

PYLOG

Tarea Programada #2

Elaborado por:

José María Rojas Carrillo

José Rolando Li Acuña

Andrés Fernández Ramírez

José Gabriel Vargas Vargas

LENGUAJES DE PROGRAMACIÓN
TI-3404



Contenido

Descripción del Problema.....	1
Diseño del Problema.....	1
Decisiones de diseño.....	1
Algoritmos usados.....	1
Descripción de principales predicados.....	1
Lenguaje escogido.....	1
Razones.....	1
Análisis de Resultados.....	2
Manual de usuario.....	2
Conclusión Personal.....	2

Descripción del Problema

El programa consiste en implementar un intérprete para un lenguaje de programación lógico similar a ProLog, en este caso se utilizará Python como lenguaje de programación para el desarrollo del mismo. ProLog es un lenguaje de programación especialmente indicado para modelar problemas que impliquen objetos y las relaciones entre ellos. Está basado en los siguientes mecanismos básicos: unificación, estructuras de datos y backtracking automático. La sintaxis del lenguaje incluye la declaración de hechos, consultas y reglas. En **PyLog** el usuario inicia el programa, le ingresa un modo y realiza las consultas o inserciones deseadas.

El objetivo del programa es resolver las consultas de usuario utilizando una base de conocimientos creada con los hechos o reglas insertadas por medio del modo `<define>`. Para salir de este modo se deberá escribir la instrucción `</define>`. El programa siempre estará corriendo en modo consulta mientras este fuera del modo `<define>` y disponible para las mismas.

Además, se deberán hacer verificaciones de aridad de predicados y correctitud léxica y sintáctica de cada uno de los mismos.

Diseño del Problema

Decisiones de diseño

El programa está diseñado de manera que cuando se ejecute inicia el intérprete, cuando el usuario digite la instrucción **<define>** el intérprete entrará en modo de definición de predicados. En este modo, se permitirá que el usuario agregue hechos o reglas a la base de conocimientos que se analizan tanto léxica como sintácticamente por separado. Todas las inserciones de hechos y reglas se realizarán a una lista llamada BaseConocimientos. Como puntos importantes se tiene:

- Toda regla o hecho válido termina con un punto.
- Toda regla o hecho puede tener de **0** a **n** argumentos
- Toda regla puede tener **n** antecedentes

Adicionalmente a los predicados ingresados por el usuario, el sistema deberá tener implementados los siguientes Built-in-predicates: **write (arg)**, **nl**. Para salir del modo de definición de predicados, el usuario ingresará la instrucción **</define>**.

Si no digita **<define>** el usuario se mantiene en modo consulta donde se revisa los hechos y reglas de la base de conocimientos, con lo que responde

YES o **NO**, dependiendo de la información que haya en la base de conocimientos y de la consulta realizada.

En el caso de que una regla tenga antecedentes, deberán validar los antecedentes antes de poder responder. Además, se trabaja el backtracking, para el proceso de resolución de metas.

Algoritmos usados

Main (): Corre el programa e indica los pasos para agregar una regla o un hecho.

MenuHechosReglas (): En cuanto se agrega un hecho

GuardarHechos (): Guarda los hecho que son calificados como válidos después de lo análisis.

AnalizaLexicoHecho: Determina si el hecho esta lexicamente correcto.

AnalizaSintaxisHecho (): Determina si el hecho esta sintácticamente correcto.

GuardarRegla (): Guarda las reglas que son calificados como válidos después de los análisis.

AnalizaLexicoRegla (): Determina si la regla ingresada esta lexicamente correcta.

AnalizaSintxisRegla (): Determina si la regla ingresada esta sintácticamente correcta.

Splitcoma (): Separa los argumentos ya sea de un hecho, regla o consulta, para poder identificarlos individualmente.

Consultas (): Función que me permite ingresar al modo consulta y realizar las mismas.

CrearLista (): Convierte lo digitado en la función anterior en una lista para poder leerla más fácilmente.

Unificar (): Unifica los argumentos en la consulta

EsRegla (): Función que me permite validar se lo digitado en el modo predicados es una regla.

