UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER UNIVERSITARIO EN TECNOLOGÍAS INFORMÁTICAS AVANZADAS

TRABAJO FIN DE MÁSTER

Hacia la difuminación de OWL/LR usando el entorno de programación lógica difusa FLOPER.

Luis Miguel Gómez Tornero

Julio de 2013

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Departamento de Sistemas Informáticos

TRABAJO FIN DE MÁSTER

Hacia la difuminación de OWL/LR usando el entorno de programación lógica difusa FLOPER.

Autor: Luis Miguel Gómez Tornero

Tutor: Dr. D. Ginés Damián Moreno Valverde

Cotutor: Dr. D. Jesús Manuel Almendros-Jiménez

Julio de 2013

Resumen

Es indudable que Internet está cambiado la forma en que nos comunicamos, pero además se encuentra en plena evolución ya que es centro de una revolución hacia una sociedad del conocimiento. En la actualidad, la mayor parte del contenido de la Web está destinado a personas donde las herramientas software tradicionales no exprimen todo el potencial de ese tipo de información, ya que implica la búsqueda de personas, revisión de catálogos en tiendas online, motores de búsqueda, información multimedia, tendencias de los usuarios, y en muchos casos incluso información difusa, es decir, información imprecisa, indefinida, o que puede presentar diversas interpretaciones.

En este escenario, Prolog, basado en el paradigma de la programación lógica, ofrece muchas ventajas en la web semántica como lenguaje de programación anfitrión para consultar y procesar información de la web, e incluso para la creación de aplicaciones. Concretamente, en este campo existen multitud de trabajos de gran relevancia dentro del grupo DEC-TAU de la UCLM y que tienen entre sus objetivos el desarrollo del entorno de programación lógico-difuso FLOPER utilizando el lenguaje MALP, y además el desarrollo de la extensión "*fuzzy*" del lenguaje XPath en el ámbito de la web semántica. Fruto de ello es posible compilar, ejecutar, optimizar, y depurar códigos MALP.

Como consecuencia, la necesidad de manejar información difusa o imprecisa en lenguajes de la Web Semántica como OWL está aumentando en importancia en los últimos años y exige una forma estándar de representar dicha información. Podemos solucionar este problema extendiendo los actuales lenguajes de la Web Semántica para trabajar con incertidumbre como es el caso de FuzzyOWL2, y proporcionando un procedimiento basado en transformaciones para representar dicha información en lenguajes y herramientas actuales basados en reglas capaces de trabajar y razonar con información difusa de manera natural como es el caso del lenguaje MALP. Este TFM no es solamente un trabajo teórico, sino que se ha profundizado en la materia para crear un prototipo de analizador difuso capaz de transformar un modelo OWL 2 difuso en un programa lógico-difuso MALP completamente funcional.

Summary

There is no doubt that Internet is changing the way we communicate, heading a revolution towards the new knowledge´s society. Nowadays, most of Web´s contents are set aside for people in the case of traditional software tools which do not exploit all the potential of this kind of information, since it implies the search of people, catalogue reviews in online shops, search engine, multimedia data, trends of users and, in a great number of cases even fuzzy information, that is, imprecise and uncertain information or information with different interpretations.

The Prolog language, which is based on the logic programming paradigm, together with its fuzzy extension MALP, offer a wide range of advantages on the Semantic Web scenario for consulting and processing Web´s information, as well as for creating powerful and flexible applications (as the one proposed in this document) in this setting. During the last decade, the DEC-TAU research group of the UCLM has focused on the design of the "Fuzzy LOgic Programming Environment for Research", FLOPER in brief, for compiling, debugging and running MALP programs. The system has served too as an excellent platform for developing the FuzzyXPath tool, consisting on both a fuzzy-interpreter and a fuzzy-debugger for the classical XPath language in the field of Semantic Web.

Consequently, the necessity of handling fuzzy or imprecise information with Semantic Web languages like OWL is becoming more important at the last years, and it requires a standard way for managing that information. We can solve this problem if we extend the current languages of Semantic Web in order to work with uncertainty, taking the case of FuzzyOWL2 and providing a procedure based on transformations to mean that information into current languages and tools based on rules with the capacity for working and thinking with fuzzy information in a natural way, as occurs for instance with the MALP language. This Master Thesis is not only a theoretical work, but it is gone deeply into the matter so as to create a fuzzy analyzer prototype capable of converting a fuzzy OWL 2 model into a fuzzy logic MALP program completely functional.

Agradecimientos

Han pasado diez años desde que terminé mi trabajo fin de carrera, y aunque en esta última década hemos presenciado unos avances tecnológicos increíbles, mis agradecimientos y dedicatorias son prácticamente los mismos que entonces. Agradecer a todos los profesores que me han impartido clase en este máster ya que considero que han enriquecido enormemente mis conocimientos, en especial a Ginés por su gran interés, y ayuda proporcionada en todo momento.

Dedicatorias

Inevitablemente dedicar esta investigación a mi novia Tere, mi familia, y en especial a mis sobrinos Juan Francisco, Óscar, Cristina, Almudena, y Estela. Por último, un saludo a todas aquellas personas que han confiado en mí y me han apoyado personal y profesionalmente.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

"*Yo tengo un sueño para la web... y consta de dos partes: En la primera parte, la web se convierte en un medio mucho más eficaz para la colaboración entre personas. En la segunda parte del sueño, la colaboración se extiende a las computadoras, y las máquinas son capaces de analizar todos los datos de la Web (el contenido, los enlaces y las transacciones entre personas y ordenadores). Cuando se alcance el sueño, la Web será un lugar donde la fantasía del ser humano y la lógica de la máquina coexistirán en una combinación ideal y poderosa.*" [Ber99].

1.1 MOTIVACIÓN

La *World Wide Web* (WWW) ha cambiado la forma en que nos comunicamos unos con otros, cómo la información se difunde y se recupera, e incluso las gestiones empresariales. Se encuentra en el centro de una revolución que está transformando el mundo desarrollado hacia un conocimiento económico y, en términos generales, hacia una sociedad del conocimiento [GHHA12]. Este desarrollo también ha cambiado la forma de utilizar los ordenadores. Originalmente se utilizaban para cálculos numéricos, pero hoy su uso mayoritario es el procesamiento de información, y aplicaciones típicas que podemos encontrarnos son sistemas de bases de datos, procesamiento del lenguaje y juegos interactivos. En la actualidad, la mayor parte del contenido de Web está destinado a personas donde las herramientas software tradicionales no exprimen todo el potencial de ese tipo de información, ya que implican la búsqueda de personas, revisión de catálogos en tiendas online, motores de búsqueda, tendencias de los usuarios, y en muchos casos incluso información difusa, es decir, información imprecisa, indefinida, o que puede presentar diversas interpretaciones. Concretamente en este campo, existen

multitud de trabajos [Pen10, Mor08, Mor13, Vaz10, Vaz11] de gran relevancia dentro del grupo DEC-TAU de la UCLM y que tienen entre sus objetivos el desarrollo del entorno de programación lógico-difuso FLOPER y el desarrollo de la extensión "*fuzzy*" del lenguaje XPath en el ámbito de la web semántica.

Una ontología formaliza un dominio (biología, aviación, transporte, vocabulario, etc.) que utiliza un lenguaje formal para definir un vocabulario permitiendo compartir el significado entre aplicaciones e inferir nuevo conocimiento siendo un paso más en la consecución de la Web Semántica. En la actualidad se pueden crear un gran número de aplicaciones utilizando ontologías en combinación con las técnicas descritas en los puntos anteriores, algunas de ellas son: video vigilancia automática (intrusiones, comportamientos anómalos, amenazas), seguridad en instalaciones (vigilancia en puertos, reconocimiento del iris), navegación de robots (reconocimiento del entorno, deliberación), control del tráfico (detección de situaciones de riesgo con peatones, seguridad vial), aplicaciones basadas en contexto (computación Ubicua e inteligencia ambiental, sistemas dependientes de contexto), etc. *Straccia* propone una sintaxis para introducir incertidumbre en las ontologías capaz de ser integrado y utilizado por herramientas de modelado como Protégé instalando previamente un plugin específico [SPG+12, PGMG11].

1.2 OBJETIVOS

Durante los últimos años a través de diversos artículos, ponencias, trabajo fin de grado [Vaz10], trabajos fin de máster [Vaz11, Mor08] y tesis doctorales [Pen10, Mor13], el grupo de investigación DEC-TAU de la UCLM ha mejorado de forma considerable el diseño, implementación y aplicación del entorno de programación lógica difusa FLOPER desarrollado además diversas técnicas de transformación en el contexto difuso.

La web (semántica) es una de las áreas de aplicación más prometedoras para SWI-Prolog [Prolog]. Además, librerías como OWL RL [Alm11a] y Thea2 [WMV09] que ofrecen soporte para trabajar con OWL2 dentro de un entorno Prolog son de gran utilidad, ya que soportan la sintaxis estructural del lenguaje, además simplifican muchas tareas de programación relacionadas con las ontologías, incluyendo consultas y procesamiento. También, se pueden utilizar para construir aplicaciones completas que dependan de ontologías más complejas.

El objetivo principal del trabajo práctico realizado es crear un prototipo de analizador difuso que acepta como entrada un fichero OWL2 difuso y utilizando la librería Thea lo transforme a un fichero MALP capaz de ser procesado por la herramienta FLOPER. Por tanto, el TFM realizado no es solamente un trabajo teórico, sino que se ha profundizado en la materia para crear un prototipo de analizador difuso capaz de

transformar un modelo OWL 2 difuso en un programa lógico-difuso MALP completamente funcional.

A modo de resumen, los principales objetivos alcanzados en esta investigación como punto de partida para continuar con esta investigación son: Estudio del estado del arte actual sobre la web semántica, MALP, FLOPER, y FuzzyOWL2. Implementación práctica de un prototipo de analizador difuso realizado en Prolog que utilizando la librería Thea2 es capaz de transformar un fichero FuzzyOWL2 a un programa MALP. Crear un modelo OWL2 difuso con Protégé basado en el modelo de un concesionario propuesto por *Umberto Straccia*. Transformar el modelo anterior y validarlo mediante FLOPER lanzando diversos objetivos, trazas, y generando árboles de desplegado. Y finalmente comenzar la difusión y validación del prototipo realizado.

1.3 ESTRUCTURA DE LA MEMORIA

De manera general, la estructura de esta memoria sigue la plantilla suministrada por la UCLM, pero con algunas modificaciones con el fin de adaptar la memoria a las distintas particularidades de mi investigación. A continuación se explica de una manera resumida la organización de los distintos capítulos abordados.

Comenzando con el capítulo primero de este trabajo fin de máster, se ha hecho una introducción contextualizando esta investigación, así como especificado los objetivos perseguidos. Seguidamente en el capítulo segundo se describen las asignaturas cursadas realizando una introducción, descripción del curso, y finalmente explicando los trabajos realizados. En el capítulo tercero y cuarto se desarrolla el estado del arte de mi investigación que versa sobre la web semántica, y la programación lógica difusa. Posteriormente en el capítulo quinto se expone con un enfoque práctico el trabajo de investigación ejecutado, explicando los antecedentes, trabajos iniciales realizados y conclusiones obtenidas empleando la herramienta FLOPER. En el capítulo sexto se hace un esfuerzo por explicar el anteproyecto de tesis doctoral, se indican las conclusiones alcanzadas, y se enumera y desarrolla la propuesta con las posibles líneas futuras de investigación o anteproyecto de posible tesis doctoral mostrando la distribución temporal prevista y explicando en detalle las distintas tareas temporalizadas. Finalmente se listan alfabéticamente, ordenadas por apellido del primer autor, las referencias bibliográficas utilizadas. También se desarrolla un breve glosario con los términos más importantes tratados en esta investigación, y se concluye con un resumen del currículum vitae en el anexo A.

CAPÍTULO 2. ASIGNATURAS CURSADAS EN EL MÁSTER

Para comenzar este capítulo deseo aclarar que he cursado el Máster Universitario en Tecnologías Informáticas Avanzadas a tiempo parcial durante dos años académicos debido a que estoy trabajando y no me ha sido posible una dedicación completa y exclusiva en un único curso académico. Por tanto, las asignaturas corresponden al curso académico 2011/2012, y el Trabajo Fin de Máster al curso 2012/2013. En concreto, las asignaturas seleccionadas durante el primer y segundo cuatrimestre son las siguientes:

- ✓ Primer cuatrimestre:
  - Generación de documentos científicos en Informática.
  - Introducción a la programación de arquitecturas de altas prestaciones.
  - Sistemas avanzados de interacción persona-computador: sistemas colaborativos y computación ubicua.

- ✓ Segundo cuatrimestre:
  - Tecnología software orientada a objetos.
  - Modelado y evaluación de sistemas.
  - Programación Internet con lenguajes declarativos multiparadigma.

A continuación se explican en detalle las asignaturas cursadas, realizando primeramente una breve introducción, se continúa con una descripción general del curso, y posteriormente se expone el trabajo o los trabajos realizados en las mismas. Se deja para el final la asignatura "*programación internet con lenguajes declarativos multiparadigma*", ya que es la que mayor incidencia tiene con este Trabajo Fin de

Máster. Para finalizar el capítulo, se explica la relación entre las distintas asignaturas con la investigación realizada.

2.1 GENERACIÓN DE DOCUMENTOS CIENTÍFICOS EN INFORMÁTICA

Los contenidos de la asignatura constan de tres bloques temáticos impartidos por los profesores Dr. José Antonio Gámez Martín, Dr. Francisco Parreño Torres, y Dr. Luis de la Ossa Jiménez. Se trata de una asignatura de carácter metodológico, suministrando las bases necesarias para la iniciación a la actividad investigadora. Está relacionada, por tanto, con la mayoría de las asignaturas del máster, no por sus contenidos, pero sí por la competencia que adquiere el alumno en la búsqueda de información, capacidad crítica para revisar trabajos y saber cómo organizar la escritura de trabajos y artículos.[1]

2.1.1 Descripción del curso

A continuación se explica en detalle los bloques temáticos tratados en la asignatura a lo largo del cuatrimestre:

- Bloque 1: Metodología de la investigación, impartido por el Dr. José Antonio Gámez Martín. En las clases se trataron temas relativos al contexto investigador y metodología de investigación, las pautas a la hora de redactar un documento científico o hacer una presentación, y las técnicas actuales existentes para medir la evaluación de la investigación.
- Bloque 2: Contraste de hipótesis estadísticas, impartido por el Dr. Francisco Parreño Torres. Los temas tratados fueron: Software estadístico, estimadores, y contraste de hipótesis paramétricos y no paramétricos.
- Bloque 3: LaTeX - Un editor de textos científico, impartido por el Dr. Luis de la Ossa Jiménez. Las clases se dedicaron a realizar una introducción a LaTeX: estructura de un documento, composición de texto, entornos de texto, fórmulas matemáticas, listado de símbolos matemáticos, elementos flotantes, gráficos, bibliografía, y presentaciones.

Respecto al software y recursos utilizados en la asignatura, destacar el uso de la herramienta R-Commander, el editor LaTeX, y el acceso a diversas bases de datos actuales de divulgación científica.

[1] Datos recogidos de la guía docente de la asignatura

2.1.2 Trabajos realizados

Primeramente, se propuso la realización de un trabajo sobre el índice H barra para los alumnos en la modalidad semi-presencial. Dicho trabajo me resultó muy interesante y pude poner en práctica los conceptos estudiados en el bloque primero realizando diversas búsquedas en la *Web of Knowledge*.

Respecto al trabajo principal y final realizado, se trataba de plantear dos problemas uno para un test paramétrico y otro no paramétrico, se debía explicar el método a utilizar, en qué consistía, y la bibliografía utilizada. El trabajo se realizaba en LaTeX, y debía crearse también una presentación con la propuesta y resolución de los problemas planteados. Concretamente, mi trabajo realizado con el test paramétrico fue sobre diferencia de medias para averiguar si pueden considerarse significativamente mejores las notas de los alumnos de Informática en 4º de la ESO de dos Institutos diferentes, y el no Paramétrico sobre el contraste de *Mann-Withney* para comparar las diferencias de precios de un producto informático en dos ciudades determinadas.

Finalmente deseo destacar que considero esta asignatura de gran importancia tanto para el resto de asignaturas como para mi labor futura como investigador.

2.2 INTRODUCCIÓN A LA PROGRAMACIÓN DE ARQUITECTURAS DE ALTAS PRESTACIONES

Los contenidos de la asignatura constan también de tres bloques temáticos y fueron impartidos por los profesores Dr. Diego C. Cazorla López, Dr. Juan José Pardo Mateo, y Dr. Enrique Arias Antúnez. En esta asignatura se tratan tres temas fundamentales relacionados con la obtención de códigos más rápidos, eficientes y portables: la programación orientada a bloques, la programación paralela y la utilización de librerías de programación orientadas a la resolución de problemas de álgebra lineal numérica.[2]

2.2.1 Descripción del curso

En esta asignatura se trataron seis temas teóricos agrupados en los tres bloques mencionados anteriormente, y tres casos prácticos (uno en cada bloque) para abordar aspectos relacionados con la optimización temporal de código secuencial y la programación paralela:

- ✓ Tema 1: Introducción a la programación de altas prestaciones

[2] Datos recogidos de la guía docente de la asignatura

- ✓ Tema 2: Programación orientada a bloques
- ✓ Tema 3: Introducción a las arquitecturas paralelas
- ✓ Tema 4: Paradigmas de computación paralela
- ✓ Tema 5: Diseño de programas en arquitecturas paralelas
- ✓ Tema 6: Software de programación de arquitecturas de altas prestaciones

- ✓ Casos prácticos:
  - Optimizaciones secuenciales
  - Programación paralela con MPI y BLAS.
  - Programación paralela con BLACS y PBLAS

Por tanto, se tratan gran cantidad de temas que se ponen en práctica con la realización de tres casos prácticos y un trabajo final que a continuación se explican.

2.2.2 Trabajos realizados

Como se ha mencionado anteriormente, los trabajos realizados en esta asignatura corresponden a cada uno de los bloques temáticos impartidos por cada profesor, así como la realización de una práctica 0 como preámbulo y familiarización con el entorno de trabajo:

- Práctica 0: Introducción al entorno de trabajo, el profesor encargado de la misma fue Dr. Diego C. Cazorla López. En esta práctica se explicaba el cluster de trabajo, debíamos conectarnos de manera remota al cluster, compilar varios programas paralelos de prueba, lanzar varias ejecuciones paralelas, y finalmente observar los resultados obtenidos.
- Práctica 1: Programación Orientada a Bloques, el profesor encargado de la misma fue Dr. Diego C. Cazorla López. La práctica consistía en implementar en lenguaje C el código que permita obtener la matriz producto y analizar los tiempos teniendo en cuenta diferentes ordenaciones de bucle, parametrizando el tamaño del bloque, y utilizando las funciones de BLAS.
- Práctica 2: Programación paralela con MPI y BLAS, el profesor encargado de la misma fue Dr. Juan José Pardo Mateo. Esta práctica se trataba de escribir el código C paralelo que permita obtener la matriz producto utilizando las primitivas MPI y las funciones BLAS. Se realizó la práctica siguiendo 2 técnicas diferentes comparando los resultados: utilizando n máquinas con 1 procesador cada una, y utilizando una sola maquina con n procesadores.
- Práctica 3: Programación paralela con BLACS y PBLAS, el profesor encargado de la misma fue Dr. Enrique Arias Antúnez. Esta práctica dadas dos matrices se

debía escribir el código en C que permita obtener la matriz producto utilizando la abstracción que proporciona ScaLAPACK + PBLAS + BLACS.

- Práctica final: "Desarrollo de algoritmos para la creación de Mosaicos de imágenes mediante programación paralela con MPI". En esta práctica quise abordar un tema de interés personal para así poner a prueba la potencia de cálculo del cluster de clase utilizando todos los nodos y procesadores al mismo tiempo. Creo que el resultado final fue muy positivo y me ayudó a conocer a fondo la programación paralela mediante el paso de mensajes.

Figura 2.1: Vista general de la práctica realizada en la asignatura IPAAP

Finalmente deseo exponer que esta asignatura me resultó muy densa, pero a la vez muy interesante, y he profundizado en la programación paralela gracias a las prácticas realizadas y el trabajo final desarrollado como puede verse en la Figura 2.1. También he apreciado las ventajas e inconvenientes de este tipo de programación desarrollando, depurando y optimizando código paralelo.

2.3 SISTEMAS AVANZADOS DE INTERACCIÓN PERSONA-COMPUTADOR

Los bloques temáticos de esta asignatura son cuatro que impartieron los profesores Dr. Miguel A. Redondo Duque, Dr. Crescencio Bravo Santos, y Dra. Ana Isabel Molina Días pertenecientes al Grupo CHICO de la Escuela Superior de Informática de Ciudad Real.

- ✓ Bloque 1: Introducción a los sistemas colaborativos y ubicuos.
- ✓ Bloque 2: Entornos de modelado y análisis de la colaboración.
- ✓ Bloque 3: Sistemas colaborativos: awareness y evaluación.
- ✓ Bloque 4: Sistemas interactivos colaborativos y ubicuos.

En esta asignatura se profundiza en el estudio de los sistemas ubicuos colaborativos con especial incidencia en aquellos específicamente diseñados para la interacción persona–computador y persona–persona en sistemas de CSCL (*Computer Supported Collaborative Learning*).[3]

2.3.1 Descripción del curso

Los dos primeros bloques indicados anteriormente fueron impartidos por el Dr. Miguel A. Redondo Duque, en ellos se abordaron temas relacionados con los sistemas colaborativos y la computación ubicua, y se mostraron diversos ejemplos de sistemas colaborativos de referencia. En el tercer bloque impartido por el Dr. Crescencio Bravo Santos, se detallaron y se especificaron diversos ejemplos de Entornos Síncronos de Aprendizaje Colaborativo mediante Modelado y Simulación, se realizó un análisis proceso-solución de los entornos CSCL, y también se mostraron diversas herramientas y técnicas para el Desarrollo de Sistemas Colaborativos. El último bloque fue impartido por la Dra. Ana Isabel Molina Días. En él se realizó un análisis y diseño de Sistemas Colaborativos, y también se estudiaron varias interfaces colaborativas.

