Proyecto de Grado

Presentado ante la ilustre Universidad de Los Andes como requisito final para obtener el Título de Ingeniero de Sistemas

Modelo de Conocimiento Basado en Datos Enlazados para Analizar el Contexto en Aplicaciones Conscientes de la Localización

Por

Br. Carlos Alejandro Jiménez Holmquist

Tutor: Prof. Dr. Jose L. Aguilar C.

Asesor: M.Sc. Marxjhony Jerez

Junio 2018

Modelo de Conocimiento Basado en Datos Enlazados para Analizar el Contexto en Aplicaciones Conscientes de la Localización

Br. Carlos Alejandro Jiménez Holmquist

Proyecto de Grado — Sistemas Computacionales, 76 páginas

Resumen: En la actualidad, los avances tecnológicos son enormes, las capacidades de computo son muy altas, y los tamaños de los sensores disminuyen continuamente. Esto ha dado paso a las Aplicaciones Conscientes del Contexto, las cuales usan la información contextual del usuario para brindar una mejor experiencia. También, la cantidad de información disponible en la web es muy grande y crece a diario. Unir la información disponible en la web con la información del contexto de los usuarios, permite generar mas información a través de mecanismos de razonamiento. El sistema propuesto en este trabajo se basa en CARMiCLOC y CameOn, el primero es un middleware reflexivo que incorpora servicios para la consciencia de contexto. Por otro lado, el segundo forma parte de CARMiCLOC, dando solución al modelado del contexto usando ontologías. Basado en ambos trabajos, el sistema propuesto busca solucionar uno de los servicios de CARMiCLOC, el razonamiento, usando la información almacenada en CameOn. Para ello, en primer lugar, el sistema propuesto extiende semanticamente a CameOn, para poder enlazar los datos del contexto de los usuarios con la Web Semántica. Luego, usando Lógica de Primer Orden y Lógica Dialéctica, el sistema realiza procesos de inferencia de conocimiento a partir de CameOn enriquecido semanticamente, de tal manera de brindar el conocimiento generado como un servicio para las demás aplicaciones que lo invoquen. Es bueno resaltar, que el uso de la Lógica Dialéctica le permite manejar la incertidumbre presente en el contexto. Los resultados obtenidos muestran el gran potencial de usar datos enlazados con Lógica Dialéctica para generar nuevo conocimiento, y además, abren el camino en el área del razonamiento con enfoques hibridos que mezclen el enfoque clásico de la Lógica de Primer Orden con

lógica del tipo Lógica Dialéctica, para manejar situaciones ambiguas que se presenten en el análisis del conocimiento.

Palabras clave: Datos Enlazados, Web Semántica, Ontologías, Consciencia de Contexto, Lógica Dialéctica

Este trabajo fue procesado en LaTeX.

El Proyecto de Grado titulado "**Modelo de Conocimiento Basado en Datos Enlazados para Analizar el Contexto en Aplicaciones Conscientes de la Localización**", realizado por Br. **Carlos Alejandro Jiménez Holmquist**, C.I. N° 23.723.364, fue presentado el día (fecha): 18 de junio de 2018, en (lugar): Salón de reuniones EISULA, ante el Jurado evaluador conformado por:

Tutor:	Prof. Dr. Jose L. Aguilar C.
Jurado:	Prof. M.Sc. Hilda Contreras
Jurado:	Prof. Alejandro Mujica

Este proyecto no tiene mención especial.

A mis abuelos,

Eneria,

Bárbara,

Otón,

Liborio.

Índice

Índice de Tablas

Índice de Figuras

Agradecimientos

En primer lugar, agradezco a Dios por todo. Luego a toda mi familia, que siempre han sido un gran apoyo. Mis padres, siempre dispuestos a ayudarme, durante tantas noches de trasnocho, tantos proyectos, tantos "búscame" y "llévame". Agradezco a todos mis amigos, que de una manera u otra, me hicieron crecer y ser quien soy ahora. Agradezco especialmente a Julio, Miguel y sobre todo a Kristell, mi compañera de tantos trasnochos, de mucho estrés, innumerables risas, y tantas tareas y proyectos. Junto a ellos tres, mitad de mi carrera se fue de un momento a otro, pero me quedo con que esa mitad de la carrera fue una mitad con muchos momentos de risa, de inventos y principalmente de disfrutar mientras aprendíamos. Agradezco a Juliana por haber sido de tanto apoyo últimamente, en los momentos que los que no quería continuar siempre estuviste ahí para darme aliento y poder continuar, por compartir conmigo tanto y siempre estar para mi. Agradezco a mi tutor, que siempre estuvo pendiente de mi proyecto, presionándome para que trabajara, para que explotara mi potencial y alcanzara mi meta de convertirme en Ingeniero. A la Universidad de Los Andes, sus profesores, sus trabajadores y todos los que hacen vida en ella y que en algún momento u otro me tope. No quisiera terminar sin agradecer a todas aquellas personas que pusieron de su parte para ayudarme con este trabajo, son muchos para nombrar a cada uno, pero siempre conté con mucho apoyo para realizar este trabajo. Gracias a todos.

