



Instituto Tecnológico de Mexicali

“Reporte de Practicas Prolog”

Carrera:

Ingeniería en Sistemas Computacionales

Nombre del Alumno(a):

Martínez Yebra Beatriz Andrea
#13490929

Nombre del profesor(a):

Batista Gaxiola Oscar Ruben

Materia:

Programación Lógica y Funcional

Horario:

09:00 p.m. – 10:00 p.m.

Mexicali Baja California, Miércoles 16 de Mayo 2018.

Introducción

El siguiente documento muestra un reporte de practicas el cual sirve como evidencia de que se han realizado los ejercicios mostrados en el salón de clases, además que se ha comprendido el desarrollo de estos mismos, se mostrará y explicará cada ejercicio realizado los cuales proporcionará el profesor de la clase Batista Gaxiola Oscar Ruben.

Se explicará brevemente la introducción general del lenguaje de programación Prolog, desde lo que es la instalación hasta el cómo guardar archivos y ejecutarlos. Comenzaremos compilando un “Hola Mundo”, Y seguiremos con el desarrollo de algunos ejemplos dependiendo del tema que se explique en clase.

Contenido

El siguiente índice de contenido muestra los temas desarrollados a lo largo del reporte. Si se desea dirigir a un tema en específico puede dar click en dicho título para ir a la página donde se encuentra la información deseada en el documento.

	Nº Página
Introducción.....	1
Contenido.....	2
Tabla de figuras.....	3
Introducción al reporte.....	4
a) Objetivo	
b) Fundamento Teórico	
c) Justificación, Aplicación e importancia	
Enunciado de las actividades.....	5- 21
• ¿Qué es Prolog?	
• Hechos en Prolog	
• Variables en Prolog	
• Reglas en Prolog	
• Operadores en Prolog	
• Instalación de Prolog	
• Practicas realizadas en clase	
Resultados.....	22
Conclusión.....	23
Bibliografía.....	24

Tabla de Figuras

La siguiente tabla muestra un listado de la ubicación de capturas de pantallas o imágenes de los ejercicios realizados a lo largo del siguiente reporte. Selecciona el título de alguna captura o imagen para ir a la página donde se encuentra.

TABLA DE DIRECCIONAMIENTO DE CAPTURAS DE PANTALLA / IMÁGENES	
“ Instalación de Prolog ”	Se nos muestra y explica la instalación de Prolog en cinco pasos una vez descargado el archivo para ejecutarlo.
Imagen 01 “Estructura y Arboles ”	Explicación de la estructura y árboles en Prolog mediante un ejemplo visual.
Imagen 02 “ Representación de Listas ”	Representación de listas en Prolog.
Imagen 03 “Pertenencia”	Pertenencia para saber si un objeto pertenece a la lista.
Imagen 04 “Recurción (Bucle)”	Representación de Recursión y error mediante bucle.
Imagen 05 “Practica Recurción, Conversación”	Prolog contesta!
Imagen 06 “Concatenar”	Ejemplos de cómo concatenar con listas
Imagen 07 “Practica Bicicleta”	Practica que muestra, recursión, concatenar, estructura, listas.
Imagen 08 “Satisfacción de Objetivos” (trace/notrace)	Practica donde se muestra y explica el buscador de objetivos Trace.
Imagen 09 “Corte: !”	Practica donde se muestra y explica el predicado Corte: !

Introducción al reporte

En el siguiente reporte se mostrará la introducción básica al lenguaje de programación Prolog. Se explicará cómo instalarlo, y como compilar programas él. Comenzaremos realizando un “Hola Mundo”, y se mostraran algunos otros ejemplos dependiendo del tema sea estructura de árboles, listas u otro.

[1] Lenguaje de programación diseñado para representar y utilizar el conocimiento que se tiene sobre un determinado dominio. Los programas en Prolog responden preguntas sobre el tema de cual tienen conocimiento. 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 basadas en árboles y backtraking automático.

“Esta práctica me permitirá conocer más sobre el lenguaje de programación Prolog, desde cómo es que se instala hasta el conocer sus elementos, su sintaxis, sus reglas y hechos y además como consultar en él. Espero comprender la sintaxis de dicho lenguaje ya que es bastante complicado dependiendo de cómo es que uno procesa la información del problema.”

Enunciados de las actividades

¿Qué es Prolog?

[3] **Lenguaje de programación declarativo**, estos se definen de los lenguajes imperativos o procedurales en que están basados en formalismo abstractos. Prolog está basado en la lógica de predicados de primer orden y LISP, otro lenguaje de programación declarativo y por lo tanto su semántica no depende de la máquina en la que se ejecutan. Las sentencias en estos lenguajes se entienden sin necesidad de hacer referencia al nivel máquina para explicar los efectos colaterales. Por lo tanto, un programa escrito en un lenguaje declarativo puede usarse como una especificación o una descripción formal de un problema. Una de las ventajas de los programas escritos en lenguajes declarativos es que pueden desarrollar y comprobar poco a poco, y pueden ser sintetizados o transformados sistemáticamente.

La sintaxis del lenguaje consiste en lo siguiente:

- Declarar hechos sobre objetos y sus relaciones.
- Hacer preguntas sobre objetos y sus relaciones.
- Definir reglas sobre objetos y sus relaciones.

Hechos en Prolog

Expresan relaciones entre objetos. Por ejemplo; Supongamos que queremos expresar el hecho de que “un coche tiene ruedas”. Este hecho, consta de dos objetos, “coche” y “ruedas”, y de una relación llamada “tiene”. La forma de representarlo en Prolog es:

Tiene (coche, ruedas).

- Los nombres de objetos y relaciones deben comenzar con una letra minúscula.
- Primero se escribe la relación, y luego los objetos separados por comas y encerrados entre paréntesis.
- Al final de un hecho debe ir un punto (el carácter “.”).

NOTA: El orden de los objetos dentro de la relación es arbitrario, pero debemos ser coherentes a lo largo de la base de hechos.

Variables en Prolog

Representan objetos que el mismo Prolog determinará. Una variable puede estar instanciada o no instanciada. Estará instanciada cuando existe un objeto determinado representado por la variable. De este modo, cuando preguntamos “¿Un coche tiene X?”, Prolog busca en los hechos cosas que tiene un coche y este respondería:

X = ruedas.

Instanciado la variable X con el objeto ruedas.

- **Los nombres de variables comienzan siempre por una letra mayúscula.**

En su caso particular es la variable anónima, representada por el carácter subrayado (“_”). Es una especie de comodín que se utiliza en aquellos lugares que debería aparecer una variable, pero no importa la idea de darle un nombre concreto ya que no se va a utilizar posteriormente.

Reglas en Prolog

Las reglas se utilizan en Prolog para significar que un hecho depende de uno o más hechos. Son la representación de las implicaciones lógicas del tipo $p \rightarrow q$ (p implica q).

- Una regla consiste en una cabeza y un cuerpo, unidos por el signo “:-”.
- La cabeza está formada por un único hecho.
- El cuerpo puede ser uno o más hechos (conjunción de hechos), separados por una coma (“,”), que actúa como el “y” lógico.
- Las reglas finalizan con un punto (“.”).

La cabeza en una regla Prolog corresponde al consecuente de una implicación lógica, y el cuerpo al antecedente. Este hecho puede conducir a errores de representación.

