Universidad de Alcalá

Escuela Politécnica Superior

Universidad de Alcalá

PRÁCTICA 2 – Gramática de cláusulas definidas

Conocimiento y Razonamiento Automatizado

Grado en Ingeniería Informática – Curso 2018/2019

Eduardo Graván Serrano – 03212337L

Marcos Barranquero Fernández – 51129104N

INTRODUCCIÓN

La práctica consiste en realizar un programa que permita analizar oraciones sintácticamente, utilizando las herramientas DCG de Prolog.

El programa debe recibir una oración, y diseccionarla en los distintos grupos que pueden conformarla (grupos nominales, adjetivales, etc).

Para ello, se definen distintas cláusulas gramáticas, que contienen la forma de los posibles distintos grupos. La frase dada se divide en sublistas de palabras y el programa busca cuadrar dichas sublistas con las cláusulas gramáticas definidas. Se busca que no se descarte ninguna palabra y todas queden en algún grupo, de forma que la frase quede analizada al completo.

Para poder diferenciar entre los distintos grupos, además de mantener la congruencia, se tienen definidas todas las palabras usadas en las frases propuestas en los distintos grupos (adjetivos, nombres, etc), además de algunas otras usadas.

El proyecto está dividido en varios archivos:

- **Datos:** contiene la definición de palabras clasificadas en sus grupos.
- **Comprobadores:** contiene las funciones utilizadas para comprobar que el género y número de las oraciones es congruente entre sí. También tiene las funciones auxiliares para contar el número de comas y de verbos, utilizado para optimizar el análisis de frases.
- **Frases:** contienen todas las cláusulas gramáticas utilizadas en el análisis.
- **Draw:** función proporcionada en el aula virtual para dibujar el análisis realizado por pantalla.
- **Main:** programa principal. Contiene las siguientes funciones:
    - **Analiza**(Oración): realiza el análisis de la oración proporcionada y lo imprime por pantalla. **Importante:** separar las comas de las palabras (Ej. “el hombre , que es un …”). **Filtro de oraciones:** contabilizando el número de comas y de verbos contenidos en la frase, permite ajustar las llamadas a GCDs al analizar una frase.
    - **Batería de oraciones de prueba:** al final del documento se encuentran comentadas.

DATOS

Como se ha explicado, este archivo contiene toda la base de datos sobre la clasificación sintáctica de las palabras empleadas.

Para algunas de ellas, tales como nombre, contiene también su género, número y persona (primera, segunda o tercera), para ser congruente con los verbos de la frase. En algunas otras, como las preposiciones, no tienen tanta información.

Este archivo contiene también los GCDs para clasificar las palabras sueltas en los grupos, de forma que las palabras puedan identificarse con su grupo.

COMPROBADORES

Para comprobar dos géneros o números, se tiene una función que recibe dos géneros y devuelve error si son distintos. Igual para el número.

Para comprobar género y número de un grupo o oración, se meten todos los géneros o números de la oración en una lista. Se va iterando sobre esta, comparando cada elemento con su siguiente. Si el género n es distinto del género n+1, se muestra un mensaje de error, género incorrecto, y se termina la ejecución. Se gestiona de forma igual para los números.

De esa forma podemos comparar todos los géneros y números de adjetivos, nombres, etc, llamando a esta función desde su sentencia GCD correspondiente.

Respecto a la persona y numero de los verbos, se tiene una función para verificar la concordancia entre el sujeto y el verbo.

Por otro lado, tenemos las funciones auxiliares de contar nº de verbos y nº de comas. Para hacerlo, se mete toda la frase en una lista, y se itera sobre ella. Si se encuentra un verbo, se añade a una lista auxiliar que luego se devuelve. Para ver el número de verbos, solo hay que ver la longitud de dicha lista auxiliar. Funciona igual para las comas.

FRASES

Este archivo contiene todos los GCD compuestos de una o varias palabras. Se distinguen 5 tipos distintos:

- Frases nominales: grupo de palabras que contienen un nombre. Pueden contener adverbios, determinantes, etc.
- Frases verbales: grupo de palabras que contienen un verbo.
- Frases adverbiales: aquellas que contienen un adverbio.
- Frases adjetivales: aquellas que contienen un adjetivo.

