



Área académica de ingeniería en computadores

Lenguajes, Compiladores e Intérpretes | CE3104

Grupo 1

Tarea 2: Programación lógica con prolog

MayCEy

Profesor: Marco Rivera Meneses

Estudiantes:

- Jose Antonio Espinoza Chaves - 2019083698
- Eduardo Zumbado Granados – 2019003973

I Semestre 2022

Índice

Descripción de los hechos y reglas implementadas	3
Hechos implementados	3
Reglas implementadas	9
BNF.....	9
MayCEy	10
Descripción de las estructuras de datos utilizadas	14
Descripción detallada de los algoritmos desarrollados	15
Problemas conocidos	15
Plan de Actividades realizadas por estudiante	16
Problemas encontrados	17
Conclusiones del Proyecto.....	17
Recomendaciones del Proyecto	18
Bibliografía consultada en todo el Proyecto	18
Enlace del repositorio	18

Descripción de los hechos y reglas implementadas

Hechos implementados

saludos('hola').

-Hecho que relaciona la palabra clave hola con saludos

saludos('buenas').

-Hecho que relaciona la palabra clave buenas con saludos

saludos('Hola').

-Hecho que relaciona la palabra clave Hola con saludos

saludos('Buenas').

-Hecho que relaciona la palabra clave Buenas con saludos

saludos('Buenas tardes').

-Hecho que relaciona la palabra clave Buenas tardes con saludos

saludos ('Buenos días').

-Hecho que relaciona la palabra clave Buenos días con saludos

saludos ('Buenas noches').

-Hecho que relaciona la palabra clave Buenas noches con saludos

despedidas('adios').

-Hecho que relaciona la palabra clave adiós con despedidas

despedidas('Adios').

-Hecho que relaciona la palabra clave Adiós con despedidas

despedidas('hasta luego').

-Hecho que relaciona la palabra clave hasta luego con despedidas

despedidas('Hasta luego').

-Hecho que relaciona la palabra clave Hasta luego con despedidas

despedidas('chao').

-Hecho que relaciona la palabra clave chao con despedidas

despedidas('Chao').

-Hecho que relaciona la palabra clave Chao con despedidas

despedidas('hasta').

-Hecho que relaciona la palabra clave hasta con despedidas

despedidas('cambio').

-Hecho que relaciona la palabra clave cambio con despedidas

agradecimientos('gracias').

-Hecho que relaciona la palabra clave gracias con agradecimientos

agradecimientos('muchas').

-Hecho que relaciona la palabra clave muchas con agradecimientos

emergencia('perdida').

-Hecho que relaciona la palabra clave perdida con emergencia

emergencia('parto').

-Hecho que relaciona la palabra clave parto con emergencia

emergencia('paro').

-Hecho que relaciona la palabra clave paro con emergencia

emergencia('secuestro').

-Hecho que relaciona la palabra clave secuestro con emergencia

emergencia('perdi').

-Hecho que relaciona la palabra clave perdi con emergencia

avionesGrandes('boing747').

-Hecho que relaciona la palabra clave boing747 con aviones grandes.

avionesGrandes('airbusa380').

-Hecho que relaciona la palabra clave airbusa380 con aviones grandes.

avionesGrandes('airbusa340').

-Hecho que relaciona la palabra clave airbusa340 con aviones grandes.

avionesMedianos('airbusa220').

-Hecho que relaciona la palabra clave airbusa220 con aviones medianos.

avionesMedianos('boing717').

-Hecho que relaciona la palabra clave boing717 con aviones medianos.

avionesMedianos('embraer190').

-Hecho que relaciona la palabra clave embraer190 con aviones medianos.

avionesPequenos('cessna').

-Hecho que relaciona la palabra clave Cessna con aviones pequeños.

avionesPequenos('beechcraft').

-Hecho que relaciona la palabra clave beechcraft con aviones pequeños.

avionesPequenos('embraerphenom').

-Hecho que relaciona la palabra clave embraerphenom con aviones pequeños.

pista('P1').

-Hecho que relaciona la palabra clave P1 con la pista 1.

pista('P2-1').

