**Conocimiento y Razonamiento Automatizado**

Práctica de Laboratorio 2

**Raúl Moratilla Núñez**

**Abel López Martínez**

**Índice**

[Reparto de tareas 2](#_Toc133430758)

[Grado de cumplimiento de los requisitos 2](#_Toc133430759)

[Estructura del código 2](#_Toc133430760)

[Mejoras Realizadas 3](#_Toc133430761)

[Tablero ASCII 3](#_Toc133430762)

[Errores y aspectos no implementados 3](#_Toc133430763)

[Bibliografía 3](#_Toc133430764)

# Reparto de tareas

El reparto de tareas en la práctica de laboratorio ha sido llevado a cabo de manera colaborativa por ambos integrantes del grupo. Durante el desarrollo de la práctica, se realizaron diversas tareas que fueron distribuidas equitativamente entre los miembros del equipo. En muchos casos, estas tareas fueron realizadas de manera simultánea, lo que permitió que se avanzara en la práctica de manera más eficiente y en un menor tiempo.

Es importante destacar que, en todo momento, se mantuvo una comunicación fluida y constante entre los integrantes del grupo para asegurarnos de que cada uno estaba realizando su tarea de manera adecuada y para coordinar los trabajos que se estaban llevando a cabo simultáneamente. Además, se fomentó el trabajo en equipo y la colaboración para que todos pudiéramos aprender de la experiencia y obtener los mejores resultados posibles.

En resumen, podemos afirmar que el reparto de tareas en la práctica de laboratorio fue realizado por ambos integrantes del grupo de manera equitativa y colaborativa, con muchas de ellas llevadas a cabo en trabajo simultáneo para optimizar el tiempo y obtener los mejores resultados.

# Grado de cumplimiento de los requisitos

Hemos completado todos los requisitos obligatorios de la práctica de manera satisfactoria. Hemos seguido cuidadosamente todas las instrucciones y criterios establecidos, prestando atención a los detalles y asegurándonos de cumplir con cada uno de ellos. Hemos trabajado juntos de manera rigurosa y metódica, lo que ha permitido lograr los resultados deseados.

# Estructura de datos: COMPOUND

Para poder realizar análisis sintáctico de oraciones de todo tipo siguiendo la sintaxis del castellano, usaremos una estructura de dato nativa de prolog (es decir, no implementada por nosotros).

Un compound es una estructura de datos que consta de un functor y uno o más argumentos. El functor es un átomo que representa el nombre del compound y los argumentos pueden ser cualquier término en Prolog, incluyendo otros compounds. Por ejemplo, el compound "padre(juan, maria)" consta del functor "padre" y dos argumentos "juan" y "maria".

Los compounds se utilizan comúnmente en Prolog para representar objetos complejos y estructuras de datos, como listas y árboles. Por ejemplo, una lista de números enteros puede representarse como un compound con el functor "lista" y los argumentos que son los elementos de la lista: "lista(1,2,3,4,5)".

Por otro lado, un functor es una estructura de datos que consta de un átomo y un número de argumentos, pero a diferencia de los compounds, los functors no tienen nombres específicos para cada argumento. En cambio, se utilizan para definir predicados y relaciones en Prolog. Por ejemplo, el functor "mujer/1" define un predicado que es verdadero si su único argumento es una mujer.

Los functors se utilizan en Prolog para definir relaciones y predicados, lo que permite al programador definir la lógica de su programa en términos de reglas y hechos. Por ejemplo, la regla "madre(X,Y) :- mujer(X), padre(Z,Y)" define que X es la madre de Y si X es una mujer y Z es el padre de Y.

En resumen, tanto los compounds como los functors son tipos de datos utilizados en Prolog para estructurar la información y definir relaciones y predicados. Los compounds se utilizan para representar objetos complejos y estructuras de datos, mientras que los functors se utilizan para definir reglas y hechos en la lógica del programa.

# Estructura del código

La práctica se descompone en los siguientes archivos de código ejecutable en Prolog:

* diccionarioESP.pl

Es un set de palabras en español diferenciadas por su tipo morfológico, es decir, determinantes (det), sustantivos (n), verbos(v), adjetivos (adj), conjunciones (conj), adverbios (adv) y preposiciones (prep). A parte, cada grupo morfológico tiene una regla al inicio, que permite definir el compound de ese tipo de palabra.

* diccionarioTRA.pl

Es el mismo set de palabras que el anterior, pero además añadiendo la traducción al inglés y las reglas para pasar de una palabra en español a una en inglés.

* draw.pl

Archivo Prolog descargado desde BlackBoard que implementa la impresión de un árbol de compounds, esto nos sirve para mostrar el árbol sintáctico que queda de analizar oraciones y así poder mostrar este análisis de una manera visual.

* analisis.pl

Es el archivo principal de la implementación de la práctica. Este, internamente, está estructurado de la siguiente manera:

* Consults

Realización de consult al resto de archivos utilizados en este (equivalente a importar una librería o programa externo en otro leguaje). En este caso consult del archivo drawl.pl para poder dibujar el árbol sintáctico tras analizar las oraciones y consult de diccionarioESP.pl para acceder a todas las palabras de las que se dispone, así como de su tipo morfológico.