Capítulo 1

Introducción

1.1 Definición del problema

Actualmente, los avances tecnológicos en cuanto a las capacidades de cómputo y al tamaño de los sensores y dispositivos móviles Aguilar et al. (2015), han dado paso a un tipo de aplicaciones que desarrollan comportamientos distintos dependiendo del contexto en el que se encuentran. Estas aplicaciones conscientes de contexto (CAA, por sus siglas en inglés *Context Awareness Applications*), constantemente monitorean y reaccionan de manera distinta, para adaptarse a la situación o al contexto en el que se encuentren.

Normalmente, las Aplicaciones Conscientes del Contexto modelan el contexto en el cual se encuentran mediante el uso de ontologías, las cuáles representan las características más importantes del entorno, tales como el usuario, el ambiente y la aplicación. Sin embargo, dado que la aplicación cambia de ámbito, es necesario realizar un alineamiento de las diferentes ontologías que pueden caracterizar el medio, para poder generar una ontología única que contenga el contexto del momento para la aplicación que lo requiera.

Además, estas ontologías generadas pueden ser enriquecidas con la inmensa cantidad de datos que existen en la Web Semántica, la cual relaciona datos semánticamente a través de vínculos. De esta manera, se crean ontologías ricas en información, y en particular, enriquecerían semánticamente la descripción del contexto

actual de la aplicación.

Por otro lado, las aplicaciones de micro-localización han aparecido como un tipo de aplicación para los dispositivos móviles, para adecuarse a la ubicación actual del usuario. Eso implica que ellas necesitan modelar el contexto de donde están ubicados, para poder actuar adecuadamente.

No obstante, al tratarse de aplicaciones que cambian de contexto y que lidian con seres humanos, las ontologías pueden presentar situaciones ambiguas que llevan a razonamientos inexactos, por lo que las inferencias sobre dichas ontologías deberán realizarse usando la lógica tradicional, extendida con lógica dialéctica, para manejar esas situaciones.

En tal sentido, en este trabajo se propone realizar un modelo ontológico de contexto enriquecido semánticamente con datos enlazados, para aplicaciones de micro-localización, con un motor de inferencia que considere, además de la lógica clásica descriptiva, procesos de razonamiento a través de los datos enlazados y la lógica dialéctica.

1.2 Justificación

Actualmente existe una gran cantidad de datos en la web, los cuales por sí solos no tienen significado; sin embargo, si estos datos son convertidos a datos enlazados semánticamente con otros datos, podemos obtener información, conocimiento y sabiduría a partir de ellos. Además, en los últimos años la cantidad de datos enlazados que han ido apareciendo en la web es enorme, lo que ha originado que hayan venido apareciendo proyectos y trabajos que están explotando el potencial de los datos enlazados.

Por otro lado, estamos en una era en la que los teléfonos inteligentes se han convertido en parte del día a día de casi todos los seres humanos. Asimismo, la cantidad de aplicaciones para teléfonos móviles que existen y que se crean a diario es enorme. Ahora bien, las aplicaciones han comenzado a comportarse de una manera específica, brindando servicios concretos dependiendo de la ubicación en la que estas actúan. A este tipo de aplicaciones se les denomina aplicaciones de micro-localización.

Al tomar estos dos temas de gran auge y combinarlos, se obtiene un tema de investigación donde las cosas por hacer aún son aun muchas. En específico, las aplicaciones móviles necesitan modelar el contexto en el que se encuentran a partir de ontologías. Si estas ontologías son además enriquecidas con datos enlazados, y un mecanismo de razonamiento que maneje la ambigüedad, se puede aumentar el conocimiento que la aplicación posea, lo cual permitirá mejorar los servicios que brindan, y en específico, que funcionen de manera más eficiente y personalizada para cada usuario.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un sistema de contextualización como servicio, basado en el enlazado de datos y en la gestión de la incertidumbre, para entornos de aplicaciones de micro-localización.

1.3.2 Objetivos específicos

- Extender la ontología de Contexto CAMeOn, para que pueda explotar el enlazado de datos.
- Desarrollar el motor de Inferencia para CAMeOn, para descubrir conocimiento, basado en el enlazado de datos.
- Extender el motor de Inferencia para CAMeOn, para que maneje la incertidumbre, usando lógica dialéctica.