La estructura de todos estos GCD es similar:

1. Se verifican las palabras con sus grupos y se extrae su género, número y persona por argumento.
2. Se comprueba la congruencia de géneros, números y persona, si procede.
3. Opcionalmente, si quedan palabras sin clasificar en los grupos de 1), se llama a otras frases con las palabras restantes que no pertenezcan a los grupos verificados.

MAIN

El main o archivo principal es desde donde se ejecuta el programa.

En primer lugar carga la librería de frases y datos, y el archivo draw.pl que permite pintar las frases analizadas en la terminal.

Se encuentran las siguientes funciones:

ANÁLISIS

Dada una oración, devuelve el análisis de esta.

ORACIÓN

Tiene por argumentos la lista de la oración analizada, la propia oración, y la lista con palabras que no ha podido analizar. Dentro, se comprueba el número de verbos y el número de comas.

- Si se tienen 0 verbos, se llama a oracion_zero, preparada para analizar oraciones sin verbo.
  - Ejemplo: “el hombre”.
- Si se tiene 1 verbo, se llama a oracion_uno, preparada para analizar oraciones simples.
  - Ejemplo: “el hombre come”.
- Si se tienen 2 verbos, se llama a oracion_dos, nos encontramos con una oración compuesta. Por ello, debemos ver si es subordinada o explicativa. Para ello, contamos el nº de comas.
  - Si se tienen 0 comas y 2 verbos, se llama a oracion_dos, siendo posiblemente una subordinada o una coordinada.
  - Si se tienen 2 comas y 2 verbos, se llama a oracion_explicativa, que analiza oraciones compuestas coordinadas explicativas.
- Si se tienen 3 verbos, se llama a oracion_tres, optimizada para analizar oraciones de tres verbos.
  - Ejemplo: “el procesador de textos es una herramienta muy potente que sirve para escribir documentos pero es bastante lento”

ANEXO – CÓDIGO

Nota: se adjutan los dos archivos con mayor funcionalidad. Los datos y los GCDs no se adjutan por no hacer absurdamente largo este documento.