Por otro lado, cada profesor propuso la realización de una tarea práctica donde se ponga en conocimiento los conceptos teóricos tratados en los contenidos. Finalmente realicé un trabajo autoexplicativo que también comentaré a continuación.

2.3.2 Trabajos realizados

El primer trabajo realizado en los dos primeros bloques consistió en una valoración crítica de dos artículos relacionados con las tecnologías actuales para el desarrollo de sistemas que soportan computación ubicua como paradigma de interacción. El siguiente trabajo correspondiente al bloque tercero trató de identificar y describir el soporte a la colaboración de Facebook y ChessBase, así como proponer un modelo para el análisis de la colaboración. Continuando con el bloque cuarto, realicé un trabajo sobre los métodos para la evaluación de sistemas groupware. Para terminar, realicé una presentación final auto-explicativa con unas reflexiones finales que reunía y relacionaba los resultados de los distintos trabajos realizados en el curso.

[3] Datos recogidos de la guía docente de la asignatura

Sin duda una asignatura muy interesante donde se trataron temas de gran actualidad, relacionados con los sistemas colaborativos y computación ubicua, aunque eché en falta poder probar de manera práctica la herramienta COLLECE para trabajo colaborativo, ya que no se encontraba operativa en ese momento.

2.4 TECNOLOGÍA SOFTWARE ORIENTADA A OBJETOS

Como en la mayoría de las asignaturas del máster, los contenidos de esta asignatura constan de tres bloques temáticos que se impartieron por los profesores Dra. Elena María Navarro Martínez, Dra. María Dolores Lozano Pérez, y Dr. Víctor Manuel Ruíz Penichet. Dentro del plan de estudios, esta asignatura cubre los siguientes aspectos: [4]

- ✓ Definir formalmente el concepto de modelo y metamodelo. Introducción a la Ingeniería Conducida por Modelos (*Model-Driven Engineering*, MDE) desde un punto de vista práctico: desde la definición de metamodelos a su instanciación, así como la construcción de herramientas de soporte en su ciclo de vida.
- ✓ Conocer los modelos específicos empleados para el desarrollo de Interfaces de Usuario y su trazabilidad a lo largo del proceso de desarrollo. Aprender las bases de MB-UIDE (*Model-Based User Interface Development Environments*) y algunas de sus aplicaciones. Conocer algunas extensiones al lenguaje de modelado OO UML para Sistemas Interactivos e Interfaces de Usuario.
- ✓ Conocer la problemática de los nuevos entornos colaborativos y las soluciones actuales aplicadas a este tipo de entornos. Nuevos métodos y herramientas de desarrollo de sistemas colaborativos (CSCW, *Groupware*) y su aplicación a casos concretos.

2.4.1 Descripción del curso

Cada profesor impartió un bloque de contenidos de la asignatura repartiéndose de la siguiente manera:

- ✓ Bloque 1: Introducción a la Ingeniería Conducida por Modelos (MDE), impartido por la Dra. Elena María Navarro Martínez. Respecto a los temas tratados fueron los siguientes:
  - Introducción al desarrollo de software dirigido por modelos (DSDM).
  - Introducción al manejo de la herramienta EMF (*Eclipse Modeling Framework*).

[4] Datos recogidos de la guía docente de la asignatura

- Introducción al manejo de la herramienta GMF (*Eclipse Graphical Modeling Framework*).
- Introducción práctica a la transformación Modelo a Modelo.
- Introducción práctica a la transformación Modelo a Texto.

✓ Bloque 2: Tendencias actuales en el desarrollo de interfaces de usuario, impartido por la Dra. María Dolores Lozano Pérez. Los temas de estudio fueron:

- Introducción al Desarrollo de Interfaces de Usuario Basada en Modelos (MB-UIDE).
- Entorno Metodológico Basado en Modelos para el Desarrollo de GUIs.

✓ Bloque 3: Introducción al Desarrollo de Entornos Colaborativos (CSCW y *Groupware*), impartido por el Dr. Víctor Manuel Ruíz Penichet y el profesor invitado Dr. Ricardo Tesoriero. Respecto a los temas abordados en este caso fueron los siguientes:

- TOUCHE: Desarrollo de Entornos Colaborativos Orientado a Tareas y Centrado en el Usuario.
- CAUCE: Desarrollo dirigido por modelos para aplicaciones sensibles al contexto en entornos de computación ubicua.

2.4.2 Trabajos realizados

En cada bloque temático se realizó uno o varios ejercicios prácticos, y también realicé un trabajo final que a continuación se indica:

✓ Subir al repositorio las prácticas realizadas con Eclipse en el Bloque 1.
✓ Valoración crítica y conclusiones principales sobre la propuesta MDD.
✓ Desarrollo de un prototipo de interfaz para una tienda virtual de venta de libros on-line.
✓ Trabajo final: "Transformaciones XSL hacia la web semántica". Las transformaciones XSL permiten generar documentos en diversos formatos a partir de documentos XML, las posibilidades y combinaciones son enormes. En este trabajo se estudiaron diversas transformaciones en la web semántica como OWL o MPEG-7 usando el entorno Eclipse.

Finalmente indicar que tanto el trabajo final realizado como gran parte de los contenidos de esta asignatura están relacionados con la temática principal de mi TFM, por tanto, considero esta asignatura de gran interés y me ha resultado de considerable ayuda.

2.5 MODELADO Y EVALUACIÓN DE SISTEMAS

Los contenidos de la asignatura constan de tres bloques temáticos y fueron impartidos por los profesores Dr. Rafael Casado González, Dr. Luis Orozco Barbosa, y Dr. Aurelio Bermúdez Marín. Respecto a los objetivos o resultados de aprendizaje esperados son los siguientes: [5]

- ✓ Conocer y manejar diferentes simuladores utilizados en el ámbito de las investigaciones en redes de interconexión, tanto cableadas como inalámbricas.
- ✓ Aprender los principios de teoría de colas y su aplicación al modelado y evaluación de sistemas.
- ✓ Conocer el concepto general de sistema y las diferentes técnicas existentes para su modelado.

2.5.1 Descripción del curso

En el primer bloque temático impartido por el Dr. Rafael Casado González, se realiza una introducción al modelado y simulación de sistemas. El segundo bloque fue impartido por el Dr. Luis Orozco Barbosa, en él se estudiaban contenidos relacionados con la evaluación de prestaciones de sistemas informáticos. Finalmente el último bloque fue impartido por el Dr. Aurelio Bermúdez Marín, es un bloque muy práctico de simuladores de redes de interconexión.

Los simuladores utilizados en la asignatura fueron NS-2 y *OPNET Modeler*, finalmente seleccioné *OPNET Modeler* para realizar mi trabajo final de la asignatura.

2.5.2 Trabajos realizados

El trabajo final realizado trata de modelar el Cluster utilizado en la asignatura Introducción a la programación de arquitecturas de altas prestaciones utilizando la herramienta de modelado OPNET como se muestra en la Figura 2.2. Para llevar a cabo dicho trabajo realicé diversos algoritmos y experimentos tanto en el cluster real como en el modelado para comprobar hasta qué punto dicho modelo se comporta como el sistema real.

[5] Datos recogidos de la guía docente de la asignatura

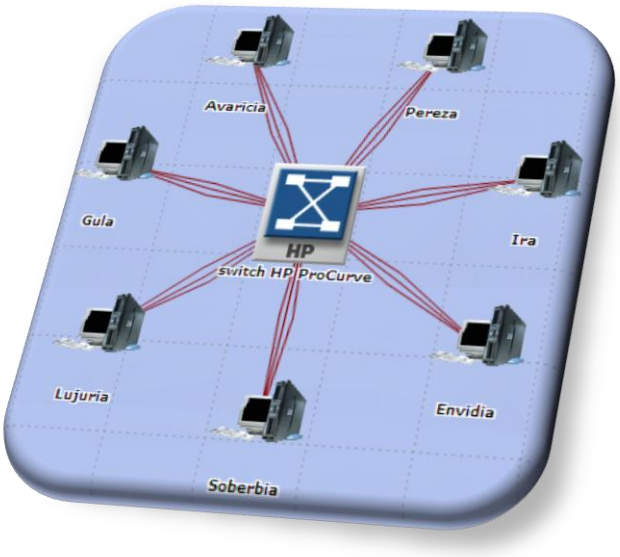


Figura 2.2: Vista general de la práctica en Modelado y Evaluación de Sistemas

Por tanto, esta asignatura me ha resultado muy interesante y gratamente satisfactoria, ya que uno de mis objetivos que perseguía era relacionar y profundizar en esta asignatura de Modelado y evaluación de sistemas con ayuda del cluster real utilizado en la asignatura del primer cuatrimestre Introducción a la programación de arquitecturas de altas prestaciones.

2.6 PROGRAMACIÓN INTERNET CON LENGUAJES DECLARATIVOS MULTIPARADIGMA

Esta asignatura se impartió por profesores pertenecientes tanto al campus de Ciudad Real como de Albacete, y fueron Dr. Pascual Julián Iranzo (campus de Ciudad Real), Dr. Francisco Pascual Romero Chicharro (campus de Ciudad Real), y Dr. Ginés Damián Moreno Valverde (campus de Albacete). Las clases del campus de Ciudad Real se siguieron mediante videoconferencia desde el campus de Albacete. En este curso se estudian lenguajes reales basados en las distintas aproximaciones declarativas y borrosas, incidiendo en sus aplicaciones prácticas y con especial énfasis en sus facilidades para la programación internet. [6]

Respecto a los objetivos que persigue la asignatura son los siguientes:

[6] Datos recogidos de la guía docente de la asignatura

- ✓ Que el alumno conozca los fundamentos y características de los lenguajes declarativos multiparadigma, que integran la programación lógica con el paradigma difuso (*fuzzy*).
- ✓ Estudiar los mecanismos operacionales y la semántica de algunas de las propuestas de integración. También, técnicas de transformación de programas que permiten optimizarlos.
- ✓ Conocer las facilidades de un lenguaje declarativo concreto para la programación internet.
- ✓ Presentar el tópico de la búsqueda en internet desde una perspectiva declarativa.

2.6.1 Descripción del curso

Los contenidos del curso se agruparon en tres bloques temáticos que a continuación explico en detalle:

- ✓ Bloque 1: Fundamentos de la programación lógica, impartido través de videoconferencia por el Dr. Pascual Julián Iranzo, se trataron los siguientes temas:
  - Fundamentos de Programación Lógica: Unificación y Resolución.
  - Programación Lógica: Semántica Operacional y Declarativa.
  - Programación lógica difusa basada en resolución débil: Teoría e implementación.
- ✓ Bloque 2: Aplicaciones de la Programación Declarativa a la búsqueda en Internet, impartido a través de videoconferencia por el Dr. Francisco Pascual Romero Chicharro. En este caso se trataron los siguientes temas:
  - La Web: Extracción de Información de la Web, *Web Mining*, Búsqueda de Información en Internet, Análisis del comportamiento y personalización, Explotación de Redes Sociales, Web Semántica, y otros temas: *Mashups*, DUM.
  - Aplicaciones: Aplicaciones de la programación Declarativa a la Web Semántica, Anotación semántica de contenidos Web mediante Bousi~Prolog, *Linked Data* en la Web.
- ✓ Bloque 3: "*programación lógica difusa con grados de verdad*", impartido por el Dr. Ginés Damián Moreno Valverde en el campus de Albacete. Finalmente los temas tratados fueron:
  - MALP "*Multi-Adjoint Logic Programming*": sintaxis y principio de ejecución.
  - El entorno FLOPER.
  - La aplicación FuzzyXPath.
  - Mejorando la eficiencia de las computaciones difusas.

- Medidas de coste computacional y depuración declarativa
- Semántica declarativa por Modelo Mínimo Difuso de *Herbrand*.
- Completitud mediante reductantes generales por evaluación parcial umbralizada y desplegado difuso.
- Nuevos modelos de igualdad.

2.6.2 Trabajos realizados

Tras consultar la bibliografía relacionada con la asignatura y documentarme sobre el estado del arte actual realicé el trabajo final sobre la representación de ontologías difusas usando OWL 2 y sus aplicaciones actuales. Este trabajo relaciona de una manera práctica las ontologías difusas con el lenguaje de programación lógico multi-adjunto que actualmente está desarrollando el grupo de investigación DEC-TAU [7] de la UCLM.

2.7 RELACIÓN DE LAS DISTINTAS ASIGNATURAS CON EL TRABAJO DE INVESTIGACIÓN

En conclusión y para finalizar el capítulo, a continuación se resume la relación de este Trabajo Fin de Máster con las asignaturas aprobadas:

- ✓ "*Generación de documentos científicos en Informática*": Me ha ayudado enormemente a la hora de buscar información científica, documentarme sobre el estado del arte, y crear documentos y presentaciones de calidad.

- ✓ "*Introducción a la programación de arquitecturas de altas prestaciones*": Al trabajar y poner a prueba el cluster de clase con el trabajo final implementado creo que puedo valorar con más profundidad de detalle las ventajas y limitaciones de los lenguajes de programación paralela frente a otros paradigmas de programación como son los lenguajes declarativos.

- ✓ "*Sistemas avanzados de interacción persona-computador*": Se trataron temas de gran actualidad relacionados con los sistemas colaborativos, la computación ubicua y el soporte a la colaboración que están fuertemente relacionados con mi TFM que trata con la Web semántica, las ontologías difusas y la reutilización de conocimiento.

[7] http://dectau.uclm.es/

- ✓ "*Tecnología software orientada a objetos*": como ya mencioné anteriormente, esta asignatura me animó enormemente en el campo de la transformación de modelos, en especial las transformaciones declarativas para la web semántica, y que es el tema central que trato en mi investigación.

- ✓ "*Modelado y evaluación de sistemas*": Esta asignatura la relacioné con la asignatura "*introducción a la programación de arquitecturas de altas prestaciones*" realizando un modelo del cluster real lo más realista posible. Además, en esta investigación he realizado un modelo con el entorno Protégé sobre un concesionario que puse a prueba mediante FLOPER lanzando distintos objetivos.

- ✓ "*Programación internet con lenguajes declarativos multiparadigma*": El trabajo final de la asignatura fue el punto de partida de esta investigación. Además, en la segunda parte del trabajo realizado se expuso un sistema basado en contexto para videovigilancia que traduce imágenes de video en código OWL donde se relacionaba con la asignatura de "*introducción a la programación de arquitecturas de altas prestaciones*" al tratarse de aplicaciones que exigen gran potencia de cálculo. De esta manera, contaríamos con la potencia de cálculo que ofrece un cluster y el motor de inferencia de la programación declarativa.

CAPÍTULO 3. WEB SEMÁNTICA

Entre este capítulo y el siguiente se explican las técnicas actuales existentes en el área de la programación lógica, y más concretamente en la programación lógica difusa para la extracción de conocimiento de la web semántica. En el primer capítulo se realiza una introducción y contextualización del problema planteado, se continuará explicando los fundamentos de la web semántica hasta alcanzar la variante difusa propuesta por Fernando Bobillo y *Umberto Straccia* [BS11b], junto con el entorno Protégé [Protege] para el diseño, edición de ontologías y adquisición de conocimiento, incluyendo además la posibilidad de trabajar con ontologías difusas. Seguidamente en el siguiente capítulo se exponen los fundamentos de la programación lógica y su importancia en la web semántica, haciendo énfasis en la librería Thea y el lenguaje de programación multi-adjunta que actualmente se encuentra en desarrollo por el grupo de investigación DEC-TAU de la UCLM.

3.1 INTRODUCCIÓN

Actualmente Internet sigue en constante cambio y evolución, ya en el año 2007 algunos estudios [Radar] apuntaban la existencia de una tendencia hacia la Web semántica como puede verse en la Figura 3.1, lo que implicaría adaptar o transformar la Web actual. La Web semántica pretende desarrollar lenguajes que faciliten la inclusión de contenido legible por las máquinas, y entre estos lenguajes destacamos OWL que está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, y para representar explícitamente el significado de términos en vocabularios y relaciones.

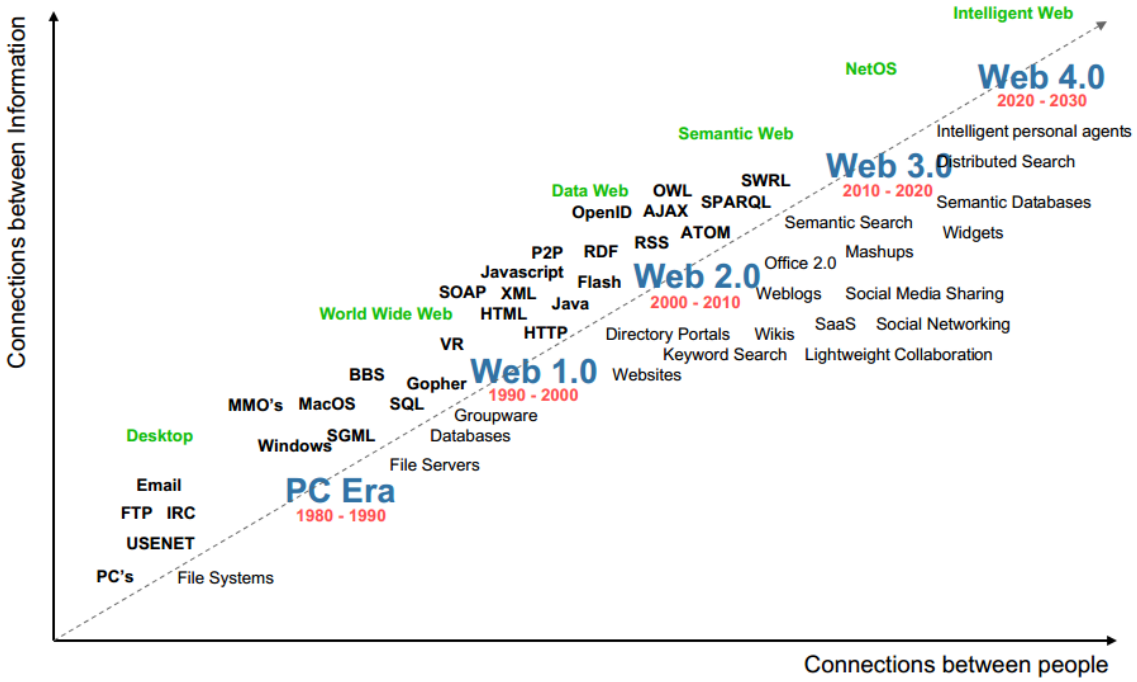


Figura 3.1: Evolución de la Web.

Los principales componentes de la Web Semántica [KRH09] son los metalenguajes y los estándares de representación XML, XML *Schema*, RDF, RDF *Schema* y OWL, así como el lenguaje SPARQL para la consulta de datos RDF. La *OWL Web Ontology Language Overview* [HD04] describe la función y relación de cada uno de estos componentes de la Web Semántica como se muestra en la Figura 3.2:

- XML aporta la sintaxis superficial para los documentos estructurados, pero sin dotarles de ninguna restricción sobre el significado.
- XML *Schema* es un lenguaje para definir la estructura de los documentos XML.
- RDF es un modelo de datos para los recursos y las relaciones que se puedan establecer entre ellos. Aporta una semántica básica para este modelo de datos que puede representarse mediante XML.
- RDF *Schema* es un vocabulario para describir las propiedades y las clases de los recursos RDF, con una semántica para establecer jerarquías de generalización entre dichas propiedades y clases.
- OWL es un lenguaje para definir ontologías mediante la descripción detallada de propiedades y clases: tales como relaciones entre clases (como disyunción), cardinalidad (por ejemplo, "únicamente uno"), igualdad, tipologías de propiedades más complejas, caracterización de propiedades (como simetría) o clases enumeradas.

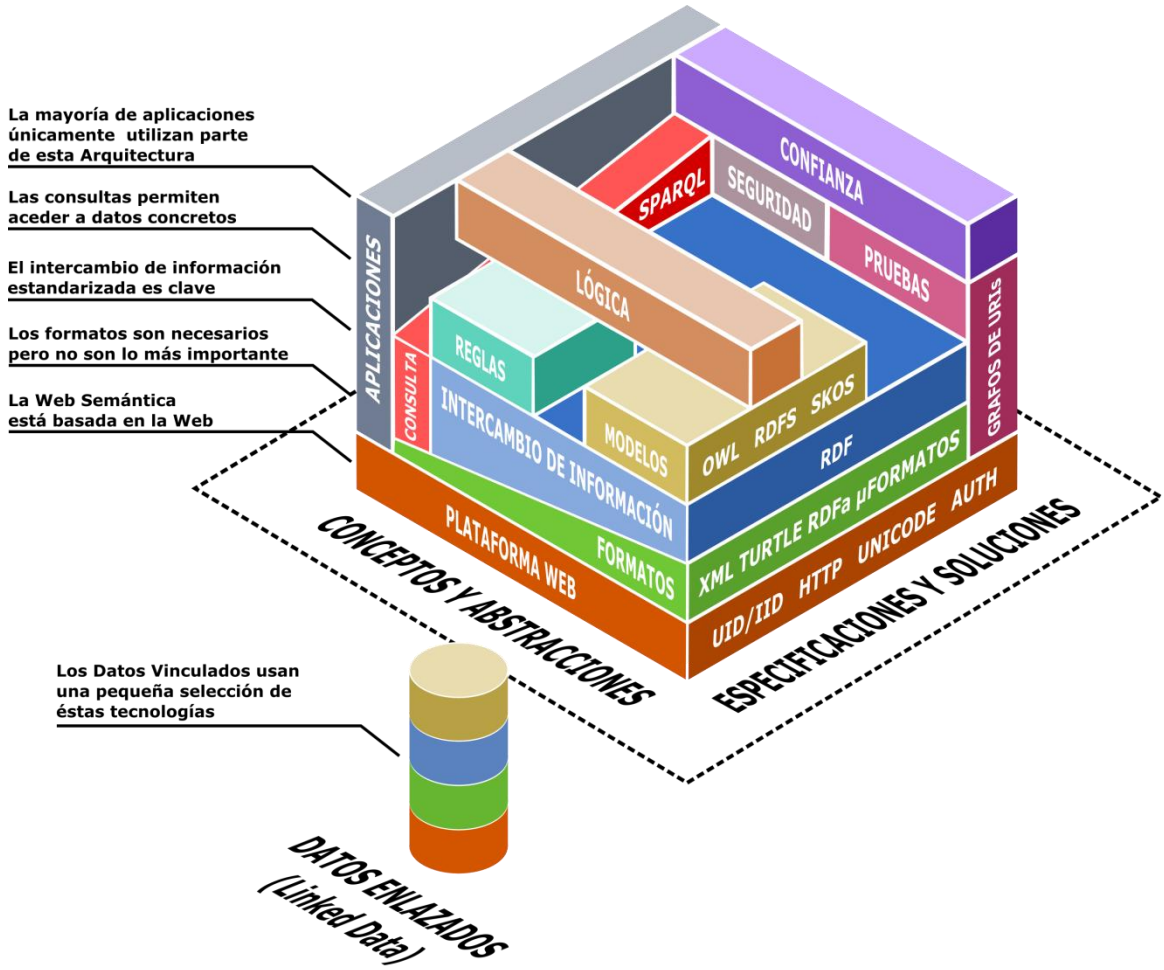


Figura 3.2: Arquitectura tecnológica de la Web Semántica

En este escenario donde están surgiendo innumerables tecnologías sustentadas por lenguajes lógicos basados en reglas y modelos de inferencia, los procedimientos para realizar transformaciones [Cri09, Alm11a, Alm11b, WMV09, GHHA12] son de vital importancia ya que permiten adaptar y transformar los contenidos de la Web a las exigencias actuales y futuras.

