****

**Pontificia Universidad Católica Madre y Maestra**

**Facultad de Ciencias de las Ingenierías**

**Departamento de Ingeniería de Sistemas y Computación**

**Proyecto Final**

Asignatura:

Programación lógica

Profesora:

Lisibonny Beato

Presentado por:

Stanley De Moya 2012-0412

Ansël E. Corona Coste 2014-0031

Indice:

* Presentación………………………………………………….1
* Indice……………………………………………………………..2
* Introducción…………………………………………………..3
* Desarrollo……………………………………………………….4
* Retos a Superar……………………………………………..10
* Conclusión……………………………………………………..11
* Bibliografía……..……………………………………………..12

Introducción:

Wikipedia define un Sistema Experto como: “Los sistemas expertos son llamados así porque emulan el razonamiento de un experto en un dominio concreto, y en ocasiones son usados por éstos. Con los sistemas expertos se busca una mejor calidad y rapidez en las respuestas, dando así lugar a una mejora de la productividad del propio experto al usar este tipo de sistemas informáticos.”

En este reporte hemos desarrollado un sistema experto de sugerencias que basado en filtros y en un presupuesto filtra los contenidos del menu de un restaurante y muestra sugerencias sobre que el usuario debería ordenar.

Desarrollo

Este proyecto esta basado en Maikito’s by De Moya, un restaurante localizado en San Francisco de Macorís, República Dominicana, el cual sirve una amplia gama de platos y bebidas con diferentes categorías que veremos más adelante.

Inicialmente se pensó tener hechos separados para los platos y las bebidas, sin embargo luego de analizar y formalizar se llegó a la conclusión de que sería mejor tenerlas todas en un mismo hecho (bajo la recomendación de la instructora).

En fin, comenzamos con un hecho “plato/3”, que recibe los siguientes parametros:

* Código
* [Ingredientes]
* Precio

Esto, luego asociamos el código de cada plato con su respectivo nombre en el hecho “codigo/2”, que recibe como primer parametro el código del plato y como segundo parametro su nombre.

Originalmente el hecho plato tenía 5 argumentos (Nombre, tipo\_plato, [Ingredientes], tipo\_plato2, precio), sin embargo se notaba cierta redundancia cuando se hablaba de ambos tipos de platos y se prefirió modificar el hecho y retirar de este el “tipo\_plato” y el “tipo\_plato2” y se creo así otro hecho, “tipo\_plato/2” que asocia el código de un plato con su respectivo tipo que puede ser: desayuno, almuerzo, cena o bebida, todo esto para simplificar el codigo fuente y alcanzar la maxima legibilidad posible, cabe también destacar que el código fuente ha sido comentado de una manera que cualquier persona que solo cuente con los conocimientos básicos de ProLog pueda ingresar nuevos platos.

Se tiene otro hecho denominado “ingrediente/3” que tiene como parámetros a:

* Nombre
* Tipo
* Tipo\_alergia (en caso de que sea un alergénico común como es el caso de la lactosa, el gluten o el huevo).

Nuestro programa cuenta con varias reglas de inferencia que filtran los alimentos para brindar una sugerencia que se aproxime lo más posible a lo que el cliente pueda consumir, por lo que se han ideado 6 filtros:

* GlutenFree.
* Vegetariano.
* EggFree.
* LactoseFree.
* SugarFree.
* AlcoholFree.

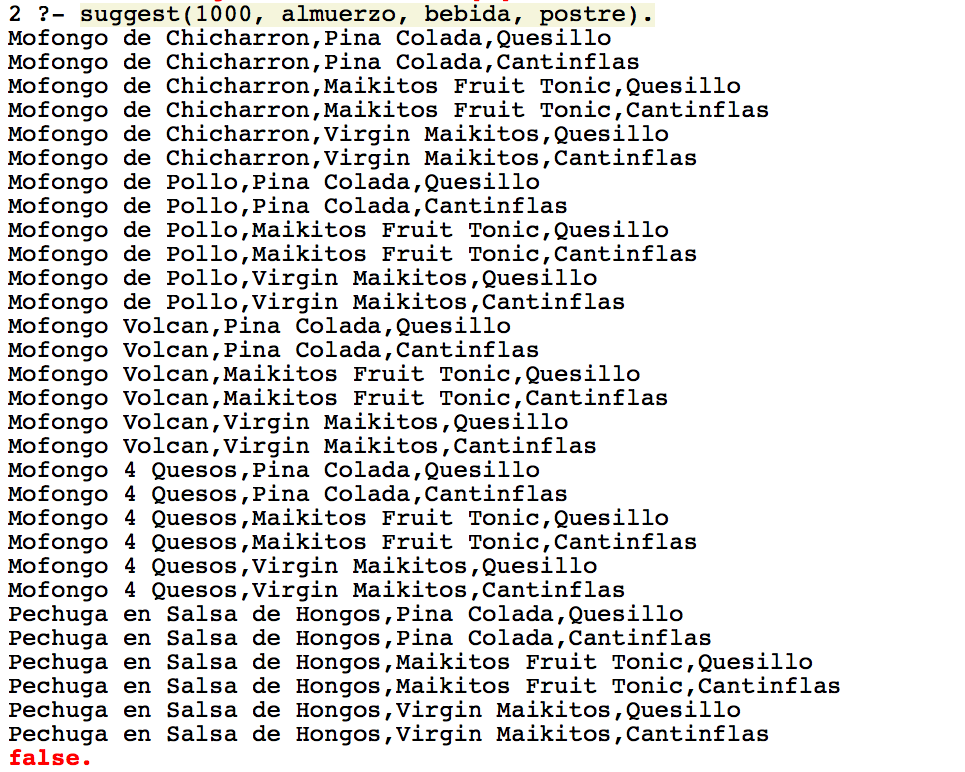
Con nombres bastantes representativos de sus funciones, reciben una instancia de una Lista y retornan una Lista con los platos que cumplan la condicion establecida, por ejemplo, si le paso una lista L = [] al filtro Vegetariano, me va a retornar en L todos los platos que no tengan carne ni huevo (puesto a que muchos vegetarianos no consumen huevos, lo cual es debatible pero para los propositos de este reporte se ha asumido que ningun vegetariano consume huevos), todos los demás filtros muestran un comportamiento similar, pues reciben una lista vacía y retornan una Lista filtrada.