1.4 Antecedentes

Los antecedentes de este trabajo son las investigaciones realizadas en los ámbitos de modelado de contexto usando ontologías, enlazado de datos, lógica de primer

orden y lógica dialéctica, y las investigaciones sobre aplicaciones de micro-localización conscientes del contexto.

En Zhong-Jun et al. (2016), los autores proponen un modelo de una Ontología para un Contexto, donde se caracterizan semánticamente todas las dimensiones de la información del mismo. El modelo provee no solo información sobre el contexto a un alto nivel (meta-contexto), sino también conceptos más específicos de los dominios en los cuáles se desarrolla el contexto, lo cual logran mediante un modelo jerárquico. En dicho trabajo, su objetivo también es definir un motor de inferencia sobre la ontología del contexto; pero para poder hacer esto deben resolver el problema de adaptarse a los contextos dinámicos e inciertos que existen en la Web Semántica. La solución propuesta es un algoritmo de inferencia sobre la Ontología de Contexto, basado en Redes Bayesianas Dinámicas.

Por su parte, Wang et al. (2004) presentan CONON, una ontología implementada en el Lenguaje de Ontologías de la Web (OWL, por sus siglas en inglés *Ontology Web Language*), para modelar contextos en entornos informáticos. CONON captura los conceptos generales del contexto, y de la misma manera que en Zhong-Jun et al. (2016), permite agregar ontologías de dominios específicos de manera jerárquica. El objetivo de este trabajo fue razonar usando información explícita de bajo nivel sobre el contexto, para obtener información implícita de alto nivel. Por último, en el trabajo se hace un análisis de la factibilidad del razonamiento lógico en aplicaciones con contextos difusos, en los cuáles es difícil conocer los recursos informáticos disponibles.

Aguilar et al. (2015), por su parte, exponen un middleware reflexivo para aplicaciones Conscientes de Contexto, basado en la computación en la nube. La arquitectura para aplicaciones Conscientes de Contexto es compuesta por los 2 niveles clásicos que se presentan en todo middleware reflexivo. El nivel Base contiene toda la información de los usuarios y aplicaciones, y el contexto que estos comparten. El nivel Meta realiza las tareas de introspección para analizar y realizar los cambios en aplicaciones basadas en el contexto, usando los servicios que presenta Aguilar et al. (2015). Entre los servicios presentados se encuentra el razonamiento del contexto, el cual infiere información sobre un contexto.

El trabajo Aguilar et al. (2017) propone un modelo ontológico, como una

extensión del trabajo Aguilar et al. (2015), representativo del contexto el cual es además independiente de la aplicación. El modelo está caracterizado por dos niveles jerárquicos: el primero es una ontología general e independiente del dominio, con seis clases principales, las cuales son el usuario, la localización, el servicio, la actividad, el tiempo y el dispositivo. El segundo nivel jerárquico corresponde a ontologías de los dominios específicos de la aplicación.

En el campo de los datos enlazados se han desarrollado diversos trabajos. El trabajo Rodriguez et al. (2017) propone una metodología para desarrollar aplicaciones web que usen datos enlazados. La metodología presenta una serie de pasos para recolectar, limpiar, publicar y enlazar datos enlazados, y crear una aplicación que consulta estos datos.

En el artículo Phalle y Salunkhe (2009) se expone un sistema consciente de contexto para comentar y buscar recursos educativos, específicamente, vídeos. Para ello, desarrollaron una ontología para las anotaciones sobre los vídeos, la cual usa otras ontologías y vocabularios. Luego, usan los datos enlazados para clasificar los vídeos adecuadamente, evitando los problemas de ambigüedad, correctitud y completitud. Posteriormente desarrollaron una interfaz para el servicio que implementaron. Los autores proponen en futuros trabajos, integrar servicios dinámicos de descubrimiento, invocación y organización.

Por otro lado, Costabello (2013) presenta una tesis que discute la influencia de la consciencia de contexto en dispositivos móviles, accediendo a la Web de Datos. El trabajo fue dividido en 3 partes. La primera parte presenta una manera de describir el contexto, declarativamente, siguiendo los lineamientos de los datos enlazados. La segunda parte habilita la adaptación de la consciencia de contexto para el consumo de los datos enlazados. La tercera parte protege el acceso a las RDF triplestores desde los dispositivos conscientes de contexto. Para resolver la primera parte, el autor propone un vocabulario para el contexto, llamado PRISSMA. En la segunda parte se realiza la adaptación de contenido, para lo que es necesario la selección, generación o transformación de contenido. Para esto, el autor propone PRISSMA, un framework basado en un algoritmo de comparación de sub-grafos, que toma en cuenta las diferencias entre el contexto percibido y las descripciones del contexto. Para la

última parte, que corresponde a proteger el acceso a los RDF triplestores en ambientes computacionales de recursos limitados, el autor propone el framework Shi3ld, el cual agrega políticas de acceso, conscientes del contexto, a los triplestores.