-Hecho que relaciona la palabra clave P2-1 con la pista 2-1.

pista('P2-2').

-Hecho que relaciona la palabra clave P2-2 con la pista 2-2.

pista('P3').

-Hecho que relaciona la palabra clave P3 con la pista 3.

emergencias('mayday').

-Hecho que relaciona la palabra clave 'mayday' con las llamadas a emergencias.

emergencias('Mayday').

-Hecho que relaciona la palabra clave 'Mayday' con las llamadas a emergencias.

emergencias('mayday,').

-Hecho que relaciona la palabra clave 'mayday,' con las llamadas a emergencias.

emergencias('Mayday,').

-Hecho que relaciona la palabra clave 'Mayday,' con las llamadas a emergencias.

emergencias('emergencia').

-Hecho que relaciona la palabra clave 'emergencia' con las llamadas a emergencias.

emergencias('Emergencia').

-Hecho que relaciona la palabra clave 'Emergencia' con las llamadas a emergencias.

emergencias('7500').

-Hecho que relaciona la palabra clave '7500' con las llamadas a emergencias.

condicion('velocidad').

-Hecho que relaciona la palabra clave 'velocidad' con las condiciones de aterrizaje.

condicion('peso').

-Hecho que relaciona la palabra clave 'peso' con las condiciones de aterrizaje.

condicion('viento').

-Hecho que relaciona la palabra clave 'viento' con las condiciones de aterrizaje.

condicion('distancia').

-Hecho que relaciona la palabra clave 'distancia' con las condiciones de aterrizaje.

matriculas('alpha').

-Hecho que relaciona la palabra clave 'alpha' con los valores de matrículas.

matriculas('bravo').

-Hecho que relaciona la palabra clave 'bravo' con los valores de matrículas.

matriculas('charlie').

-Hecho que relaciona la palabra clave 'charlie' con los valores de matrículas.

matriculas('delta').

-Hecho que relaciona la palabra clave 'delta' con los valores de matrículas.

matriculas('echo').

-Hecho que relaciona la palabra clave 'echo' con los valores de matrículas.

matriculas('foxtrot').

-Hecho que relaciona la palabra clave 'foxtrot' con los valores de matrículas.

matriculas('golf').

-Hecho que relaciona la palabra clave 'golf' con los valores de matrículas.

matriculas('hotel').

-Hecho que relaciona la palabra clave 'hotel' con los valores de matrículas.

matriculas('india').

-Hecho que relaciona la palabra clave 'india' con los valores de matrículas.

matriculas('juliet').

-Hecho que relaciona la palabra clave 'juliet' con los valores de matrículas.

matriculas('kilo').

-Hecho que relaciona la palabra clave 'kilo' con los valores de matrículas.

matriculas('lima').

-Hecho que relaciona la palabra clave 'lima' con los valores de matrículas.

matriculas('mike').

-Hecho que relaciona la palabra clave 'mike' con los valores de matrículas.

matriculas('november').

-Hecho que relaciona la palabra clave 'november' con los valores de matrículas.

matriculas('oscar').

-Hecho que relaciona la palabra clave 'oscar' con los valores de matrículas.

matriculas('papa').

-Hecho que relaciona la palabra clave 'papa' con los valores de matrículas.

matriculas('quebec').

-Hecho que relaciona la palabra clave 'quebec' con los valores de matrículas.

matriculas('romeo').

-Hecho que relaciona la palabra clave 'romeo' con los valores de matrículas.

matriculas('sierra').

-Hecho que relaciona la palabra clave 'sierra' con los valores de matrículas.

matriculas('tango').

-Hecho que relaciona la palabra clave 'tango' con los valores de matrículas.

matriculas('uniform').

-Hecho que relaciona la palabra clave 'uniform' con los valores de matrículas.

matriculas('victor').

-Hecho que relaciona la palabra clave 'victor' con los valores de matrículas.

matriculas('whiskey').

-Hecho que relaciona la palabra clave 'whiskey' con los valores de matrículas.

matriculas('xray').