* Regla aplanar\_args

A

* Regla aplanar\_comp

B

* Regla aplanar\_iter

C

* Regla args\_aplanados

D

* Regla add\_args

E

* Regla buscar\_subordinada

Dentro del árbol sintáctico pasado busca una oración de relativo (subordinada) tanto en sus nodos hijos como en sus nodos hermanos.

* Regla ajustar\_compuestas\_args

Sirve para ayudar a iterar el árbol sintactico buscando si hay y en su caso donde se encuentra una posible oración de relativo, sirve de ayuda para la regla ajustar\_compuestas y para ello hace uso de la regla anterior buscar\_subordinada.

* Regla ajustar\_compuestas

Con las reglas sintácticas que hemos definido, no podemos ver donde hay una oración subordinada dentro de otra en una primera pasada, por lo que estas siempre se toman como oraciones simples, pese a ser compuestas, por lo que con esta función podemos realizar una pasada posterior recorriendo todo el árbol sintáctico, y con la ayuda de la regla anterior buscar si hay oraciones subordinadas para cambiar la oración padre de esta de simple a compuesta.

* Regla separar

Otro de los requisitos de la práctica era “La simplificación de oraciones coordinadas, subordinadas de relativo o compuestas consistirá en la conversión de cualquiera de estas oraciones en tantas oraciones simples como se componga la original.”. Para esto sirve esta regla, que busca oraciones simples dentro de una oración compuesta mayor (oración padre) y mediante esto separar en tantas oraciones simples como sea necesario. Es posible que se encuentren guiones “-” entre las oraciones simples de salida, lo cual no significa nada pero era necesario por la implementación realizada de la gramática.

* Regla separar\_args

Sirve de ayuda para la regla anterior, con el fin de poder iterar el árbol tanto en niveles horizontales (nodos hermanos) como en niveles verticales (nodos hijos).

* Regla copy\_compound

Sirve para copiar desde el primer compound pasado como argumento hacia el segundo compound pasado como argumento, pero eligiendo desde que posición quieres copiar en el primero y desde que posición quieres pegar en el segundo.

* Regla concatenar\_compound

Mediante la ayuda de la anterior regla concatena dos compounds de entrada devolviéndolo en el tercer argumento de la regla.

* Las reglas gramaticales son las siguientes
* Regla oracion
* Regla compuesta
* Regla simple
* Regla coordinada
* Regla subordinada
* Regla complementos
* Regla g\_verbal
* Regla g\_nominal
* Regla g\_adjetival
* Regla g\_adverbial
* Regla g\_preposicional
* Regla o\_prueba

Almacena mediante un índice y una lista de palabras cada una de las frases de entrada que se pedían como casos en el pdf de enunciado de la práctica

* Regla ejecutar\_pruebas

Itera para la frase i-ésima de la regla anterior, y para ella, saca el análisis sintáctico mediante compounds, dibuja el árbol de salida llamando a draw, separa las oraciones compuestas en oraciones simples y las muestra y por último se vuelve a llamar recursivamente con la oración con el índice i+1.

* traductor.pl

Es el archivo de una de las mejoras de la práctica que permite traducir oraciones de español a inglés y viceversa. Su estructura es la siguiente:

* Consults

En este caso consult del archivo diccionarioTRA.pl para acceder a todas las palabras de las que se dispone tanto en español como en inglés, así como de su tipo morfológico.

* Las mismas reglas gramaticales que en el archivo analisis.pl, pero con estas tanto para palabras en español, como para palabras en inglés.
* Regla traducir

Que dado el idioma desde el que se quiere traducir, hacia el que se quiere traducir y una oración, pasa esta de un idioma a otro, mostrándolo por pantalla.

* Reglas o\_pruebaESP, o\_pruebaENG y ejecutar\_pruebasESP y ejecutar\_pruebasENG

Son igual que en el archivo anterior, con la diferencia de que, en vez de mostrar el árbol del análisis sintáctico de cada oración, la traduce del español al inglés y viceversa, mostrándola en ambos idiomas.

# EJECUCIÓN

# Mejoras Realizadas

Las mejoras extras realizadas respectos a los elementos obligatorios de la práctica son los siguientes.

## Tablero ASCII

Se ha implementado una salida por pantalla mediante ASCII ART, usando distintos símbolos para dar un

# Errores y aspectos no implementados

Se informa que todos los aspectos solicitados en la práctica han sido implementados adecuadamente y sin errores. Se ha prestado atención a los detalles y se ha garantizado que se cumplan todas las especificaciones establecidas. No se ha encontrado ningún error o aspecto solicitado que no haya sido implementado correctamente, lo que demuestra un alto nivel de calidad en el trabajo realizado. En conclusión, se puede afirmar que la práctica ha sido completada con éxito en todos sus aspectos.

# Bibliografía

Apuntes de teoría y laboratorio disponibles en Blackboard.