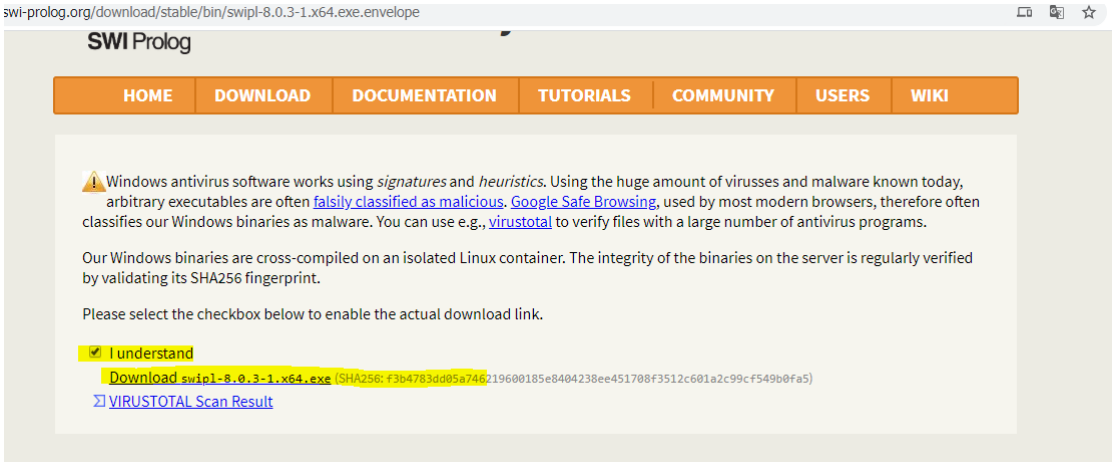
Fecha de entrega	24/04/2020
Solo se enviará al Whatsapp, solo en el **Grupo No de manera personal**, el link de GitHub, por ejemplo: Pedro Hernández Hernández = https://github.com/UsuariosCreado	

1-Resumen del siguiente archivo a libreta y posteriormente escanear y guardar en formato. PFD, después subir a GitHub

2.- Descargar e Instalar Swi-Prolog. Manual de instalación en PDF.

https://www.swi-prolog.org/download/stable/bin/swipl-8.0.3-1.x64.exe.envelope



PROLOG

Introducción

El lenguaje de programación PROLOG (“PROgrammation en LOGique”) fue creado por Alain Colmerauer y sus colaboradores alrededor de 1970 en la Universidad de Marseille-Aix1.

PROLOG usa Lógica de Predicados de Primer Orden (restringida a cláusulas de Horn) para representar datos y conocimiento, utiliza encadenamiento hacia atrás y una estrategia de control retroactiva sin información heurística (backtracking).

Una de las ventajas de la programación lógica es que se especifica qué se tiene que hacer (programación declarativa), y no cómo se debe hacer (programación imperativa). A pesar de esto, Prolog incluye algunos predicados predefinidos meta-lógicos, ajenos al ámbito de la Lógica de Primer Orden, (var, nonvar, ==, ...), otros extra-lógicos, que tienen un efecto lateral, (write, get, ...) y un tercer grupo que nos sirven para expresar información de control de como realizar alguna tarea (el corte, ...).

Por tanto, Prolog ofrece un sistema de programación práctico que tiene algunas de las ventajas de claridad y declaratividad que ofrecería un lenguaje de programación lógica y, al mismo tiempo, nos permite un cierto control y operatividad.

Elementos del lenguaje

- Hechos (átomos).

- Reglas (cláusulas de Horn).

- Preguntas u objetivos (conjunciones ó disyunciones de átomos).

PREDICADOS

Se utilizan para expresar propiedades de los objetos, predicados monádicos, y relaciones entre ellos, predicados poliádicos. En Prolog los llamaremos **hechos**. Los hechos son las sentencias más sencillas. Un hecho es una fórmula atómica o átomo: **p(t1, ..., tn)** e indica que se verifica la relación

Debemos tener en cuenta que:

• Los nombres de todos los objetos y relaciones deben comenzar con una letra minúscula.

• Primero se escribe la relación o propiedad: predicado

• Y los objetos se escriben separándolos mediante comas y encerrados entre paréntesis: argumentos.

• Al final del hecho debe ir un punto (".").

Ejemplo : simbolo_de_predicado(arg1,arg2,...,argn).