-Hecho que relaciona la palabra clave 'xray' con los valores de matrículas.

matriculas('yankee').

-Hecho que relaciona la palabra clave 'yankee' con los valores de matrículas.

matriculas('zulu').

-Hecho que relaciona la palabra clave 'zulu' con los valores de matrículas.

hora ('X: XX').

-Hecho que relaciona los valores desde 0:00 a 24:00 con los valores de hora.

Reglas implementadas

BNF

oracion(S0,S):

-Regla que define la oración para el BNF, tiene variantes, para cuando la oración es conformada por sintagma nominal y sintagma verbal, cuando es adverbio, sintagma nominal y verbal, solo sintagma nominal, solo sintagma verbal, solo adverbio o solo palabra clave.

sintagmaNominal(S0,S):

-Regla que define el sintagma nominal, el cual tiene dos posibles tipos, solamente conformado por un pronombre o conformado por un complemento directo.

sintagmaVerbal(S0,S):

-Regla que define el sintagma verbal, tiene múltiples variantes, para cuando este esta formado por un verbo conjugado, un infinitivo, verbo conjugado y complemento directo o infinitivo y complemento directo

complemento(S0,S):

-Regla que define el complemento directo. El cual puede estar hecho de un sustantivo, adjetivo, articulo, matriculas, articulo y sustantivo, artículo, sustantivo y adjetivo, o múltiples sustantivos.

palabraClave(S0,S):

-Regla que define la palabra clave, se definen también los hechos de sustantivos que se utilizan como palabras clave. Estas abarcan palabras claves de saludo, despedida, agradecimiento y solicitud de emergencia.

eliminarCaracter(S,C,X) :

-Regla que elimina un carácter de un string.

my_read(List):

-Regla standard utilizada para leer la oración escrita, dividirla en palabras e insertarlas en una lista para futuro análisis.

MayCEy

primerElemento([X|_],X)

-Regla para obtener el primer elemento de una lista.

eliminar(_,[],[]).

-Regla para eliminar un elemento de la lista.

revisarEntrada(X):

-Regla que revisa la primera entrada del usuario, revisa si es un saludo o una emergencia.

revisarSaludo(Lista,Entrada):

-Regla que recorre recursivamente la lista de palabras hasta encontrar una palabra clave que coincida con un saludo.

revisarEmergencia(Lista, Entrada):

-Regla que recorre recursivamente la lista de palabras hasta encontrar una palabra clave que coincida con una emergencia.

revisarSolicitud(X):

-Regla que revisa si hay una solicitud de aterrizar o despegar en la entrada del usuario.

revisarAterrizaje(Lista, Entrada):

-Regla que revisa si en la entrada del usuario se pide un aterrizaje

revisarDespegue(Lista, Entrada):

-Regla que revisa si en la entrada del usuario se pide un despegue.

revisarEmergencia(X):

-Regla que llama a la regla tipoEmergencia y funciona como controlador de secuencia.

tipoEmergencia(Lista, Entrada):

-Regla que recorre la lista recursivamente hasta encontrar la palabra clave de una emergencia.

revisarIdentificacion(X,Y):

-Regla que revisa la identificación del usuario con matrícula y avión

revisarMatricula(Lista,Entrada,Matricula,X,Y):

-Regla que lee la entrada y busca los cuatro valores de matricula para aceptar la matricula del usuario.

revisarAvion(Lista,Entrada,Pista,Y)

-Regla que revisa que el avión entrado es un avión pequeño, mediano o grande. De igual manera recorre la lista de palabras recursivas hasta encontrarlo. De no encontrarlo retorna error.

leerHora(X,Y,Matricula,Pista):

-Regla que llama a revisarHora(), se utiliza como control de flujo de trabajo.

revisarHora(Lista,Entrada,Y,Matricula,Pista,Hora):

-Regla que revisa en la entrada del usuario, la hora que se requiere, utilizado para definir la hora de llegada o salida.

revisarHora(Lista,Entrada,Y,Matricula,Pista,Hora):

