de mantener un registro coherente del tiempo.

# Instrucciones de uso

La implementación de este laboratorio fue realizada a través del entorno virtual de prolog https://swish.swi-prolog.org/ por ser desarrollado en un equipo computacional con sistema operativo macOS Monterrey. Por lo anterior se recomienda la utilización de este entorno virtual para realizar las pruebas de funcionamiento del código entregado.

Para la ejecución de los algoritmos propuestos, debe abrir el archivo lab2\_13452929\_Bustamante.pl, seleccionar todo el contenido, copiar y pegar en el editor virtual de prolog, <https://swish.swi-prolog.org/>

Para hacer las pruebas de funcionamiento debe ingresar en el apartado para las consultas del intérprete de prolog, lo que se encuentra en el contenido del archivo, pruebas\_13452929\_Bustamante.pl

# Resultados y evaluación

A continuación, se muestra la autoevaluación de los requerimientos funcionales requeridos por el laboratorio, y el grado de logro alcanzado para cada uno de ellos.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RF Nº | Requerimiento funcional | Grado de alcance | Tipo o cantidad de pruebas | | Éxitos | | Fracasos | | Razones de fallo | |
| 1 | TDA’s | 1 |  |  | |  | |  | |
| 2 | TDA Option - constructor | 1 | 2 | 2 | | 0 | |  | |
| 3 | TDA Flow - constructor | 1 | 3, un intento de duplicidad | 2 | | 1 | | Agregaba elementos duplicados, se soluciona. | |
| 4 | TDA Flow - modificador | 1 | 1 | 1 | | 0 | |  | |
| 5 | TDA chatbot - constructor | 1 | 2, un intento de duplicidad | 1 | | 1 | | Agregaba elementos duplicados, se soluciona. | |
| 6 | TDA chatbot - modificador | 1 | 3, un intento de duplicidad | 2 | | 1 | | Agregaba elementos duplicados, se soluciona. | |
| 7 | TDA system - constructor | 1 | 2 | 2 | | 0 | |  | |
| 8 | TDA system – modificador systemAddChatbot | 1 | 2 intentos duplicidad | 2 | | 0 | |  | |
| 9 | TDA system – modificador systemAddUser | 0,5 | 2 | 0 | | 2 | | Stack limit (0.2Gb) exceeded | |
| 10 | TDA system – modificador systemLogin |  |  |  | |  | |  | |
| 11 | TDA system – modificador systemLogout |  |  |  | |  | |  | |
| 12 |  |  |  |  | |  | |  | |
| 13 |  |  |  |  | |  | |  | |
| 14 |  |  |  |  | |  | |  | |

**Tabla 1.-** Muestra los requerimientos funcionales y el nivel de logro.

# Conclusiones

Este proyecto se enfoca en abordar la necesidad de crear un chatbot que brinde respuestas estructuradas a través de acciones específicas, utilizando programación lógica y el lenguaje de programación Prolog. El chatbot, categorizado como ITR (Iteración a Través del Lenguaje Natural), permitirá a los usuarios interactuar de manera eficiente utilizando palabras clave, sinónimos, números o letras. El análisis del problema ha proporcionado una visión clara de los objetivos, requisitos, alcance y limitaciones del proyecto, guiando así la estrategia de implementación para lograr un chatbot funcional y adaptado a las necesidades identificadas.

# Referencias

* Swi-prolog, Reference manual. https://www.swi-prolog.org/pldoc/doc\_for?object=manual
* Campus Virtual, (2023). Paradigmas de Programación. https://uvirtual.usach.cl/moodle/