UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA

LENGUAJE Y MÉTODOS DE PROGRAMACIÓN

BUSCADOR EN PROGRAMACION LOGICO

Profesor: Roberto González
Alumno: Roberto Orellana

Fecha de Entrega: 10/11/2017

Santiago de Chile

10 - 11 – 2017

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCION

El siguiente informe trata de explicar cómo llegar a la solución del laboratorio N°3 de Lenguaje y métodos de Programación.

1.1 ANTECEDENTES Y MOTIVACIÓN

El laboratorio consta de crear un Buscador en cuatro lenguajes distintos de programación, siendo este el tercero que trata sobre Lenguaje de Programación Lógico (Prolog) encontrándonos con Cláusulas de Horn, metas, dominios y predicados lo cual para mí ya es familiar, para conocer todo este mundo de métodos y lenguajes de Programación.

El siguiente paradigma se basa en predicados que caracterizan o relacionan a los individuos involucrados y la deducción de las posibles respuestas a una determinada consulta.
Los individuos son aquellas cosas sobre las que versa el conocimiento que queremos expresar
Individuos simples**:** los átomos
Individuos compuestos**:** tienen otros individuos adentro, y se pueden ver o acceder a cada componente.

1.2 DESCRIPCIÓN DEL PROBLEMA

El Problema consiste en crear un sistema de recuperación de Información en el lenguaje de programación Lógico (Prolog) el cual consiste en la implementación de los Requerimientos No funcionales, requerimientos funcionales obligatorios y/o requerimientos extra, implementar la base de conocimiento para nuestro buscador y las funciones que nos servirán para realizar diferentes consultas,

1.3 SOLUCIÓN PROPUESTA

La solución propuesta a este problema radica inicialmente en el análisis de los requerimientos funcionales y no funcionales contenidos en el enunciado del Laboratorio, y de esta forma llegar a una posible solución comenzando por la implementación apropiada para los problemas a través de dominios y predicados, metas primarias y secundarias, hechos y relaciones, con la creación de nuestra base de conocimiento los siguientes predicados Stopwords, Documentos e Índice utilizando la representación más adecuada, además de las funciones para realizar las siguientes consultas; consultar por un término y documentos que lo contienen, consultar por una frase y documentos que lo contienen, consultar una frase y una lista de todos los documentos que la contienen, consultar por una lista de resultados y mostrar los documentos con sus respectivos

títulos y consultar por una lista de resultados y mostrar la cantidad de documentos que lo contienen, además de otros requerimientos extras como son la consulta de un término y su frecuencia en el documento y la consulta de un término con los documentos que no contienen ese termino con sus respectivos títulos.

1.4 OBJETIVOS Y ALCANCES DEL PROYECTO

1.4.1 Objetivo General

El Objetivo general del proyecto consta en lograr el correcto funcionamiento del programa en el lenguaje de programación Prolog, utilizando lo aprendido en clases y laboratorio sobre paradigma de programación Lógico

1.4.2 Objetivos Específicos

Lograr que el usuario (con conocimientos básico en programación de Logico (SWI-Prolog)) pueda realizar diferentes consultas al Índice, consultar por un término y documentos que lo contienen, consultar por una frase y documentos que lo contienen, consultar una frase y una lista de todos los documentos que la contienen, consultar por una lista de resultados y mostrar los documentos con sus respectivos títulos y consultar por una lista de resultados y mostrar la cantidad de documentos que lo contienen, además de otros requerimientos extras como son la consulta de un término y su frecuencia en el documento y la consulta de un término con los documentos que no contienen ese término con sus respectivos títulos.

1.4.3 Alcances

Obtener un correcto funcionamiento del programa gracias a los objetivos y problemas que se plantearon anteriormente.
Obtener un buscador (realizar consultas) de Índices que pueda ser ejecutado por personas con conocimientos básico sobre prolog.