A nivel de aplicaciones conscientes de contexto, basadas en micro-localización, tenemos el trabajo Posdorfer y Maalej (2016), el cual propone un sistema de encuestas consciente de contexto mediante el uso de Beacons. Los autores proponen un sistema que monitorea el contexto mediante el uso de dispositivos Bluetooth Low Energy (BLE, por sus siglas en inglés) Beacons, para saber la localización de los usuarios, las actividades que realizan, entre otros datos. Los autores reportan un resultado satisfactorio, aunque indican que se deben investigar e integrar otras maneras de monitorear el contexto. También, indican que al incrementar los datos que se poseen sobre el contexto, el servidor que desarrollaron necesitará mejoras para poder procesarlo. Por último, la aplicación nativa que desarrollaron necesita ser convertida, o adaptada, para trabajar en otros sistemas operativos móviles, ya que al momento de publicar el artículo solo estaba desarrollada en el sistema iOS.

En de Freitas et al. (2016), los autores exponen Bluewave, un sistema para compartir el contexto a través de Bluetooth, permitiendo de esta manera que los dispositivos puedan publicar sus contextos y obtener el contexto de los demás usuarios. Esto es logrado, ya que los dispositivos publican su contenido en un servidor, y publican unas credenciales temporales. Luego, los dispositivos que quieren acceder a ese contenido, deben hacer una petición al servidor donde está publicado el contenido, mediante el uso de las credenciales y pidiendo los campos que necesita del contexto.

Por último, al estar trabajando con ontologías es natural trabajar con lógica, por eso que en este proyecto proponemos usar además de la Lógica de Primer Orden, la Lógica Dialéctica, la cual tiene la ventaja que permite manejar ambigüedad. En ese sentido, en Pelletier et al. (2017) se describe un sistema para razonamiento usando lógica dialéctica, la cual es un tipo de lógica que permite a las sentencias ser verdaderas, falsas, o a diferencia de la lógica clásica, ambas al mismo tiempo. En ese trabajo se propone un probador de teoremas de lógica dialéctica de primer orden, el cual se desarrolló traduciendo sentencias de lógica dialéctica a lógica clásica de primer orden.

1.5 Metodología

El proyecto de grado será realizado usando la metodología MIDANO, propuesta por Pacheco et al. (2014); Aguilar et al. (2013), y la Metodología para el desarrollo de Aplicaciones Web utilizando Datos Enlazados (MEDAWEDE), propuesta en Rodriguez et al. (2017).

La metodología propuesta en Rodriguez et al. (2017), consta de seis fases para construir una aplicación web que use datos enlazados:

1. Especificación: donde se decide los datos que se usarán, y se definen sus URIs, entre otras cosas;
2. Modelado: donde se crea el modelo de datos que describe los datos que se usan;
3. Generación: donde se transforman los datos que se usarán al lenguaje RDF;
4. Vinculación: donde se vinculan los datos a otros datos enlazados para aumentar el valor de ellos;
5. Publicación: donde se publican los datos en un RDF Store para poder ser usados por otros.
6. Explotación de la información: donde se definen las interfaces que permitan consumir los datos publicados.

Así, dicha metodología propone una serie de fases para construir una aplicación web que use datos enlazados. Entre las primeras fases están la obtención, preparación y transformación de los datos a datos enlazados. Luego, al tener los datos que serán usados en la web semántica conectados con otros datos, se procede a publicar y construir los servicios e interfaces que consumirán estos datos. Las primeras cuatro fases de MEDAWEDE corresponden a la etapa de "Modelar los datos de la organización", y los últimas dos a la etapa de "Desarrollar la aplicación web".

Por otro lado, la metodología MIDANO es una metodología para el desarrollo de aplicaciones de minería de datos, basada en el análisis organizacional. Esta metodología propone tres fases:

1. Identificación de fuentes para la extracción de conocimiento en una organización, donde la finalidad de la etapa es realizar un proceso de ingeniería de conocimiento orientado a organizaciones de las cuáles se tiene poca información sobre los problemas, o procesos a estudiar. También, en esta fase se define la tarea de minería de datos, como el modelo de datos a usar.