La parte interesante es aquella en la que tenemos las reglas suggest/2, suggest/3 y suggest/4 que reciben como parametros (Presupuesto, Comida), (Presupuesto, Comida1, Comida2), (Presupuesto,Comida1,Comida2,Comida3) respectivamente.

La estructura de estas reglas es generar sugerencias a partir de los parametros establecidos, el punto de las mismas es responder a la pregunta siguiente:

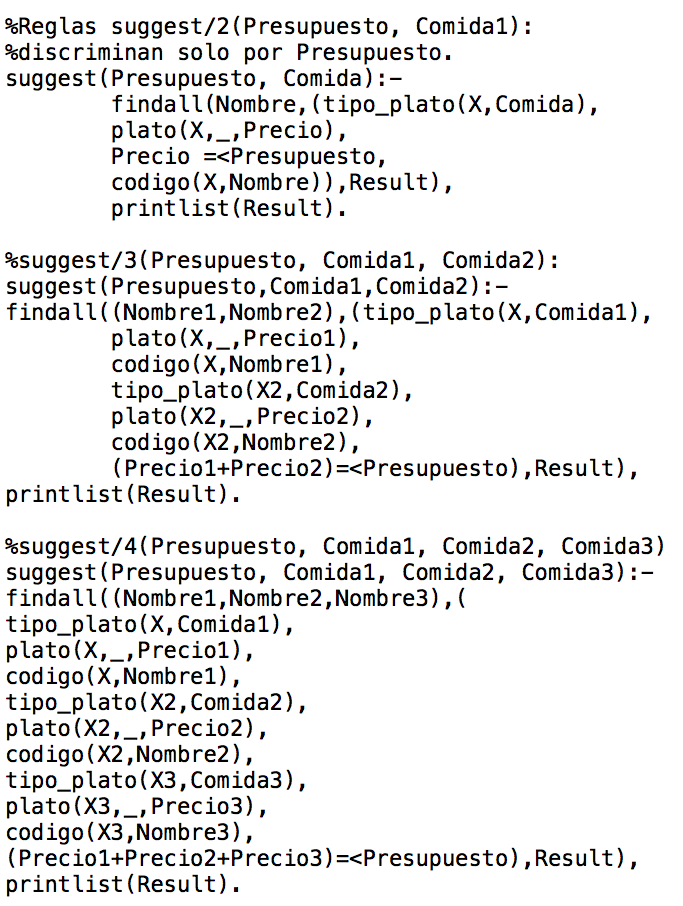
**“¿Que puedo comer hoy si tengo X pesos?”**

y sus variantes, “Pero quisiera poder almorzar y ademas comerme un postre, y tomarme una bebida tambien, mas solo tengo 1000 pesos”, entonces en un caso como este ejecutamos un query sobre la base de conocimientos que iría de la siguiente forma “?- suggest(1000, almuerzo, bebida, postre).” A lo que nuestro SWI-ProLog respondería:



En términos combinatorios consigue la permutacion entre las opciones que caen en los renglones seleccionados, el total de respuestas será igual a la permutacion de los diferentes elementos, algo como Total = (5[numero de comidas]\*3[numero de bebidas]\* 2[numero de postres]), para un total de 30.