3.2 OWL2 DIFUSO

HTML mezcla el contenido y la presentación de la información, la metainformación que permite introducir es muy pobre, y no trata la semántica de la información. Para intentar solucionar estos problemas se propuso XML que permite estructurar la información, definir nuevas etiquetas, definir nuevos lenguajes de marcas, y actualmente es la base de muchas tecnologías actuales. Por ejemplo, XML permite definir nuevos vocabularios que dotarían de semántica a un documento, siempre y cuando todas las partes compartan el mismo vocabulario.

OWL (*Web Ontology Language*) o "Lenguaje de Ontologías para la Web", está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos [Mar07, Gru93]. OWL puede usarse para representar explícitamente el significado de términos en vocabularios y las relaciones entre aquellos términos. En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo [HD04]. Por tanto, OWL posee más funcionalidades para expresar el significado y semántica que XML, RDF, y RDFS. Además OWL va más allá que estos lenguajes pues ofrece la posibilidad de representar contenido de la Web interpretable por una máquina al dotar a cada atributo de un significado [Gru95].

En concreto, la primera versión de OWL ofrece tres sublenguajes de expresión incremental diseñados para ser usados por comunidades específicas de desarrolladores y usuarios según el nivel de expresividad que precisen como se muestra en la Figura 3.3:

- ✓ **OWL Lite**, adecuado para aquellos usuarios que necesiten una clasificación jerárquica y funcionalidades restringidas. Por ejemplo, solo permite valores de cardinalidad de 0-1.

- ✓ **OWL DL** (*Descriptions Logics*), basado en la lógica de primer orden, adecuado para aquellos usuarios que quieren la mayor expresividad sin perder la funcionalidad computacional, ya que todas las implicaciones se pueden computar, y es decidible porque todas las computaciones acaban en un tiempo finito.

- ✓ **OWL Full**, pensado para usuarios que quieren la mayor expresividad y la libertad de RDF sin garantías computacionales. Por ejemplo, en OWL Full, una clase puede ser tratada de manera simultánea como una colección de individuos y como un individuo propiamente dicho.

Figura 3.3: Sublenguajes de OWL

Por tanto, cada sublenguaje o perfil ofrece una determinada funcionalidad como pueden verse en las recomendaciones publicadas por el *World Wide Web Consortium*

(W3C) [8]. Debido a que tiene gran relación con la investigación realizada, en la Tabla 3.1 se describen de manera resumida los distintos elementos o componentes más representativos que componen una ontología OWL Lite, y en la Tabla 3.2 los que se añaden y completan con OWL DL y Full.

Características del esquema RDF	
Class	Define un grupo de individuos que comparten propiedades.
Thing	Clase de todos los individuos, es una superclase de todas las clases.
Nothing	Clase que no tiene instancias.
rdfs:subClassOf	Define que una clase es subclase de otra.
rdf:Property	Define propiedades RDF.
rdfs:subPropertyOf	Define que una propiedad es subpropiedad de otra.
rdfs:domain	Define el dominio de la propiedad.
rdfs:range	Define el rango de la propiedad.
Individual	Define instancias de clases o propiedades.
Igualdad y Desigualdad	
equivalentClass	Define que dos clases son equivalentes.
equivalentProperty	Define que dos propiedades son equivalentes.
sameAs	Define que dos instancias son iguales, es decir, son instancias equivalentes.
differentFrom	Define una instancia que debe ser diferente a las instancias especificadas.
AllDifferent	Define que todos los miembros de un conjunto son distintos mutuamente.
distinctMembers	Indica que todas las instancias de una lista son distintas.
Características de Propiedad	
ObjectProperty	Relaciona objetos con otros objetos, y propiedades con valores de objetos.
DatatypeProperty	Relaciona objetos con valores datatype, y propiedades con valores datatypes.
inverseOf	Define que una propiedad es inversa a otra.
TransitiveProperty	Define que una propiedad es transitiva.
SymmetricProperty	Define que una propiedad es simétrica.
FunctionalProperty	Define una propiedad que tiene un valor único.
InverseFunctionalProperty	Define que la inversa de la propiedad es funcional.
Restricciones de Propiedad	
Restriction	Define restricciones locales a la clase.
onProperty	Define la restricción de la propiedad que se aplicará.
allValuesFrom	Significa que todas las instancias de una propiedad son instancias de esa clase.
someValuesFrom	Significa que algunas instancias de una propiedad son instancias de esa clase.
Cardinalidad restringida	
minCardinality	Indica que una propiedad tiene al menos 0 ó 1 valores.
maxCardinality	Indica que una propiedad tiene a lo más 0 ó 1 valores.
cardinality	Indica que una propiedad sólo puede tener 0 ó 1 valor.
Información de cabecera	
Ontology	Permite adjuntar información a las ontologías.
imports	Indica la URI de una ontología que se va a incorporar a la ontología actual.
Intersección de clases	
intersectionOf	Permite establecer intersecciones entre clases identificadas y restricciones.
Tipos de datos	
xsd:datatypes	OWL usa los mecanismos de RDF para los tipos de datos.

[8] www.w3.org

Control de versiones	
versionInfo	Describir la información de control de versiones.
priorVersion	Contiene una referencia a otra ontología. Esto identifica la ontología especificada como una versión previa.
backwardCompatibleWith	Todos los identificadores de la versión anterior tienen las mismas interpretaciones esperados en la nueva versión.
incompatibleWith	Identifica la ontología como incompatible.
DeprecatedClass	Permite mantener la compatibilidad mientras se elimina una clase antigua.
DeprecatedProperty	Permite mantener la compatibilidad mientras se elimina una propiedad antigua.
Propiedades de Anotaciones	
rdfs:label	Define el nombre del recurso.
rdfs:comment	Define la descripción del recurso.
rdfs:seeAlso	Define la información acerca del recurso.
rdfs:isDefinedBy	Define quien realiza la definición del recurso.
AnnotationProperty	Permite añadir información adicional a las propiedades.
OntologyProperty	Contiene información sobre la propia ontología.

Tabla 3.1: OWL Lite

Axiomas de clase	
oneOf, dataRange	Las clases se pueden describir mediante la enumeración de los individuos que la componen.
disjointWith	Establece que las clases son disjuntas unas de otras.
equivalentClass	Define dos clases como equivalentes, por tanto, tienen las mismas instancias.
rdfs:subClassOf	Define que una clase es subclase de otra.
Combinaciones booleanas de expresiones de clase	
unionOf	Permite combinaciones booleanas arbitrarias de clases y de restricciones, en este caso de unión.
complementOf	Permite seleccionar individuos que no pertenecen a una clase.
intersectionOf	Selecciona todos los individuos que no pertenecen a dos clases.
Cardinalidad arbitraria	
minCardinality	Describe una clase que tienen al menos un número distintos valores.
maxCardinality	Describe una clase que tienen como máximo un número distintos valores.
Cardinality	Permite realizar declaraciones de cardinalidad para números enteros no negativos.
Asignación de información	
hasValue	Describe una clase para los que la propiedad en cuestión tiene al menos un valor determinado.

Tabla 3.2: OWL DL y Full

Para mejorar su expresividad y eficacia se desarrolló OWL 2 [KPP+09] que constituye la siguiente versión de OWL e incorpora recursos expresivos no incluidos en OWL, totalmente compatible con la primera versión y basado en la lógica descriptiva *SROIQ(D)* [HMP+08]. Un perfil (*profile*) o fragmento de OWL2 [CHW+09] es una versión reducida de OWL2 como restricciones sintácticas con mejores complejidades computacionales. Así, existen dos perfiles generales que engloban tres perfiles independientes entre sí, cada uno de los cuales logra la eficiencia de una manera diferente y ofrecen importantes ventajas en escenarios de aplicaciones específicas como se muestra

en la Figura 3.4. La elección entre los diferentes perfiles dependerá de la estructura y tamaño de las ontologías, la prioridad y tareas de razonamiento sobre las clases o los datos, y expresividad requerida por la aplicación:

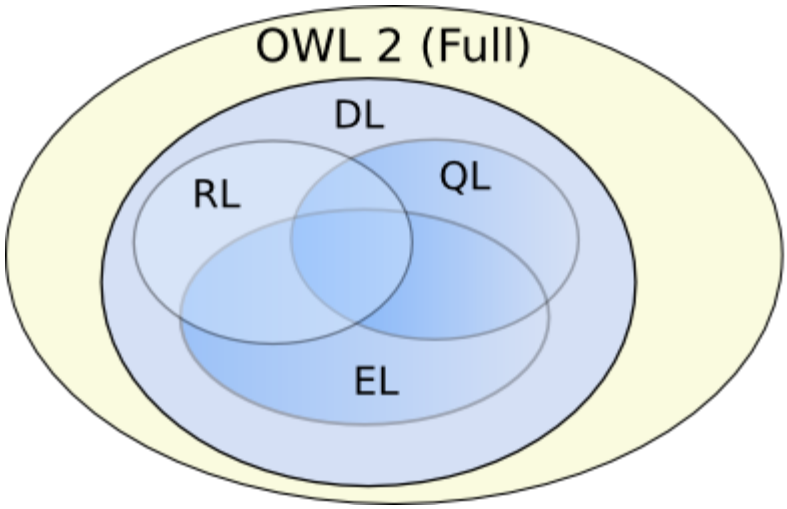


Figura 3.4: Diagrama de Venn de los perfiles de OWL 2 [W3C]

- ✓ **OWL 2 Full**: Incluye toda la funcionalidad de OWL 2, sin embargo, puede convertir el lenguaje en indecidible, lo cual hace que la construcción de razonadores sea una tarea complicada, y además no resulta práctico para muchas aplicaciones.

- ✓ **OWL 2 DL**: Versión restringida de OWL 2 Full, en este caso sí existen razonadores que cubren toda la expresividad del lenguaje.

- ✓ **OWL2 EL**: Es particularmente útil en aplicaciones con ontologías que contienen un gran número de propiedades y/o clases con baja complejidad computacional. Por tanto, es aconsejable cuando las ontologías son pesadas y lo que se busca es rendimiento y escalabilidad renunciando a una mayor expresividad.

- ✓ **OWL2 QL**: Se utiliza en ontologías con un gran número de instancias y búsqueda de respuestas como razonamiento. La potencia expresiva de este perfil es muy limitada, aunque incluye la mayoría de las características principales de los modelos conceptuales tales como diagramas de clases UML y diagramas Entidad-Relación.

- ✓ **OWL2 RL**: Se dirige a aplicaciones que requieren razonamiento escalable sin sacrificar en exceso el poder expresivo. Puede implementarse utilizando lenguajes de regla estándares junto a razonadores basados en reglas, y se define

estableciendo restricciones en la estructura de las ontologías OWL2. El acrónimo RL refleja el hecho de que el razonamiento puede ser implementado utilizando el *Standard Rule Language*. Las herramientas que ofrecen soporte para OWL 2 RL son: *Oracle Database 11g OWL Reasoner, Jena, FaCT++, Hermit y Pellet* [Agu11]. Pero OWL 2 RL tiene algunas restricciones sobre los constructores y los axiomas en comparación con OWL2 Full [Daw11]. No obstante, estas restricciones no impiden ofrecer garantías de cómputo en aplicaciones con razonadores utilizando inferencia basada en reglas. Una aplicación basada en reglas puede operar directamente sobre tripletas RDF y así se puede aplicar a cualquier grafo.

Al igual que se ha realizado anteriormente, en la Tabla 3.3 se describen de manera resumida las distintas mejoras introducidas en OWL 2 para aumentar el nivel expresivo, ampliar los tipos de datos y hacer las declaraciones comunes más fáciles.

Declaraciones comunes	
DisjointUnion	Define una clase como la unión de otras clases las cuales son disjuntas dos a dos.
DisjointClasses	Permite indicar que un conjunto de clases son disjuntas dos a dos.
NegativeObjectPropertyAssertion	Establece que una determinada propiedad no se cumple para los individuos dados.
Nuevos constructores	
ObjectHasSelf	En OWL 2 se permite afirmar reflexividad local.
ReflexiveObjectProperty	Define restricciones sobre el número de casos de una propiedad.
IrreflexiveObjectProperty	Permite asegurar que una expresión de propiedad de objeto es irreflexiva.
AsymmetricObjectProperty	Permite asegurar que una expresión de propiedad de objeto es asimétrica.
DisjointObjectProperties	Afirma que las propiedades de varios objetos son pares incompatibles.
ObjectPropertyChain	Define una propiedad como la composición de varias propiedades.
Haskey	Permite definir claves para una clase dada.
Tipos de datos	
Extra datatypes	Añade soporte para un gran rango de tipos de datos, como rdf:PlainLiteral, valores booleanos, datos binarios, IRI, instante de tiempo, etc.
Datatype Restrictions	Especifica restricciones sobre los tipos de datos imponiendo restricciones sobre el rango.
N-ary Datatypes	Incluye constructores sintácticos necesarios para los tipos de datos n-arios.
Datatype Definitions	Define tipos de datos para utilizar varias veces en una ontología.
Capacidades simples de meta modelización	
Punning	Permite mayor flexibilidad en el nombramiento de entidades.
Capacidades extendidas de anotación	
AnnotationAssertion	Permite citar entidades de la ontología y personas anónimas.
Annotation	Se utiliza para las anotaciones de los axiomas y las ontologías.
AnnotationPropertyDomain	Propiedades de anotación que tiene un dominio.
AnnotationPropertyRange	Rango de las propiedades de anotación.
SubAnnotationPropertyOf	Participa en la jerarquía de la propiedad de anotación.

Innovaciones	
Declaraciones	Asocia una categoría de entidad (class, datatype, object property, data property, annotation property, o individual).
Propiedades Bottom y Top	Proporciona propiedades sobre los objetos y las propiedades.
IRIs (Internationalized Resource Identifiers)	Identifica las ontologías y otros elementos. Es el resultado de concatenar el valor del espacio de nombres con "#" y el identificador.
Imports y control de versiones	Una ontología OWL2 puede tener una versión del IRI que se utiliza para identificar una determinada versión de la ontología.
Individuos anónimos	Se identifican usando nodos ID.
Propiedades inversas	Se pueden usar directamente en expresiones de clase.

Tabla 3.3: Mejoras OWL2

En esta investigación se profundiza en la variante difusa de OWL2 (*Fuzzy OWL2*) propuesta por *Umberto Straccia* y Fernando Bobillo en su artículo "*Fuzzy ontology representation using OWL 2*" [BS11b] que toma como base el lenguaje OWL2 para incorporarle elementos difusos, y que se encuentra actualmente en estado de aprobación por el consorcio W3C. A partir del modelo propuesto por *Gruber* [Gru95], una ontología difusa, como se muestra en la Figura 3.5, se compone de:

- **Conceptos:** conjunto difuso de objetos del dominio con características comunes.

- **Individuos:** objetos o instancias pertenecientes a una clase con un grado de verdad.

- **Roles**: conexiones binarias entre individuos relacionados con un grado de verdad.

- **Axiomas:** restricciones sobre las características de conceptos, instancias y roles.

En el que la base de conocimiento se forma por dos tipos de conocimiento diferenciado [San06]: descripción del universo o **ABox**, que es un conjunto de axiomas asertivos, y **TBox** que son el conjunto de definiciones y especializaciones, es decir, las reglas que describen un sistema mediante un vocabulario controlado.

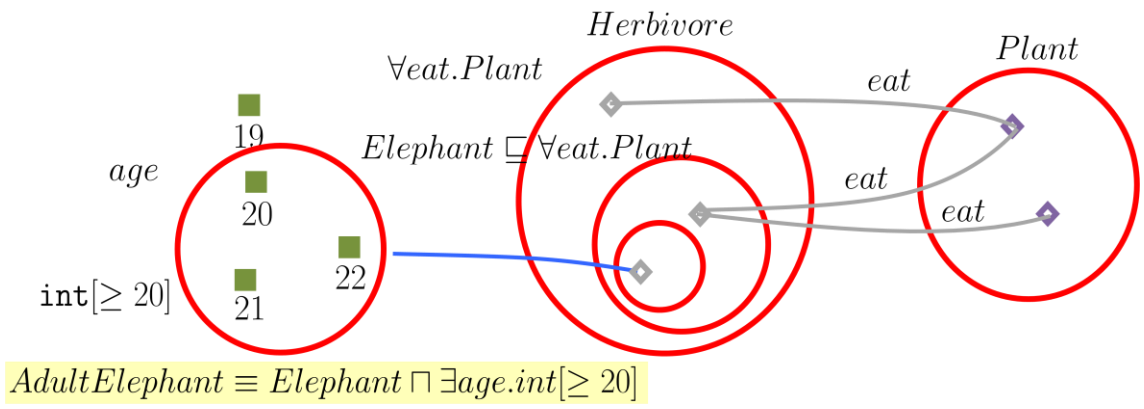


Figura 3.5: Ejemplo de representación de una ontología difusa

Respecto a los requerimientos sintácticos, existen diferencias frente a las ontologías no difusas, ya que algunos modificadores, tipos de datos, conceptos, o roles difusos no tienen equivalencia, y además algunos axiomas requieren un signo de desigualdad o un grado de verdad. Respecto a la sintaxis de las anotaciones, en lugar de utilizar la anotación por defecto de OWL 2, se utiliza la propiedad de anotación *fuzzyLabel*. La etiqueta de inicio es <fuzzyOwl2>, con un atributo para especificar el elemento difuso que se etiqueta *fuzzyType*, y la etiqueta de cierre </fuzzyOwl2>. Respecto a los modificadores lingüísticos, el valor de *fuzzyType* es "*modifier*", y hay una etiqueta "<MODIFIER>" con un atributo "*type*" (que puede ser lineal o triangular), y atributos "a, b, c" dependiendo del tipo de modificador. Los tipos de datos difusos pueden ser *left* (D1), *right* (D2), *triangular* (D3), *trapezoidal* (D4), o modificado (D5) que permite expresar, por ejemplo, que una persona es joven, como se muestra en la Figura 3.6. Respecto a los conceptos, roles y axiomas difusos, se permiten conceptos modificados, por ejemplo, *very*(C), peso en los conceptos, suma ponderada de los conceptos, o conceptos nominales. Pero existen restricciones, ya que por el momento sólo se soporta modificadores de roles difusos, y sólo se permiten algunos axiomas difusos como por ejemplo, alto >= 1.8. Finalmente, se puede especificar la lógica difusa a emplear mediante el valor de *fuzzyType*, hay una etiqueta <FuzzyLogic>, con un atributo "*logic*", que especifica la lógica difusa predeterminada a utilizar en la semántica de la ontología difusa.

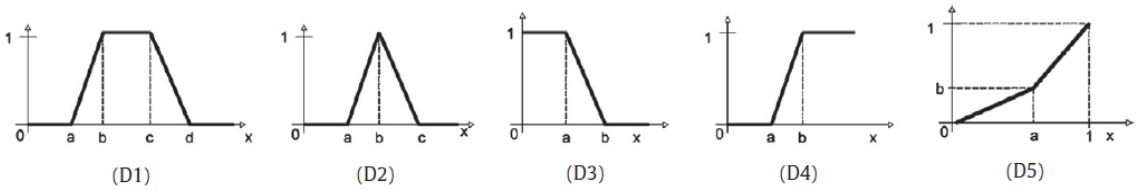


Figura 3.6: Tipos de datos difusos [BS11b, BS11a]

3.3 PROTÉGÉ

Protégé [Protege] es un editor libre de código abierto y también un sistema de adquisición de conocimiento. Al igual que Eclipse, Protégé es un entorno para el cual otros proyectos sugieren *plugins*. La aplicación está escrita en lenguaje Java y usa la librería *Swing* para crear la interfaz gráfica. Como dato se puede mencionar que cuenta con una comunidad de más de 224.000 usuarios registrados en su web. La versión actual es Protégé 4.3, y se ha desarrollado por la Universidad de *Stanford* en colaboración con la Universidad de Mánchester. A continuación se muestra en la Figura 3.7 una vista general del entorno.

