Tarea 2: Documentación

Paradigma Lógico: RestauranTEC

José Julián Camacho Hernández

Jose Fabián Mendoza Mata

Sergio Andrés Ríos Campos

Área de Ingeniería en Computadores, Instituto Tecnológico de Costa Rica

CE-3104: Lenguajes, Compiladores e Intérpretes

Profesor: Marco Rivera Meneses

14 de octubre de 2020

Tabla de contenidos

1. Breve descripción del proyecto

RestauranTEC es un sistema experto (SE) que se comporta como un humano. Se encarga de recomendar restaurantes a los usuarios que le indican sus preferencias y busca en sus cercanías el que más coincidencias tiene. Cuenta con una interfaz completamente natural utilizando el lenguaje español implementado utilizando BNF.

1.1. Descripción de los hechos y reglas implementadas.

Base de datos

1. Restaurantes: hechos que definen un restaurante por su nombre.
    - restaurante("Nombre del restaurante").

```
restaurante("Bella Italia").
restaurante("Italianisimo").
restaurante("McBurguesa").
```

2. Disposiciones: hecho que define las disposiciones comunes de los restaurantes.
    - disposiciones("Nombre del restaurante", “Disposición”).

```
disposiciones("Bella Italia",  "Solo se permiten
burbujas y durante la espera se debe utilizar
mascarilla").
disposiciones("Italianisimo", "Utilizar mascarilla").
disposiciones("McBurguesa", "Solo se permiten
burbujas").
```

3. Menús: hechos que definen el contenido de los menús de los restaurantes.

- mennu(“Nombre”, “Tipo de menú”, “[Comida | [Tipos específicos]]”).

```
mennu("Bella Italia", "Italiano", ["Pizza", ["Jamón y
Queso", "Suprema", "Hawaiana"], "Calzone",
"Espagueti"]).

mennu("Italianisimo","Italiano" ,["Pizza",
["Pepperoni"], "Calzone", "Espagueti"] ).


mennu("McBurguesa","Comida Rapida" ,["Hamburguesas",
"Tacos ", "papas"] ).
```

4. Pizza: hechos que definen los tipos de pizzas específicos de los restaurantes o los restaurantes que tienen pizza.

- pizza(“Nombre del restaurante”, “[Tipos específicos]”).

```
pizza("Bella Italia", ["Jamón y Queso", "Suprema",
"Hawaiana"]).

pizza("Italianisimo", ["Pepperoni"]).
```

- pizza(“Restaurante 1”, “Restaurante 2”).

```
pizza("Italianisimo","Bella Italia").
```

5. Comidarapida: hecho que define el tipos de comida rápida específica de los restaurantes.

- comidarapida(“Nombre del restaurante”, “[Tipos específicos]”).

```
comidarapida("McBurguesa",["Hamburguesas", "Tacos",
"papas"]  ).
```

6. Dirección: hechos que definen las direcciones de los restaurantes.

- direccion(“Nombre del restaurante”, “Dirección”).

```
direccion("Bella Italia","300m Sur de la entrada
principal de la Universidad Nacional" ).
```

```
direccion("Italianisimo", "50m Sur de la entrada Banco
de Costa Rica" ).

direccion("Bella Italia","100m Norte de la entrada
principal del TEC" ).
```

7. Lugar: hechos que definen el lugar donde se ubican los restaurantes.

- lugar(“Nombre del restaurante”, “Lugar donde se ubica”).

```
lugar("Bella Italia", "Heredia").

lugar("Italianisimo", "Alajuela").

lugar("McBurguesa", "Cartago").
```

8. Capacidad: hechos que definen la capacidad máxima de los restaurantes.

- capacidad(“Nombre del restaurante”, Capacidad máxima).

```
capacidad("Bella Italia", 10).

capacidad("Italianisimo", 5).

capacidad("McBurguesa", 20).
```