UnificacionArgumentos (): Unifica los argumentos correctos después de la consulta.

Descripción de principales predicados

Lenguaje escogido

Se escogió **Python** ya que es un lenguaje de programación multiparadigma, que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Esto ayuda a una mejor interpretación y digitación de código para realizar el intérprete de Prolog.

Razones

No se utilizaron librerías para la revisión léxica y sintáctica del lenguaje Prolog, ya que se decidió implementarlas desde cero para obtener un mayor aprendizaje del intérprete de Prolog.

Además, la escogencia de Python se dio ya que los integrantes tienen un mayor conocimiento de la sintaxis y manejo del lenguaje de programación.

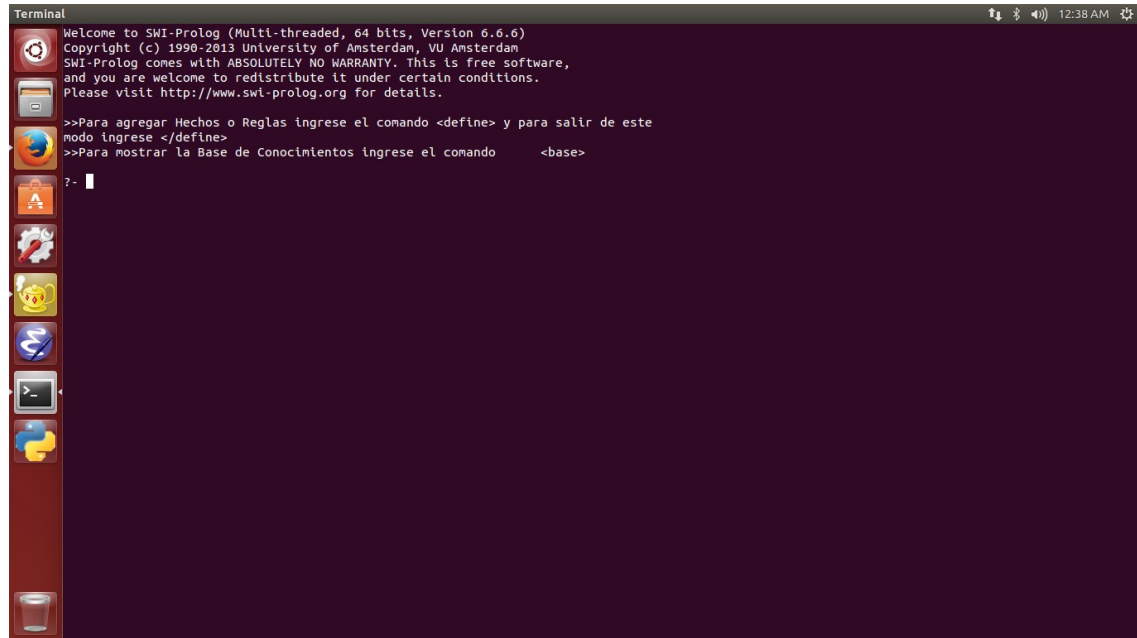
Análisis de Resultados

El programa maneja de una manera eficiente y eficaz los algoritmos ya que presenta al intérprete de Prolog (Pylog) de una forma casi idéntica al programa original. Brinda opciones de escogencia del modo Definición y Consultas, además de conocer lo almacenado en la base de datos.

La descripción de como digitar de manera correcta los hechos o reglas, ayuda al usuario a utilizar el programa con gran facilidad y rapidez.

Manual de usuario

- I. Instale el programa Python 3.3 en su computadora. (En caso de tenerlo omita esta instrucción).
- II. Ejecute el programa desde la ubicación en la que se encuentra.
- III. Se inicia el modo de Consulta automáticamente.