2. Preparación y tratamiento de los datos, en la cual se plantea realizar una preparación de los datos, para generar datos listos para ser usados;

3. Desarrollo de herramientas de minería de datos, donde se desarrollan las tareas que consumirán los datos.

Como podemos ver, ambas metodologías poseen fases parecidas, por lo que se combinarán ambas metodologías para la formulación, desarrollo y experimentación con la herramienta que será implementada.

1.6 Alcance

Diseñar y desarrollar un sistema que se comporte como servicio, que sea capaz de navegar a través de las diferentes ontologías que posea, para conformar una ontología única correspondiente al contexto actual en la que se desarrolla la aplicación de micro-localización que consume el servicio. Además, sobre esta ontología, el sistema debe ser capaz de inferir, tanto información como conocimiento, usando un motor de inferencia. Ese motor de inferencia debe usar los datos enlazados en su proceso de razonamiento, así como resolver conflictos de ambigüedad y de incertidumbre que se presentan al trabajar con la ontología del contexto. En ese sentido, en la tesis se desarrolla un prototipo de dicho sistema.

Capítulo 2

Marco Teórico

2.1 Lógica

Al buscar la palabra "lógica" en la Real Academia Española (2014), se encuentra que la lógica es la "ciencia que expone las leyes, modos y formas de las proposiciones en relación con su verdad o falsedad.".

En Manzano y Huertas (2000) se describe a la lógica como el estudio de la consecuencia, o el estudio de los razonamientos válidos o correctos. Los autores caracterizan a la lógica como el estudio de los conjuntos de creencias consistentes.

Si vamos más alla, la lógica es la ciencia cuyo objeto de estudio es la inferencia, el cual es el proceso del cual se derivan conclusiones a partir de premisas. Al trabajar con ontologías y con información que ha sido enriquecida semánticamente, estamos trabajando con premisas, por lo que la inferencia es un proceso natural que se hace sobre las ontologías y sobre la información enriquecida semánticamente.

Al ser una ciencia, la lógica es un campo muy extenso, donde hay diferentes ramas para estudiar distintas maneras de razonar. En las siguientes secciones se tratara de explicar brevemente las dos ramas de la lógica que se usarán en el trabajo.

2.1.1 Lógica de Primer Orden

La lógica de primer orden es una vertiente de la lógica clásica, que según Manzano y Huertas (2000), se diferencia de la lógica clásica en que la lógica clásica es caracterizada

por su rigor y precisión; la verdad que ella dicta es absoluta y el tiempo no existe. La Lógica de Primer Orden una lógica que se basa en predicados (expresiones lingüísticas) con cuantificadores y conectores entre ellos, para formar una oración. La lógica de Primer Orden permite el razonamiento matemático, y en particular, esta tiene un poder expresivo muy superior al de la lógica proposicional.

En la lógica clásica, el lenguaje usado es tal que solo intervienen conectores, operaciones como la conjunción (una condición **y** otra condición) y la disyunción (una condición **u** otra condición). Los lenguajes de primer orden son aquellos que constan de variables, constantes, funciones, relaciones, cuantificadores, y símbolos lógicos como $\rightarrow$, $\leftrightarrow$, $\wedge$, $\vee$, entre otros. Las variables se refieren solo a los individuos, y los argumentos de las funciones (o predicados) son solo constantes o variables.

Un ejemplo de una oración en lógica de primer orden esta en la ecuación 2.1.

$$Publicacion(X) \wedge tieneAutor(X, Y) \wedge tieneAutor(X, Z) \rightarrow cooperoCon(Y, Z) \quad (2.1)$$

En el ejemplo 2.1 existe una variable X, la cual es una publicación, segun lo indica el predicado $Publicacion(X)$. Luego se afirma que la variable X tiene un autor Y y un autor Z, mediante los predicados $tieneAutor(X, Y)$ y $tieneAutor(X, Z)$. Con estos predicados, podemos concluir que Y cooperó con Z, lo cual es expuesto por el predicado $cooperoCon(Y, Z)$.

En general, se puede decir que la Lógica de Primer Orden es una extensión de la lógica clásica, que permite describir enunciados complejos donde intervienen variables y predicados.

2.1.2 Lógica Dialéctica

La Lógica Dialéctica es otra rama de la lógica clásica. En el trabajo de Pelletier et al. (2017), los autores la describen como una lógica que permite a sus fórmulas ser verdaderas o falsas; o a diferencia de la lógica clásica, verdadera y falsa al mismo tiempo; y los conectores son interpretados en términos de estos tres valores.

Una lógica paraconsistente es una lógica que trata las contradicciones, atenuándolas, ya que es tolerante a inconsistencias y permite que las contradicciones sean validas. Este tipo de lógica también es usada en los teoremas dialécticos.