1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

Las herramientas y técnicas utilizadas para la solución final del problema constan de lo aprendido en clases, investigación propia del alumno y las metodologías enseñadas por el profesor, analizando el problema y teniendo una abstracción de cómo resolverlo.
El Software utilizado en mi caso fue la de SWI-Prolog pese a su complicada compilación y ejecución, logra los objetivos necesarios para la realización del proyecto.

CAPITULO 2. FUNDAMENTOS TEORICOS

2.1 Método

El Método Utilizado para representar o modelar las funciones, fue a través de dominios y predicados, metas primarias y secundarias, hechos y relaciones.

2.2 Definiciones

- **Paradigma de programación Lógica:** La programación lógica gira en torno al concepto de predicado, o relación entre elementos. Resuelve problemas descriptos como las relaciones de un conjunto de datos, sobre las que aplica reglas de deducción y a partir de tales premisas genera conclusiones aceptadas como válidas.

- **Constantes:** se utilizan para referirse a objetos

- **Predicados:** expresan relaciones entre los objetos.

- **Hechos:** es una relación entre objetos, formados por combinación de predicados y constantes.

- **Reglas:** es un conjunto de hechos que consta de dos parte (cabeza:-cuerpo), en el cuerpo puede contener varios hecho y en la cabeza solo uno.

CAPITULO 3. DESARROLLO DE LA SOLUCION

3.1 Análisis

En el análisis la deducción principal fue la representación de nuestra base de conocimiento en la cual utiliza todas las consultas que realiza el buscador y la implementación de los Hechos y Reglas que el buscador (que realiza consultas) utilizara y combinara para llegar a la solución del problema. Es muy importante elegir una buena representación para el Índice.

3.2 Diseño

Para desarrollar este proyecto se decidió optar por una representación utilizada en los paradigmas anteriores, y es la de un hecho con dos argumentos, un término y una lista dispersa que contiene todos los documentos y sus respectivas frecuencias ej. índice (termino, [0, 2, 0, 1, 0]).
Debido a la representación elegida resulta necesario generar reglas que operen con ella, obteniendo la información necesaria para entregar al usuario que realizar las consultas al buscador.

Para este proyecto se almacenan en la base de conocimientos del programa, una lista de stopWords, 5 documentos y aproximadamente 186 Índices que contienen diferentes términos que pertenecen a los 5 documentos.

3.3 Construcción

Para este proyecto se implementa primero como hechos y base de conocimiento las Stopwords, Documentos e index que utiliza las siguientes representaciones:

stopWords

que utiliza una lista de elementos donde cada elemento corresponde a una palabra.

```
stopWords([palabra1,palabra2,....,palabraN]).
```

```
%La representacion utilizada para las StopWordds fue la de una lista de
% elementos donda cada elemento corresponde a una palabra
% stopWords([palabra1,palabra2,....,palabraN]).
stopWords([a, about, above, across, after, afterwards, again, against, al
ny, anyhow, anyone, anything, anyway, anywhere, are, around, as, at, back
between, beyond, bill, both, bottom, but, by, call, can, cannot, cant, co
leven, else, elsewhere, empty, enough, etc, even, ever, every, everyone,
```

Imagen 3.1. Base de conocimientos stopWords

Documentos

que utiliza 5 argumentos o átomos.

```
documentos(ID, Titulo, Autor, Biblio, Texto).
```

```
%Representacion para Documentos
%documentos(ID,Titulo,Autor,Biblio,Texto).
documentos(1,
"experimental investigation of the aerodynamics of a
wing in a slipstream .",
"brenckman,m",
"j. ae. scs. 25, 1958, 324",
"experimental investigation of the aerodynamics of a
wing in a slipstream .
  an experimental study of a wing in a propeller slipstream was
made in order to determine the spanwise distribution of the lift
increase due to slipstream at different angles of attack of the wing
and at different free stream to slipstream velocity ratios .  the
results were intended in part as an evaluation basis for different
theoretical treatments of this problem .
  the comparative span loading curves, together with
supporting evidence, showed that a substantial part of the lift increment
produced by the slipstream was due to a /destalling/ or
boundary-layer-control effect .  the integrated remaining lift
increment, after subtracting this destalling lift, was found to agree
well with a potential flow theory .
  an empirical evaluation of the destalling effects was made for
the specific configuration of the experiment").
documentos(2,
"simple shear flow past a flat plate in an incompressible fluid of small
viscosity",
```