Gramáticas libres de contexto BNF

Análisis de oraciones relacionadas con alimentos

- alimento(Oración, Oración preliminar, Palabra(s) clave).

```
alimento(S0,S,Claves):-
    pronombre(Num,S0,S1),
    sintagma_verbal(Num,Estado,S1,S2),
    sintagma_nominal(_Gen2,Num,Estado,S2,S, Claves).
                  %Género, número, estado del nombre
alimento(S0,S, Claves):-
    sintagma_verbal(Num,Estado,S0,S1),
    sintagma_nominal(_Gen2,Num,Estado,S1,S, Claves).

alimento(S0,S, Claves):-
    sintagma_nominal(_Gen2,Num,Estado,S0,S, Claves).

alimento(_S0,_S,_Claves):-
    write("Lo sentimos, no se conoce algún restaurante con
ese tipo de comida"),
    nl, nl,
    restaurantec().
```

Análisis de oraciones relacionadas con ubicaciones

- ubicación(Oración, Oración preliminar, Palabra(s) clave).

```
ubicacion(S0,S,S1):-
    preposicion(S0,S1),
    lugares(_,S), !.

ubicacion(S0,S,S0):-
    lugares(_,S), !
```

Análisis de oraciones relacionadas con cantidad de personas

- personas(Oración, Oración preliminar, Palabra(s) clave).

```
personas(S0,S,S1):-
    preposicion(S0,[S1|_]),
    cantidad(_,S2),
    person(S2,S), !.

personas(S0,S,S1):-
    preposicion(S0,S1),
    cantidad(_,S), !.

personas(S0,S,S0):-
    cantidad(_,S1),
    person(S1,S), !.

personas(S0,S,S0):-
    cantidad(_,S), !.
```

Análisis de sintagmas

- sintagma_nominal(Género, Número, Estado, Oración preliminar, Oración, Palabra clave).

```
sintagma_nominal(Gen,Num,Estado,S0,S, S1):-
    determinante(Gen,Num,S0,S1),
    nombre(Gen,Num,Estado,S1,S2),
    adjetivo(Gen,Num,S2,S).

sintagma_nominal(Gen,Num,Estado,S0,S, S1):-
    nombre(Gen,Num,Estado,S0,S1),
    adjetivo(Gen,Num,S1,S).

sintagma_nominal(Gen,Num,Estado,S0,S, S1):-
    determinante(Gen,Num,S0,S1),
    nombre(Gen,Num,Estado,S1,S).

sintagma_nominal(Gen,Num,Estado,S0,S, S0):-
    nombre(Gen,Num,Estado,S0,S).
```

- sintagma_verbal(Género, Número, Estado, Oración preliminar, Oración, Palabra clave).

```
sintagma_verbal(Num,_Estado,S0,S):-verbo(Num,S0,S).
sintagma_verbal(Num,Estado,S0,S):-
    verbo(Num,S0,S1),
    infinitivo(Estado,S1,S).
```

Gramáticas de cláusulas definidas

- determinante(Género, Número, Determinante, Oración).

```
determinante(femenino, singular, [una|S],S).
determinante(femenino, plural, [unas|S],S).
determinante(masculino, singular, [un|S],S).
determinante(masculino, plural, [unos|S],S).
```

- pronombre(Número, Pronombre, Oración).

```
pronombre(singular,[yo|S],S).
```

- adjetivo(Género, Número, adjetivo, Oración).

```
adjetivo(femenino,singular,[rápida|S],S).
```

- nombre(Género, Número, Estado, Nombre, Oración).

```
nombre(masculino, singular, solido, [calzone|S],S).
nombre(masculino, singular, solido, [espagueti|S],S).
nombre(femenino, singular, solido, [pizza|S],S).
nombre(femenino, singular, solido, [comida|S],S).
nombre(femenino, singular, liquido, [bebida|S],S).
```

- lugares(_, Oración). Admite cualquier lugar

```
lugares([_|S],S).
```

- person(personas, Oración).

```
person([personas|S],S).
```

- cantidad(_, Oración). Admite cualquier cantidad

```
cantidad([_|S],S).
```

- verbo(Número, Verbo, Oración).

```
verbo(singular,[quiero|S],S).
```

- infinitivo(Estado, Infinitivo, Oración).

```
infinitivo(solido, [comer|S],S).
infinitivo(liquido, [tomar|S],S).
```

- preposición(Preposición Oración).

```
preposicion([en|S],S).
preposicion([para|S],S).
```

Análisis del input del usuario

Hecho base

```
parseInput([],[]).
```

Se hace una lista de las palabras ingresadas por el usuario como átomos

```
parseInput([C|InputList], [A|Result]):-
    atom_string(A,C),
    parseInput(InputList,Result).
```