UNIFICACIÓN

La unificación, es el algoritmo que se encarga de resolver las igualdades lógicas. Aunque la unificación en Prolog sustituye a la asignación de los lenguajes procedurales, no hay que confundirla con esta, son operaciones totalmente distintas. La operación de unificación se representa mediante el operador **=**.

TERMINOS

Los términos pueden ser constantes o variables, y suponemos definido un dominio no vacío en el cual toman valores (Universo del Discurso).

Las **constantes** se utilizan para dar nombre a objetos concretos del dominio, dicho de otra manera, representan individuos conocidos de nuestro Universo. Además, como ya hemos dicho, las constantes atómicas de Prolog también se utilizan para representar propiedades y relaciones entre los objetos del dominio. Hay dos clases de constantes:

• **Átomos**: existen tres clases de constantes atómicas:

- Cadenas de letras, dígitos y subrayado (_) empezando por letra minúscula.

- Cualquier cadena de caracteres encerrada entre comillas simples (').

- Combinaciones especiales de signos: "?-", ":-", ...

• **Números**: se utilizan para representar números de forma que se puedan realizar operaciones aritméticas. Dependen del ordenador y la implementación.

- Enteros: en la implementación de Prolog-2 puede utilizarse cualquier entero que el intervalo [-223,223-1]=[-8.388.608,8.388.607].

- Reales: decimales en coma flotante, consistentes en al menos un dígito, opcionalmente un punto decimal y más dígitos, opcionalmente E, un «+» o «-» y más dígitos.

Las **variables** se utilizan para representar objetos cualesquiera del Universo u objetos desconocidos en ese momento, es decir, son las incógnitas del problema. Se diferencian de los átomos en que empiezan siempre con una letra mayúscula o con el signo de subrayado (_). Así, deberemos ir con cuidado ya que cualquier identificador que empiece por mayúscula, será tomado por Prolog como una variable. Para trabajar con objetos desconocidos cuya identidad no nos interesa, podemos utilizar la variable anónima (_). Las variables anónimas no están compartidas entre sí.

CONECTIVAS LÓGICAS

Para trabajar con sentencias más complejas, fórmulas moleculares, que constarán de fórmulas atómicas combinadas mediante conectivas. Las conectivas que se utilizan en la Lógica de Primer Orden son: conjunción, disyunción, negación e implicación.

La conjunción, “y”, la representaremos poniendo una coma entre los objetivos “,” y consiste en objetivos separados que Prolog debe satisfacer, uno después de otro:

Por ejemplo **X , Y.**

La **disyunción**, “o”, tendrá éxito si se cumple alguno de los objetivos que la componen. Se utiliza un punto y coma “;” colocado entre los objetivos.

Por ejemplo **X ; Y.**

La **negación** lógica no puede ser representada explícitamente en Prolog, sino que se representa implícitamente por la falta de aserción : “no”, tendrá éxito si el objetivo X fracasa. No es una verdadera negación, en el sentido de la Lógica, sino una negación “por fallo”. La representamos con el predicado predefinido not o con \+.

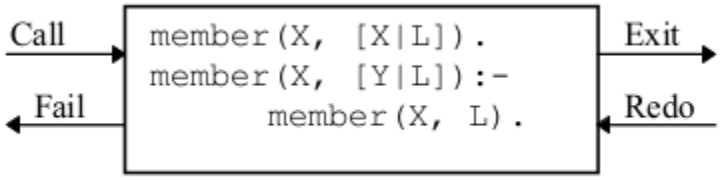
Por ejemplo: **not(X) \+ X .**

La **implicación o condicional**, sirve para significar que un hecho depende de un grupo de otros hechos. En castellano solemos utilizar las palabras “si ... entonces ...”. En Prolog se usa el símbolo “:-” para representar lo que llamamos una regla:

Por ejemplo : **cabeza_de_la_regla :- cuerpo_de_la_regla**.

EL DEPURADOR DE PROLOG

En Prolog, los predicados se consideran “cajas negras” que tienen cuatro puertos, dos de entrada y dos de salida.



Puertos:

• Call (llamada): Comienza la ejecución del objetivo.

• Exit (salida): Salida con éxito del objetivo.

• Redo (reintentar): Reintentar el predicado utilizando otra alternativa.

• Fail (fallo): Salida con fallo del objetivo. No se encontraron soluciones.