Imagen 3.2. Base de conocimientos documentos

Index

que utiliza como argumento 2 elementos, un término y una lista dispersa de frecuencias

```
index(termino, [doc1, doc2, doc3,....,docN]).
```

```
%Mi representación para el Indice consta de 2 argumento
%un termino y una lista que es el Indice Disperso.
%index(termino, [doc1, doc2, doc3,....,docN]).
index(experimental, [3, 0, 0, 0, 0]).
index(investigation, [2, 0, 0, 0, 0]).
index(aerodynamics, [2, 0, 0, 0, 0]).
index(wing,[4, 0, 0, 0, 0]).
index(slipstream, [6, 0, 0, 0, 0]).
```

Imagen 3.3. Base de conocimientos index

Y reglas que operan sobre ella, reglas básicas como por ejemplo:
Regla singleTermQuery que nos permite saber si un término se encuentra dentro de algún documento.

```
% ------------------------------------------------------------------------
% Consulta un termino si se encuentra en un documento.
% meta Primaria y secundaria en algunos casos
% singleTermQuery(vorticity, Result).
singleTermQuery(Term, Result):-not(eliminaStopWords(Term)),index(Term,IDisperso),doc(IDisperso,Result).

%meta secundaria
doc([Doc|_],1):-Doc =\= 0.
doc([_|Docs],NoDoc):-doc(Docs,ND),NoDoc is ND + 1.

%meta secundaria
%comprueba si la palabra se encuentra en las palabras comunes(StopWords)
eliminaStopWords(Term):-stopWords(X),comparaStopWords(X,Term).

%meta secundaria
comparaStopWords([X|_],X):-!.
comparaStopWords([_|Xs],Term):-comparaStopWords(Xs,Term).
```

Imagen 3.4. Reglas singleTerm]Query.

Regla bestMatch que nos permita consultar por una frase y saber que documentos los contiene

```
% ------------------------------------------------------------------------
% Se Consulta una lista de palabras y los documentos que lo contienen
% meta Primaria
%bestMatch([karman,small,wing],Result).
bestMatch(Phrase, Rasult):-palabras(Phrase, Rasult).

%meta secundaria
palabras([Pal|_],Result):-singleTermQuery(Pal, Result).
palabras([_|Pals],Result):-palabras(Pals, Result).
```

Imagen 3.5. Regla bestMatch

Regla exactMatch que nos permita consultar por una frase y nos responde con una lista de todos los documentos.

```
% ------------------------------------------------------------------------
% Se Consulta una lista de palabras y una lista de Documentos
% meta Primaria
%exactMatch([karman,small,wing],Result).
exactMatch(Phrase, Results):-listPal(Phrase, Results).

%meta secundaria
listPal([],[]).
listPal([Pal|Pals],Result):-singleTermQuery(Pal, R),listPal(Pals, RL),append([R],RL,Result).
```

Imagen 3.6. Regla exactMatch

Regla Documents que me permite Consulta por una lista de documentos y responde con una lista de ID y título de documentos

```
% ------------------------------------------------------------------------
% Consulta por una lista de documentos y responde con una lista de ID y
% titulo de documentos
% meta Primaria
% documents([1,2,3], Documents).
documents(Result, Documents):-text(Result,Documents).

%meta secundaria
text([],[]).
text([R|Rs],F):-documentos(R,Te,_,_,_),text(Rs,Tex),append([R],[Te],T),append(T,Tex,F).
```

Imagen 3.7. Regla documents.