Input es la entrada de texto del usuario R será la entrada en formato analizable

```
getInput(Input,R):-
    split_string(Input," ",".",R1),
    parseInput(R1,R).
```

1.2. Descripción de las estructuras de datos desarrolladas.

La principal estructura utilizada en el programa, son listas. Estas se emplearon para almacenar los datos relacionados a los restaurantes, tales como menú, dirección, lugar, capacidad y disposiciones. Por ejemplo:

pizza("Bella Italia", ["Jamón y Queso", "Suprema", "Hawaiana"]).

Donde este hace referencia a los tipos de pizza que ofrece un restaurante en particular, este hecho, contiene el nombre del restaurante y una lista con todos los tipos y opciones del alimento que este ofrece. Esta misma estrategia se aplicó a datos como los menús y los tipos de comida de cada restaurante, ya que facilita su búsqueda a la hora de correr el programa.

1.3. Descripción detallada del algoritmo de solución desarrollado.

El algoritmo utilizado se enfoca en el sistema experto, el cual hará una serie de preguntas al usuario, para ir " Filtrando " los restaurantes que puede recomendar al usuario. Se ideó un plan para hacer la recomendación de forma ordenada y precisa siguiendo un plan : Filtrar el restaurante a recomendar y luego aplicar validaciones:

- Filtrar el restaurante a recomendar:
    - Primero se le pregunta al usuario qué desea comer, esto para tener información general del tipo de comida a recomendar
    - Luego se le pregunta qué tipo de comida específico desea comer, esto para revisar si en el menú de los posibles restaurantes ofrecen el tipo de comida que indicó.

      Con las dos informaciones previas y, por como están definidos los restaurantes, ya podemos saber qué restaurante debemos recomendar al usuario, lo que faltaría serían hacer algunas validaciones respecto a ese restaurante.
- Aplicar validaciones:
    - Se le pregunta el lugar donde quiere comer y se revisa si el restaurante se encuentra en esa locación.
    - Se le pregunta cuántas personas irán a comer y se revisa si el restaurante tiene capacidad mayor o igual a la solicitada por el usuario.

      Si se consigue validar todos los datos necesarios, se procede a hacer la recomendación al usuario del restaurante, dar su dirección y también

informarle de las disposiciones del Ministerio de Salud, sino retorna false, indicando que los restaurantes de la base de datos no cumplen con los criterios del usuario.

1.4. Problemas sin solución: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

De manera general, se logró implementar la totalidad del proyecto y fue posible solucionar la mayoría de los problemas que fueron detectados.

1.5. Problemas solucionados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

i) **Análisis del input del usuario:** la interpretación del input del usuario fue un problema que necesitó varias horas de trabajo para ser solucionado. Esto debido a que las reglas del BNF para obtener las palabras clave recibían solo oraciones en una lista separando cada palabra por comas, por ejemplo, [yo, quiero, comer, pizza]. Sin embargo, para ser más amigable con el usuario, fue necesario investigar métodos para que el texto pudiera ser insertado entre comillas y de manera normal, por ejemplo, “yo quiero comer pizza”. El punto tres de la bibliografía planteaba una solución similar a la deseada.

ii) **Modelado de la base de datos:** se presentó un problema al modelar la base de datos, ya que se probaron varias opciones hasta que se encontró una solución viable. Para solucionar este problema, se implementó de la manera que fue mencionada en este documento en la sección de hechos y reglas desarrolladas.

1.6. Plan de Actividades realizadas por estudiante: Planeamiento de las actividades, descripción de la tarea, tiempo estimado de completitud, responsable a cargo y fecha de entrega.