MAIN

```
:-['draw.pl'].
:-['frases.pl'].

% ----- Configuración de oraciones -----

% Nota: separar las comas (Ej. ' el procesador de textos , que es
una herramienta...')
analiza(Oracion):-
    atomic_list_concat(L, ' ', Oracion),
    oracion(LR, L, []), draw(LR).


% Clausula que llama al tipo de oracion correspondiente en funcion
del numero de verbos
oracion(ListaResuelta, Oracion, ListaSobras):-
    numero_verbos(Oracion, ListaVerbos),
    length(ListaVerbos, X),
    (
        %write("Hay "),
        %write(X),
        %writeln(" verbos. "),
        X=0 -> oracion_zero(ListaResuelta, Oracion, ListaSobras);
        X=1 -> oracion_uno(ListaResuelta,Oracion,ListaSobras);
```

```
        X=2 ->
        numero_comas(Oracion, ListaComas),
        length(ListaComas, CuantasComas),
        (
            %write("Hay "),
            %write(X),
            %writeln(" comas. "),
            CuantasComas=0 ->
oracion_dos(ListaResuelta,Oracion,ListaSobras);
            CuantasComas=2 -> oracion_explicativa(ListaResuelta,
Oracion, ListaSobras)
        );
        X=3 -> oracion_tres(ListaResuelta,Oracion,ListaSobras);
        writeln('Oracion no valida para su analisis'), fail
    ).


% Oraciones simples
% Solo FN
oracion_zero(oracion(SublistaNominal)) -->
    frase_nominal(SublistaNominal,_,_,_).

% Solo FV
oracion_uno(oracion(SublistaVerbal)) -->
    frase_verbal(SublistaVerbal,_,_,_).

% FN + FV
oracion_uno(oracion(SublistaNominal, SublistaVerbal)) -->
    frase_nominal(SublistaNominal, PersonaN, NumeroN, GeneroN),
    frase_verbal(SublistaVerbal, PersonaV, NumeroV, GeneroV),
    {concordancia_sujeto_verbo_persona(PersonaN, PersonaV)},
    {concordancia_sujeto_verbo_numero(NumeroN, NumeroV)},
    {comprobar_genero([GeneroN, GeneroV])}.

% Oracion subordinada sin nexo
oracion_dos(oracion_subordinada(Oracion1, subor(Oracion2))) -->
    oracion_uno(Oracion1),
    oracion_uno(Oracion2).

% Oracion compuesta con nexo
oracion_dos(oracion_compuesta(oracion1(Oracion1), nexo(Nx),
oracion2(Oracion2))) -->
    oracion_uno(Oracion1),
    nexo(Nx),
    oracion_uno(Oracion2).

oracion_tres(oracion_compuesta(oracion1(Oracion1), nexo(Nx1),
oracion2(Oracion2), nexo(Nx2), oracion3(Oracion3))) -->
    oracion_uno(Oracion1),
```

```
    nexo(Nx1),
    oracion_uno(Oracion2),
    nexo(Nx2),
    oracion_uno(Oracion3).

% Oracion explicativa
oracion_explicativa(oc(gn(SublistaNominal, Coma1, Nexo,
SublistaVerbal1, Coma2),gv(SublistaVerbal2)))-->
    frase_nominal_coordinada_explicativa(SublistaNominal, Coma1,
Nexo, SublistaVerbal1, Coma2, PersonaN, NumeroN, GeneroN),
    frase_verbal(SublistaVerbal2, PersonaV, NumeroV, GeneroV),
    {concordancia_sujeto_verbo_persona(PersonaN, PersonaV)},
    {concordancia_sujeto_verbo_numero(NumeroN, NumeroV)},
    {comprobar_genero([GeneroN, GeneroV])}.




% ORACIONES DE PRUEBA %
% --- APARTADO 3 --- %
%1-  oracion(X, [el, hombre, come, una, manzana], []), draw(X).
%2-  oracion(X, [la, mujer, come, manzanas], []), draw(X).
%3-  oracion(X, [maria, come, una, manzana, roja], []), draw(X).
%4-  oracion(X, [juan, ama, a, maria], []), draw(X).
%5-  oracion(X, [el, gato, grande, come, un, raton, gris], []),
draw(X).
%6-  oracion(X, [juan, estudia, en, la, universidad], []), draw(X).
%7-  oracion(X, [el, alumno, ama, la, universidad], []), draw(X).
%8-  oracion(X, [el, gato, come, ratones], []), draw(X).
%9-  oracion(X, [el, raton, que, cazo, el, gato, era, gris], []),
draw(X).
%10- oracion(X, [la, universidad, es, grande], []), draw(X).

% --- ANEXO --- %
%1-  oracion(X, [el, hombre, grande, come, la, manzana, roja], []),
draw(X).
%2-  oracion(X, [el, hombre, con, un, tenedor, grande, come, la,
manzana, roja], []), draw(X).
%3-  oracion(X, [juan, y, maria, comen, la, manzana, roja, con, un,
tenedor, y, un, cuchillo], []), draw(X).
%4-  oracion(X, [ella, hace, la, practica, de, juan], []), draw(X).
%5-  oracion(X, [el, canario, de, juan, y, maria, canta], []),
draw(X).
%6-  oracion(X, [la, blanca, paloma, alzo, el, vuelo], []),
draw(X).
%7-  oracion(X, [esta, muy, lejos, de, madrid], []), draw(X).
%8-  oracion(X, [el, es, lento, de, reflejos], []), draw(X).
%9-  oracion(X, [juan, habla, muy, claramente], []), draw(X).
```

```
%10- oracion(X, [la, esperanza, de, vida, de, un, nino, depende,
de, su, lugar, de, nacimiento], []), draw(X).
%11- oracion(X, [el, hombre, que, vimos, en, la, universidad, era,
mi, profesor], []), draw(X).
%12- oracion(X, [juan, ',', que, es, muy, delicado, ',', come,
solamente, manzanas, rojas], []), draw(X).
%13- oracion(X, [el, procesador, de, textos, ',', que, es, una,
herramienta, muy, potente, ',', sirve, para, escribir, documentos],
[]), draw(X).
%14- oracion(X, [juan, es, moreno, y, maria, es, alta], []),
draw(X).
%15- oracion(X, [juan, recoge, la, mesa, mientras, maria, toma, un,
cafe], []), draw(X).
%16- oracion(X, [compre, un, pantalon, y, una, corbata, negros],
[]), draw(X).
%17- oracion(X, [juan, y, hector, comen, patatas, fritas, y, beben,
cerveza], []), draw(X).
%18- oracion(X, [irene, canta, y, salta, mientras, juan, estudia],
[]), draw(X).
%19- oracion(X, [irene, canta, y, salta, y, sonrie, alegre], []),
draw(X).
%20- oracion(X, [el, procesador, de, textos, es, una, herramienta,
muy, potente, que, sirve, para, escribir, documentos, pero, es,
bastante, lento], []), draw(X).
```

