**Prolog**

Prolog es un lenguaje de programación que está orientada a la programación lógica. Su estructura se basa en dos principales elemento, hechos y clausulas. Las clausulas son proposiciones que pueden ser determinadas como falsas o verdaderas de acuerdo a un conjunto de hechos establecidos. En términos más simples prolog es un sistema al cual se alimenta de información que pueden ser reglas, procedimientos o hechos. Todas las relaciones que encuentra luego son utilizadas para resolver alguna consulta formulada por el usuario.

**Uso de prolog**

Prolog también ofrece herramientas que incluyen el “Backtracking” (retroceso). El Backtracking es una característica que se basa en determinar un objetivo y retroceder buscando los hechos que sean necesarios para lograr cumplirla. Esto puede ser empleado al momento de encontrar una falla en alguna tarea. Esta característica única ha sido muy útil en campos como la inteligencia artificial.

Otra función con la que cuenta prolog son las consultas. Una consulta se basa en la entrada de una proposición la cual contiene hechos que pueden ser variados y prolog se encarga de retornar todos aquellos hechos que hacen la proposición verdadera. Por esta misma característica es por la cual Prolog también puede ser empleada como un lenguaje para la manipulación de base de datos.

**Introducción a Prolog**

La sintaxis es una pieza fundamental el cual forma parte de cualquier lenguaje de programación. Este es un elemento que puede cambiar dependiendo de las reglas que establezca el lenguaje. Muchas veces el programador no encontrara gran desafío al adaptarse a una nueva sintaxis, esto es debido a que a pesar de ser diferentes presentan un enfoque similar al momento de cómo resolver un problema. Sin embargo los lenguajes orientados a una programación lógica no cumplen con esta misma relación. Un ejemplo de ello es Prolog. El cual trabaja bajo un enfoque especial. El cual no consiste en “Como” resolver un problema sino en segmentarlo y reducirlo a proposiciones que pueden ser determinadas como verdaderas o falsas y luego relacionarlas para generar estructuras más complejas.

La sintaxis de Prolog trabaja con dos principales elementos, variables y constantes. Mediante estos elementos se pueden construir hechos y clausulas. Para una clara comprensión primero es importante definir algunos elementos básicos del lenguaje el cual ayudara a manipular o elaborar estos dos elementos.

/\*

Los siguientes son algunos de elementos del lenguaje de prolog más

comunes y a partir de ellos se pueden construir estructuras basadas

en proposiciones complejas.

\*/

% :- --- Implicación (de derecha a izquierda)

% . --- Significa el final de la proposición

% == --- Operador de igualdad

% \== --- Operador de distinto

% \+ --- Operador de negación

% , --- Operador AND

% ; --- Operador OR

Los Hechos son una proposición que puede ser conformada por uno o un conjunto de constantes o variables.

color(rojo).

color(verde).

color(azul).

color(amarillo).

Las Clausula son relaciones entre Hechos estas relaciones usan algo conocido como “Pattern Matching” (Coincidencia por patrón), Los cuales verifican que hechos cumplen con una determinada regla declarada dentro una clausula.

rgb(X) :-

X == rojo;

X == verde;

X == azul.

A partir de una herramienta como SWi-Prolog se puede correr programas escritos usando la gramática de Prolog. A partir de la consola podemos usar la clausula anterior de dos distintas maneras.

?- rgb(x).

?- rgb(verde).

En la primera manera se hace uso de la clausula para preguntarle al programa de prolog cuales Hechos hacen verdadera la proposición. En la segunda forma preguntamos a prolog si la proposición declarada es verdadera o falsa.

Otro elemento importante de prolog son sus listas. Las listas en Prolog a principio pueden parecer semejantes a cualquier lenguaje de programación conocido en la actualidad. Estas pueden contener cualquier elemento previamente mencionado o incluso más listas. Sin embargo cuentan con una característica especial. Cada lista tiene dos piezas que la conforman, estas son la cabeza (Header) y la cola (Tail). En una lista la cabeza sería el primer elemento contenido y la cola seria otra lista con los demás elementos. Esta característica brinda gran apoyo durante la manipulación de una lista mediante recursión.