Por ejemplo:

tiempo(lluvioso) \rightarrow suelo(mojado)
suelo(mojado)

Que el suelo esté mojado, es una condición suficiente de que el tiempo sea lluvioso, pero no necesaria. Por lo tanto, a partir de ese hecho, no podemos deducir mediante la implicación, que esté lloviendo (pueden haber regado las calles). La representación correcta en Prolog, sería:

```
suelo(mojado): - tiempo(lluvioso).  
Suelo(mojado).
```

Adviértase que la regla esta “al revés”. Esto es así por el mecanismo de deducción hacia atrás que emplea Prolog. Si cometiéramos el error de representarla como:

```
tiempo(lluvioso): - suelo(mojado).  
Suelo(mojado).
```

Prolog, partiendo del hecho de que el suelo esta mojado, deduciría incorrectamente que el tiempo es lluvioso. Para generalizar una relación entre objetos mediante una regla, se utilizan variables.

Ámbito de las variables

Cuando en una regla aparece una variable, el ámbito de esa variable es únicamente esa regla. Supongamos las siguientes reglas:

- (1) hermana_de (X, Y): - hembra(X), padres (X, M, P), padres (Y, M, P).
- (2) puede_robar (X, P): - ladron(X), le_gusta_a (X, P), valioso(P).

Aunque en ambas aparece la variable X (y la variable P), no tiene nada que ver la X de la regla (1) con la de la regla (2), y, por lo tanto, la instanciación de la X en (1) no implica la instanciación en (2). Sin embargo, todas las X **de una misma regla** sí que se instanciar n con el mismo valor.

Operadores en Prolog

Son predicados predefinidos en Prolog para las operaciones matemáticas básicas. Su sintaxis depende de la posición que ocupen, pudiendo ser infijos o prefijos. También se disponen de predicados de igualdad o desigualdad.

$X = Y$	Igual
$X \neq Y$	Distinto
$X < Y$	Menor
$X > Y$	Mayor
$X \leq Y$	Menor o Igual
$X \geq Y$	Mayor o Igual

Al igual que en otros lenguajes de programación es necesario tener en cuenta la precedencia y asociatividad de los operadores antes de trabajar con ellos. En cuanto a precedencia, es la típica.

Por ejemplo:

$3+2*6$ se evalúa como $3+(2*6)$. En lo referente a la asociatividad, Prolog es asociativo por la izquierda. Así, $8/4/4$ se interpreta como $(8/4)/4$. De igual forma, $5+8/2/2$ significa $5+((8/2)/2)$.

El operador “is”.

Es un operador infijo, que en su parte derecha lleva un término que se interpreta como una expresión aritmética, contrastándose con el término de su izquierda.

Por ejemplo, la expresión ‘6 is 4+3.’ es falsa. Por otra parte, si la expresión es ‘X is 4+3.’, el resultado será la instanciación de X: $X = 7$

Ejemplo de regla Prolog: $\text{densidad}(X,Y) \text{ :- población}(X,P), \text{area}(X,A), Y \text{ is } P/A.$

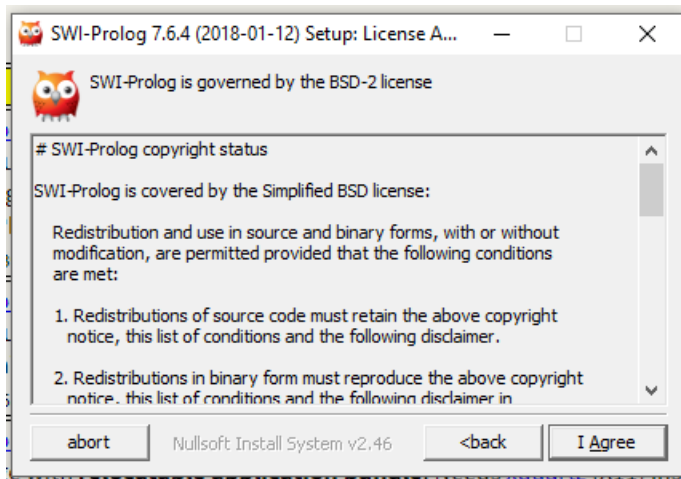
Instalación de Prolog

Se entra a la página oficial de Prolog: <http://www.swi-prolog.org/> y nos dirigimos a download donde se despliega una lista de selección y elegimos SWI-Prolog, nos aparecerá una lista de la cual elegimos “Stable release”, enseguida nos aparecerán los links de descargas de Prolog más actualizado hasta la fecha y podemos escoger entre el sistema operativo deseado. En mi caso Elijo, Windows 64 bits.

Esperamos a que se descargue nuestro archivo y lo ejecutamos.

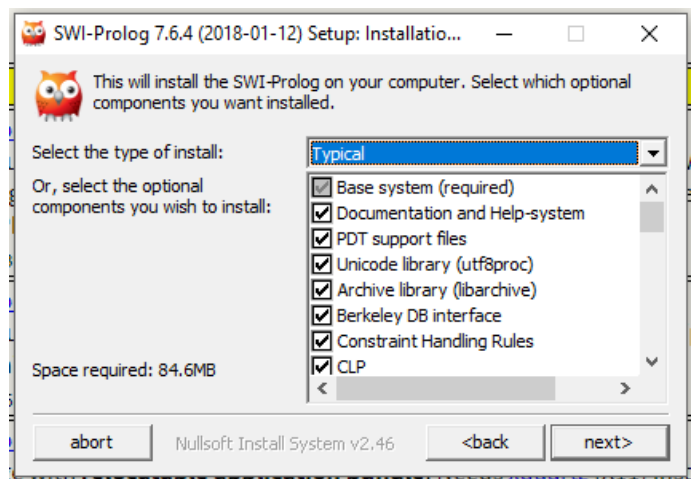
Y realmente la instalación es sencilla, desde que comienza hasta que termina solo hacemos click en next, ya que es recomendado dejar lo que está habilitado por default.

A continuación se muestra la instalación en cinco imágenes.