ESTRUCTURA DE UN PROLOG

El hecho de programar en Prolog consiste en dar al ordenador un Universo finito en forma de hechos y reglas, proporcionando los medios para realizar inferencias de un hecho a otro. A continuación, si se hacen las preguntas adecuadas, Prolog buscará las respuestas en dicho Universo y las presentará en la pantalla. La programación en Prolog consiste en:

• declarar algunos HECHOS sobre los objetos y sus relaciones,

• definir algunas REGLAS sobre los objetos y sus relaciones, y

• hacer PREGUNTAS sobre los objetos y sus relaciones.

Programa Prolog: Conjunto de afirmaciones (hechos y reglas) representando los conocimientos que poseemos en un determinado dominio o campo de nuestra competencia.

Ejecución del programa: Demostración de un Teorema en este Universo, es decir, demostración de que una conclusión se deduce de las premisas (afirmaciones previas).

PREGUNTAS

Las preguntas son las herramientas que tenemos para recuperar la información desde Prolog. Al hacer una pregunta a un programa lógico queremos determinar si esa pregunta es consecuencia lógica del programa. Prolog considera que todo lo que hay en la Base de Datos es verdad, y lo que no, es falso. De manera que si Prolog responde “yes” es que ha podido demostrarlo, y si responde “no” es que no lo ha podido demostrar (no debe interpretarse como “falso” si no que con lo que Prolog conoce no puede demostrar su veracidad).

Cuando se hace una pregunta a Prolog, éste efectuará una búsqueda por toda la Base de Datos intentando encontrar hechos que coincidan con la pregunta. Dos hechos “coinciden” (se pueden unificar) si sus predicados son el mismo (se escriben de igual forma) y si cada uno de los respectivos argumentos son iguales entre sí.

?- simbolo_de_predicado(arg1,arg2,...,argn).

SINTAXIS

COMENTARIOS.

Para escribir un comentario lo encerraremos entre los signos /* y */ o desde el símbolo % hasta el final de línea. Así, Prolog pasa por alto los comentarios, pero los debemos añadir a nuestros programas para aclararlos y que el propio programa quede documentado

/* ... comentario ... */

% comentario de una sola línea

CARACTERES

Los nombres de constantes y variables se construyen a partir de cadenas de caracteres. Prolog reconoce dos tipos de caracteres:

• Imprimibles: hacen que aparezca un determinado signo en la pantalla del ordenador.

Se dividen en cuatro categorías:

letras mayúsculas: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U,V, W, X, Y, Z.

letras minúsculas: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

dígitos numéricos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

signos: !"#$%&'()=-^|/\{}[]_@+*;:<>,.?

• No imprimibles: no aparecen en forma de signo en la pantalla, pero realizan una determinada acción: nueva línea, retorno de carro, ... Cada carácter tiene un entero entre 0 y 127 asociado a él, este es su código ASCII ("American Standard Code for Information Interchange").

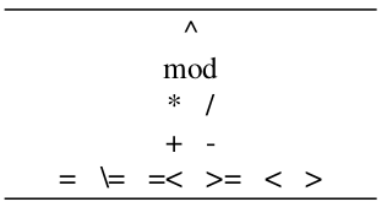
ESTRUCTURAS

Una estructura es un único objeto que se compone de una colección de otros objetos, llamados componentes, lo que facilita su tratamiento. Una estructura se escribe en Prolog especificando su nombre, y sus componentes (argumentos). Las componentes están encerradas entre paréntesis y separadas por comas; el nombre se escribe justo antes de abrir el paréntesis:

nombre (comp1, comp2, ..., compn)

OPERADORES

En Prolog están predefinidos los operadores aritméticos y relacionales típicos, con la precedencia habitual entre ellos:



Para poder leer expresiones que contengan operadores necesitamos conocer los siguientes atributos:

* POSICIÓN: Prefijo: el operador va delante de sus argumentos.

Infijo: el operador se escribe entre los argumentos.

Postfijo: el operador se escribe detrás de sus argumentos.

* PRECEDENCIA: Nos indica el orden en que se realizan las operaciones. El operador más prioritario tendrá una precedencia 1 y el menos, 1201 (depende de la implementación).

*ASOCIATIVIDAD: Sirve para quitar la ambigüedad en las expresiones en las que hay dos operadores, uno a cada lado del argumento, que tienen la misma precedencia.