**Prolog para el Desarrollo Web**

Como antes se mencionaba Prolog es una poderosa herramienta para resolver consultas. Es por esto que llega a ser una gran opción cuando se refiere a ser implementado para la elaboración de una página web. Varias plataformas ya lo hacen, un ejemplo de ello es la página de Price Charting, Price Charting es una página de venta de videojuegos clásicos, El uso de prolog destaca al momento de manejar la base de datos de todas las características que maneja cualquiera de estos títulos. Gracias a prolog podemos manejar cualquier petición generado por el cliente desde el servidor. Para esto se hará uso de la librería “http” que ofrece y está documentada por la página oficial de prolog.

:- use\_module(library(http/thread\_httpd)).

:- use\_module(library(http/http\_dispatch)).

:- initialization

http\_server(http\_dispatch, [port(8080)]).

Mediante estas librerías se establecerá al lenguaje de programación de prolog como el principal componente del backend del sistema. El código que se ejecuta en la zona superior será el encargado de correr el servidor en el puerto 8080 de nuestra red local, y podremos acceder a ella desde cualquier navegador a través del siguiente link: http://localhost:8080/.

**Manejadores**

Para programar nuestro primer "Hola Mundo" primero se necesita abarcar el tema de los “Manejadores” (Handlers). Un Manejador es una herramienta que nos ofrece la librería de http para el manejo de rutas. Esto quiere decir que un manejador se encargara de retornar o ejecutar una tarea dependiendo de la ruta del servidor web donde se encuentre el cliente. Normalmente estas peticiones generadas son conocidas como peticiones GET.

Un ejemplo de estas rutas es la misma dirección del servidor web. Cuando el cliente intenta acceder a la raíz de la página web los manejadores son los encargado de entregarle el documento html el cual se mostrara en el navegador. Para programar un manejador se necesita usar la siguiente proposición.

:- http\_handler('/', index, []).

La anterior proposición presenta 3 elementos, el primero se refiere la ruta del cual el manejador será responsable responder. El segundo elemento será la tarea ejecutada y que por lo general será encargada de retornar el documento HTML o cualquier otro archivo. El último argumento se refiere al tipo de petición.

El siguiente paso será desarrollar el segundo elemento mencionado, en el caso del ejemplo será definido como index.

index(\_Request) :-

format('Content-Type: text/html~n~n'),

format('<h1>Hello World!</h1>~n').

El código anterior retornara al manejador un archivo de HTML el cual se mostrara en el navegador.

**Generar HTML Controlado**

Sin embargo el manipular etiquetas de HTML a través de un texto plano no es la mejor manera de hacerlo, por eso es que la librería de http también ofrece una opción con la que se puede crear etiquetas de HTML a través de la declaración de hechos.

index(Request):-

reply\_html\_page(

[title('Mi Título')],

[\page\_content(Request)]

).

page\_content(\_Request) -->

html(

[

h1('Esto es el Encabezado de la Página'),

p('Soy un Párrafo')

]

).

En el código anterior se segmentó la cláusula y ahora se puede observar un mayor control sobre los componentes del HTML. La primera regla se encarga de responder con un documento HTML. El documento HTML es declarado como una lista, En la primera parte de la lista se observa el uso de la etiqueta title utilizado para declarar un nombre al documento. Por lo general esta etiqueta pertenece al conjunto del Head de la estructura. En la segunda parte de la lista observamos el uso de una función que llama un fragmento de contenido. El fragmento de contenido también mantiene la misma estructura donde etiquetas HTML son declaradas como reglas y contenidas en una lista.

Con este programa dentro de nuestro servidor podemos obtener un mayor control sobre la estructura HTML y facilitarnos el desarrollo al reutilizar fragmentos con diseño similar.

Otra funcionalidad que nos ofrece el manejo de estos elementos es la declaración de atributos. Los atributos de las etiquetas son agregados de una manera muy similar a como es declarado el cuerpo de la etiqueta.

page\_content(\_Request) -->

html(

[

p(['Soy otro Párrafo pero con ', b('Negrita!')),

p([style='font-size: 36pt', class='styleText'],'Soy otro Párrafo pero con Estilo'),

]

).