-Regla que llama a buscarDireccion(), se utiliza como control de flujo de trabajo, de manera que si retorna un valor esperado se puede avanzar en las entradas.

buscarDireccion(Lista,Entrada,Y,Matricula,Pista,Hora):

-Regla que busca recursivamente este u oeste, si es así es porque se ocuparan una de las dos pistas 2, mientras que, de ser vacío, se escoge pista 1 o pista 3.

revisarDireccion(Entrada,Pista):

-Revisa recursivamente el orden de encuentro de las palabras este y oeste y brinda un valor a la pista 2.

asignarPista(Matricula,Pista,Hora,Y):

-Regla que asigna la pista dependiendo de la dirección de llegada o salida y el tamaño del avión. Tiene múltiples variantes para cubrir todos los posibles casos.

asignarHoraPXX(Hora, HoraC, Hora2):

- Regla que asigna una hora para la pista, en caso de que no este disponible la hora dada por el usuario, se asigna la más próxima disponible.
- Los valores XX varían, ya que hay un método por cada pista y por cada orientación.

asignarSiguienteHora(Hora, ListaHoras, Hora2):

- Regla para asignar la hora libre más cercana a la hora dada, utiliza una lista de horas.

asignarPistaNuevaX(PistaNueva):

- Regla para asignar una pista nueva, en caso de que la pista solicitada se encuentre ocupada, se verifica cual es la pista que se encuentra ocupada y cual se asignará.
- El valor X varía según si se trata de la pistas 1, 2 o 3.

revisarPAB(X):

- Regla que revisa si las pistas están llenas, los valores AB cambian en la implementación, pues se tiene una regla por cada pista.

leerAgradecimiento(X):

- Regla para revisar el mensaje de agradecimiento del usuario, llama a la regla “revisarAgradecimiento(Lista, Entrada, Respuesta)”

revisarAgradecimiento(Lista, Entrada, Respuesta):

- Regla para revisar mensajes de despedida, si encuentra una palabra clave de agradecimiento, envía el mensaje de respuesta.

leerDespedida(X):

- Esta Regla se utiliza para revisar mensajes de despedida, llama a la regla “revisarDespedida”.

revisarDespedida(Lista, Entrada, Answer):

- Regla para procesar los mensajes de despedida del usuario, se encarga de cerrar el chat si así es necesario y prepararse para recibir mensajes de otro usuario diferente.

chequearCierre(X):

- Regla para revisar si la entrada del usuario es una palabra clave de cierre, para detener la ejecución del programa.

chequearEntrada(X):

- Regla que llama a “my_read” en la base de datos para dividir la oración y convertirla en una lista, y luego de eso, se llama al BNF para verificar si la sintaxis es correcta.

primeraEntrada(X):

- Regla para revisar la primera entrada del usuario y determinar si es un saludo o una emergencia.

segundoValorSaludo(X):

- Revisa la segunda entrada en caso de que la primera no fuese un saludo.

segundoValorEmergencia(X):

- Revisa la primera fuese una emergencia.

tercerValorSaludo(X):

- Revisa la tercera entrada para ambos casos, sin importar que tipo de mensaje sea.

tercerValorEmergencia(X):

- Revisa la tercera entrada para el caso en que es un mensaje de emergencia.

cuartoValorSaludo(X, Y, Matricula, Pista):

- Revisa la cuarta entrada, solo para las conversaciones normales, pues las emergencias fueron cubiertas anteriormente.

quintoValorSaludo(X, Y, Matricula, Pista, Hora):

- Revisa la quinta entrada, analiza la dirección dada por el usuario.

sextaEntrada(X):

- Revisa la sexta entrada del usuario, relacionada con la despedida o cierre del chat.

septimaEntrada(X):

- Revisa la ultima entrada posible, relacionada con el cierre, en caso de que el usuario no cierre o no se despida correctamente

mayCEy():

- Regla inicial para comenzar el código, se debe escribir en la consola para iniciar la ejecución.