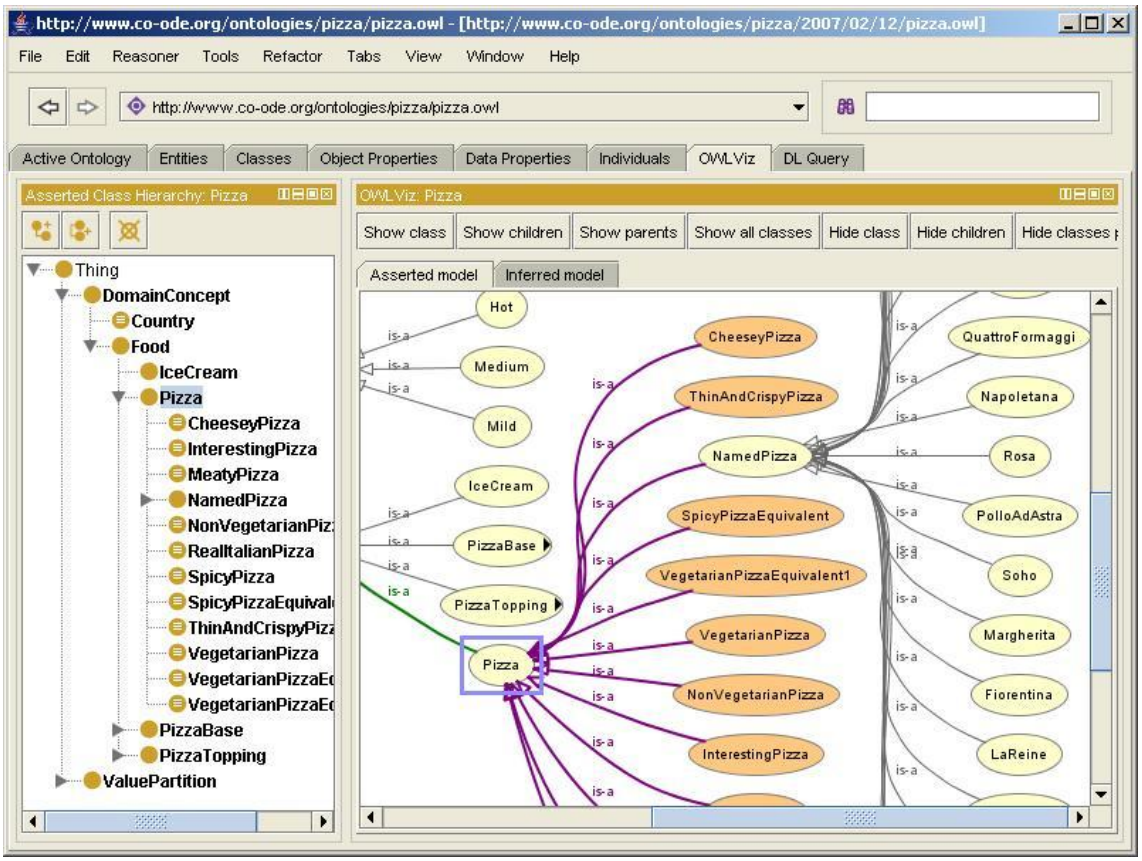


Figura 3.7: Captura de pantalla de Protégé 4 con la ontología Pizza

Entre los aspectos más destacados cabe mencionar que cuenta con una Interfaz de usuario con un diseño de componentes configurables, creación, importación y exportación de pestañas de usuario personalizables, múltiples puntos de vista de la misma ontología, clonación de componentes, atajos de teclado, soporte *drag and drop*, componentes de carga en segundo plano para mejorar la velocidad y el uso de memoria. También tiene un API para trabajar con ontologías OWL 2 ofreciendo un modelo eficiente en memoria, basado en la especificación OSGI (*Open Services Gateway Initiative*) que define una serie de API's básicas para el desarrollo de servicios fácilmente extensibles, y donde la aplicación principal está separada del editor. Respecto a la modularidad, hace un uso inteligente de repositorios locales y globales para gestionar las dependencias, puede cargar múltiples ontologías en un único espacio de trabajo, cambio dinámico entre ontologías, indicaciones y sugerencias en la interfaz de usuario, fusión de ontologías y eliminación de importaciones redundantes, y también permite movimiento de axiomas entre ontologías. Existe un historial de navegación en el que es posible realizar búsquedas globales y locales. Además cuenta con herramientas de refactorización para reestructurar el código fuente, renombrado, manejadores disjuntos, creación rápida de clases definidas, transformaciones sobre las restricciones, conversión de ID a etiquetas. Incluye también un razonamiento donde es posible inferir axiomas, consultas DL basadas en la lógica descriptiva para comprobar expresiones de clase, el razonador *FACT++* y el *Pellet* integrados en el menú, y la posibilidad de instalar *plugins* para

añadir más funcionalidades. Edición de ficheros OWL, con representación coherente de entidades ontológicas utilizando fragmentos de *URI* o anotaciones, analizador sintáctico, soporte integrado que permite deshacer cambios combinados, autocompletado, histórico, resaltado de sintaxis, y edición de reglas *SWRL*. Por tanto, su arquitectura es altamente personalizable con soporte de multitud de *plugins* incluyendo tipos de vistas, acciones en el menú, razonadores, preferencias, etc., y notificaciones para actualizar esos *plugins* a las nuevas versiones.

En concreto, para trabajar con OWL2 difuso es necesario añadir el *plugin* "*Fuzzy OWL 2 Protégé plug-in*" [BS12] quedando el entorno configurado como se muestra en la Figura 3.8. En el siguiente capítulo se explican los pasos para su correcta integración con el entorno. Por tanto, al instalar el *plugin* FuzzyOWL2 se añade un nuevo menú de trabajo llamado *FuzzyOWL* donde se pueden crear nuevos modificadores difusos, anotar reglas difusas en el modelo, o definir la lógica difusa utilizada mediante un intuitivo interfaz gráfico.

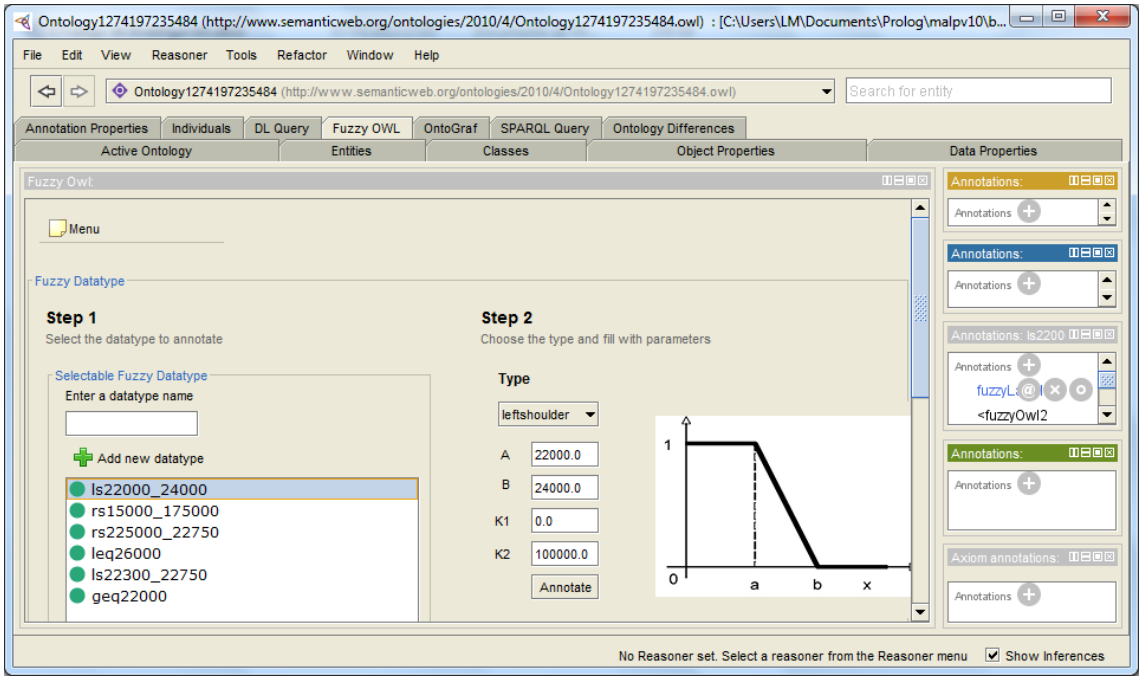


Figura 3.8: Plugin Fuzzy OWL para Protégé

CAPÍTULO 4. PROGRAMACIÓN LÓGICA

A continuación se explica en primer lugar la librería Thea que permite manipular ficheros OWL2 utilizando el entorno de programación lógico SWI-Prolog[9] [Wie09], después se describen los fundamentos de la programación lógica multi-adjunta (MALP) como un potente y flexible lenguaje que combina la programación lógica tradicional con la lógica difusa. Finalmente se muestra la herramienta FLOPER que viene desarrollando ya varios años la UCLM dentro del grupo de investigación DEC-TAU.

4.1 INTRODUCCIÓN

La programación lógica es un tipo de paradigma declarativo (como también lo es la programación funcional, que gira en torno al concepto de función) basada en fragmentos de la lógica de predicados, siendo la aproximación más popular la basada en la *Lógica de Cláusulas de Horn*. Como ejemplo más destacado de lenguaje de programación lógica contamos con PROLOG [MN00, SS94] (proveniente del francés *PROgrammation en LOGique*) ideado a principios de los años 70 y que está basado en la lógica de predicados de primer orden, en mecanismos de unificación, resolución SLD, estructuras de datos basadas en árboles, y *backtracking* automático, lo que lo convierte en un lenguaje muy potente y flexible para tratar con estándares de la Web Semántica como OWL.

Para hacer frente a las exigencias actuales, se han desarrollando varias librerías Prolog bajo el intérprete SWI-Prolog que por medio de reglas podemos inferir nuevos conocimientos a partir de una ontología dada, y por tanto son capaces de trabajar con

[9] http://www.swi-prolog.org/

ontologías cada vez más ricas y dinámicas. En este caso cabe citar la librería Thea2 desarrollada por *Jan Wielemaker*, *Chris Mungally*, y *Vangelis Vassilades* [WMV09] que trata directamente con OWL como si fueran hechos Prolog, y la librería OWL RL desarrollada por Jesús Almendros [Alm11a] basada en la librería RDF y que opera directamente sobre las tripletas OWL. En este TFM se ha utilizado la librería Thea2 que a continuación se explica, y se deja como trabajo futuro adaptar el analizador difuso a partir de la librería desarrollada por Jesús Almendros para analizar las ventajas y desventajas comparando los resultados.

4.2 THEA

Thea es una librería de Prolog que proporciona soporte completo para consultar y procesar ontologías OWL2 directamente desde los programas Prolog. Utiliza la librería RDF de SWI-Prolog para analizar y convertir ontologías, pero el núcleo es independiente de RDF y se basa en la sintaxis de estilo funcional OWL2, lo que permite la manipulación directa de los axiomas de la ontología a través de la base de datos de Prolog en lugar de hacerlo indirectamente a través de los tripletes RDF. Mientras que las versiones de Thea anteriores a la 0.5.5 de Thea [Thea] proporcionaban un reducido conjunto de predicados y entidades OWL básicas (clases, propiedades y personas), Thea 2 fue rediseñado para soportar OWL2 donde existe una correspondencia uno a uno entre cada axioma de la ontología con los hechos en la base de datos Prolog, lo que permite realizar consultas en la Ontología simplemente consultando la base de datos Prolog usando objetivos con variables como argumentos. Thea tiene el respaldo de la *Semantic Web Rule Language* (SWRL), y también ofrece capacidades adicionales, incluyendo un enlace con la API OWL de Java y la transformación de ontologías a programas de la Lógica Descriptiva.

En la Figura 4.1 se representa la estructura general donde destaca el analizador, generador, convertidor y razonador OWL. El analizador OWL utiliza la librería de la Web Semántica de SWI-Prolog para analizar trazas RDF / XML de documentos OWL en tripletas RDF y luego construir una representación de la ontología OWL con la sintaxis definida en la Web Semántica y la especificación de OWL. La sintaxis abstracta de la ontología OWL se implementa como términos Prolog. Thea ha sido ampliamente probado con muchos casos de prueba y en casi todos los casos, genera construcciones sintácticamente correctas.

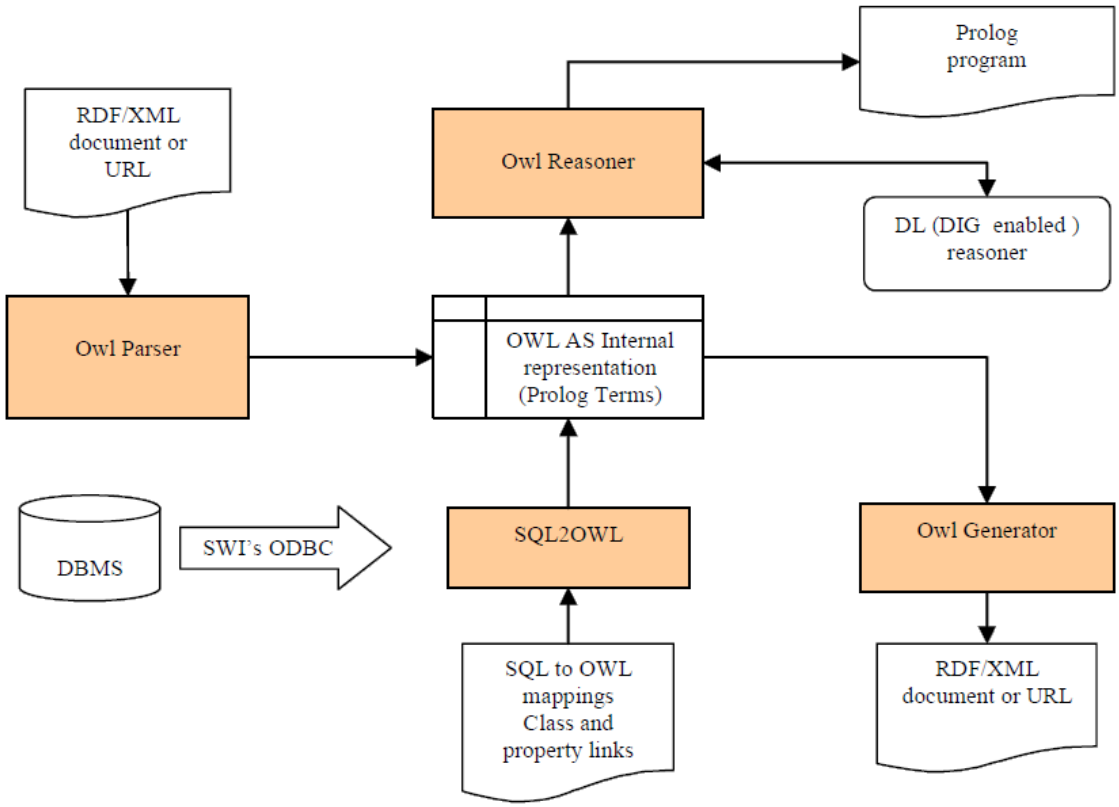


Figura 4.1: Estructura general de Thea2

A modo de ejemplo, en la Tabla 4.1 se muestra una comparación de la representación de un axioma OWL2 mediante sintaxis estructural y la forma nativa de inserción en la base de datos Prolog.

OWL2	**Prolog**
EquivalentClasses(*forebrain_neuron* *intersectionOf(neuron* *someValuesFrom(partOf forebrain)))*	*equivalentClasses(* *[forebrain_neuron,* *intersectionOf([neuron,* *someValuesFrom(partOf, forebrain)])]).*

Tabla 4.1: Comparación de la sintaxis de OWL2 frente a la de Prolog

Por su parte, el generador OWL utiliza tripletas RDF para las construcciones sintácticas abstractas de los términos Prolog y para guardar el modelo RDF resultante en un archivo RDF / XML. El conversor Thea se utiliza para generar registros en una base de datos relacional a partir de hechos OWL. Concretamente, SQL2OWL utiliza el paquete ODBC de SWI-Prolog para acceder al RDBMS (*Relational Database Management System*). La conversión es guiada por una asignación entre las entidades relacionales (tablas y columnas) y las construcciones de OWL (clases y propiedades), la asignación se define de una manera declarativa por medio de términos Prolog.

El uso de lenguajes de programación declarativos de alto nivel puede ser una ventaja cuando se trabaja con modelos ontológicos ricos y complejos. Algunos ejemplos destacados de Prolog junto con Thea para realizar diferentes tareas son:

- ✓ Consultas ontológicas: no hace falta ninguna API específica para consultar o manipular ontologías OWL2 utilizando Thea2. La coincidencia de patrones y manipulación de símbolos Prolog son suficientes, además es fácil crear nuevas reglas nombrando eficazmente las consultas.
- ✓ Procesamiento de ontologías: Aunque las ontologías pueden ser creadas y mantenidas usando entornos de desarrollo como Protégé que proporciona una interfaz gráfica de usuario para que los expertos puedan ver, crear y editar axiomas, con frecuencia existe la necesidad de automatizar tareas que serían tediosas y repetitivas haciéndolas manualmente.
- ✓ Generación de etiquetas: Uno de los retos en el desarrollo de una ontología es mantener las etiquetas de clase consistentes y que se ajusten a las normas de la comunidad. Teniendo en cuenta así los axiomas de equivalencia apropiados, se puede generar automáticamente las etiquetas.
- ✓ Traducción desde y hacia otras fuentes: Las ontologías pueden construirse de forma manual o automática. En este último caso, la ontología puede ser construida a partir de otras fuentes de datos como archivos planos, XML, o datos relacionales. En este caso, se puede sacar un buen partido a Prolog para tareas de traducción de datos debido a su naturaleza basada en reglas para analizar coincidencia de patrones.
- ✓ Aplicaciones y servicios Web: Además de proporcionar un medio expresivo de consulta, procesamiento y realización de traducciones ontológicas, es posible escribir aplicaciones completas utilizando Thea2 a través de la librería http de SWI-Prolog. Además, la distribución Thea contiene ejemplos sencillos, incluyendo un navegador básico de axiomas basado en la web.

4.3 MALP

La lógica difusa fue introducida por *Lotfi Asker Zadeh* en el año 1965 [Zad65], como una extensión de la lógica tradicional que utiliza conceptos de pertenencia más parecidos a la manera de pensar humana. En este caso, un grado de verdad puede ser un valor numérico de un intervalo, o incluso una etiqueta lingüística [Zad75, VM]. Posteriormente aparecieron diversas lógicas difusas como puede verse en la Figura 4.2 (los términos de t-norma y t-conorma se usan frecuentemente en contextos fuzzy para referirse a conjunciones y disyunciones, respectivamente).

Family	t-Norm $\alpha \otimes \beta$	t-Conorm $\alpha \oplus \beta$	Negation $\ominus\alpha$	Implication $\alpha \Rightarrow \beta$
Zadeh	$\min\{\alpha, \beta\}$	$\max\{\alpha, \beta\}$	$1 - \alpha$	$\max\{1 - \alpha, \beta\}$
Gödel	$\min\{\alpha, \beta\}$	$\max\{\alpha, \beta\}$	$\begin{cases} 1, & \alpha = 0 \\ 0, & \alpha > 0 \end{cases}$	$\begin{cases} 1, & \alpha \leq \beta \\ \beta, & \alpha > \beta \end{cases}$
Łukasiewicz	$\max\{\alpha + \beta - 1, 0\}$	$\min\{\alpha + \beta, 1\}$	$1 - \alpha$	$\min\{1 - \alpha + \beta, 1\}$
Product	$\alpha \cdot \beta$	$\alpha + \beta - \alpha \cdot \beta$	$\begin{cases} 1, & \alpha = 0 \\ 0, & \alpha > 0 \end{cases}$	$\begin{cases} 1, & \alpha \leq \beta \\ \beta/\alpha, & \alpha > \beta \end{cases}$

Figura 4.2: Algunas de las lógicas difusas más populares [BS11b]

Desde entonces, la expresión "lógica difusa" ha tenido diversas interpretaciones (Moreno, 2008; Moreno, 2013), pero la que se trata en este trabajo es una extensión de la lógica multivaluada basada en el paradigma de inferencia bajo imprecisión con infinitos valores de verdad en un intervalo cerrado con el orden parcial. Este tipo de lógica sigue el espíritu de la lógica clásica al definir nociones semánticas y perseguir la especificación de sistemas deductivos (axiomáticos) que preserven la corrección y completitud. Y en concreto, la Programación Lógica Multi-Adjunta, MALP [OVM01a, OVM01b, OVM04], puede verse como un conjunto de reglas con un grado de verdad asociado, y un objetivo es una pregunta al sistema más una sustitución que son evaluados en dos fases computacionales encadenadas:

- ✓ Fase operacional: se aplican de manera sistemática los pasos admisibles mediante un proceso de razonamiento hacia atrás, de manera análoga a la resolución SLD en la programación lógica clásica. Finalmente se devuelve una sustitución computada junto con una expresión donde todos los átomos han sido explotados.

- ✓ Fase interpretativa: Posteriormente la expresión resultante en la fase anterior es interpretada en un retículo concreto, devolviéndose un par (grado de verdad; sustitución).

En programación lógica multi-adjunta [OVM04, MPI09], se trabaja con un lenguaje lógico de primer orden, $\mathcal{L}$, que contiene variables, símbolos de función, símbolos de predicado, constantes, cuantificadores (∀ y ∃), y varios conectores como implicaciones ($\leftarrow_1$, $\leftarrow_2$, ..., $\leftarrow_m$), conjunciones ($\&_1$, $\&_2$, ..., $\&_k$), disyunciones ($\vee_1$, $\vee_2$, ..., $\vee_l$), y operadores de agregación o agregadores ($@_1$, $@_2$, ... , $@_n$), que se utilizan para combinar y propagar los valores de verdad a través de las reglas, y por lo tanto aumentar la expresividad del lenguaje. Los operadores de agregación cuando se interpreta como una función de verdad puede tener un significado aritmético, de suma ponderada o, en general, de cualquier aplicación monótona cuyos argumentos son valores de un retículo $\mathcal{L}$ completo acotado. Además, el **lenguaje** $\mathcal{L}$ contiene valores del retículo multi-adjunto en la forma $\langle \mathcal{L}, \preccurlyeq, \leftarrow_1, \&_1, ..., \leftarrow_n, \&_n \rangle$, equipado con una colección de pares adjuntos $\langle \leftarrow_i, \&_i \rangle$ en donde cada & es un conjuntor destinado a la evaluación del modus ponens.