Generalmente, en la literatura se citan paradojas semánticas, como por ejemplo "Esta oración es falsa", la cual es falsa si es verdadera, y es verdadera si es falsa. Otro ejemplo comúnmente citado por lo filósofos es: ¿Puede Dios crear una piedra tan pesada que incluso él no pueda levantar? o cuando una persona camina bajo el marco de una puerta ¿Está dentro de la habitación? o ¿Está fuera de la habitación?

La lógica dialéctica busca atacar estas vaguedades en los predicados permitiendo los tres valores de verdad, mediante el uso de cuatro casos:

1. *Declaraciones contingentes sobre el futuro*: Un contingente es un evento que puede suceder o no suceder, por lo que, en este caso, se busca describir la posible ocurrencia de eventos en el futuro. Ejemplos de ello son las frases: Mi madre irá a Londres, Habrá una competencia marítima mañana.

2. *Vaguedad que existe en el lenguaje natural*: En el lenguaje natural existen distintos casos de vaguedad, como los casos limítrofes (alto, bajo, pesado, liviano, experto, entre muchos otros), la vaguedad léxica donde existen significados no precisos en un enunciado y la ambigüedad o la generalidad en los enunciados. Ejemplos de sentencias de este tipo son, Juan es alto, pero, ¿qué es alto? Juan y Pedro son cuñados, ¿son cuñados de otras personas? o ¿cuñados el uno del otro?

3. *Fallo de una preposición*: Por ejemplo, a partir de un hecho realizar aseveraciones sobre su contexto, que después resultan que son falsas. Un ejemplo de lo anterior se puede ver en la frase: "Él es un bribón que robó las tartas", al asumir que: Hay un bribón, hubo tartas, alguien robó las tartas; y después resulta que alguna de esas cosas no suceden realmente en el contexto, como por ejemplo, que nunca haya habido tartas.

4. *Discurso ficticio*: Cuando se habla de entidades que no existen en un dominio actual, sino en otros dominios posibles. También, cuando se habla de entidades ficticias y de cómo podrían ser. Por ejemplo, las narrativas en novelas y películas que plasman los novelistas y directores de películas.

Esta lógica tiene cabida en el mundo real, ya que en ella existen inconsistencias en los cuales hay acciones que son verdaderas y falsas al mismo tiempo.

2.2 Ontologías

Filósofos como Aristoteles se han preocupado por saber qué existe y cómo podemos describirlo. Una vez que se sabe que algo existe, el siguiente paso es encontrarle un lugar entre todas las demás cosas. Para la filosofía, esto se conoce como ontología. En la Real Academia Española (2014), la ontología es definida como "las relaciones existentes entre los conceptos de un dominio o área del conocimiento.".

Para la informática, las ontologías son una definición formal de tipos, propiedades y relaciones entre entidades que existen en un dominio, o en otras palabras, es un modelo usado para describir el mundo. Se puede decir que una ontología cataloga las variables requeridas en un sistema computacional, y las relaciona, logrando así, poder ser usadas para resolver preguntas.

El software que opera con información enriquecida semánticamente necesita modelos del mundo de los cuales pueda, a partir del conocimiento que se posea, darles sentido y operar sobre ellos; para ello que se usan las ontologías. Una ontología provee un vocabulario preciso para representar el conocimiento que existe en un contexto. Este vocabulario permite representar y agrupar entidades, y también, las relaciones entre ellas.

Formalmente, una ontología $O = \{C, P, R, I\}$ es un conjunto de conceptos C sobre un dominio específico, con sus propiedades P, entre los cuales existen relaciones R, y los cuales poseen instancias I asociadas. Las instancias, en este caso, son las fuentes de datos que ingresan al sistema.

En especifico, los componentes de una ontología son:

- Los Conceptos o Clases (C), los cuales son los recursos que la ontología describe (entidades, objetos, personas, etc.).
- Las Propiedades (P), que se refieren a los aspectos, rasgos, características, entre otros, que las clases pueden tener.
- Las Relaciones (R), que describen la forma en la que las clases y los individuos se relacionan entre si.

- Las Restricciones, las cuales establecen descripciones formales de lo que deben cumplir los conceptos y/o sus relaciones.
- Los Individuos, los cuales son instancias de las clases o conceptos.
- Los Axiomas, los cuales son aserciones sobre los conceptos y sus relaciones descritas en una forma lógica, generalmente Lógica de Primer Orden. Los axiomas definen toda la teoría que describe la ontología.