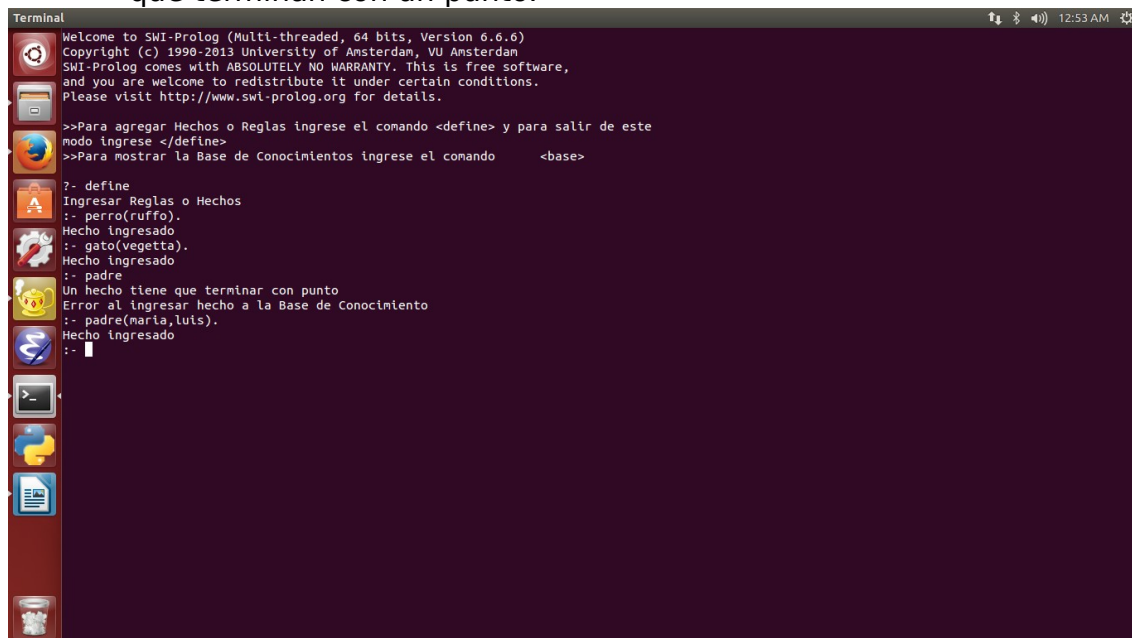
```
Terminal
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

>>Para agregar Hechos o Reglas ingrese el comando <define> y para salir de este
modo ingrese </define>
>>Para mostrar la Base de Conocimientos ingrese el comando    <base>

?- 
```

Nota: Digite **define** para acceder al modo de definición de predicados

IV. En ese modo, usted puede agregar un hecho o una regla. Recuerde que terminan con un punto.



```
Terminal
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

>>Para agregar Hechos o Reglas ingrese el comando <define> y para salir de este
modo ingrese </define>
>>Para mostrar la Base de Conocimientos ingrese el comando    <base>

?- define
Ingresar Reglas o Hechos
:- perro(ruffo).
Hecho Ingresado
:- gato(vegetta).
Hecho Ingresado
:- padre
Un hecho tiene que terminar con punto
Error al ingresar hecho a la Base de Conocimiento
:- padre(maria,luis).
Hecho Ingresado
:- 
```

Nota: Dará error si el analizador léxico o sintáctico encuentra alguna falla.

V. Para regresar al modo consulta digite </define>.

VI. Digite la consulta que desea con respecto a lo ingresado anteriormente

- VII.** Si desea puede consultar la base de conocimientos digitando **base.**

Conclusión Personal

Con la realización de este proyecto se pudo concluir que es de suma importancia, antes de empezar a programar, tener conocimiento sobre el campo en el que se está trabajando, investigar sobre diferentes funciones que son útiles para la realización del trabajo, y comprender su funcionamiento. Analizar y entender cuáles son las necesidades, qué es lo que se busca y el resultado al que se quiere llegar.

Para finalizar, es relevante mencionar que para facilitar la confección del programa, un método que se puede poner en práctica es la realización de un plan de trabajo que permita ir modelando un esquema o un diagrama de las ideas para la elaboración del sistema, y así poder ir implementando los algoritmos que se establecieron y afinar los detalles para una buena producción del programa deseado.