Una **regla** es una fórmula "A $\leftarrow_i$ B with α", donde A es una fórmula atómica (normalmente llamada cabeza), B (que se llama cuerpo) es una fórmula construida a partir de fórmulas atómicas B_1, ..., B_n ($n \geq 0$), los valores de verdad de $\mathcal{L}$ y conjunciones, disyunciones y agregaciones, y finalmente, $\alpha \in \mathcal{L}$ es el peso o grado de verdad de la regla. Las reglas cuyo cuerpo es τ se llaman **hechos** (habitualmente, se representa un hecho como una regla con cuerpo vacío). Un **objetivo** es un cuerpo planteado como una pregunta al sistema. Las variables de las reglas están universalmente cuantificadas. El conjunto de valores de verdad $\mathcal{L}$ puede ser el soporte de cualquier retículo acotado completo, como ocurre por ejemplo con el conjunto de números reales en el intervalo [0,1] y su correspondiente ordenamiento $\leq$.

Por ejemplo, el siguiente programa de $\mathcal{P}$ se compone por tres reglas con un retículo multi-adjunto asociado $\langle [0,1], \leq, \leftarrow_P, \&_P \rangle$, donde la etiqueta p denota la lógica del producto con las siguientes definiciones de sus conectivos:

$$\leftarrow_P(x,y) = \min(1,x/y) \qquad \&_P(x,y) = x * y \qquad |_P(x,y) = x + y - x * y$$

$$\begin{array}{llll} R_1: & p(X) & \leftarrow_P \; q(X,Y) \,|_P\, r(Y) & \textit{with} \;\; 0.8 \\ R_2: & q(a,Y) & \leftarrow & \textit{with} \;\; 0.9 \\ R_3: & r(b) & \leftarrow & \textit{with} \;\; 0.7 \end{array}$$

Para describir la semántica procedural de MALP, a continuación se denota por C[A] una fórmula donde A es una sub-expresión (normalmente un átomo) que se produce en el contexto posiblemente vacío C[] mientras que C [A / A'] significa la sustitución de A por A' en el contexto C[], y mgu(E) es el unificador más general de un conjunto de ecuaciones E. El par $\langle$ Q; σ $\rangle$ compuesto por un objetivo y una sustitución se llama un estado. Por lo tanto, teniendo en cuenta un programa $\mathcal{P}$, un paso admisible de computación se formaliza como un sistema de transición de estados, cuya relación de transición $\rightsquigarrow^{AS}$ procede de forma muy similar a los pasos de resolución de Prolog: primero se selecciona una regla de programa *"H* $\leftarrow_i$ *B with α",* después se unifica su cabeza *H* con el átomo seleccionado *A* obteniéndose un unificador σ, a continuación se reemplaza el átomo *A* por el grado de verdad de la regla junto con su cuerpo (lo que viene dado por la expresión "$\alpha \;\&_i\; B$") y finalmente se aplica σ sobre el nuevo estado.

La siguiente derivación ilustra la definición anterior:

$$\begin{array}{ll} \langle p(X);\{\}\rangle & AS^{R1} \\ \langle 0.8 \;\&_P\; (q(X_1,Y_1) \,|_P\, r(Y_1));\{X/X_1\}\rangle & AS^{R2} \\ \langle 0.8 \;\&_P\; (0.9 \,|_P\, r(Y_2));\{X/a, X_1/a, Y_1/Y_2\}\rangle & AS^{R3} \\ \langle 0.8 \;\&_P\; (0.9 \,|_P\, 0.7);\{X/a, X_1/a, Y_1/b, Y_2/b\}\rangle & \end{array}$$

La fórmula final se puede interpretar directamente en el retículo $\mathcal{L}$ para obtener la respuesta computada difusa o f.c.a. definitiva. Así, puesto que $0.8 \,\&_P\, (0.9 \,|_P\, 0.7) = 0.8 * (0.9 + 0.7 - (0.9 * 0.7)) = 0.776$, el objetivo p(X) es cierto en un 77,6% cuando X es a.

4.4 FLOPER

FLOPER (*Fuzzy LOgic Programming Environment for Research*) es un intérprete de programas difusos realizado en Prolog que además permite dibujar árboles de ejecución parciales. La programación lógica difusa es una interesante y creciente área de investigación que aglutina los esfuerzos de la lógica difusa en la programación lógica, con el fin de incorporar más recursos expresivos a estos lenguajes para hacer frente a la incertidumbre y el razonamiento aproximado [10]. Concretamente, el enfoque de la Programación Lógica Multi-adjunta representa un paradigma reciente y extremadamente flexible de la lógica difusa para el que se ha diseñado la herramienta FLOPER [MM08a, Mor08, MM08b, Mor13, Vaz11]. Actualmente la herramienta es un prototipo es capaz de traducir directamente programas de lógica difusa en el código de Prolog con el fin de ejecutarlo en cualquier intérprete Prolog estándar de una manera completamente transparente al usuario final.

Por otra parte, la herramienta también genera una representación a bajo nivel con capacidades de depuración (trazas) y abre la puerta al diseño de tareas más avanzadas para la manipulación de programas como la optimización y especialización de programas. Por tanto, el diseño e implementación de estos lenguajes difusos y las herramientas de programación asociadas podrían desempeñar un papel importante en el desarrollo de aplicaciones software avanzadas para la medicina, control industrial, videovigilancia, procesamiento del lenguaje, motores de búsqueda, etc. como también lo demuestra la aplicación realizada en este TFM.

A continuación en la Figura 4.3 se observa la interfaz de usuario donde el menú principal está en la parte superior para gestionar los ficheros y configurar las opciones del entorno, barra de herramientas debajo del menú principal que contiene accesos directos a las opciones más comunes, el panel de proyecto en la parte izquierda donde se muestra el proyecto y sus archivos, el panel de edición en la parte central superior para mostrar los archivos del proyecto en diferentes pestañas, y ventana de salida en la parte central inferior que muestra los resultados de salida que se van produciendo.

[10] http://dectau.uclm.es/floper/

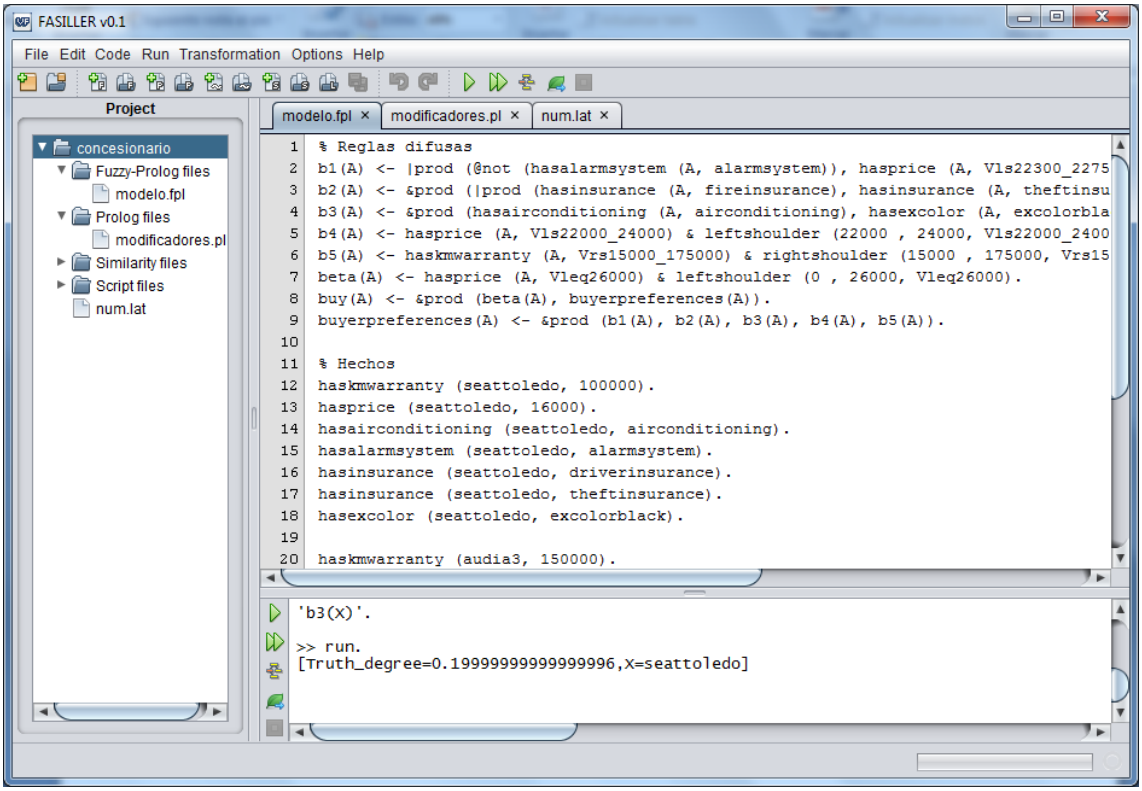


Figura 4.3: Vista general interfaz gráfica FLOPER

Por tanto, de manera gráfica se puede administrar proyectos, lanzar objetivos difusos, usar retículos, ejecutar scripts y programas, mostrar árboles de desplegados y depurar trazas de ejecución. En el siguiente capítulo se detalla los pasos de ejecución y traza con el código generado a partir del analizador difuso implementado en esta investigación. También se realiza una demostración práctica de la herramienta lanzando varios objetivos, generando trazas y árboles de desplegado.

CAPÍTULO 5. TRABAJO DE INVESTIGACIÓN: "*Hacia la difuminación de OWL/LR usando el entorno de programación lógica difusa FLOPER*".

La web (semántica) es una de las áreas de aplicación más prometedoras para SWI-Prolog [Prolog], además Prolog maneja el modelo RDF de forma natural, donde RDF proporciona un modelo estable para la representación del conocimiento con semánticas compartidas. Actualmente ya existen implementadas diversas librerías Prolog específicas capaces de trabajar con archivos OWL. En concreto, en los trabajos de Jesús Almendros [Alm11a, Alm11b, SB13, Alm12] se describe el desarrollo de una librería Prolog para OWL RL. OWL RL ha sido recientemente propuesto por el consorcio W3C como un subconjunto de OWL 2 para que el razonamiento se pueda hacer eficientemente. De este modo, por medio de reglas Prolog podemos inferir nuevos conocimientos a partir de una ontología dada. La librería OWL RL trabaja dentro del intérprete SWI-Prolog y se basa en la librería RDF de SWI-Prolog, de tal manera que los tripletes OWL se calculan y almacenan en memoria secundaria. Otra librería de gran relevancia y aceptación es Thea [WMV09], se trata de una librería creada para Prolog y que se puede encontrar dentro de la web oficial de SWI-Prolog [Prolog], y que proporciona soporte completo para consultar y procesar ontologías OWL2 directamente desde programas Prolog.

5.1 ANTECEDENTES

El objetivo principal del trabajo práctico realizado es crear un prototipo de analizador difuso que acepta como entrada un fichero OWL2 difuso y utilizando la librería Thea lo transforme a un fichero MALP capaz de ser procesado por la herramienta FLOPER. Hoy en día una piedra fundamental de la web semántica para su uso en el diseño de ontologías son las lógicas de descripción. Las lógicas de descripción [CMN+10, LS08, Stra01], también llamadas lógicas descriptivas (*DL* por *Description Logics*) son una familia de lenguajes de representación del conocimiento que pueden ser usados para representar conocimiento terminológico de un dominio de aplicación de una forma estructurada y formalmente bien comprendida [San06]. Para una explicación más detallada y precisa del trabajo práctico realizado, se ha desarrollado un modelo difuso construido a partir del modelo de un concesionario descrito por *Umberto Straccia* [BS11b] en el que un comprador (*Buy*) tiene una serie de preferencias (*B1 a B5*) y requerimientos a la hora de comprar un automóvil como es el precio, color, sistema de alarma, aire acondicionado, kilómetros de garantía, o tipos de seguros. A continuación se comienza presentando el modelo general seguido con la notación de la Lógica Descriptiva [Stra05, Stra01, Alm11b]:

Buy ≡ BuyerRequirements ∩ BuyerPreferences

BuyerRequirements ≡ ∃hasPrice.leq26000

BuyerPreferences ≡B1 ∩ B2 ∩ B3 ∩ B4 ∩ B5

B1 ≡ (¬ ∃hasAlarmSystem.AlarmSystem) ∪ ∃hasPrice.ls22300-22750

B2 ≡ (∃hasInsurance.DriverInsurance) ∩ ∃hasInsurance.(TheftInsurance ∪ FireInsurance)

B3 ≡ (∃hasAirConditioning.AirConditioning) ∩ ∃HasExColor.(ExColorBlack)

B4 ≡ ∃hasPrice.ls22000-24000

B5 ≡ ∃hasKMWarranty.rs15000-175000

Además, no todas las preferencias son igual de importantes (recordamos que el grado de verdad en nuestro caso está definido como un número real en el intervalo [0,1]), así la preferencia *B1* tiene una importancia de 0.3, *B2* tiene 0.4, *B3* tiene 0.2, *B4* tiene 0.8, y *B5* tiene una importancia de 0.6, es decir, el doble que la preferencia *B1*. Esto se implementa añadiendo un peso (*weight*) en la etiqueta difusa *fuzzyOwl2*. Por otro lado, *leq26000*, *ls22300–22750*, *ls22000–24000*, y *rs15000–175000* son modificadores difusos [YK95, CAF+13] o variables lingüísticas [Zad75], y representa un rango de valores posibles para un hecho concreto. Continuando con el desarrollo del modelo, utilizando la herramienta de modelado Protégé [Protege] se construye el modelo gráficamente como se puede apreciar en la Figura 5.1. Posteriormente, instalando el *plugin* propuesto por *Straccia* [BS12] se trabaja con elementos difusos a partir de una variante de OWL denominada FuzzyOWL2 como se muestra en la Figura 5.2.

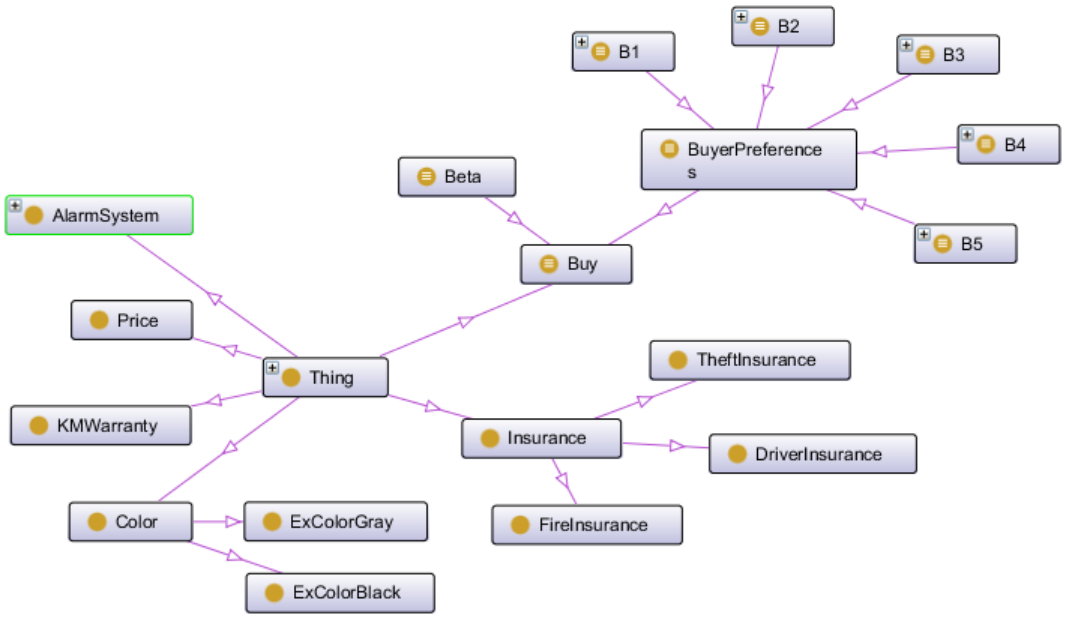


Figura 5.1: Vista general del modelo Concesionario con Protégé

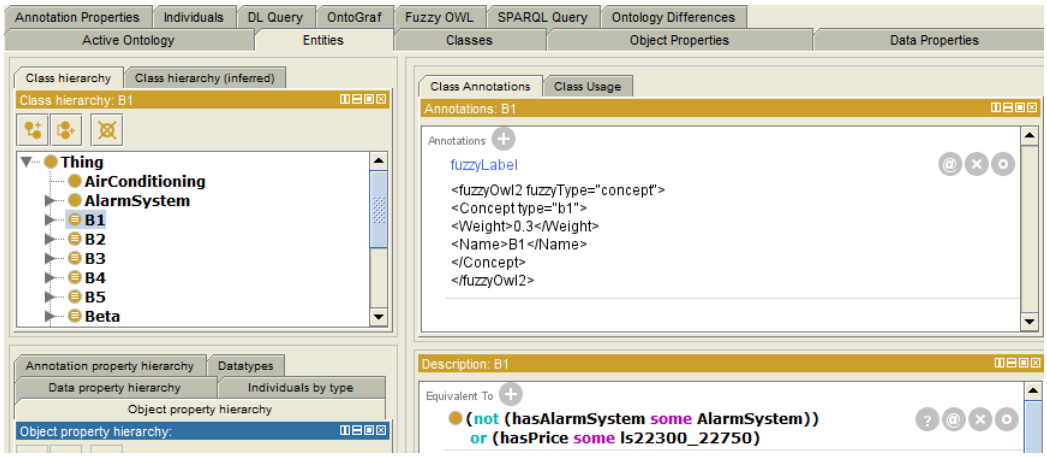


Figura 5.2: Añadiendo elementos difusos con Protégé

Clase	Descripción	Peso
B1	**(not (hasAlarmSystem some AlarmSystem)) or (hasPrice some ls22300_22750)**	0.3
B2	**((hasInsurance some FireInsurance) or (hasInsurance some TheftInsurance)) and (hasInsurance some DriverInsurance)**	0.4
B3	**(hasAirConditioning some AirConditioning) and (hasExColor some ExColorBlack)**	0.2
B4	**hasPrice some ls22000_24000**	0.8
B5	**hasKMWarranty some rs15000_175000**	0.6
Beta	**hasPrice some leq26000**	1

Tabla 5.1: Resumen de las preferencias y requerimientos del modelo

Finalmente, a modo de resumen se muestra en la Tabla 5.1 las preferencias y requerimientos obtenidos del modelo desarrollado en Protégé.

5.2 TRABAJOS INICIALES: PROTOTIPO DE ANALIZADOR DIFUSO

Como se ha indicado anteriormente, el objetivo principal es crear un analizador que reciba como entrada un fichero OWL2 difuso basado en la sintaxis que *Umberto Straccia* describe en su artículo [BS11b] y en su web [BS12], para generar como salida un fichero lógico-difuso con sintaxis MALP. En concreto, el analizador es capaz de trabajar y generar reglas difusas, hechos, modificadores difusos (variables lingüísticas), e incluso se puede indicar la lógica difusa a emplear. De manera general, el procedimiento seguido se puede clasificar en tres fases:

1) **Inicialización**: En esta fase se detallan las herramientas y configuraciones necesarias para trabajar con los modelos FuzzyOWL2.

2) **Transformación:** Tras cargar el analizador difuso en el entorno SWI-Prolog y lanzar su ejecución invocando a la función "*main*", se carga un fichero FuzzyOWL2 (con extensión *.owl*) y se obtiene un fichero de salida MALP (con extensión *.flp*).

3) **Análisis de resultados:** Finalmente, el fichero *.flp* resultante se carga en FLOPER para estudiar los resultados ejecutando distintos objetivos y generando diversas trazas gráficas. Esta fase se explica en detalle en el apartado de conclusiones.

Fase 1: Inicialización: Como paso previo, se debe instalar en el editor de modelos Protégé el plugin para poder trabajar con ficheros FuzzyOWL2 [BS12], esto permite trabajar con el modelo de una manera gráfica mucho más cómoda y precisa que editando directamente el fichero fuente. Para integrar el plugin FuzzyOWL2 en Protégé básicamente debemos seguir los siguientes pasos:

- Descargar el archivo comprimido.
- Extraer el archivo FuzzyOWL2PlugIn.jar y copiarlo dentro del directorio plugins de Protégé.
- Copiar el directorio dlib y todo su contenido dentro del directorio Protégé.

Una vez aplicados correctamente los pasos anteriores, tras volver a iniciar el editor de modelos Protégé se puede observar una nueva pestaña en la Figura 5.3 llamada "*Fuzzy OWL*" y que nos permite definir gran cantidad de elementos difusos.