Descripción de las estructuras de datos utilizadas

En el proyecto se hace uso de las listas para almacenar y procesar las entradas del usuario. Se utilizaron reglas para poder manipular dichas listas, cuando se requiera obtener el primer elemento se llama a la regla “primerElemento ([X| _], X).” Además de que se utiliza la regla “eliminar” en caso de retirar elementos de la lista. El uso principal de la estructura de lista se da a la hora de analizar la línea introducida por el usuario, ya que esta es pasada por my_read (), la cual divide la oración en palabras, la pasa a minúsculas y las introduce en una lista, la cual es llamada por casi todas las reglas del sistema experto, de manera que se aplican recorridos a la lista para encontrar palabras claves y así saber que acción aplicar. Otro uso que se da es que se definió la lista de horasPosibles (en la base de datos), la cual almacena las horas disponibles para las pistas, y se le aplican reglas de borrado, actualización y revisión.

Descripción detallada de los algoritmos desarrollados

Para la solución del sistema experto se utilizó una gramática libre de contexto, la cual se describe utilizando Backus Naur-Form (BNF), la cual es una notación para describir este tipo de gramáticas. Esta notación consiste en realizar derivaciones partiendo de una oración inicial, tal como se muestra en las reglas del documento “BNF.pl”, donde se tiene como entrada una oración y se busca que la misma siga un orden específico buscando por los diferentes sintagmas, verbos, adjetivos o demás componentes que conformen una solicitud del usuario. Se puede notar como se realizan derivaciones hasta llegar a componentes atómicos de la oración, como lo son en este caso los verbos, adjetivos, sustantivos, artículos o pronombres.

Además, tenemos que en la mayoría de las reglas del sistema experto se hace un recorrido recursivo de la lista de palabras hasta encontrar la palabra clave. Esto lo hace revisando el primer elemento de la lista, si no coincide con la palabra clave buscada, elimina el primer elemento y llama la regla de nuevo con la lista nueva. Este algoritmo se repite en todas las reglas de manejo de listas.

Problemas conocidos

Uno de los principales problemas es que el lenguaje no reconoce los acentos naturales del idioma español, por ejemplo, las tildes. Usualmente no es problema, ya que, al no encontrar la palabra con acentuación en la base de datos, devuelve un mensaje que el input es incorrecto, sin embargo, para el caso de despedida, “adiós” debería de poderse leer, sin embargo, lo toma como inválido. Esto pues el lenguaje tiene un comportamiento de cambiar la tilde por un símbolo que es difícil de incluir en el análisis de las palabras.

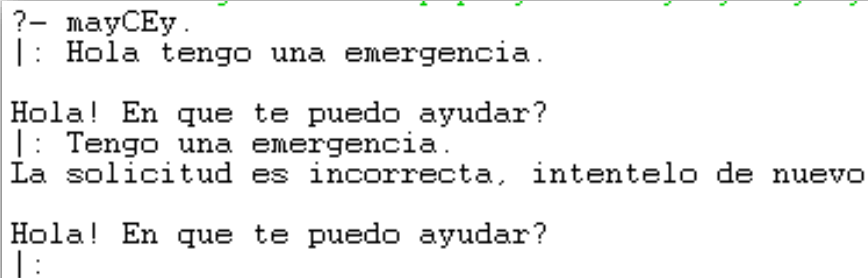
Otro problema que se presentó es que, cuando el usuario ingresa “Hola” seguido de “tengo una emergencia” o cualquier otra forma de solicitar ayuda, el sistema se confunde, pues no espera recibir una emergencia después de un saludo, por lo que para las solicitudes de emergencias se ingresa únicamente la solicitud sin saludo. Es importante corregir dichos errores, pues el propósito de un sistema experto es que el usuario pueda interactuar sin ser capaz de determinar si se trata de un programa o una persona.