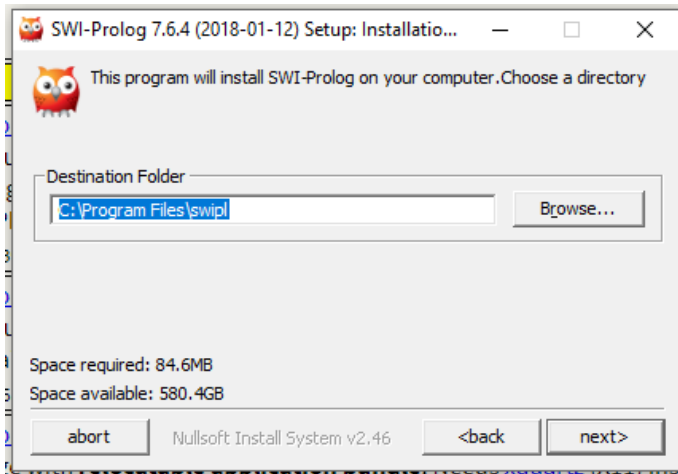
Instalación SWI-Prolog Parte 1

Condiciones de Licencia



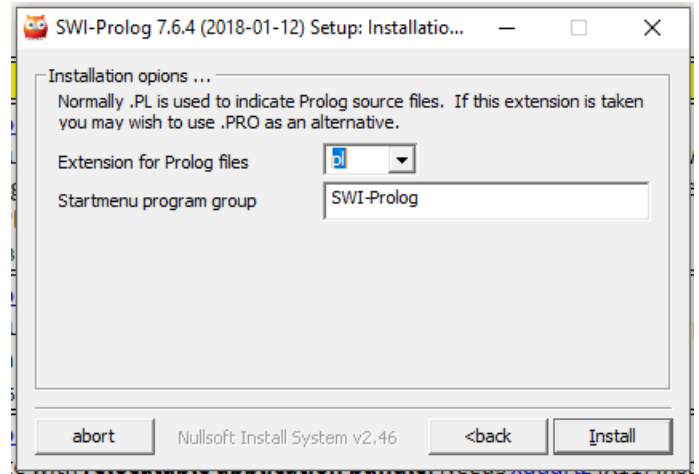
Instalación SWI-Prolog Parte 2

Componentes que a instalar



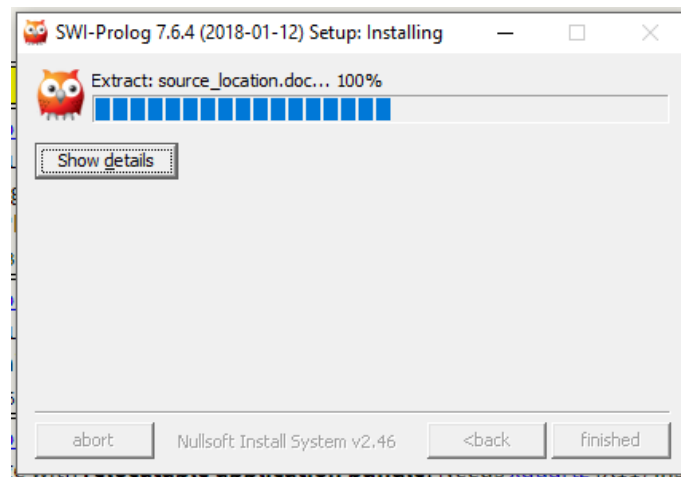
Instalación SWI-Prolog Parte 3

Destino de la carpeta donde se guardará



Instalación SWI-Prolog Parte 4

Extensión de cómo se guardará



Instalación SWI-Prolog Parte 5

Comienza instalación y una vez terminada solo seleccionamos finished.

Practicas Realizadas en clase

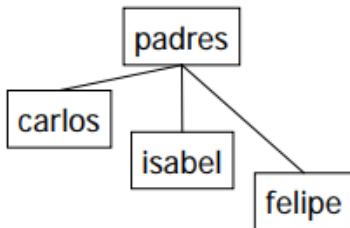
A continuación, se mostrarán practicas realizadas en clase respecto a Prolog ya enfocado en los puntos del temario.

- Estructura de árboles
- Listas
- Operaciones con Listas
- Pertenencias
- Recursión
- Concatenar
- Satisfacción de objetivos
- Predicados

Estructura y Árboles

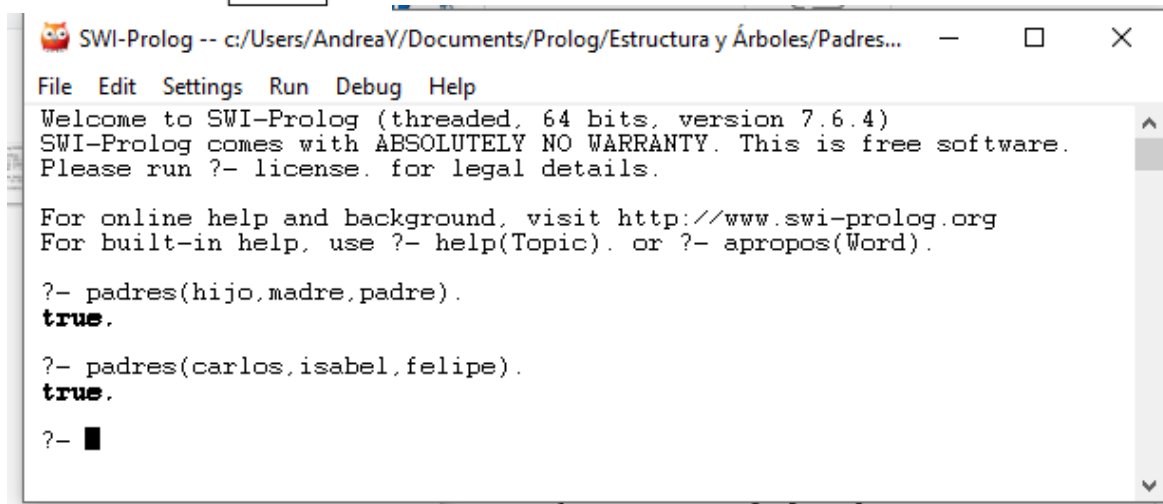
[4] Una estructura de datos en PROLOG se representa mediante un árbol.

`padres(hijo, madre, padre)`
`padres(carlos, isabel, felipe)`



- Entonces en este ejemplo podemos ver un pequeño árbol familiar con su jerarquía. Al insertar estas líneas de código en un notepad y guardarlo con cualquier nombre y la terminación `pl`, para ejecutarlo en Prolog SWI.

“ Imagen 01 – Estructura y Arboles ”



Listas

[5] Una lista es una secuencia “ordenada” de términos (constantes, variables, estructuras e, incluso otras listas).

- Usos de Listas: análisis sintáctico, gramáticas, diagramas de flujo, grafos, etc.
- Manejo específico de listas -> LISP.
- La lista es un caso particular de estructuras en PROLOG => definido recursiva.

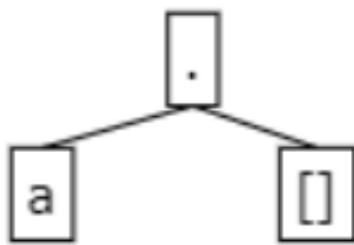
Tipo particular de árbol: cabeza y cola

La estructura asociada es “.”

El final de la lista es “[]”

Listas de un solo elemento:

. (a, [])

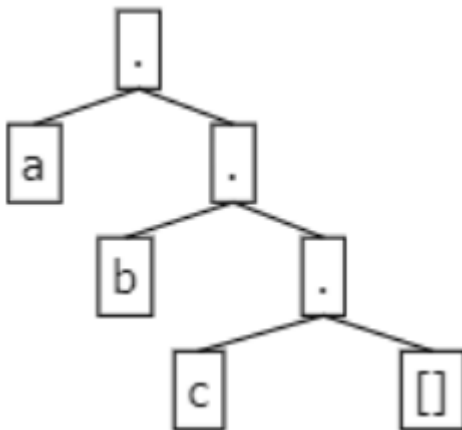


Representación habitual de las listas:

[a,b,c]

[los, hombres, [van, a, pescar]]

Como se puede ver, la cabeza es el primer término. Y la cola es una lista que contiene al resto. Una manera de instanciar a una lista con cabeza X y cola Y sería: [X|Y]



Lista: a,b,c

.(a, .(b, .(c, [])))

Imagen 02 – “Representación de Listas”

Pertenencia

[6] Saber si un objeto pertenece a lista.

[carlos_i, felipe_ii, felipe_iii, felipe_iv, carlos_ii]

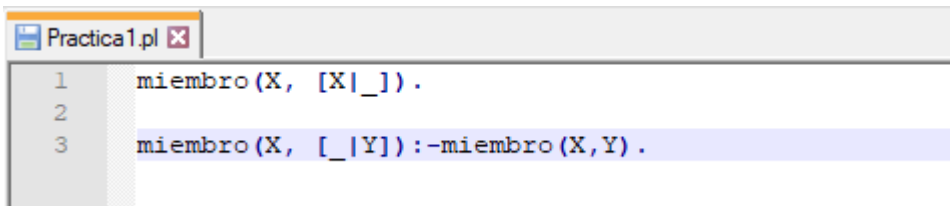
Construir el predicado “miembro”:

Miembro(X,[X|_]). (miembro(X,[Y|_]):-X=Y.)