Regla numDocuments que me permite consulta por un término y la cantidad de frecuencia en un documento.

```
% ------------------------------------------------------------------------
%consulta una lista de Doc y la cantidad de resultados lo hace verdadero
%meta Primaria
%numDocuments([1,3,5],3)
numDocuments(Results,Count):-length(Results,Count).
```

Imagen 3.8. Regla numDocuments.

Tambien parte de los requerimientos extras:

Regla termTotalCount que me permite consulta por un termino y la cantidad de frecuencia en un documento

```
% ------------------------------------------------------------------------
%consulta por un termino y la cantidad de frecuencia en un documento
%meta Primaria
%termTotalCount(wing,Frequency).
termTotalCount(Term,Frequency):-not(eliminaStopWords(Term)),index(Term,IDisperso),freqDoc(IDisperso,Frequency).

%meta secundaria
freqDoc([Doc|_],Doc):-Doc =\= 0.
freqDoc([_|Docs],NoDoc):-freqDoc(Docs,NoDoc).
```

Imagen 3.9. Regla termTotalCount

Regla docNotCointain que a partir de la consulta de un termino se obtiene los documentos que NO contienen la palabra entregando una lista con ID y Titulo

```
% ------------------------------------------------------------------------
% A partir de la consulta de un termino se obtiene los documentos que no
% contienen la palabra su ID y Titulo
% meta Primaria
% docNotContain(vorticity, Documents).
docNotContain(Term, Documents):-not(eliminaStopWords(Term)),index(Term,IDisperso),length(IDisperso,Largo),
noDoc(IDisperso,NoDoc,Largo),reverse(NoDoc,NoIDyTitle),text(NoIDyTitle,Documents).

%meta secundaria
noDoc([],[],0).
noDoc([Doc|NoDoc], NNDoc,Cont):-Doc =\= 0,noDoc(NoDoc,NNDoc,C),Cont is C + 1,!.
noDoc([_|Xs], [Cont|NoDoc],Cont):- noDoc(Xs, NoDoc,C),Cont is C + 1.
```

Imagen 3.10. Regla docNotCointain

En el manual de usuario se encuentran las pruebas del programa y la forma se usó.

CAPITULO 4. RESULTADO

Se da cumplimiento a los objetivos del proyecto, creando las funciones requeridas; consultar por un término y documentos que lo contienen, consultar por una frase y documentos que lo contienen, consultar una frase y una lista de todos los documentos que la contienen, consultar por una lista de resultados y mostrar los documentos con sus respectivos títulos y consultar por una lista de resultados y mostrar la cantidad de documentos que lo contienen, además de otros requerimientos extras como son la consulta de un término y su frecuencia en el documento y la consulta de un termino se obtiene los documentos que NO contienen la palabra entregando una lista con ID y Titulo.
Si en alguna de las consultas retorna true y luego false, es porque luego de encontrar una coincidencia (true) sigue buscando hasta el final de base de conocimiento y retornando finalmente false (porque ya no encontró más coincidencias) (mundo cerrado).

CAPITULO 5. CONCLUSION

Se cumple con los objetivos obligatorios del proyecto además de algunos objetivos extras, gracias a su método declarativo que se basa en consultas se logra crear un buscador más eficaz, debido a que el índice es parte de su base de conocimiento,

Gracias a lo aprendido en clases y en este laboratorio he adquirido nuevos conocimientos sobre este lenguaje de programación lógico, conocimiento sobre predicados, metas primarias, secundarias, Hechos, Reglas y como operan de forma lógica. Además de ser un lenguaje bastante complicado permite minimizar en gran cantidad las líneas de código que en cualquier otro lenguaje de programación sería bastante largas como en el caso de los paradigmas antes visto.

CAPITULO 6. REFERENCIAS

https://mumuki.io/chapters/8-programacion-logica
http://www.udesantiagovirtual.cl/moodle2/course/view.php?id=9365
https://es.wikipedia.org/wiki/Prolog
Escrig, M.T, Pacheco J, Toledo J, 2001, ***El Lenguaje de Programación PROLOG.***