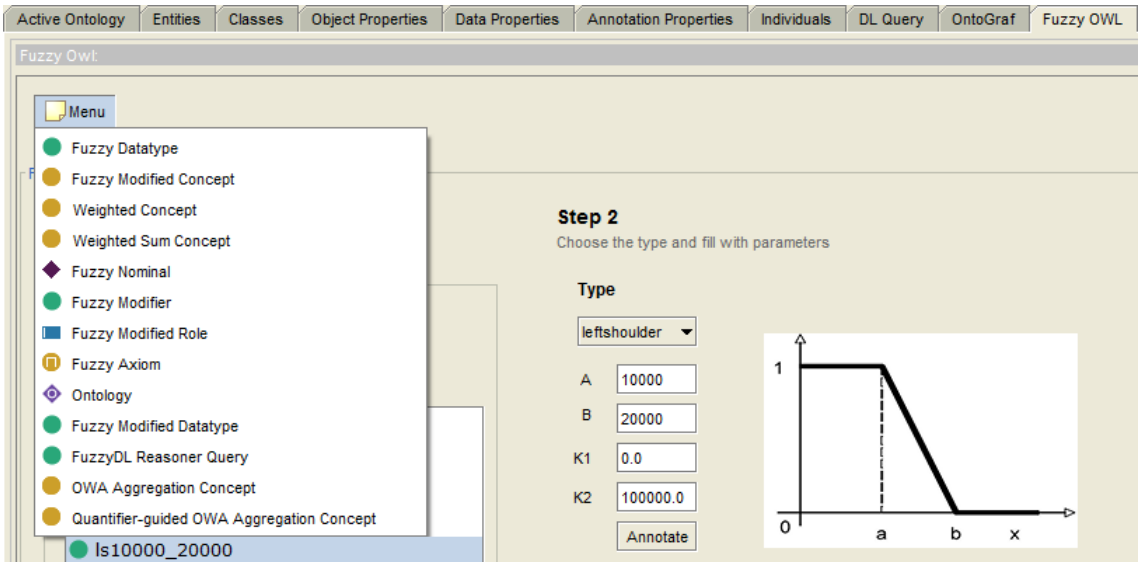


Figura 5.3: Pestaña para añadir modificadores difusos con Protégé

Fase 2: Transformación: La mayor carga del trabajo práctico realizado en este TFM ha consistido en crear un programa Prolog para SWI-Prolog que empleando la librería Thea permita analizar y transformar un fichero de entrada FuzzyOWL2 a un fichero de salida MALP analizando el contenido de etiquetas OWL2 como *fuzzylabel*, *annotationAssertion, someValuesFrom, equivalentClasses, propertyAssertion, owl:Class, owl:intersectionOf, owl:unionOf, owl:complementOf, owl:Ontology, owl:NamedIndividual,* y *rdfs:Datatype* en busca de información difusa (se puede consultar el capítulo anterior donde se explican resumidamente estas etiquetas). A continuación se procede a describir las tareas y transformaciones más importantes realizadas.

a) **Reglas difusas**: En este caso, se toma como ejemplo la regla "*b1*" para indicar los pasos de transformación realizados a partir del fichero de entrada FuzzyOWL2 hasta generar las reglas difusas MALP correspondientes. Para tener una visión más comprensible del modelo, sin entrar en detalle en el código OWL, se muestra a continuación en la Figura 5.4 una captura del modelo Protégé donde se aprecia la etiqueta difusa *fuzzylabel* que asocia a la regla *b1* un peso de 0.3.

Posteriormente, utilizando las librerías Thea de SWI-Prolog para la web semántica se transforma el fichero FuzzyOWL2 a Prolog (PL), a falta de extraer posteriormente la información difusa:

```
equivalentClasses(B1, unionOf(complementOf(someValuesFrom(hasAlarmSystem,
    AlarmSystem)), someValuesFrom(hasPrice,ls22300_22750))).
annotationAssertion(fuzzyLabel,B1,literal(<fuzzyOwl2
    fuzzyType="concept">\n<Concept type="b1">\n
    <Weight>0.3</Weight>\n<Name>B1</Name>\n</Concept>\n</fuzzyOwl2>)).
class(B1).
```

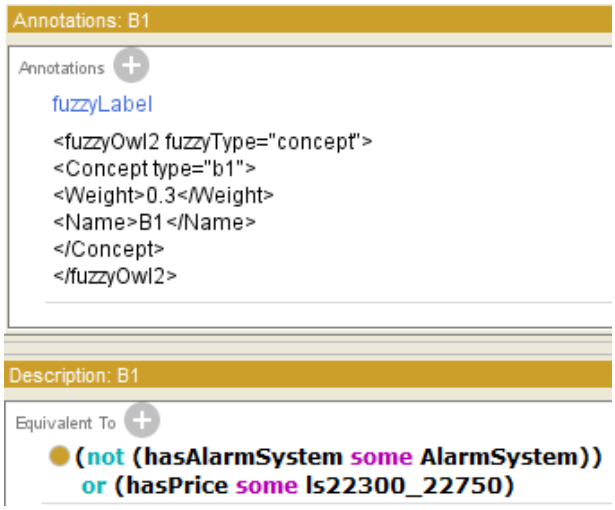


Figura 5.4: Regla difusa b1 del modelo Concesionario

Se resalta en negrita el fragmento donde se encuentra la información difusa, y después de realizar con el analizador las transformaciones necesarias que a continuación se resumen se obtiene la regla *b1* con una sintaxis MALP mucho más comprensible:

```
b1(A) <- | (@not (hasalarmsystem (A, alarmsystem)),
             hasprice (A, ls22300_22750)) with 0.3.
```

- ✓ Las librerías Thea sólo inserta hechos Prolog, pero determinados hechos pueden generar **reglas** MALP con una variable.

  equivalentClasses(B1,… Se transforma en: *b1(A)* <-

- ✓ Para indicar el **grado de verdad** Straccia emplea la etiqueta *fuzzylabel* con una anotación y la palabra *Weight* (peso):

  <fuzzyLabel fuzzyType="concept">…<Weight>0.3</Weight>…

  Por esta razón se inserta un nuevo hecho *fuzzylabel (b1, 0.3).*

- ✓ Si se encuentra la **etiqueta** *someValuesFrom* (algunos valores de) se transforma con la variable de la regla del siguiente modo:

  someValuesFrom(hasAlarmSystem,AlarmSystem)

  Se transforma en: *hasalarmsystem (A, alarmsystem)*

b) **Hechos**: En este caso se han modelado dos individuos que corresponden a dos coches "*Seat Toledo*" y "*Audi A3*" con sus propiedades particulares. A continuación se explica el hecho "*Seat Toledo*" como se puede ver en la Figura 5.5.

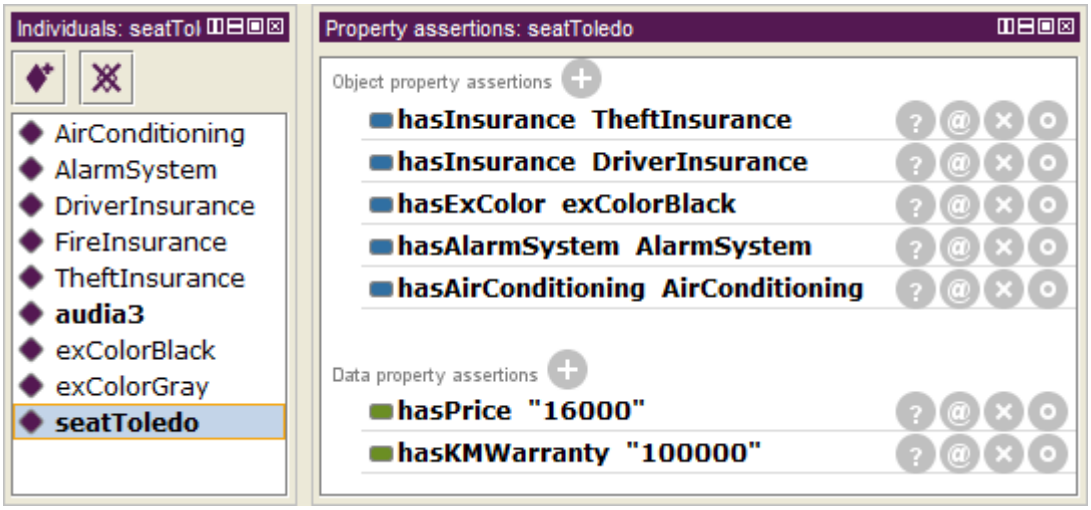


Figura 5.5: Hechos del modelo Concesionario

A partir del código OWL y utilizando la librería Thea se obtienen los hechos donde las transformaciones necesarias en este caso son directas, ya que cada etiqueta *propertyAssertion* en OWL genera un hecho Prolog. Así obtenemos los siguientes hechos A-Box con sintaxis MALP:

```
haskmwarranty (seattoledo, 100000).
hasprice (seattoledo, 16000).
hasairconditioning (seattoledo, airconditioning).
hasalarmsystem (seattoledo, alarmsystem).
hasinsurance (seattoledo, driverinsurance).
hasinsurance (seattoledo, theftinsurance).
hasexcolor (seattoledo, excolorblack).
```

c) **Lógica difusa**: *Umberto Straccia* también contempla la posibilidad de especificar la lógica difusa utilizada. Se muestra en la Figura 5.6 un ejemplo donde se especifica que la lógica difusa es la definida por *Zadeh* (repasar de nuevo la primera fila de la Figura 4.2):

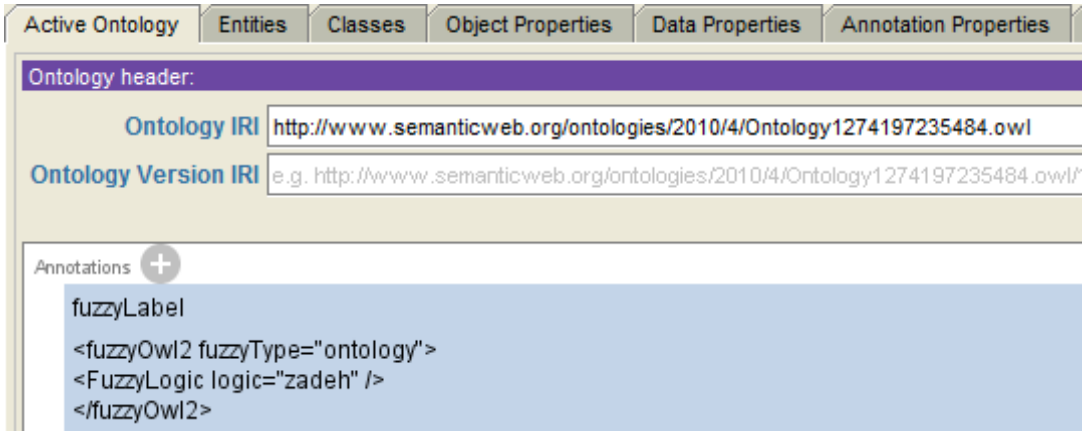


Figura 5.6: Lógica difusa del modelo Concesionario

Concretando el ejemplo para la regla *b1* anteriormente expuesta, las transformaciones que utilizan la librería Thea generan los siguientes hechos:

```
annotationAssertion(fuzzyLabel,http://www.semanticweb.org/ontologies/2010/4/
   Ontology1274197235484.owl,literal(<fuzzyOwl2 fuzzyType="ontology">\n
   <FuzzyLogic logic="zadeh" />\n</fuzzyOwl2>)).

annotationAssertion(fuzzyLabel,B1,literal(<fuzzyOwl2
   fuzzyType="concept">\n<Concept type="b1">\n<Weight>0.3</Weight>\n
   <Name>B1</Name>\n</Concept>\n</fuzzyOwl2>)).
```

Se localiza la ontología empleada, y se extrae la información de la lógica difusa dentro de la etiqueta fuzzyOwl2:

...fuzzyType="ontology">\n<FuzzyLogic logic="zadeh...

Para finalmente insertar un hecho *fuzzylabel (owl, zadeh).*

Además, el cuerpo de las reglas MALP afectadas con una conjunción o una disyunción se modifican para la lógica difusa localizada, y en caso de no encontrarse se dejan los operadores por defecto.

```
b1(A) <- |zadeh (@not (hasalarmsystem (A, alarmsystem)),
         hasprice (A, ls22300_22750)) with 0.3.
```

d) Modificadores lingüísticos y variables lingüísticas. Estas dos características del lenguaje MALP, se refieren a potentes recursos expresivos con fuerte sabor "*fuzzy*" donde, en esencia, las variables lingüísticas (asociadas a adjetivos como "joven" o "cercano") se refieren a predicados difusos modelados en FLOPER como conjuntos

difusos, mientras que los modificadores difusos (como los adverbios “raramente”, “muy” o “extremadamente”) se modelan como conectivos de un retículo multi-adjunto cuya aplicación altera el grado de pertenencia de un determinado valor a un conjunto difuso [BS11a, VM].

En este caso se explica detalladamente la variable lingüística de hombro izquierdo (*leftshoulder*), en concreto *leq26000* como se puede verse en la Figura 5.7.

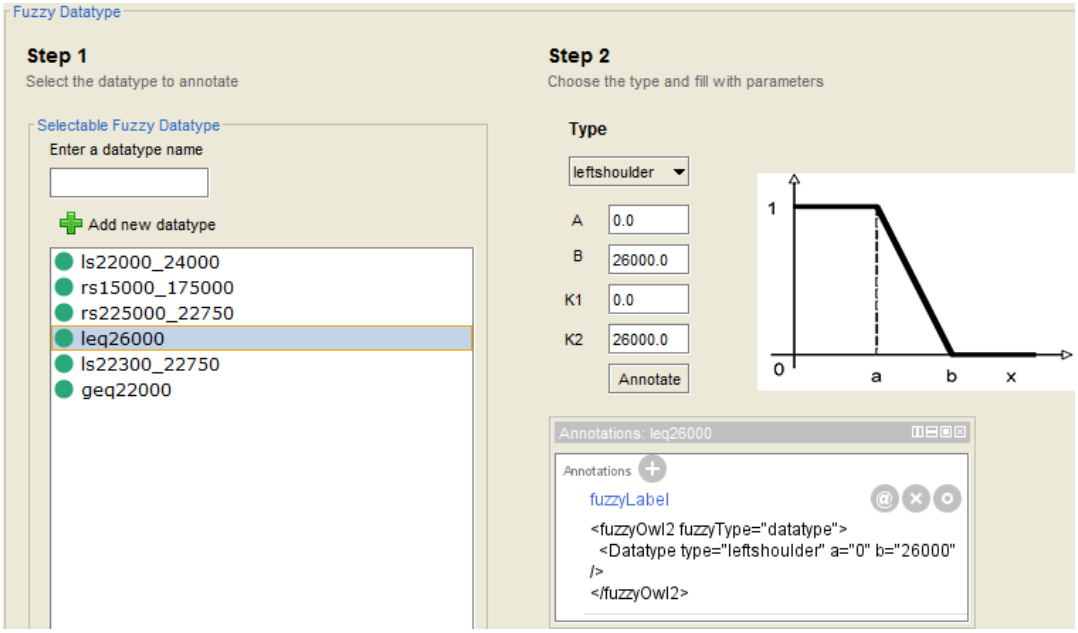


Figura 5.7: Modificadores difusos del modelo Concesionario

A continuación se muestra el fragmento de código resultante tras realizar la transformación:

```
equivalentClasses(Beta,intersectionOf(PassengerCar,
   someValuesFrom(hasPrice,leq26000))).
annotationAssertion(fuzzyLabel,leq26000,literal(<fuzzyOwl2
   fuzzyType="datatype">\n  <Datatype  type="leftshoulder"  a="0"  b="26000"
   />\n</fuzzyOwl2>)).
```

Finalmente las reglas MALP resultantes se transforman del siguiente modo:

```
beta(A) <- hasprice (A, Vleq26000) & leftshoulder (0 , 26000, Vleq26000).
```

En este caso, también es necesario incluir en un fichero Prolog el código que define la regla *leftshoulder* para este caso.

```
leftshoulder(A,B,In,1):- In =< A.
leftshoulder(A,B,In,0):- In >= B.
leftshoulder(A,B,In,Out):- In > A, In < B, Out is (B-In)/(B-A).
```

5.3 CONCLUSIONES

Tras realizar todas las transformaciones anteriores, puede observarse en la Figura 5.8 y Figura 5.9 el código MALP generado donde destaca la sencillez de su sintaxis para incluir y trabajar con información difusa:

```
% Reglas difusas
b1(A) <- |prod (@not (hasalarmsystem (A, alarmsystem)),
     hasprice (A, Vls22300_22750) &
     leftshoulder (22300 , 22750, Vls22300_22750)) with 0.3.
b2(A) <- &prod (|prod (hasinsurance (A, fireinsurance),
     hasinsurance (A, theftinsurance)),
     hasinsurance (A, driverinsurance)) with 0.4.
b3(A) <- &prod (hasairconditioning (A, airconditioning),
     hasexcolor (A, excolorblack)) with 0.2.
b4(A) <- hasprice (A, Vls22000_24000) &
     leftshoulder (22000 , 24000, Vls22000_24000) with 0.8.
b5(A) <- haskmwarranty (A, Vrs15000_175000) &
     rightshoulder (15000 , 175000, Vrs15000_175000) with 0.6.
beta(A) <- hasprice (A, Vleq26000) &
     leftshoulder (0 , 26000, Vleq26000).
buy(A) <- &prod (beta(A), buyerpreferences(A)).
buyerpreferences(A) <- &prod (b1(A), b2(A),
     b3(A), b4(A), b5(A)).
```

Figura 5.8: Código MALP con las reglas difusas obtenidas

```
% Hechos
haskmwarranty (seattoledo, 100000).
hasprice (seattoledo, 16000).
hasairconditioning (seattoledo, airconditioning).
hasalarmsystem (seattoledo, alarmsystem).
hasinsurance (seattoledo, driverinsurance).
hasinsurance (seattoledo, theftinsurance).
hasexcolor (seattoledo, excolorblack).

haskmwarranty (audia3, 150000).
hasprice (audia3, 23000).
hasairconditioning (audia3, airconditioning).
hasalarmsystem (audia3, alarmsystem).
hasinsurance (audia3, driverinsurance).
hasinsurance (audia3, theftinsurance).
hasinsurance (audia3, fireinsurance).
hasexcolor (audia3, excolorgray).
```

Figura 5.9: Código MALP con los hechos obtenidos

Fase 3: Análisis de resultados: El fichero MALP generado en la fase anterior, junto con el retículo, y el fichero Prolog con las variables lingüísticas se cargan en un proyecto FLOPER con la finalidad de analizar los resultados lanzando ejecuciones y generando trazas gráficas. Concretamente, el proyecto se llama "*concesionario.vfp*", y está compuesto por una serie de carpetas y ficheros como se muestra en la Figura 5.10.

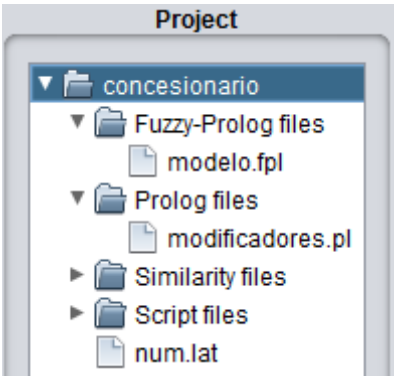


Figura 5.10: Proyecto Concesionario en FLOPER

- ✓ **modelo.fpl** que contiene las reglas y hechos MALP,
- ✓ **modificadores.pl** que contiene las variables lingüísticas utilizadas en el modelo, y
- ✓ **num.lat** que contiene la información del retículo.

Por tanto, para mostrar la funcionalidad del entorno FLOPER, se lanzan dos ejecuciones con distintos objetivos (*b3* y *beta*) y así analizar los resultados que se obtienen. Las reglas y hechos afectados en este caso pueden observarse en Figura 5.11 y corresponden a una versión reducida del modelo del concesionario:

```
%Reglas difusas
b3(A) <- &luka (hasairconditioning (A, airconditioning),
     hasexcolor (A, excolorblack)) with 0.2.
beta(A) <- hasprice (A, Vleq26000) &
      leftshoulder (0 , 26000, Vleq26000).

%Hechos
hasprice (seattoledo, 16000).
hasairconditioning (seattoledo, airconditioning).
hasexcolor (seattoledo, excolorblack).
hasprice (audia3, 23000).
hasairconditioning (audia3, airconditioning).
hasexcolor (audia3, excolorgray).
```

Figura 5.11: Código MALP con las reglas b3 y beta

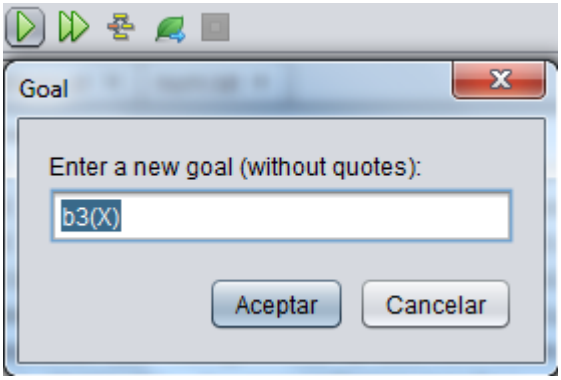


Figura 5.12: Ejecución del objetivo b3(X) en FLOPER

Primero se lanza la ejecución del objetivo b3(X) como se muestra en la Figura 5.12, obteniendo la siguiente salida:

```
'b3(X)'.
>> run.
[Truth_degree=0.19999999999999996,X=seattoledo]
```

Por tanto, se obtiene un resultado aproximado a 0.2 para el coche "*Seat Toledo*", y no se obtiene ningún resultado para el coche "*Audi A3*". Esto es así ya que aunque ambos coches disponen de aire acondicionado, sólo el coche "*Seat Toledo*" es de color negro.

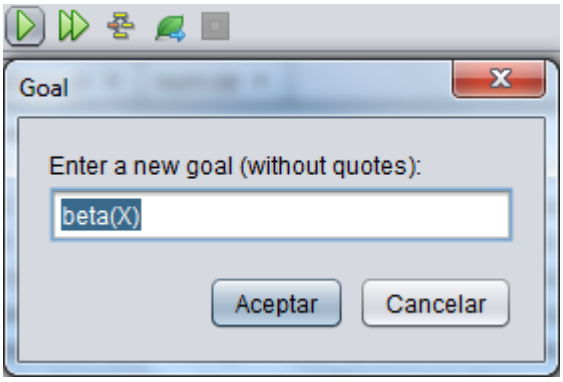


Figura 5.13: Ejecución del objetivo beta(X) en FLOPER

En la siguiente ejecución se relaciona la influencia de las variables lingüísticas con grado de verdad resultante. Como se muestra en la Figura 5.13, se lanza la ejecución del objetivo *beta(X)*, obteniendo en siguiente resultado:

```
'beta(X)'.
>> run.
[Truth_degree=0.11538461538461542,X=audia3]
[Truth_degree=0.3846153846153846,X=seattoledo]
```

En este caso se obtienen grados de verdad positivos para ambos individuos, pero es mayor el grado de verdad para el caso del coche "*Seat Toledo*" debido al funcionamiento y utilización del modificador difuso de hombro izquierdo como se aprecia en la Figura 5.14.