Solo comprueba coincidencia con la cabeza

Miembro(X,[_|Y]):-miembro(X,Y).

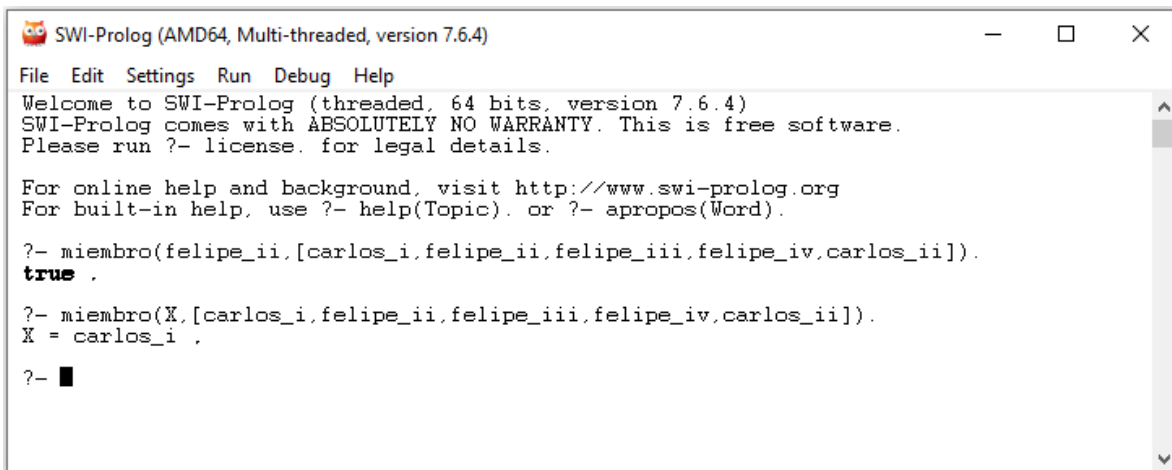
Verifica la coincidencia a la cola y de forma recursiva va operando sobre otra lista progresivamente más pequeña.



```
1  miembro(X, [X|_] ).  
2  
3  miembro(X, [_|Y]) :- miembro(X,Y) .
```

A continuación, en la consulta nos muestra la lista de miembros, donde preguntamos si felipe_ii, es parte de esa lista, lo cual así es y por eso el resultado es “true”.

En la segunda consulta es donde en la condición estamos pidiendo que nos arroje el primer miembro de la lista.



```
SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)  
File Edit Settings Run Debug Help  
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit http://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?- miembro(felipe_ii,[carlos_i,felipe_ii,felipe_iii,felipe_iv,carlos_ii]).  
true .  
  
?- miembro(X,[carlos_i,felipe_ii,felipe_iii,felipe_iv,carlos_ii]).  
X = carlos_i .  
  
?-
```

Imagen 03 – “Pertenencia”

Pertenencia II

- Para evitar definiciones circulares:

padre(X,Y):-hijo(Y,X).

hijo(A,B):-padre(B,A).

Esto entra en un bucle infinito)

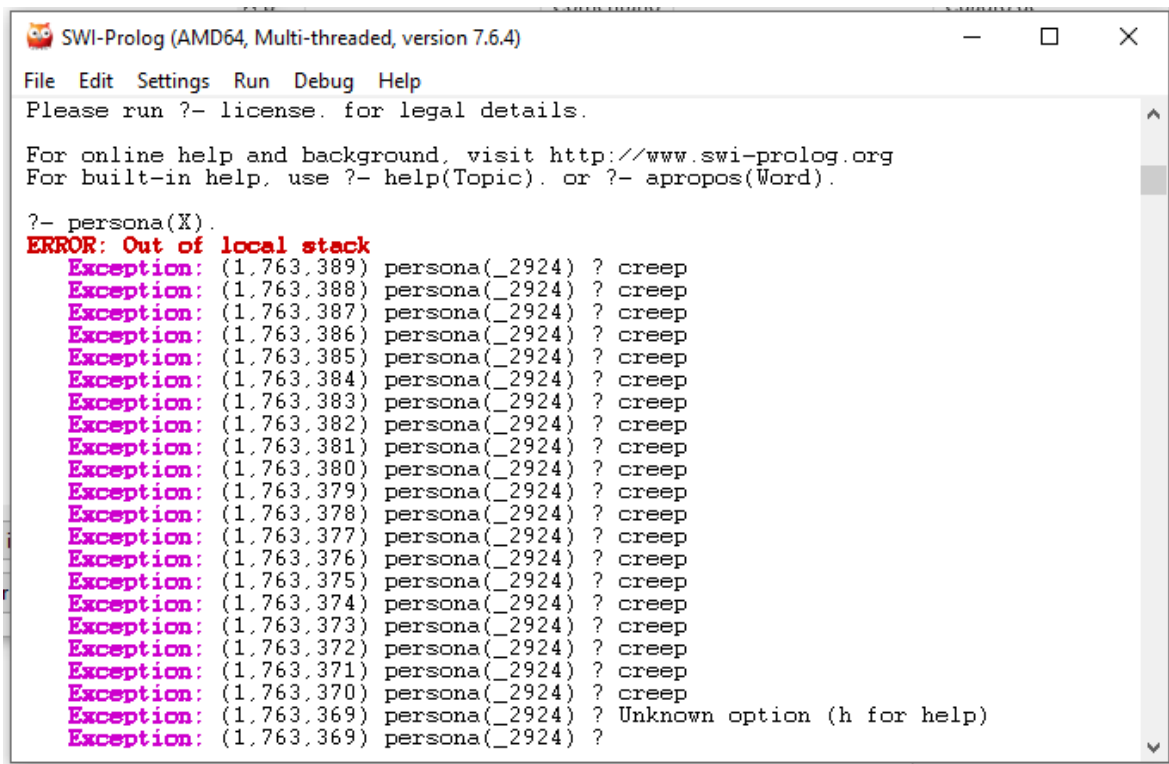
- Recursión por la izquierda (Siempre saldrá ERROR).

Ya que se entra en un bucle infinito.

persona(X):-persona(Y),madre(X,Y).

persona(adan).

?- persona(X).



```
SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)
File Edit Settings Run Debug Help
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- persona(X).
ERROR: Out of local stack
Exception: (1,763,389) persona(_2924) ? creep
Exception: (1,763,388) persona(_2924) ? creep
Exception: (1,763,387) persona(_2924) ? creep
Exception: (1,763,386) persona(_2924) ? creep
Exception: (1,763,385) persona(_2924) ? creep
Exception: (1,763,384) persona(_2924) ? creep
Exception: (1,763,383) persona(_2924) ? creep
Exception: (1,763,382) persona(_2924) ? creep
Exception: (1,763,381) persona(_2924) ? creep
Exception: (1,763,380) persona(_2924) ? creep
Exception: (1,763,379) persona(_2924) ? creep
Exception: (1,763,378) persona(_2924) ? creep
Exception: (1,763,377) persona(_2924) ? creep
Exception: (1,763,376) persona(_2924) ? creep
Exception: (1,763,375) persona(_2924) ? creep
Exception: (1,763,374) persona(_2924) ? creep
Exception: (1,763,373) persona(_2924) ? creep
Exception: (1,763,372) persona(_2924) ? creep
Exception: (1,763,371) persona(_2924) ? creep
Exception: (1,763,370) persona(_2924) ? creep
Exception: (1,763,369) persona(_2924) ? Unknown option (h for help)
Exception: (1,763,369) persona(_2924) ?
```

Imagen 04 – “Recursión (Bucle)”

“Práctica Conversación con Prolog!”