COMPROBADORES

```
:-['draw.pl'].
:-['datos.pl'].

% Archivo con los comprobadores de concordancia %

% Comprobador de concordancia entre generos %
comprobar_genero([G1, G2|TG]):-
    concordancia_generos(G1, G2),   % Compruebo la concordancia
entre los dos primeros de la lista
    comprobar_genero([G2|TG]).    % Llamo a recursion con el
segundo de la lista y el resto de la lista

comprobar_genero([_|[]]).

concordancia_generos(Gen1, Gen2):-Gen1=Gen2.
concordancia_generos(Gen1, Gen2):-Gen1\=Gen2,writeln('El genero no
concuerda.'), fail.
```

```
% Comprobador de concordancia entre numeros %
comprobar_numero([N1, N2|TN]):-
    concordancia_numeros(N1, N2),  % Compruebo la concordancia
entre los dos primeros de la lista
    comprobar_numero([N2|TN]).   % Llamo a recursion con el segundo
de la lista y el resto de la lista

comprobar_numero([_|[]]).

concordancia_numeros(Num1, Num2):-Num1=Num2.
concordancia_numeros(Num1, Num2):-Num1\=Num2,writeln('El numero no
concuerda.'), fail.

% Comprobador de concordancia entre las personas del S.Nom y el
S.Verb %
concordancia_sujeto_verbo_persona(PersonaN, PersonaV):-
PersonaN=PersonaV.
concordancia_sujeto_verbo_persona(PersonaN, PersonaV):-
PersonaN\=PersonaV, fail.

% Comprobador de concordancia entre el numero de la persona del
S.Nom y del S.Verb %
concordancia_sujeto_verbo_numero(NumeroN, NumeroV):-
NumeroN=NumeroV.
concordancia_sujeto_verbo_numero(NumeroN, NumeroV):-
NumeroN\=NumeroV, fail.


% ----------------------------------------------------------------
------------ %
% Dada una frase, itera sobre esta añadiendo verbos o comas a una
tercera lista. %
% Después, se ve la longitud de dicha lista para poder optimizar
las pruebas.    %
% ----------------------------------------------------------------
------------ %

numero_verbos(Frase, Verbos):- num_ver_aux(Frase, [],Verbos).

num_ver_aux([Cabeza|Resto], ListaVerbos,Verbos):-
    v(Cabeza,_,_) ->
        num_ver_aux(Resto, [Cabeza|ListaVerbos], Verbos);
        num_ver_aux(Resto, ListaVerbos, Verbos).


num_ver_aux([], V,V).

% Contador de comas (oraciones explicativas)
numero_comas(Frase, Comas):- num_comas_aux(Frase, [], Comas).
```

```
num_comas_aux([Cabeza|Resto], ListaComas, Comas):-
    com(Cabeza) ->
        num_comas_aux(Resto, [Cabeza|ListaComas], Comas);
        num_comas_aux(Resto, ListaComas, Comas).

num_comas_aux([],C,C).
```