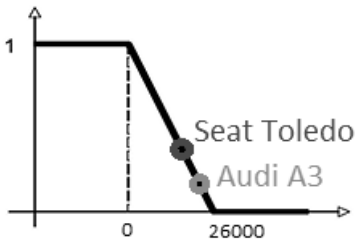


Figura 5.14: Grados de verdad obtenidos para los individuos

Posteriormente, se genera un árbol de desplegado para apreciar de manera gráfica cómo se comporta el modelo propuesto, y también explicar de manera separada la fase operacional y la interpretativa descrita en el capítulo anterior. El caso analizado es consecuencia de lanzar el objetivo b3(X) como se muestra en la Figura 5.15.

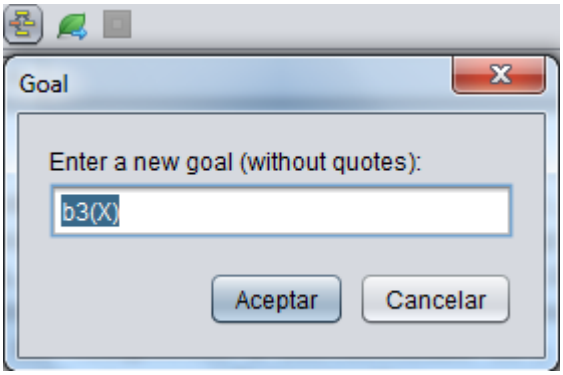


Figura 5.15: Lanzar el árbol de desplegado con objetivo b3(X) en FLOPER

```
>> tree.
R0 < b3(X), {} >
 R1 < &luka(0.2,&luka(hasairconditioning(X,airconditioning),
      hasexcolor(X,excolorblack))), {A1/X} >
 R4 < &luka(0.2,&luka(1,hasexcolor(seattoledo,excolorblack))),
      {X/seattoledo,A1/seattoledo} >
 R5 < &luka(0.2,&luka(1,1)), {X/seattoledo,A1/seattoledo} >
 is < &luka(0.2,1), {X/seattoledo,A1/seattoledo} >
 is < 0.19999999999999996, {X/seattoledo,A1/seattoledo} >
 R7 < &luka(0.2,&luka(1,hasexcolor(audia3,excolorblack))),{X/audia3,A1/audia3}>
 R0 < &luka(0.2,&luka(1,0)), {X/audia3,A1/audia3} >
 is < &luka(0.2,0), {X/audia3,A1/audia3} >
 is < 0, {X/audia3,A1/audia3} >
```

El árbol de desplegado de profundidad 5 para el programa anterior y el átomo *b3(X)* se puede apreciar de forma gráfica (tal y como lo muestra FLOPER) en la Figura 5.16.

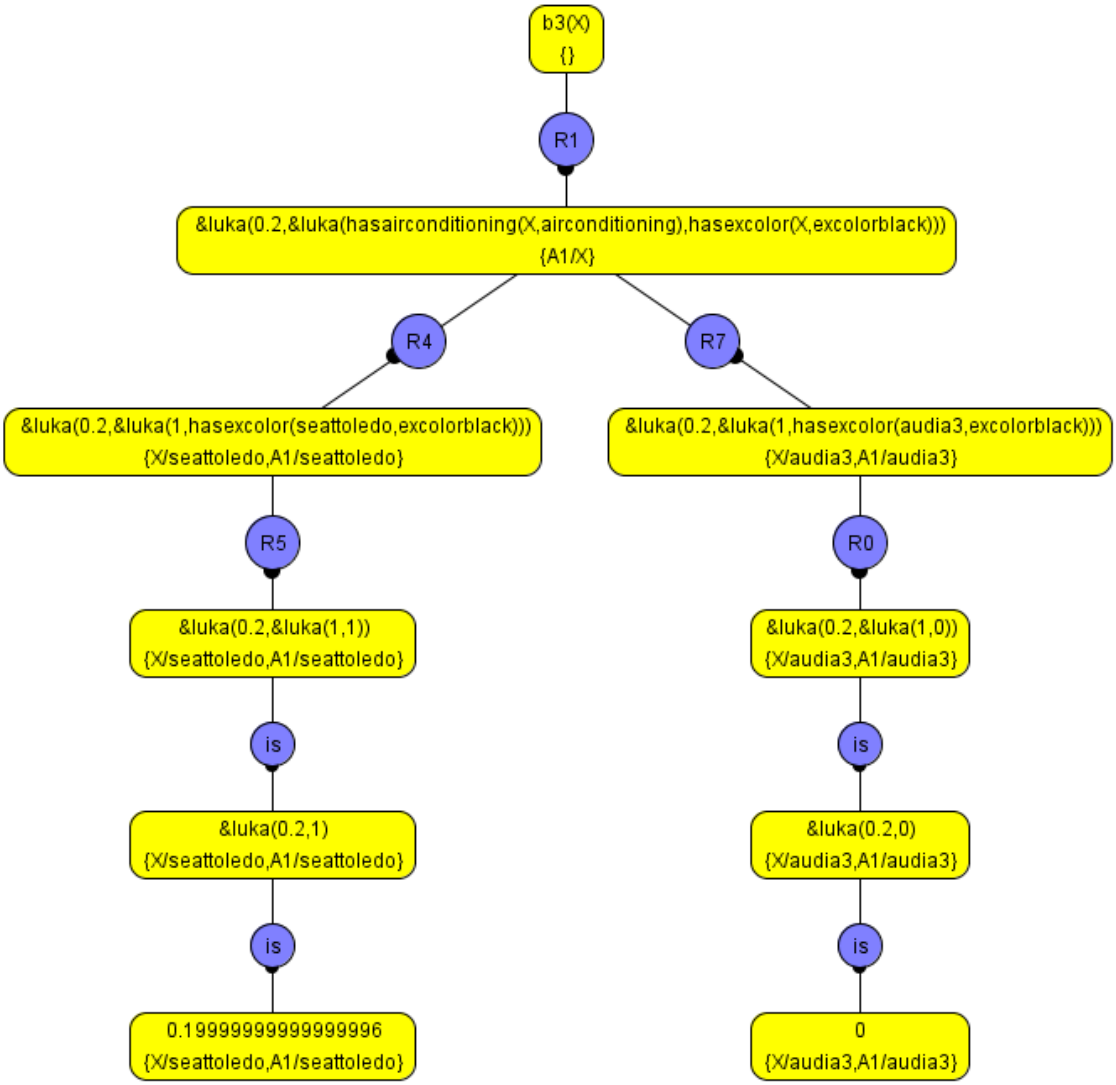


Figura 5.16: Árbol de desplegado de profundidad 5

Se comienza con un paso admisible partiendo del objetivo *b3(X)* y unificando con la regla *R1*, para generar el siguiente estado como se observa en la Figura 5.17.

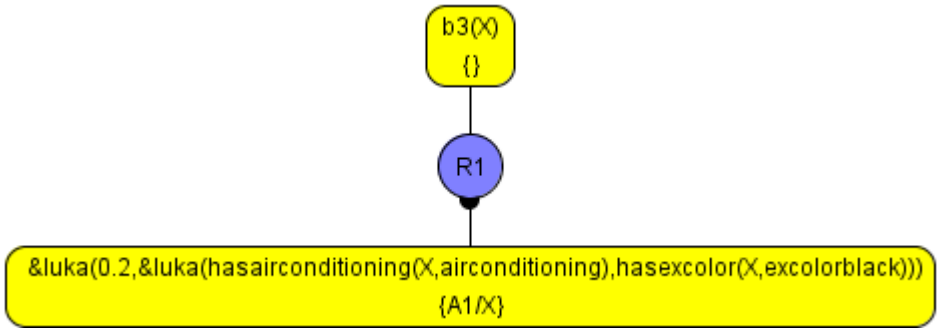


Figura 5.17: Primer paso admisible a partir del objetivo b3(X)

A continuación el árbol se divide en dos ramas mostradas en la Figura 5.18, esto es debido a que en el programa aparecen dos hechos *hasairconditioning* (*seattoledo* y *audia3*) que unifican con el sub-objetivo o átomo seleccionado.

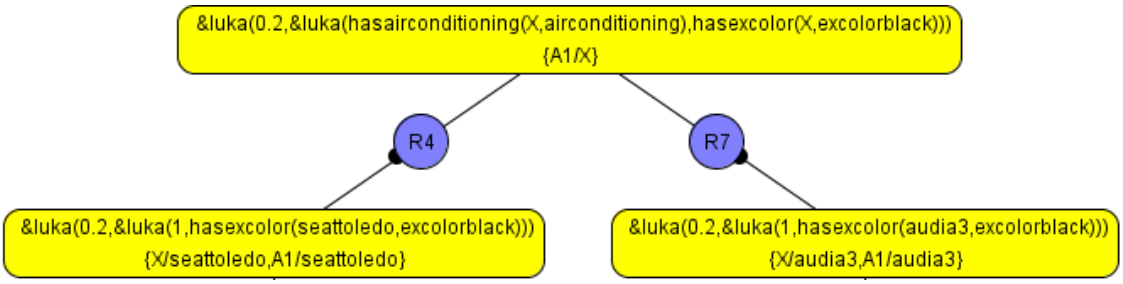
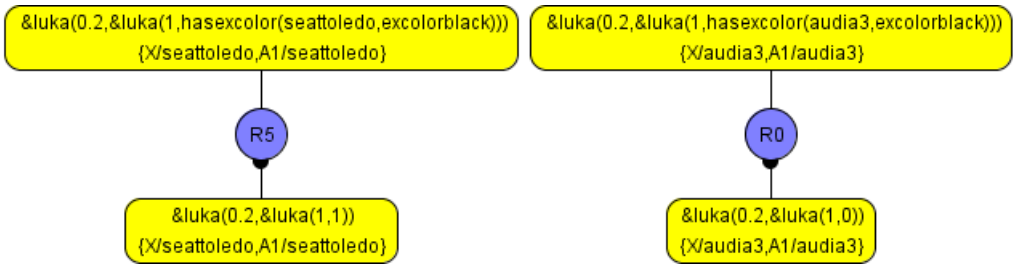
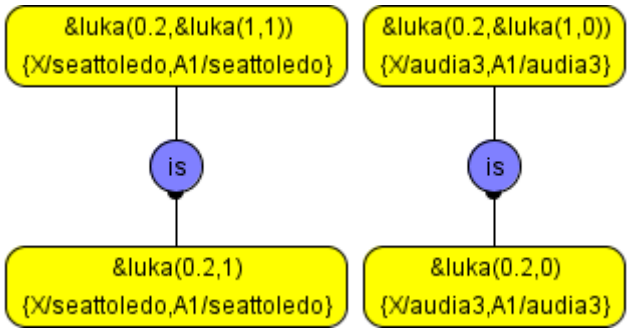


Figura 5.18: División del árbol de desplegado en dos ramas

En este punto, dependiendo del camino seguido, se obtienen dos resultados, uno para cada rama del árbol, ya que en este ejemplo existen dos hechos. Concretamente, para la rama izquierda del árbol se encuentra el hecho en la base de datos *hasexcolor(seattoledo,excolorblack)*, pero para la rama derecha del árbol este hecho no existe como puede verse a continuación.



Aquí acaba la fase operacional y comienza la fase interpretativa donde la expresión resultante en la fase anterior es interpretada en un retículo concreto para obtener un par compuesto por el grado de verdad final junto con la substitución que asocia valores concretos a las variables del objetivo inicial. Enlazando con la fase anterior, se opera con cada rama de manera separada para obtener cada solución. En la rama izquierda tras realizar la operación en el retículo *&luka(1,1)* se obtiene un 1. Por otro lado, en la rama derecha se resuelve *&luka(1,0)* dando un resultado de 0.



Sólo resta aplicar un paso para cada rama del árbol de desplegado. Operando en el retículo con la rama izquierda se resuelve la operación *&luka(0.2,1)* obteniendo un valor aproximado de 0,2. Por el contrario, al resolver en la rama derecha la operación *&luka(0.2,0)* se obtiene un valor de 0 indicando que no hay solución en este caso como finalmente puede verse en la Figura 5.19.

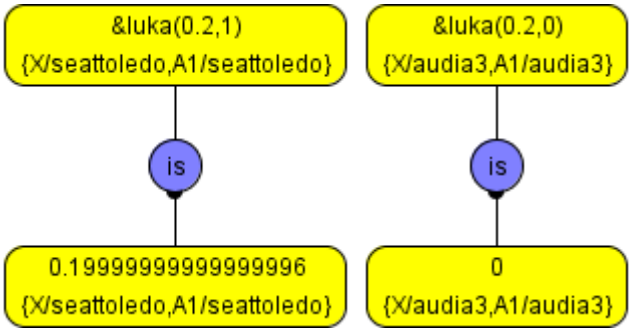


Figura 5.19: Fase interpretativa del árbol de desplegado

Por tanto, se concluye que para el objetivo lanzado *b3(X)* se obtiene un resultado aproximado de 0.2 cuando el valor de la variable *X* es "*seattoledo*" que corresponde al coche "*Seat Toledo*".

CAPÍTULO 6. ANTEPROYECTO DE TESIS DOCTORAL

6.1 CONCLUSIONES

La Web Semántica es una visión de la Web en el que se proporciona información con significado explícito, haciendo más fácil para las máquinas procesar automáticamente e integrar la información disponible en la Web. El éxito y la proliferación de la Web Semántica dependen en gran medida de la construcción de las ontologías Web. Sin embargo, los enfoques clásicos en la construcción de ontologías no son suficientes para el manejo y manipulación de información imprecisa e incierta que se encuentra comúnmente en muchos dominios de aplicación [MYZ13]. Por esta razón, en los últimos años se han hecho grandes esfuerzos en la construcción de ontologías difusas para hacer frente al conocimiento incierto y dinámico en la Web Semántica.

El grupo de investigación DEC-TAU de la UCLM presenta tres líneas de trabajo futuro que se explican en la tesis de Pedro Morcillo [Mor13]: Integración de los lenguajes lógicos difusos basados en pesos y relaciones de similaridad, transformaciones umbralizadas para programas integrados difusos, y reforzamientos de FLOPER y aplicaciones sobre la web semántica. Este último punto tiene gran relación con este TFM, ya que también se debe destacar que actualmente se está desarrollando FuzzyXPath que implementa una función difusa de XPath / XQuery [LMA11, LMA12b, CGSA12] para la consulta flexible de documentos XML. Además de lo anterior, sería interesante interactuar con sistemas como "*WordNet*" que trata con relaciones semánticas entre los conjuntos de sinónimos para incorporar en FLOPER la posibilidad del tratamiento de ontologías de este tipo.

Para el desarrollo del analizador difuso se ha utilizando la librería Thea de SWI-Prolog, pero podría adaptarse del prototipo para trabajar con otra librería que es capaz de trabajar con ficheros OWL RL y que actualmente está desarrollando Jesús Almendros en la Universidad de Almería [Alm11a, Alm11b, SB13, Alm12]. Sería muy interesante poder validar y comparar los resultados obtenidos utilizando ambas librerías o incluso realizar una fusión de ambas librerías.

6.2 ANTEPROYECTO DE TESIS DOCTORAL

La comunidad *World Wide Web* vislumbra una interacción sin problemas entre personas y ordenadores, interoperabilidad transparente en el intercambio de información entre las distintas aplicaciones web, y una rápida y precisa identificación e invocación a los servicios Web correspondientes [LMA12a]. A medida que el trabajo con la web semántica y los servicios web son más ambiciosos, aumenta la necesidad de enfoques de representación más formales y de razonamientos con incertidumbre. El término incertidumbre abarca una gran variedad de formas de conocimiento incompleto, incluyendo incompletitud, inconcluso, vaguedad, ambigüedad, y otros. En este escenario, los retos actuales más relevantes mediante razonamiento con incertidumbre incluyen:

- ✓ **Incertidumbre en la información disponible**: Mucha de la información de la Web tiene incertidumbre, por ejemplo, previsiones meteorológicas o las probabilidades en las apuestas.
- ✓ **Información incompleta** extraída de las grandes redes de información como Internet. La capacidad para explotar información parcial puede ser muy útil para la identificación de fuentes de servicio o información, por ejemplo, ofertas con tarjetas de felicitación en servicios online pueden ser evidencia de que también se comercializan productos de papelería.
- ✓ **Información inexacta**: La información web también es a menudo incorrecta o sólo parcialmente correcta, lo que plantea cuestiones relacionadas con la confianza y la credibilidad. Una representación y razonamiento adecuado de la incertidumbre puede ayudar a resolver el conflicto entre diversas fuentes de información que a menudo cuentan con diferentes niveles de confianza, y también puede facilitar la fusión de información contradictoria obtenida de diversas fuentes.
- ✓ **Mapeados de ontologías inciertas**: La visión de la Web Semántica implica que coexistirán múltiples ontologías diferentes, pero conceptualmente superpuestas y que interactúan. En este tipo de escenarios, se puede asignar un peso o grado de verdad por categorías a cada ontología acerca de la pertenencia a una clase.
- ✓ **Información indefinida en los servicios Web**: La dinamicidad de los servicios Web requieren la identificación de los recursos en tiempo de ejecución y la resolución de los objetivos establecidos. Técnicas de razonamiento con

incertidumbre pueden ser útiles para resolver situaciones en las que la información existente no es definitiva.

Por tanto, la incertidumbre es una característica intrínseca en muchas de las tareas cotidianas de la Web y de la Web Semántica, y una completa comprensión de la *World Wide Web* como fuente de datos y servicios procesables exige formalismos capaces de representar y razonar en situaciones de incertidumbre. Desafortunadamente, ningunas de estas necesidades en un principio pueden ser abordadas con los estándares web actuales. Y aunque en cierta medida es posible utilizar semántica en los lenguajes de marcado como OWL y RDF para representar información difusa cualitativa y cuantitativa, todavía no hay fundamentos suficientemente probados, y los enfoques existentes son muy limitados.

Por otro lado, los tradicionales lenguajes de programación orientados a objetos pueden ser difíciles de usar cuando se trabaja con ontologías, lo que lleva a la creación de lenguajes específicos diseñados concretamente para el procesamiento de ontologías [WMV09]. En este escenario, Prolog, basado en el paradigma de la programación lógica, ofrece muchas ventajas en la web semántica como lenguaje de programación anfitrión para consultar y procesar ontologías, e incluso para la creación de aplicaciones. En particular, el entorno SWI-Prolog [Prolog] incluye librerías RDF que han sido utilizadas con éxito para desarrollar complejas aplicaciones AJAX en la web semántica. En esta línea actualmente también se están desarrollando implementaciones como FuzzyXPath [LMA11, LMA12b, CGSA12, FXZ+09] utilizando el lenguaje lógico-difuso MALP y la herramienta FLOPER.

Como consecuencia, la necesidad de manejar información difusa o imprecisa en lenguajes de la Web Semántica está aumentando en importancia en los últimos años y, exige una forma estándar de representar dicha información [BS11b]. Podemos solucionar este problema extendiendo los actuales lenguajes de la Web Semántica para trabajar con incertidumbre como es el caso de FuzzyOWL2, y proporcionando un procedimiento basado en transformaciones para representar dicha información en lenguajes y herramientas actuales basados en reglas capaces de trabajar y razonar con información difusa de manera natural como es el caso del lenguaje MALP. En el resto del capítulo de expone la secuencia de actividades profundizando en cada una de las tareas.

6.3 CRONOGRAMA PARA LA TESIS DOCTORAL

Como se acaba de mencionar, en este punto se enumeran y desarrollan la propuesta con las posibles líneas futuras de investigación o anteproyecto de posible tesis doctoral, cuya distribución temporal puede verse en la Tabla 6.1, y que seguidamente se explican en detalle.

	Primer año	Segundo año	Tercer año	Cuarto año
Estado del arte				
Creación prototipo analizador				
Evaluación y mejora prototipo analizador				
Automatización tareas e integración				
Validación y difusión				

Tabla 6.1: Planificación de actividades

- ✓ **Tarea 1. Estado del arte**: Los contenidos más relevantes investigados están relacionados con la web semántica, MALP, fuzzyOWL2, y los lenguajes lógicos de transformaciones sobre OWL existentes en la actualidad. Esta tarea se ha empezado a realizar en este TFM y se seguirá realizando durante toda la tesis doctoral de manera periódica, introduciendo nuevos conceptos que vayan relacionándose y interés para la investigación.

- ✓ **Tarea 2. Creación de un prototipo de analizador difuso**: Esta tarea se ha realizado durante todo un año pero con dedicación parcial y pone de manifiesto que en este TFM se ha llevado a cabo un importante trabajo práctico, ya que el prototipo desarrollado es totalmente funcional para ficheros OWL2 difusos generados mediante la herramienta de modelado Protégé, y permite transformar un fichero FuzzyOWL2 (con reglas difusas, hechos, modificadores difusos, e incluso especificando la lógica difusa utilizada) a un programa con sintaxis MALP listo para ser cargado, ejecutado y depurado por FLOPER.

- ✓ **Tarea 3. Evaluación y mejora del prototipo de analizador difuso creado**: Esta tarea se empezará a realizar en el siguiente año, el objetivo es poner a prueba el prototipo para detectar y depurar deficiencias, añadir funcionalidades, evaluar su comportamiento y rendimiento ante determinadas circunstancias límite, y también adaptarlo a las nuevas funcionalidades que el grupo de investigación DEC-TAU de la UCLM está actualmente desarrollando. UCLM está actualmente desarrollando. En este último sentido, resultará interesante investigar el uso potencial de "relaciones de similaridad" en nuestra aplicación.

- ✓ **Tarea 4. Automatización de tareas e integración**: Esta tarea se llevará a cabo en dos etapas. La primera consistirá en integrar dentro de FLOPER la funcionalidad para cargar un fichero FuzzyOWL2, y la segunda destinada a crear una aplicación Web capaz de tratar directamente con ficheros difusos y realizar razonamientos mediante consultas.