Para poder escribir estas ontologías, la Web Semántica ha definido como estándar el lenguaje ***Web Ontology Language*** (conocido como OWL). Este consiste en un lenguaje de marcado que permite publicar y compartir ontologías en la WWW, esta construido sobre RDF (véase la sección 2.3.1) y codificado en XML.

2.3 Web Semántica

Del nombre **Web Semántica**, probablemente podemos deducir que se trata de algo relacionado con la ya conocida **World Wide Web** (WWW), y que además, tiene algo que ver con la semántica. La semántica trata de entender la naturaleza del significado, como es explicado en Dean Allemang (2011).

Por otro lado, gracias a la expansión de la WWW es de esperarse que casi toda persona conozca sobre ella, y esté familiarizado con la idea de la red de información que se comparte alrededor del mundo. A través de la WWW, cualquier persona puede difundir sus ideas al mundo. Para esto, basta con que un usuario se anime a publicar una página web, y la cargue de análisis, presentaciones, resúmenes, imágenes y demás contenido sobre el tema que le apasione.

De esta manera, la WWW puede sentirse demasiado grande y demasiado pequeña a la vez. Suponga que usted busca información sobre restaurantes en la WWW y consigue una página donde los usuarios recomiendan cierto plato de un restaurante, y al usted dirigirse al sitio web del restaurante, consigue que el lugar no ofrece esa clase de platos. Usted busca la sucursal de una tienda en la web de la ciudad y consigue su dirección, pero cuando busca en la web de la tienda si la sucursal está en esa dirección, ya dicha tienda no está en ese lugar. Le gustaría ir a ver una película al cine, por

lo que busca la película en una web sobre los cines de su ciudad, y al momento de llegar al cine la película no esta siendo proyectada, luego revisa la web del cine y la película efectivamente no esta siendo proyectada. Entonces, ¿Cómo podemos hacer que la experiencia en la web sea más íntegra y consistente? ¿Necesitamos aplicaciones web más inteligentes? O tal vez ¿Necesitamos infraestructura web más inteligente?

Actualmente, la web está llena de aplicaciones web inteligentes; todos los días hay innovaciones en el área. Podemos verlo cuando hacemos búsquedas en la web y los buscadores dan resultados intuitivos, también cuando los sitios de compras en la web nos hacen recomendaciones que podrían gustarnos basadas en lo que hemos comprado recientemente; y más ejemplos como estos salen a la web diariamente. Entonces, es lógico pensar que si construimos infraestructura inteligente que incluya todas estas aplicaciones y más, la web mejore. Mientras más inteligente es la infraestructura, mejor es la experiencia en la web ¿no? Podríamos pensar esto, pero no sería práctico ni posible debido a la gran cantidad y variedad de aplicaciones inteligentes que podemos conseguir. Por lo tanto, podemos decir que el comportamiento inteligente de la web proviene de las aplicaciones que existen, no de la infraestructura existente.

Las aplicaciones inteligentes en la web son tan inteligentes como se lo permitan los datos que existen en la misma. Datos inconsistentes resultarán en información inconsistente, y por ende, en una experiencia inconsistente. La razón de mejorar la infraestructura web es permitirle a las aplicaciones web hacer uso de todo su potencial, permitiéndoles conseguir todos los datos que necesitan, de manera que la información sea consistente y completa.

La Web Semántica no hace los datos “inteligentes”, ya que datos “inteligentes” son de lo que ella se alimenta para cumplir su objetivo, el cual consiste en llevar los datos correctos al lugar correcto, de manera que las aplicaciones web puedan hacer su trabajo.

Podemos concluir que el reto de la Web Semántica no es desarrollar infraestructura web inteligente, sino más bien, desarrollar infraestructura web que permita, apropiada y fácilmente, la integración de los datos para que las aplicaciones puedan hacer uso de ellos.