Tarea	Tiempo Estimado	Responsable	Fecha de entrega
Base de la aplicación para obtener y analizar el input del usuario	3 horas	Julián Camacho	5/10/2020
Programa identificador de palabras clave	12 horas	Julián Camacho	5/10/2020
Definición de reglas y deducciones para las sugerencias	10 horas	Fabián Mendoza Sergio Ríos	8/10/2020
Definición del conjunto de restaurantes	4 horas	Fabián Mendoza	8/10/2020
Desarrollo de la interfaz de usuario	5 horas	Fabián Mendoza Segio Ríos	13/10/2020

1.7. Conclusiones.

i) Se concluye que desarrollar una aplicación con el comportamiento experto utilizando Prolog, es una opción factible y favorable, debido a las propiedades y ventajas que brinda un paradigma lógico.

ii) Se aplicaron los conceptos y habilidades aprendidas durante el estudio del paradigma de programación lógico, como la declaración de hechos y reglas para satisfacer el objetivo.

iii) Se logró modelar el problema mediante la implementación de diversos hechos y reglas para manipular las listas con las que fue modelada la base de datos de los restaurantes. Entre ellas, la función miembro, la obtención de las cabezas de las listas, entre otras.

iv) Se logró implementar un sistema experto que se encarga de recomendar al usuario restaurantes a partir de las preferencias brindadas, analizando su gramática y realizando deducciones a partir de la base de datos.

1.8. Recomendaciones.

a. Prolog brinda características favorables para el desarrollo de sistemas expertos utilizando gramáticas libres de contexto, por lo que se recomienda la implementación de aplicaciones que involucren este tipo de sistemas utilizando un paradigma lógico.
b. Para la utilización de la aplicación, se recomienda leer el manual de usuario antes de utilizar la aplicación, esto con el fin de comprender las gramáticas válidas y las distintas partes que componen el programa.
c. Recordar utilizar comillas a la hora de escribir el input al programa, ya que de no utilizarlas el programa no funciona correctamente.
d. Se recomienda el uso de oraciones claras, que contengan estructuras con verbos y predicados definidos, esto con el fin de facilitar la interacción con el Sistema Experto .
e. Es necesario escribir las instrucciones en minúscula, para que el Sistema Experto pueda entender de la mejor manera los deseos del usuario.

1.9. Bibliografía consultada en todo el proyecto

Soler, F (2020). *Programación de gramáticas clausales en Prolog.* Recuperado el 4 de Octubre de 2020, de https://personal.us.es/fsoler/papers/05capgram.pdf

Valverde, J. (2020). *Gramáticas de cláusulas definidas - JARV's blog.* Recuperado el 4 de Octubre de 2020, de http://jariaza.es/blog/post/prolog/gramaticas-de-clausulas-definidas.md

SWI-Prolog -- split_string/4. (2014). *Converting a user input string to list*. Recuperado el 5 de Octubre de 2020, de https://www.swi-prolog.org/pldoc/man?predicate=split_string/4

SWI-Prolog -- *atom_string/2*. (2020). Recuperado el 13 de Octubre de 2020, de https://www.swi-prolog.org/pldoc/man?predicate=atom_string/2

Prolog: *how to convert string to integer?*. (2011). Recuperado el 13 de Octubre 2020, de https://stackoverflow.com/questions/6782007/prolog-how-to-convert-string-to-integer

2. Bitácora en digital, donde se describen las actividades realizadas.

Fecha	Estudiante(s)	Actividad
4/10/2020	Todos	Primera reunión para comprender, compartir ideas y organización: División del trabajo correspondiente a cada uno de los integrantes de acuerdo con la dificultad estimada de este.
4/10/2020	Julián Camacho	Definición de hechos y reglas para el BNF. Para iniciar este desarrollo, fue necesario consultar el punto uno y dos de la bibliografía, para reforzar los conceptos y comprender el funcionamiento de las gramáticas libres de contexto.
5/10/2020	Julián Camacho	Implementación del análisis del input del usuario. Actualización del BNF. Para este análisis se necesitó investigar en la página web del punto tres de la bibliografía.
6/10/2020	Todos	Reunión de organización y actualización del trabajo realizado hasta el momento.
12/12/2020	Fabián Mendoza Julián Camacho	Reunión para la definición de la base de datos y las reglas para la búsqueda en la misma.
13/10/2020	Sergio Ríos Fabián Mendoza	Reunión para terminar de definir la búsqueda, reglas y hechos para el comportamiento experto de la aplicación.
13/10/2020	Julián Camacho	Implementación del análisis gramático y obtención de palabras claves para oraciones relacionadas con ubicaciones y capacidad. Se utilizaron las fuentes cuatro y cinco de la bibliografía para solucionar bugs relacionados con inconsistencias e incompatibilidad.