Diálogo:

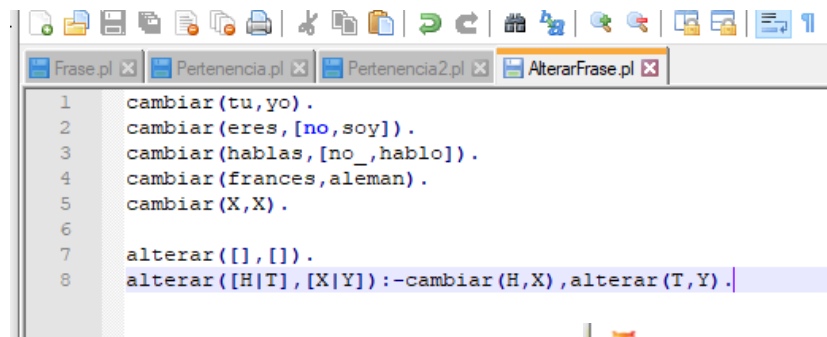
- Usuario: tu eres un ordenador
- Prolog: yo no soy un ordenador
- Usuario: hablas frances
- Prolog: no_hablo alemán

Reglas:

- Aceptar frase en formato lista:

[tu, eres, un ordenador]

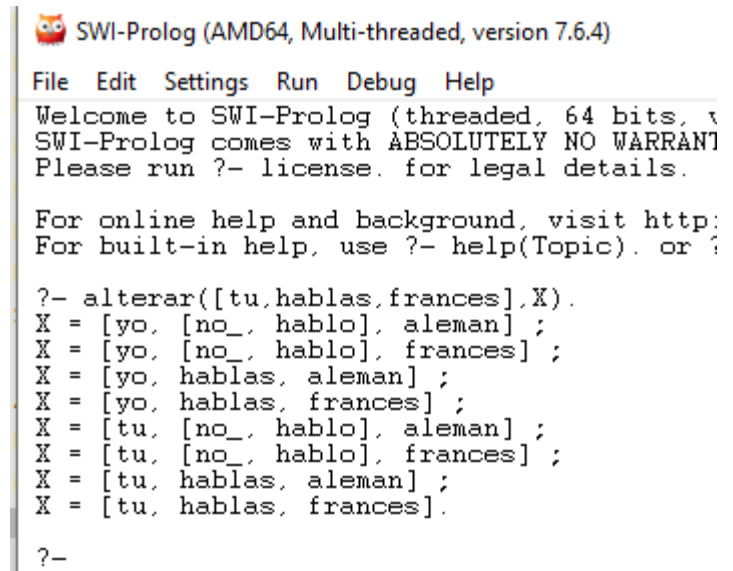
- Cambiar cada eres por un no soy.
- Cambiar cada habla por un no_hablo.
- Cambiar cada frances por un aleman



```
1  cambiar(tu,yo).
2  cambiar(eres,[no,soy]).
3  cambiar(hablas,[no_,hablo]).
4  cambiar(frances,aleman).
5  cambiar(X,X).
6
7  alterar([],[]).
8  alterar([H|T],[X|Y]):-cambiar(H,X),alterar(T,Y).
```

Archivo en notepad con las reglas.

Prolog nos muestra el resultado de la pregunta hecha al terminar el punto y las posibles contestacones al seguir con un punto y coma.



```
SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, x86_64)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?

?- alterar([tu,hablas,frances],X).
X = [yo, [no_, hablo], aleman] ;
X = [yo, [no_, hablo], frances] ;
X = [yo, hablas, aleman] ;
X = [yo, hablas, frances] ;
X = [tu, [no_, hablo], aleman] ;
X = [tu, [no_, hablo], frances] ;
X = [tu, hablas, aleman] ;
X = [tu, hablas, frances].

?-
```

Imagen 05 – “Practica Conversación Prolog”

Recursión

¿Cómo insertar un elemento en una lista en Prolog?

Si queremos introducir algún elemento en alguna parte de una lista, por ejemplo, en el primer lugar, este tiene que ser la cabeza de nuestra lista.

Como si realizamos,

Definición:

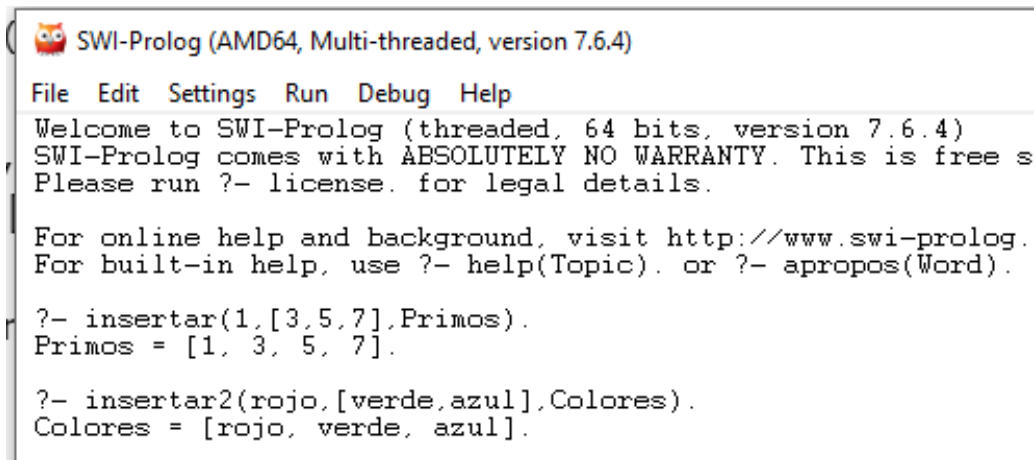
`insertar(X,lista,resultado):-resultado = [X|Lista].`

`insertar2(X,lista,[X|Lista]).`

Ejemplos:

En el ejemplo uno podemos observar como “1” está fuera de la lista, pero con la condición del resultado que deseamos que nos arroje `X = “1”`, esta anexado a la lista final mostrando que los numero Primos son los [1,3,5,7].

En el ejemplo dos podemos observar como en un ejemplo similar mostramos fuera de la lista un color “Rojo”, este nuevamente será nuestra X, y de igual manera pedimos que nos imprima una lista de colores, el resultado son los dos colores dentro de la lista y nuestra cabeza que esta fuera de esta siendo el Rojo el color, mostrando al final [rojo, verde, azul].



```
SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free s
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- insertar(1,[3,5,7],Primos).
Primos = [1, 3, 5, 7].

?- insertar2(rojo,[verde,azul],Colores).
Colores = [rojo, verde, azul].
```

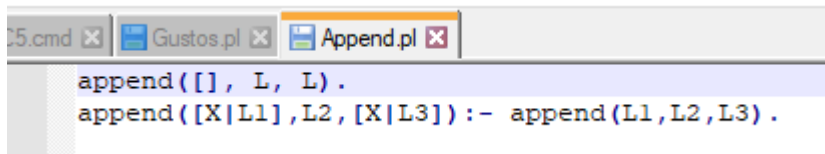
Imagen 05 – “Concatenar, Listas”

Concatenar

Predicado predefinido **append**, el cual nos sirve para concatenar en Prolog.