Regresando a los fragmentos de Contenido también cabe mencionar una herramienta que será de gran utilidad sobre todo si queremos hacer nuestras páginas web. Dentro de las funciones podemos declarar reglas de forma muy similar de como si fuera una variable. Es decir que podemos diseñar un fragmento el cual puede cambiar de acuerdo de los argumentos de la proposición. Un ejemplo es el siguiente segmento de código. El cual representa el diseño implementado en cada bloque de código en esta misma página.

code\_txt(X) -->

html(

pre(

[class = 'codetxtClass'],

[code(X)]

)

).

**Mas Rutas**

Como antes se mencionaba, un manejador es el responsable de llevar a cabo una tarea dada una ruta, para definir mas rutas entonces solo tendremos que definir más manejadores. Existen varias maneras de escribir la ruta de un manejador. Se puede hacer uso de reglas haciendo una estructura jerárquica comenzando con el ROOT, añadiéndole un argumento para indicar una subruta dentro de la ruta padre. Otra manera más simple es el agregar la dirección en forma de cadena, de la manera convencional tal como se vería en un navegador.

:- http\_handler(root(.), index , []).

:- http\_handler(root('nombre'), name, []).

:- http\_handler('./nombreCompleto'), fullname, []).

Cada tarea se encargara de ejecutar y retornar un documento HTML diferente, o al menos así debería ser. Incluso se podría crear rutas para archivos distintos como imágenes las cuales se hospedarían dentro de una ruta única en el servidor para su uso dentro de un documento HTML gracias al componente de img y la ruta de la misma. Para esto se haría uso de una nueva herramienta brindada por la librería.

%Rutas de Imagenes

http:location(images, root(images), []).

user:file\_search\_path(imgs, './imgs').

:- http\_handler(images('ejemplo.png'), ejemplo, []).

ejemplo(Request) :-

http\_reply\_file(imgs('ejemplo.png'),[unsafe(true)],Request).

En el código superior se observan dos nuevas proposiciones. La primera, “file\_search\_path”, se encargara de generar una comunicación con el servidor y la carpeta contenedora. El primer argumento es el alias generado según la ruta declarada en el segundo parámetro. La segunda proposición, http\_reply\_file, Ejecutara una tarea muy similar al http\_reply\_html visto en programas anteriores. El primer argumento hace uso del alias de la carpeta contenedora antes establecida, mientras que los siguientes parámetros hacen referencia a opciones o peticiones durante la ejecución de la proposición.

**Petición Post**

Durante la mayor parte del informe se habló sobre las peticiones GET las cuales son generadas cuando el usuario intenta acceder a una dirección del servidor. La única información que recibe esta petición es la ruta establecida, Es una petición la cual está limitado a solo trabajar con lo que ya sabe.

La petición POST nos ofrece una forma en la que el cliente puede mandar información que podemos estructurar con un formulario, Esta información luego es enviada al servidor y prolog se encarga de la manipulación y ejecución de cualquier tarea que se necesite hacer con ella. El ejemplo más claro para el uso de una petición Post se da cuando el cliente introduce información personal para la creación de una cuenta.

La librería de http no se queda atrás y nos ofrece las herramientas necesarias para llevar a cabo esta tarea. Para ello es necesario construir un formulario. Un formulario es una estructura de un documento HTML el cual contienen casillas para ser rellenadas con la información respectiva del usuario. Dentro de los atributos de los usuarios tendremos que establecer la acción que se ejecutara y el nombre de la petición.

web\_form() -->

html(

[

form([action='/reply', method='POST'], [

p([], [

label([for=name],'Nombre:'),

input([name=name, type=textarea])

]),

p([], [

label([for=acc],'#Cuenta:'),

input([name=acc, type=number])

]),

p([], [

label([for=email],'Email:'),

input([name=email, type=textarea])

]),

p([], input([name=submit, type=submit, value='Enviar'], []))

])

]

).