- ✓ **Tarea 5. Validación y difusión**: Finalmente esta tarea se ha empezado a realizar al crear el prototipo, pero considero que debe seguir validándose y difundiendo los resultados y funcionalidades añadidas hasta la finalización de la tesis doctoral mediante la publicación de diversos artículos científicos relacionados con el tema de la investigación. Añadir que debido a que en la actualidad existen varias librerías capaces de trabajar con OWL RL [Alm11a], sería interesante adaptar o fusionar el prototipo para que fuera capaz de trabajar con esas librerías y así poder analizar y comparar los resultados obtenidos.

Finalmente puntualizar que estas tareas se realizarán y refinarán periódicamente con la finalidad de introducir mejoras y evaluar el trabajo realizado.

BIBLIOGRAFÍA

LIBROS Y ARTICULOS

[Agu11] Norka Natalia Aguirre Helguero. Un agente basado en un razonador de ontologías, 2011, Tesis. Universidad de Buenos Aires.

[Alm11a] Jesús M. Almendros-Jiménez. "A Prolog library for OWL RL", *LID '11 Proceedings of the 4th International Workshop on Logic in Databases*, pp. 49-56, Marzo 2011.

[Alm11b] Jesús M. Almendros-Jiménez. "A Prolog-based Query Language for OWL", *Electronic Notes in Theoretical Computer Science (ENTCS)*, vol. 271, pp. 3-22, Marzo 2011.

[Alm12] Jesús M. Almendros-Jiménez. "OWL RL in logic programming: querying, reasoning and inconsistency explanations", RuleML'12 Proceedings of the 6th international conference on Rules on the Web: research and applications, vol. 7438, pp. 248-255, 2012.

[Ber99] Tim Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web*, Primera ed., HarperBusiness, Ed. New York, USA: Harper Collins, 1999.

[BS11a] Fernando Bobillo, Umberto Straccia. "Aggregation Operators and Fuzzy OWL 2", FUZZ-IEEE 2011 Proceedings of the 20th IEEE International Conference on Fuzzy Systems, pp. 1727-1734, Junio 2011.

[BS11b] Fernando Bobillo, Umberto Straccia. "Fuzzy ontology representation using OWL 2", *International Journal of Approximate Reasoning*, vol. 52, no. 7, pp. 1073-1094, Octubre 2011.

[CAF+13] Paulo C.G. Costa, Claudia d'Amato, Nicola Fanizzi, Kathryn B. Laskey, Kenneth J. Laskey, Thomas Lukasiewicz, Matthias Nickles, Michael Pool (Eds.), Fernando Bobillo. *Uncertainty Reasoning for the SemanticWeb II*, International Workshops URSW 2008-2010 Held at ISWC and UniDL 2010 Held at Floc, Ed.: Springer, 2013.

[CGSA12] Rafael Caballero, Yolanda García-Ruiz, Fernando Sáenz-Pérez, Jesús M. Almendros-Jiménez. "XPath Query Processing in a Functional-Logic Language", Electronic Notes in Theoretical Computer Science (ENTCS), vol. 282, pp. 19-34, Mayo 2012.

[CHW+09] Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, Carsten Lutz, Boris Motik. "OWL 2 Web Ontology Language Profiles", W3C Recommendation (MIT, ERCIM, Keio), Abril 2009.

[CMN+10] D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider, F. Baader. *The Description Logic Handbook: Theory, Implementation, and Applications*, Segunda ed.: Cambridge University Press, 2010.

[Cri09] Luis Criado Fernández. Procedimiento semi-automático para transformar la web en Web Semántica, 2009, Tesis Doctoral. UNED.

[Daw11] Zaid Dawood Issa. Identificación y explotación de expresiones temporales para entornos SIG, 2011, Proyecto Fin de Carrera. Universidad Politécnica de Madrid.

[FXZ+09] Wei Fang, Xuefeng Xian, Shukui Zhang, Pengpeng Zhao, Zhiming Cui. "Extension of OWL with Dynamic Fuzzy Logic", Advances in Web and Network Technologies, and Information Management, vol. 5731, pp. 67-76, 2009.

[GHHA12] Paul Groth, Frank van Harmelen, Rinke Hoekstra, Grigoris Antoniou. *A Semantic Web Primer*, Tercera ed., MIT Press, Ed.: Cambridge, MA, 2012.

[Gru93] Thomas R. Gruber. "A translation approach to portable ontology", Knowledge Acquisition, vol. 5, no. 2, pp. 199-220, Junio 1993.

[Gru95] Thomas R. Gruber. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", International Journal of Human-Computer Studies, vol. 43, no. 5-6, pp. 907-928, Noviembre/Diciembre 1995.

[HD04] Frank van Harmelen, Deborah L. McGuinness. "OWL Web Ontology Language Overview", W3C Recommendation (MIT, ERCIM, Keio), Febrero 2004.

[HMP+08] Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, Ulrike Sattler, Bernardo Cuenca Grau. "OWL 2: The next step for OWL", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 6, no. 4, pp. 309-322, Noviembre 2008.

[KPP+09] Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, Sebastian Rudolph, Pascal Hitzler. "OWL 2 Web Ontology Language Primer", W3C Recommendation (MIT, ERCIM, Keio), 2009.

[KRH09] Markus Krötzsch, Sebastian Rudolph, Pascal Hitzler. *Foundations of Semantic Web Technologies*, KI 2009 Knowledge Representation for the Semantic Web, Ed. Paderborn, Alemania: Chapman & Hall/CRC, 2009.

[LMA11] Alejandro Luna, Ginés Moreno, Jesús M. Almendros-Jiménez. "A Flexible XPath-Based Query Language Implemented with Fuzzy Logic Programming", RuleML'2011 Proceedings of the 5th international conference on Rule-based reasoning, programming, and applications, Springer LNCS vol. 6826, pp. 186-193, Julio 2011.

[LMA12a] Alejandro Luna, Ginés Moreno, Jesús M. Almendros-Jiménez. "A XPath Debugger Based on Fuzzy Chance Degrees", OTM 2012 Workshops Lecture Notes in Computer Science, Springer LNCS vol. 7567, pp. 669–672, 2012.

[LMA12b] Alejandro Luna, Ginés Moreno, Jesús M. Almendros-Jiménez. "Fuzzy Logic Programming for Implementing a Flexible XPath-based Query Language", Electronic Notes in Theoretical Computer Science (ENTCS), vol. 282, pp. 3-18, Mayo 2012.

[LS08] Thomas Lukasiewicz, Umberto Straccia. "Managing uncertainty and vagueness in description logics for the Semantic Web", Web Semantics, vol. 6, no. 4, pp. 291-308, Noviembre 2008.

[Mar07] Santiago Márquez Solís. *La Web Semántica*, Primera ed., Santiago Márquez Solís, Ed.: Lulu.com, 2007.

[MM08a] Gines Moreno, Pedro J. Morcillo. "Programming with Fuzzy Logic Rules by Using the FLOPER Tool", RuleML '08 Proceedings of the International Symposium on Rule Representation, Springer LNCS vol. 5321, pp. 119-126, Octubre 2008.

[MM08b] Gines Moreno, Pedro J. Morcillo. "The Fuzzy Logic Programming Environment FLOPER", 15th International Symposium on formal Methods (FM 2008), vol. Turku, Finlandia, pp. 26-30, Mayo 2008.

[MN00] Jan Maluszynski, Ulf Nilsson, *Logic. Programming and Prolog*, Segunda ed.: John Wiley & Sons Ltd, 2000.

[Mor08] Pedro J. Morcillo. Diseño de un entorno de programación lógica difusa, 2008, Trabajo Fin de Máster. UCLM.

[Mor13] Pedro J. Morcillo. Semánticas Operacionales Avanzadas para Programas Lógicos Difusos, 2013, Tesis Doctoral. UCLM.

[MPI09] Ginés Moreno, Jaime Penabad, Pascual J. Iranzo. "On the Declarative Semantics of Multi-Adjoint Logic Programs", IWANN '09 Proceedings of the 10th International Work-Conference on Artificial Neural Networks, Springer LNCS vol. 5517, pp. 253–260, Junio 2009.

[MYZ13] Z. M. Ma, Li Yan, Fu Zhang. "Construction of fuzzy ontologies from fuzzy XML models", Knowledge-Based Systems, vol. 42, pp. 20–39, Abril 2013.

[OVM01a] Manuel Ojeda-Aciego, Peter Vojtas, Jesús Medina. "Multi-Adjoint Logic Programming with Continuous Semantics", LPNMR '01 Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning, vol. 2173, pp. 351–364, Enero 2001.

[OVM01b] Manuel Ojeda-Aciego, Peter Vojtas, Jesús Medina. "A Procedural Semantics for Multi-Adjoint Logic Programming", EPIA '01 Proceedings of the10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, vol. 2258, pp. 290–297, 2001.

[OVM04] Manuel Ojeda-Aciego, Peter Vojtas, Jesús Medina. "Similarity-based unification: a multi-adjoint approach", Fuzzy Sets and Systems, vol. 146, no. 1, pp. 43–62, Agosto 2004.

[Pen10] Jaime Penabad. Desplegado de Programas Lógicos difusos, 2010, Tesis doctoral. UCLM.

[PGMG11] Miguel A. Patricio, Jesús García, José M. Molina, Juan Gómez-Romero. "Ontology-based context representation and reasoning for object tracking and scene interpretation in video", Expert Systems with Applications, vol. 38, no. 6, pp. 7494-7510, Junio 2011.

[San06] Antonia M. Sánchez Huertas. "Lógicas para la red", Colecciones Azafea, vol. 8, pp. 85-102, Marzo 2006.

[SB13] Umberto Straccia, Fernando Bobillo. "Finite Fuzzy Description Logics and Crisp Representations", Uncertainty Reasoning for the Semantic Web (URSW), vol. 7123, pp. 99-118, 2013.

[SPG+12] Miguel A. Serrano, Miguel A. Patricio, Jesús García, José M. Molina, Juan Gómez-Romero. "Context-based scene recognition from visual data in smart homes: an Information Fusion approach", Personal and Ubiquitous Computing, vol. 16, no. 7, pp. 835-857, Octubre 2012.

[SS94] Ehud Y. Shapiro, Leon S. Sterling. *The Art Of Prolog*, Segunda ed., MA Cambridge, Ed. Londres, Inglaterra: MIT Press, 1994.

[Stra01] Umberto Straccia. "Reasoning within Fuzzy Description Logics", Journal of Artificial Intelligence Research, vol. 14, no. 1, pp. 137-166, Enero 2001.

[Stra05] Umberto Straccia. "A Fuzzy Description Logic for the Semantic Web", ISTI-CNR, vol. 3532, pp. 167-181, 2005.

[Vaz10] Carlos Vázquez Pérez-Íñigo. Extensiones prácticas sobre el entorno de programación lógica difusa FLOPER, 2010, Trabajo Fin de Grado. UCLM.

[Vaz11] Carlos Vázquez Pérez-Íñigo. Modelado de retículos difusos avanzados sobre el entorno FLOPER, 2011, Trabajo Fin de Máster. UCLM.

[VM] Carlos Vázquez, Ginés Moreno. "Fuzzy Logic Programming in Action with FLOPER", En proceso de revisión en la revista "Information Sciences" de Elsevier.

[Wie09] Jan Wielemaker. *SWI-Prolog 5.8. Reference Manual*. Ámsterdam, Países Bajos: University of Amsterdam, 2009.

[WMV09] Jan Wielemaker, Chris Mungall, Vangelis Vassilades. "Processing OWL2 ontologies using Thea: An application of logic programming", OWLED'09 Proceedings of the 6th International Workshop on OWL: Experiences and Directions, vol. 529, p. 34, Marzo 2009.

[YK95] Bo Yuan, George J. Klir. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, US ed ed., Prentice Hall PTR, Ed. USA: Patti Guerrieri, 1995.

[Zad65] Lotfi Asker Zadeh. "Fuzzy Sets. Information and Control", vol. 8, no. 3, pp. 338-353, 1965.

[Zad75] Lotfi A. Zadeh. "The concept of a linguistic variable and its application to approximate reasoning", Information Sciences, vol. 8, no. 3, pp. 199-249, 1975.

ENLACES DE INTERNET

[BS12] Fernando Bobillo, Umberto Straccia. (2012, Junio) Web Fuzzy Ontology Representation using OWL 2. http://gaia.isti.cnr.it/straccia/software/FuzzyOWL/index.html

[Protege] Stanford Center for Biomedical Informatics Research. (2013, Junio) The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/

[Prolog] SWI-Prolog. (2013, Junio) Web SWI-Prolog. http://www.swi-prolog.org/web/index.html

[Radar] Radar Networks & Nova Spivack. (2007, Marzo) Radar Networks. http://www.novaspivack.com/tag/radar-networks

[Thea] Vangelis Vassiliadis. (2006) Thea. A Web Ontology Language - OWL Library for [SWI] Prolog. http://www.semanticweb.gr/TheaOWLLib/Thea%20OWL%20Lib.pdf

[W3C] W3C OWL Working Group. (2012, Diciembre) OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl2-overview/

GLOSARIO DE TÉRMINOS

ABox	Descripción del universo, conjunto de axiomas asertivos.
Axiomas	Restricciones sobre los conceptos, instancias y roles.
Cláusula de Horn	Fórmula lógica en lógica proposicional.
Conceptos	Conjunto de objetos del dominio con características comunes.
DL	*Description Logics.* Familia de lenguajes de representación del conocimiento.
Fase interpretativa	Resultado de interpretar una expresión en un retículo concreto.
Fase operacional	Resultado de aplicar de manera sistemática pasos admisibles.
FLOPER	*Fuzzy LOgic Programming Environment for Research*. Intérprete de programas difusos.
FuzzyOWL2	Plugin para Protégé que permite trabajar con ontologías OWL 2 difusas.
FuzzyXPath	Implementación difusa de XPath / XQuery.
Individuos	Objetos o instancias pertenecientes a una clase.
Lógica difusa	Extensión de la lógica para incluir elementos difusos.
MALP	*Multi-Adjoint Logic Programming*. Lenguaje que combina la programación lógica tradicional con la lógica difusa.
Modificador lingüístico	Conectivo de un retículo multi-adjunto.
OWL	*Web Ontology Language*. Lenguaje de Ontologías para la Web.
OWL 2	Segunda versión de OWL, se añaden elementos expresivos.
Plugin	Aplicación adicional que es ejecutada por la aplicación principal.
Prolog	*PROgrammation en LOGique*. Lenguaje basado en la lógica de predicados de primer orden.
Protégé	Editor libre y también un sistema de adquisición de conocimiento.
RDF	*Resource Description Framework*. Lenguaje para especificar metadatos.
RDFS	*RDF Schema*. Extensión semántica de RDF.
resolución SLD	Mecanismo muy potente de demostración en programación lógica.
Roles	Conexiones binarias entre individuos relacionados.
SPARQL	Lenguaje para la consulta de datos RDF.
SWI-Prolog	Implementación en código abierto del lenguaje de programación Prolog.
SWRL	*Semantic Web Rule Language.*
TBox	Reglas que describen un sistema mediante un vocabulario controlado.
Thea	Librería con un reducido conjunto de predicados y entidades OWL.
Thea 2	Mejora de Thea que soporta toda la funcionalidad de OWL2.
Variable lingüística	Predicado difuso modelado en FLOPER como un conjunto difuso.
W3C	*World Wide Web Consortium*.
Web Semántica	Extensión de la Web para incluir contenido legible por las máquinas.
WWW	*World Wide Web.*
XML	*eXtensible Markup Language*. Lenguaje de Marcas Extensible.
XML Schema	Lenguaje para definir la estructura de los documentos XML.
XPath	*XML Path Language.*
XQuery	Lenguaje de consulta diseñado para colecciones de datos XML.

ANEXO A. **CURRÍCULUM VITAE**

DATOS PERSONALES

- **Apellidos:** Gómez Tornero
- **Nombre:** Luis Miguel
- **Sexo:** Varón

FORMACIÓN ACADÉMICA

- **INGENIERO INFORMÁTICO** por la Escuela Universitaria Superior de Albacete en 2003, obteniendo Matrícula de Honor en el Proyecto Fin de Carrera.

- **INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS** por la Escuela Universitaria Superior de Albacete en 1998.

- **INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN** por la UNED en 2007.

FORMACIÓN COMPLEMENTARIA

- Taller Centro Regional de Formación del Profesorado: "Formación Coordinadores de Formación de Seminarios" en el 2013.
- Taller Centro Regional de Formación del Profesorado: "El currículum de las inteligencias múltiples" en el 2013.
- Taller Centro Regional de Formación del Profesorado: "Aplicaciones didácticas de los formularios de Google" en el 2013.
- Taller Centro Regional de Formación del Profesorado: "Primeros auxilios II" en el 2013.
- Taller Centro Regional de Formación del Profesorado: "Códigos QR en el aula" en el 2013.
- Taller Centro Regional de Formación del Profesorado: "Secuencias digitales en el aula con LAMS" en el 2013.
- Coordinador Seminario de centro: "Puesta en marcha y manejo de pizarras digitales y Netbook" en el 2013.
- Coordinador Evaluación de Diagnóstico: "Responsable de la prueba del IES Virrey Morcillo" en el 2011.
- Coordinador Grupo de Trabajo en el CEP de Villarrobledo: "Moodle: Creación de actividades para moodle", con una duración de 30 horas en el 2011.
- Proyectos de Innovación: "Prácticas de innovación para la evaluación de diagnóstico censal 2010. (IES Virrey Morcillo)", con 2 créditos en el 2010.
- Coordinador Grupo de Trabajo en el CEP de Villarrobledo: "Moodle para docentes de un centro educativo", con una duración de 30 horas en el 2010
- Grupo de Trabajo: "Publicación de contenidos web en Joomla", con una duración de 28 horas en el 2010.
- Tutoría de alumnos en prácticas: "Tutores de profesores funcionarios en prácticas", con 5 créditos en el 2009.
- Grupo de Trabajo: "Plataforma educativa Moodle", con una duración de 20 horas en el 2009.
- Grupo de Trabajo: "Desarrollo curricular del título de formación profesional de técnico en sistemas microinformáticos y redes" con una duración de 70 horas en el 2008.
- Curso CEP-Villarrobledo: "Redes de Area Local - Aplicaciones y Servicios Windows", con una duración de 50 horas en el 2008.

- Curso: "CNICE: Bibliotecas escolares: Uso didáctico", con 20 horas en el 2008.
- Grupo de Trabajo: "Molinux como proveedor de servicios de una Intranet-Internet", con una duración de 20 horas en el 2008.
- Grupo de Trabajo: "Evaluación de la lectoescritura. Mejora mediante el uso de las TIC", con una duración de 20 horas en el 2008.
- Curso: "Redes de área local: Aplicaciones y servicios en Windows", con una duración de 20 horas en el 2008.
- Proyectos de Innovación con Alumnos: "Olimpiada Informática", con un crédito en el 2007.
- Curso: "Formación en el programa Papas. IES Virrey Morcillo", con una duración de 20 horas en el 2007.
- Coordinador Grupo de Trabajo en el CEP de Granada: "Proyecto Curricular Ciclo Administración de Sistemas Informáticos", con 35 horas en el 2007.
- Encuentro de grupos de trabajo "La autoformación del profesorado: Plan de lectura y bibliotecas", con 25 horas en el CEP de Granada en el 2007.
- Jornadas "Conmemoración del 150 aniversario de la ley Moyano", con una duración 12 horas en el CEP de Granada en el 2007.
- Jornadas "II Jornadas de coeducación: Educar en igualdad", con una duración 10 horas en el CEP de Granada en el 2007.
- Jornadas "La formación profesional específica y el mercado de trabajo andaluz:", con una duración 7 horas en el CEP de Granada en el 2007.
- Jornadas "Formación para el Practicum", con una duración de 10 horas en el CEP de Granada en el 2007.
- Jornadas "II Jornadas de la familia profesional de informática", con una duración de 8 horas en el 2006.
- Curso CEP-Jaén "Administrador de servicios de red bajo software libre", con una duración de 30 horas en el 2005.
- Grupo de Trabajo: "Diseño de página Web del IES Virgen del Carmen", con una duración de 45 horas en el 2005.
- Curso CEP-Jaén "Redes inalámbricas. Wireless", con 30 horas en el 2005.
- Curso CEP-Jaén "Software avanzado. Servicios de red bajo Windows 2000 Server", con una duración de 30 horas en el 2004.
- Certificación Cisco CCNA- Organizado por Fundación Iniciativas de Futuro en el 2004.

- Curso CEP: "Introducción a las aplicaciones didácticas en Flash", con una duración de 30 horas desarrollado en Torrijos (Toledo) en el 2003.
- "Certificado de Aptitud Pedagógica" a través del Instituto de Ciencias de la Educación de la Universidad de Murcia en el 2002.
- Escuelas de verano "Sistemas distribuidos: Modelos y Aplicaciones". Organizado por la UCLM, con duración de 20h en el 2001.
- Escuelas de verano "Administración de redes locales corporativas con servidores NT". Organizado por la UCLM, con una duración de 20 horas.
- Escuelas de verano "Nuevas tecnologías y desarrollo local". Organizado por la UCLM, con una duración de 20 horas.
- Escuelas de verano "IX Escuela de verano de informática: Tendencias en redes de altas prestaciones", organizado por el Vicerrectorado de Extensión Universitaria y la UCLM, con duración de 20h en 1999.
- Escuelas de verano "VIII Escuela de verano de informática: Sistemas expertos probabilísticos". Organizado por el Vicerrectorado de Extensión Universitaria y la UCLM, con duración de 20h en 1998.
- "Instalación y mantenimiento de servidores de información en Internet". Organizado por la Sección de Tecnología de la Información del IDR y el Departamento de Informática de la UCLM, con duración de 20h en 1998

EXPERIENCIA LABORAL

- ✓ 6 años como **profesor técnico de FP** en el IES Virrey Morcillo de Villarrobledo con destino definitivo.

- ✓ 3 años como **profesor técnico de FP** en Andalucía (Jaén y Granada) como funcionario de carrera.

- ✓ 1 año como **profesor interino de Secundaria** en Fuensalida.

- ✓ 1 año como **monitor de cursos de la JCCM** destinados a desempleados y trabajadores en activo.