Plan de Actividades realizadas por estudiante

Tarea	Tiempo estimado	Responsable a cargo	Fecha de entrega
Definir sintagmas nominales	4 horas	Jose Antonio	13/05/2022
Definir sintagmas verbales	4 horas	Jose Antonio	13/05/2022
Definir los nombres y determinantes	2 horas	Eduardo	12/05/2022
Crear las tablas de saludos, aviones, pistas...	2 horas	Eduardo	12/05/2022
Identificar aviones con matrícula y tamaño	3 horas	Eduardo	15/05/2022
Iniciar con la documentación	2 horas	Eduardo	18/05/2022
Validar la solicitud del usuario	4 horas	Jose Antonio	19/05/2022
Programar procesado de emergencias	8 horas	Jose Antonio	22/05/2022
Programar selección y asignación de pistas	4 horas	Eduardo	23/05/2022

Problemas encontrados

Como se mencionó anteriormente, el mayor problema encontrado en el desarrollo del sistema experto es que cuando el sistema espera la primera entrada del usuario, esta puede ser un saludo o una emergencia, donde la palabra clave para el saludo puede ser “hola” o “buenas”, y para la emergencia es “mayday” o “7500” o “emergencia”. Por lo que al escribir en la primera línea: “Hola, tengo una emergencia”, el sistema lo lee como un saludo, ya que recorre la oración de izquierda a derecha. Por lo que lee primero la palabra “Hola” y sigue el flujo del trabajo para conversación normal. No se pudo solucionar este problema ya que el sistema del experto completamente se basa en que para la primera entrada solo puede ser un saludo o una emergencia. Se adjunta imagen del error.



```
?- mayCEy.  
|: Hola tengo una emergencia.  
  
Hola! En que te puedo ayudar?  
|: Tengo una emergencia.  
La solicitud es incorrecta, intentelo de nuevo  
  
Hola! En que te puedo ayudar?  
|:
```

Conclusiones del Proyecto

El uso del paradigma de programación lógica permitió la implementación de un sistema experto que no requiere una experiencia algorítmica, ya que utiliza una base de datos e inferencias lógicas para determinar que acciones tomar. Por otra parte, al utilizar una base de datos se permite que el sistema sea escalable, es decir, se facilita el agregar más hechos y reglas a las ya disponibles.

Finalmente, se logró implementar una gramática libre de contexto, fundamental para el sistema experto, ya que esta procesa la entrada del usuario y la separa gramaticalmente, esto hace que se realice un análisis más natural. De esta manera, teniendo un BNF muy completo, que implique un sin número de artículos, sustantivos, verbos, adjetivos, etc.; se obtiene un sistema el cual puede entender lenguaje natural sin ningún problema.

Recomendaciones del Proyecto

Primeramente, se recomienda ampliar el conjunto de elementos terminales al BNF, para poder reconocer más sinónimos de palabras o poder procesar más solicitudes, así como implementar una solución para reconocer los signos de puntuación naturales del lenguaje español y tener una comunicación más natural. De la mano con lo anterior, se recomienda el agregar más reglas de derivación, para así poder manejar más oraciones diferentes y minimizar los casos en los que no se entienda la entrada del usuario y a su vez, minimizar los errores en tiempo de ejecución.

Por otro lado, se recomienda que se busque una manera de hacer las reglas, para que dentro de una misma regla se tomen varios casos posibles, ya que en esta tarea se hizo una regla para cada caso posible, lo cual promueve el orden en el código, pero hace que el código se extienda mucho e innecesariamente.

Bibliografía consultada en todo el Proyecto

- [1] Escrig, T., Pacheco, J., Toledo, F. “El lenguaje de programación Prolog”. Castellón: Universidad Jaume I de Castellón. [2001]. Disponible:
<https://tecdigital.tec.ac.cr/dotlrn/classes/IDC/CE3104/S-1-2022.CA.CE3104.1/file-storage/view/lenguajes%2Flibros%2FLibroPROLOG.pdf>
- [2] Universidad autónoma de baja california. “Programación lógica”. Facultad de Ciencias Químicas e Ingeniería, Universidad autónoma de baja california. Disponible:
<http://fcqi.tij.uabc.mx/usuarios/ardiaz/prior/prolog/ManualdeLaboratorio.pdf>
- [3] Borland International. Turbo Prolog Reference Guide V 2.0. Borland International, 1988

Enlace del repositorio

<https://github.com/JoseA4718/MayCEy>