Dentro del formulario podemos observar la existencia de dos etiquetas la cuales serán necesarias para generar las casillas que el usuario tendrá que rellenar. Estas son Input y label. La etiqueta label indica al usuario la información que deberá ir dentro la casilla, mientras que input hace referencia al espacio donde deberá agregarlo. Cabe destacar que los inputs deben contener el atributo de “name”. El atributo name ayudara a facilitar la manipulación de la información dentro el servidor, brindando la opción de acceder fácilmente al dato enviado.

El siguiente paso es hacer el manejador de la respuesta que ejecutara el servidor. Para esto se seguirá el mismo proceso que se hizo para declarar nuevas rutas.

:- http\_handler('/reply', reply, []).

reply(Request) :-

member(method(post), Request), !,

http\_read\_data(Request, Data, []),

member(email=Email,Data),

member(name=Name,Data),

member(acc=Acc,Data),

reply\_html\_page(

[title('Informe Prolog')],

[

h1('Descripción de los Datos Enviados.'),

p([b('Nombre: '), Name]),

p([b('#Cuenta: '), Acc]),

p([b('Email: '), Email])

]

).

Dentro de la ejecución del manejador se puede observar varias nuevas proposiciones. “Member” será el encargado de brindarnos la información de la petición entrante. Con ella se puede verificar el tipo de la petición y acceder a los datos enviados mediante el “name” definido previamente. Antes de poder acceder a los datos será necesario leer la petición mediante la proposición de “http\_read\_data”. Abajo se puede observar el formulario creado con su respectiva respuesta.

Principio del formulario

Nombre:

#Cuenta:

Email:



Final del formulario

**Más detalles del servidor**

El primer detalle a mencionar será acerca de la codificación del documento HTML generado. Una instrucción el cual es importante declarar sobre todo para aquellas páginas con público hispano hablantes es el uso de la codificación utf-8. Utf-8 ayuda a reconocer algunos caracteres especiales como las vocales con tildes. Para indicarle al programa trabajar en esta codificación será necesario declarar la siguiente proposición.

:- encoding(utf8).

Como último punto a mencionar será el cierre del servidor. Para cerrar el servidor que se ha creado solo será necesario usar el comando “halt.” dentro de la consola de SWI-PL.

**Conclusiones**

1. El uso de prolog como un lenguaje de programación ha significado la construcción de un nuevo paradigma el cual se enfoca en construir predicados y relaciones entre ellos para crear estructuras complejas de información. Este proceso ha sido muy útil en ramas como inteligencia artificial y matemáticas.
2. La sintaxis de prolog es muy distinta respecto a los lenguajes de programación convencionales. La razón tras de ello es por el enfoque empleado al momento de construir soluciones. Mientras un lenguaje de programación convencional se encarga de resolver un problema de manera secuencial, prolog necesita descomponer el problema y generar relaciones.
3. La funcionalidad de prolog al momento de responder consultas le ha dado la posibilidad de ser implementado para la creación de una plataforma web. Un servidor el cual usa prolog como su sistema de respuesta de consultas presenta una fuerte estructura la cual no necesita miles líneas de código para considerarse segura.
4. La librería de “http” hace de prolog una opción a tener en cuenta durante el desarrollo de una plataforma web. En ella se puede encontrar piezas fundamentales como los manejadores, que son una herramienta poderosa que nos ayudara durante la gestión de las respuestas para la declaración de una ruta. También podemos hacer uso de etiquetas HTML en la forma de Hechos para una manipulación más limpia y controlada de nuestros documentos HTML.
5. La librería cuenta con muchas más herramientas de las planteadas en este informe. Se puede crear una ruta virtual donde se pueden hospedar todas las imágenes que necesitara la página. También se puede implementar un sistema de autenticación segura para el inicio de sesión de algún usuario. Cuenta con opciones para manejar peticiones como PUT y DELETE , ofrece la posibilidad de implementar javascript, permite el uso de json para la manipulación de información y muchas cosas más. Sin lugar a dudas prolog brinda todas las herramientas necesarias para elaborar una plataforma web rápida y segura.