Ejemplo:

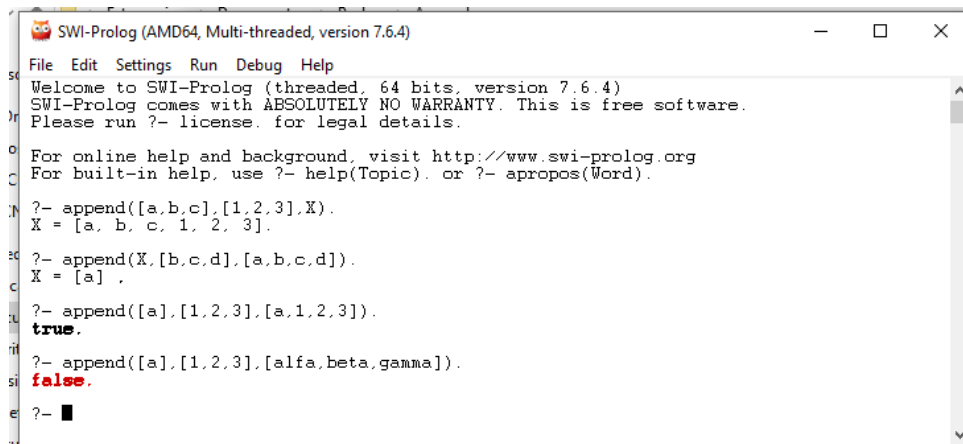
Condiciones en Notepad.



```
append([], L, L).  
append([X|L1], L2, [X|L3]) :- append(L1, L2, L3).
```

- En la primera pregunta mostramos dos listas las cuales queremos concatenar y como resultado nos muestra una sola lista con las letras de ambas listas.
- En la segunda mostramos como es que tenemos nuestra X como cabeza de las listas fuera de dos listas diferentes. Como resultado se nos muestra la letra “a”, que es la cabeza de la lista uno L1.
- En la tercera mostramos tres listas diferentes donde X es nuestra a L1 de una de las listas según la condición, en la pregunta solamente realizamos la pregunta como si esto fuera verdad y nos imprime un “True”.
- En la cuarta mostramos como es que ahora tenemos tres listas diferentes, una con una letra, la segunda lista con tres números y la tercera lista con tres palabras, de igual manera como en la pregunta tres, hacemos una pregunta como si fuera verdad lo que estamos estableciendo, solamente que esta vez nos regresa un “False”, porque alfa no es nuestra cabeza de la lista.

En ninguna de las preguntas se dejó de concatenar tanto en la respuesta como en la pregunta por las condiciones establecidas.



```
SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)  
File Edit Settings Run Debug Help  
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit http://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?- append([a,b,c],[1,2,3],X).  
X = [a, b, c, 1, 2, 3].  
  
?- append(X,[b,c,d],[a,b,c,d]).  
X = [a].  
  
?- append([a],[1,2,3],[a,1,2,3]).  
true.  
  
?- append([a],[1,2,3],[alfa,beta,gamma]).  
false.  
?-
```

Imagen 06 – “Practica Concatenar”

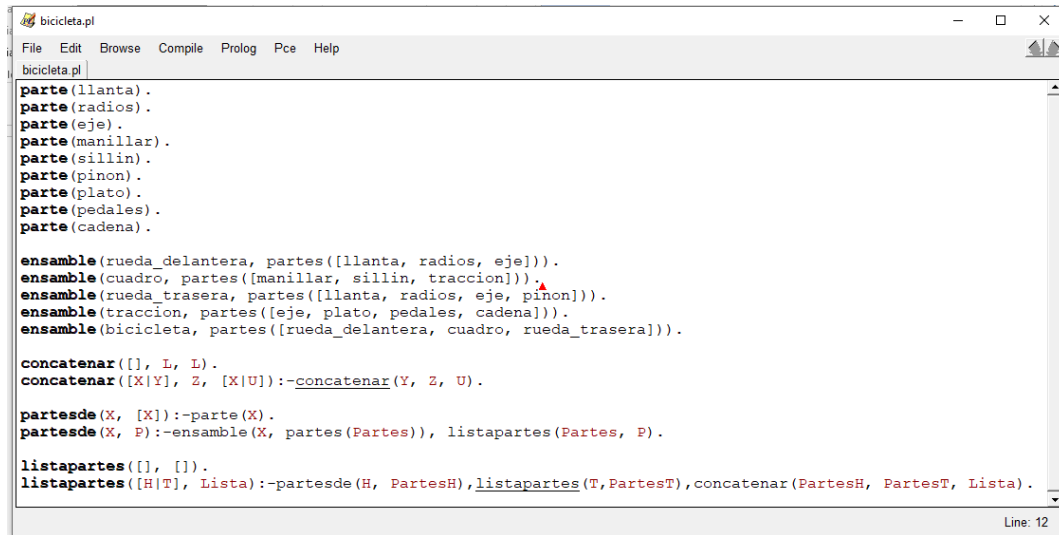
Practica lista de partes

- Definir árbol mediante las relaciones:

parte(cadena).

ensamble(bicicleta, [rueda_delantera, cuadro, rueda_trasera]).

- Construir relaciones “partes”, que sirva para obtener la lista de artes para construir un ensamble (o toda) la bicicleta.



```
bicicleta.pl
File Edit Browse Compile Prolog Pce Help
bicicleta.pl
parte(llanta).
parte(radios).
parte(eje).
parte(manillar).
parte(sillin).
parte(pinion).
parte(plato).
parte(pedales).
parte(cadena).

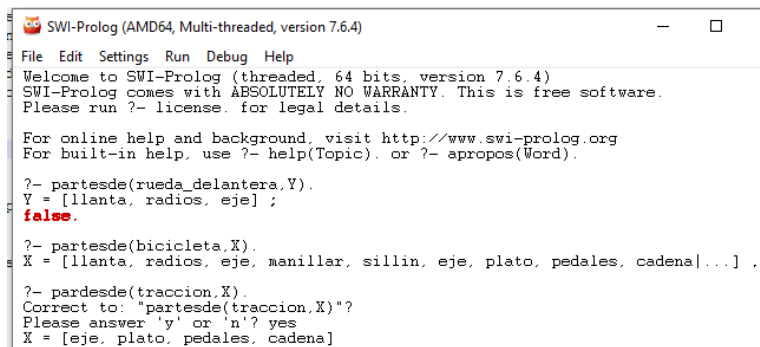
ensamble(rueda_delantera, partes([llanta, radios, eje])).
ensamble(cuadro, partes([manillar, sillin, traccion])).
ensamble(rueda_trasera, partes([llanta, radios, eje, pinon])).
ensamble(traccion, partes([eje, plato, pedales, cadena])).
ensamble(bicicleta, partes([rueda_delantera, cuadro, rueda_trasera])).

concatenar([], L, L).
concatenar([X|Y], Z, [X|U]):-concatenar(Y, Z, U).

partesde(X, [X]):-parte(X).
partesde(X, P):-ensamble(X, partes(Partes)), listapartes(Partes, P).

listapartes([], []).
listapartes([H|T], Lista):-partesde(H, PartesH), listapartes(T, PartesT), concatenar(PartesH, PartesT, Lista).
```