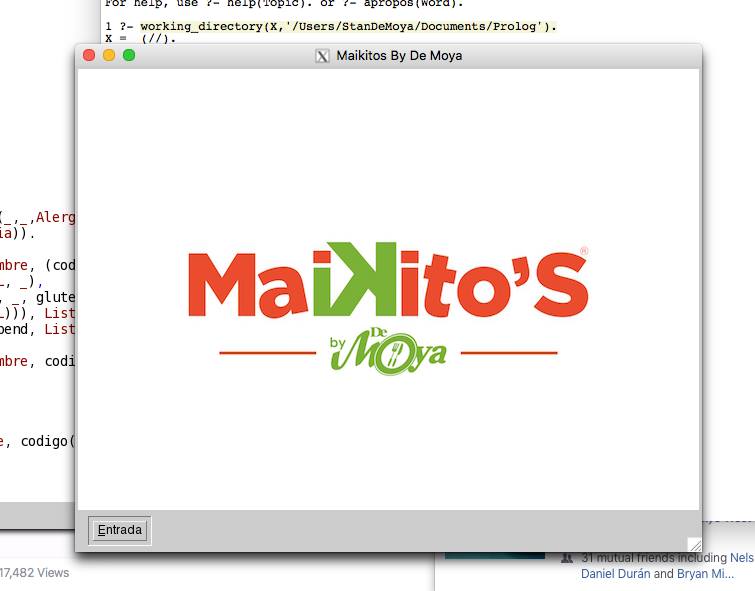
Esta función logra su cometido gracias a la regla nativa “findall/3”, con la cual adquirimos la lista de combinaciones y la imprimimos al usuario, el código de las funciones suggest/2,3,4 va de la siguiente forma:



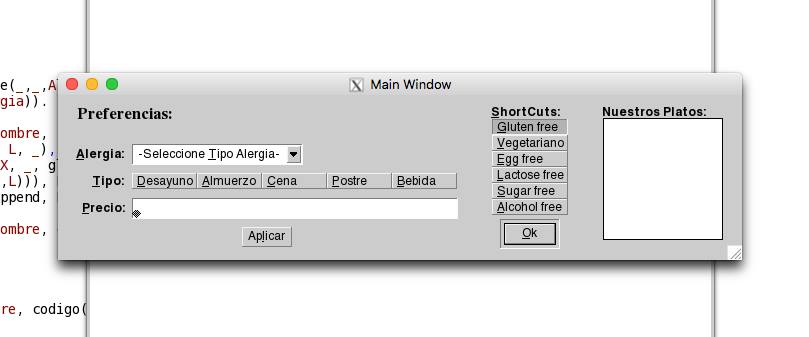
Que encuentra los nombres de los platos cuya sumatoria de precios sea menor o igual al presupuesto ingresado por el cliente, eso sin tomar en consideración los discriminantes por alergenicos que son los siguientes:



Hubo un problema con la implementacion de esta regla y es que originalmente esa regla debe filtrar los resultados entregados por la otra regla, para mostrarlos a los clientes, sin embargo el retorno de una Lista se complicó, en especial porque cuando se intentaba filtrar por el tipo de ingrediente se replicaban los elementos en la lista resultante, por lo que hubo que *tacklear* cada ingrediente de manera individual, que dificultaría la ingresión de nuevos platos, ya que de ingresar nuevos alergenos habría que identificar el ingrediente directamente.

Capturas de pantalla del gráfico:

Pantalla inicial.



Pantalla de los filtros

**Retos a Superar:**

1. Agregar categorías de personas, es algo que se ponderó pero no se llevo a cabo porque no pudimos llegar a un acuerdo sobre la lógica a implementar.
2. Agregar filtros eficientes.
3. Completar aquellas cosas que no se pudieron terminar.
4. Asertar a hechos dinamicos sugerencias ya realizadas, de manera que el acceso se facilite.
5. Tener un mejor manejo de otras herramientas de programación para de esa manera poder trabajar y entregar dentro de los *deadlines* establecidos.

Conclusión

Definitivamente es un reto para uno trabajar con herramientas que uno no maneja tan bien, como XPCE/Prolog, si bien la parte logica no se complica mucho, (pues se maneja con Prolog, que es un lenguaje con el que se tiene cierta afinidad), la parte visual se complica bastante principalmente por la falta de documentación respecto al tema, es sorprendente que tecnologías tan *explotables* como ProLog esten tan mal documentadas en la red, y mas la curva de aprendizaje tan alta que representa XPCE, supongo que podemos admitir que hemos subestimado el problema, lo cual de seguro afectará a la calificación a obtener en este reporte y subsecuentemente en la asignatura, lo cual es bien triste considerando lo mucho que se aprendió durante el poco tiempo que estuvimos interactuando con el lenguaje.

Bibliografía:

* <https://es.wikipedia.org/wiki/Sistema_experto>
* <http://stackoverflow.com/questions/10750316/prolog-xpce-dialog-return-a-list-of-selections>
* <https://es.wikipedia.org/wiki/Prolog>
* Catedras impartidas por la Instructora.
* Ayudas solicitadas a compañeros.