A continuación, se muestra el editor Prolog donde se muestra “parte” donde podemos observar las partes que tiene la bicicleta, después “ensamble” que muestra por medio de listas las partes de la bicicleta, pero por categoría, separando cada parte, el “concatenar” muestra cómo se unirán estas categorías dependiendo de las listas ordenadas, “partesde” nos muestra como vinculamos ensamble con las partes de la bicicleta, con nuestras partes dividida por categorías también en “listapartes”. De esta manera le podemos preguntar a Prolog sobre una categoría de la bicicleta, mencionándole cierta categoría y la letra de la lista que contiene ciertas partes, entonces Prolog nos regresara una lista con las partes que la bicicleta tiene en esa categoría de igual manera si nosotros le preguntamos sobre una letra (lista), que no corresponde a esa categoría por así decirse, Prolog nos regresa una lista con “false” o “true” dependiendo de la lista.



```
SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- partesde(rueda_delantera,Y).
Y = [llanta, radios, eje] ;
false.

?- partesde(bicicleta,X).
X = [llanta, radios, eje, manillar, sillin, eje, plato, pedales, cadena] ;

?- partesde(traccion,X).
Correct to: "partesde(traccion,X)"?
Please answer 'y' or 'n'? yes
X = [eje, plato, pedales, cadena]
```

Imagen 07 – “Practica Bicicleta”

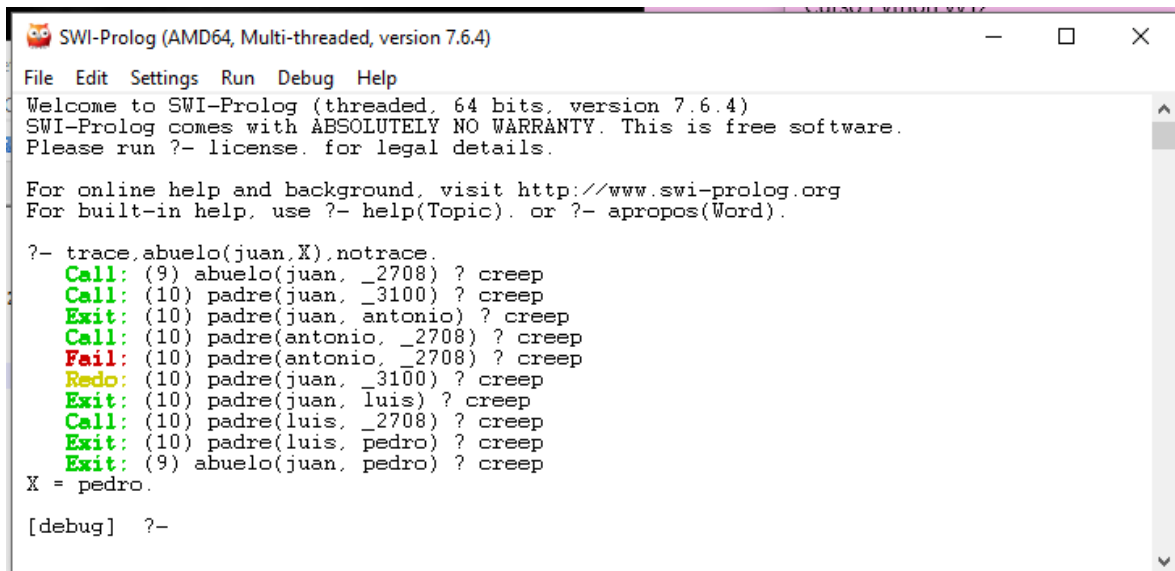
Satisfacción de Objetivos

A continuación, utilizaremos “trace”/”no trace”, el cual funciona como rastreador de objetivos. “Trace” enciende el objetivo y comienza a buscar y “notrace” lo detiene.

Escribimos en una base de datos pequeña donde se muestra una familia. Nosotros queremos preguntar quién es el abuelo de Juan. Al preguntar en Prolog empezando la pregunta colocamos “trace”, seguido de una coma y la pregunta, terminamos con una coma y cerramos nuestro trace, con un “notrace” y un punto. Esto nos mostrara los posibles resultados en la búsqueda de trace.

```
padre(juan,antonio).  
padre(juan,luis).  
padre(luis,pedro).  
padre(luis,ana).
```

```
abuelo(X,Y):-padre(X,Z),padre(Z,Y).
```



```
SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)  
File Edit Settings Run Debug Help  
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit http://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?- trace,abuelo(juan,X),notrace.  
Call: (9) abuelo(juan, _2708) ? creep  
Call: (10) padre(juan, _3100) ? creep  
Exit: (10) padre(juan, antonio) ? creep  
Call: (10) padre(antonio, _2708) ? creep  
Fail: (10) padre(antonio, _2708) ? creep  
Redo: (10) padre(juan, _3100) ? creep  
Exit: (10) padre(juan, luis) ? creep  
Call: (10) padre(luis, _2708) ? creep  
Exit: (10) padre(luis, pedro) ? creep  
Exit: (9) abuelo(juan, pedro) ? creep  
X = pedro.  
[debug] ?-
```

Imagen 08 – “Satisfacción de Objetivos (trace/notrace)”

Predicado Corte: !

Este predicado nos muestra el flujo de control, evita explorar ramas del árbol de búsqueda, eficiencia, con un correcto funcionamiento, aunque altera el significado declarativo del programa. El predicado corte siempre se satisface, definición operacional, meta padre: meta que redujo con la cláusula que contiene el predicado corte, si se alza el predicado corte se satisface eliminando de la meta, si se vuelve al predicado corte (backtracking), devuelve control da la meta anterior a la meta padre. Obliga a mantener todas las elecciones realizadas entre la meta padre y el corte, pues las posibles alternativas no se consideran.

Por ejemplo:

En una biblioteca tenemos lo que son:

- Libros existentes
- Libros prestados y a quién.
- Fecha de devolución de préstamos

Servicios básicos (accesibles a cualquiera):

- Biblioteca de referencias o mostrador de consulta

Servicios adicionales(regla):

- Préstamo normal o interbibliotecario.

Regla: No permitir servicios adicionales a personas con libros pendientes de devolver.



```
predicado.pl
File Edit Browse Compile Prolog Pce Help
predicado.pl
libros_por_devolver(c_perez,libro10089).
libros_por_devolver(a_ramos,libro29907).

cliente(a_ramos).
cliente(c_perez).
cliente(p_gonzalez).

servicio_basico(referencia).
servicio_basico(consulta).

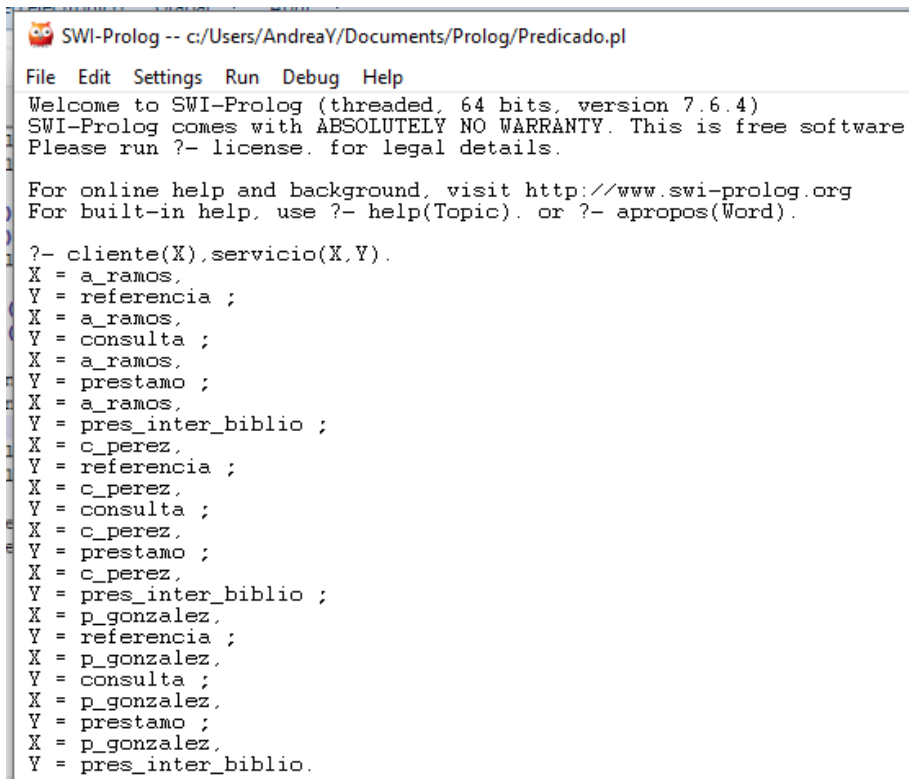
servicio_adicional(prestamo).
servicio_adicional(pres_inter_biblio).

servicio_general(X):-servicio_basico(X).
servicio_general(X):-servicio_adicional(X).

servicio(Pers,Serv):-cliente(Pers),libros_por_devolver(Pers,libro),servicio_basico(Serv).
servicio(Pers,Serv):-cliente(Pers),servicio_general(Serv).
```

En la captura de pantalla podemos observar como tenemos nuestros libres_por_devolver, nuestros clientes, servicio_basico, servicio_adicional, servicio_general y el servicio de la condición de clientes con los libros.

A continuación, podemos observar como en la pregunta tenemos el cliente X con el servicio que le pertenece, Prolog no arroja a los clientes con su respectivo servicio.



```
SWI-Prolog -- c:/Users/AndreaY/Documents/Prolog/Predicado.pl
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- cliente(X),servicio(X,Y).
X = a_ramos,
Y = referencia ;
X = a_ramos,
Y = consulta ;
X = a_ramos,
Y = prestamo ;
X = a_ramos,
Y = pres_inter_biblio ;
X = c_perez,
Y = referencia ;
X = c_perez,
Y = consulta ;
X = c_perez,
Y = prestamo ;
X = c_perez,
Y = pres_inter_biblio ;
X = p_gonzalez,
Y = referencia ;
X = p_gonzalez,
Y = consulta ;
X = p_gonzalez,
Y = prestamo ;
X = p_gonzalez,
Y = pres_inter_biblio.
```

Imagen 09 – “Corte: !”

Resultados

La instalación como cualquier otra en Windows como SO fue sencilla, tuve algunos problemas al comenzar guardando los archivos de notas, ya sea por la sintaxis de los hechos o por no guardarlo como “.pl”, al acostumbrarme cómo funcionaba el abrir primero los archivos en Prolog y después hacer una pregunta inteligente relacionada con cómo había escrito dicho programa en el notepad. Aunque una vez que se domina como compilar, ya solo lo más difícil es pensar cómo solucionar los problemas y además el cómo escribir la sintaxis correcta, porque de ella depende mucho el cómo es que se va a preguntar para que te regrese un resultado como uno lo tiene pensado o simplemente para que te regrese la contestación que el problema espera. Si es bastante complicado, mucho más que Haskell, varias veces tuve que recurrir a internet para resolver los ejercicios porque no me salían las listas como yo imaginaba que estaban bien. Cuando avanzaba algo de alguno de los ejemplos siempre que veía como sería uno de los resultados era completamente diferente y sentía que volvía desde cero y nuevamente no entendía nada.

Conclusión

A pesar de tanto buscar resultados y leer en internet para poder resolver los ejercicios realizados en clase, tengo la idea de en qué está enfocado Prolog, siento que, si comprendí, aunque sea lo básico y la sintaxis, aunque es un lenguaje completamente diferente a lo que normalmente he acostumbrado si me di topes en la pared porque se le tiene que pensar de una manera diferente para poder resolver el problema, siento que a pesar de que no es mi lenguaje favorito, la unidad de la materia con este lenguaje cumplió su cometido de conocer Prolog y al menos lo básico y la idea me quedo.

Bibliografía

- [1] Introducción a Prolog, *Dicha fuente de información fue proporcionada de <https://www.ecured.cu/Prolog>*
- [2] **Enunciado de las actividades.** *Dicha fuente de información de toda introducción a Prolog, sintaxis, reglas, hechos y demás que se proporcionó en el desarrollo de este reporte fue proporcionada por página con nombre “**Inteligencia Artificial 2**” – **Lenguaje de Programación Prolog**. Disponible en: <https://ia2gerardolloor.blogspot.mx/2014/12/lenguaje-de-programacion-prolog.html>*
- [3] **Estructura y Árboles.** *Dicha fuente de información sobre la explicación de Estructura de datos en Prolog fue proporcionada por diapositivas de la página: <http://docplayer.es/39955088-Estructuras-de-datos-y-listas-en-prolog-1-estructuras-y-arboles-2-listas-3-operaciones-con-listas-4-ejercicios-5-practicas.html>*
- [4] **Listas.** *Dicha fuente de información sobre la explicación de Listas en Prolog fue proporcionada por diapositivas de la página: <http://docplayer.es/39955088-Estructuras-de-datos-y-listas-en-prolog-1-estructuras-y-arboles-2-listas-3-operaciones-con-listas-4-ejercicios-5-practicas.html>*
- [5] **Pertenencia.** *Dicha fuente de información sobre la explicación de Listas en Prolog fue proporcionada por diapositivas de la página: <http://docplayer.es/39955088-Estructuras-de-datos-y-listas-en-prolog-1-estructuras-y-arboles-2-listas-3-operaciones-con-listas-4-ejercicios-5-practicas.html>*
- [6] **Reducción, Concatenar y Corte.** *Dicha fuente de información sobre la explicación de estos temas en Prolog fue proporcionada por diapositivas de la página: <https://www.infor.uva.es/~calonso/Ingenieria%20ConocimientoGrado%20Informatica/Practicas/Practica%20I%20Prolog.pdf>*

Instalación de Prolog: **SWI Prolog**, página oficial de Prolog <http://www.swi-prolog.org/>

Páginas extras donde leí ideas y significados diferentes de los subtemas para el desarrollo de toda información sobre Prolog proporcionada en este reporte.

- **Prolog Tutorial:** https://www.cpp.edu/~jrfisher/www/prolog_tutorial/intro.html
- **Wikipedia:** <https://en.wikipedia.org/wiki/Prolog>
- **EcuRed:** [https://www.ecured.cu/Prolog_\(Lenguaje_de_programaci%C3%B3n\)](https://www.ecured.cu/Prolog_(Lenguaje_de_programaci%C3%B3n))
- **Introducción a Prolog:** https://www.cs.us.es/~jalonso/pub/2006-int_prolog.pdf
- **Oefa.blogspot.mx:** <https://oefa.blogspot.mx/2008/04/programacion-logica-swi-prolog.html>