En la figura 2.1 podemos ver la arquitectura de la Web Semántica. En la

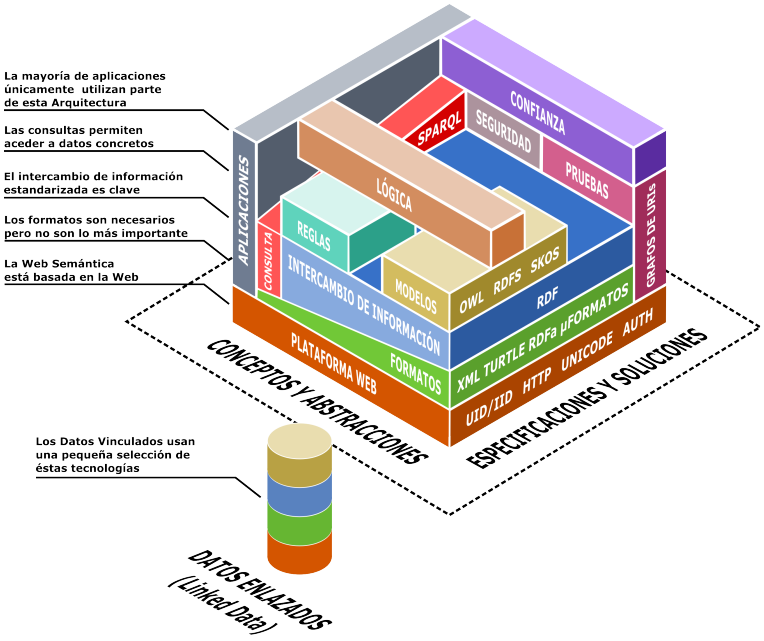

Figura 2.1: Arquitectura de la Web Semántica

base encontramos la plataforma web, en ella se utiliza el protocolo HTTP (*Hypertext Transfer Protocol*, HTTP por sus siglas en inglés) y los UID (*Unique Identifier*, UID por sus siglas en inglés). Sobre ella, están los formatos necesarios para poder estandarizar la manera de compartir la información y usa XML (Extensible Markup Language, XML por sus siglas en inglés), entre otros. Luego de tener la información estandarizada, es necesario para la Web Semántica poder compartir e intercambiar la información, para lo cual se usa el lenguaje RDF. Luego, sobre esta información estandarizada, y también extendida con más información (usando OWL, RDFS, etc.). Es importante tener una manera de consultarla, para lo que se usa SPARQL (ver sección 2.3.2). Todos estos componentes conforman la arquitectura de la Web Semántica; aunque la mayoría de las aplicaciones no utilicen todos los componentes.

2.3.1 RDF

Una posible pregunta para las personas que ven la Web Semántica y los Datos Enlazados por primera vez sería ¿Cómo publico la información? ¿Cómo hago que la información esté disponible para las demás personas? Es aquí donde entra en juego

el ***Resource Description Framework*** (RDF, por sus siglas en inglés).

El RDF es una recomendacion del ***World Wide Web Consortium*** (W3C, por sus siglas en inglés), que según Segaran et al. (2009), provee una manera estándar de expresar grafos de datos (que son **Datos Enlazados**), y también, una manera de compartir estos datos. Pero tal vez la contribución más importante del RDF no es ninguna de las mencionadas anteriormente. En la sección anterior, también se habló de datos que las máquinas entiendan, y es esta la contribución más importante del RDF, la de proveer una manera de que los datos que fueron enriquecidos por los usuarios de la Web Semántica sean también entendidos por las máquinas (con máquinas nos referimos a programas, aplicaciones web, dispositivos, etc.).

Una gran cantidad de herramientas y servicios han emergido alrededor del RDF por ser este una recomendación del W3C, lo que lo hace un estándar de la industria, o de la Web Semántica.

En sí, RDF es un lenguaje para modelado de datos. Este permite expresar dichos modelos de datos usando oraciones, o tripletas. Las tripletas son un conjunto de sujeto, predicado y objeto, donde el sujeto debe ser un recurso representado por una URI, el predicado debe ser un verbo, y el objeto puede ser tanto un literal (una cadena de caracteres) u otro recurso. Además, RDF introduce varios conceptos que hacen estos modelos de tripletas más precisos y robustos, reduciendo ambigüedad en la transmisión de datos semánticos entre las máquinas.

Como es mencionado en Dean Allemang (2011), RDF es uno de los lenguajes de representación básicos de la web, además, es el cimiento de los otros dos, OWL y ***Resource Description Framework Schema*** (RDFS, por sus siglas en inglés). Un ejemplo de tripletas en formato Notation3 o N3, es expuesto en la Figura 2.2.

```
@prefix ex: <http://www.example.com/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
ex:CarlosJimenez rdf:type foaf:Person;
        foaf:name "Carlos Jimenez";
```

Figura 2.2: Ejemplo de tripleta en formato Notation3

En el ejemplo de la Figura 2.2 podemos ver varios nombres: `ex` es un

vocabulario para referirse a los recursos, propiedades e instancias definidas en el URI `http://www.example.com/`. Por su lado, `foaf` es otra ontología que describe personas, sus actividades, y relaciones con otras personas y objetos. Finalmente, `rdf` define la sintaxis para evaluar la sentencia que viene de ser escrita. Todos estos están antecedidos por la palabra `@prefix`, indicando que son un espacio de nombres. Los nombres que les siguen indican que entidad está definida en ellos. En el ejemplo, tenemos una entidad, que es un recurso, llamado `CarlosJimenez`, perteneciente al espacio de nombre `ex` (o `http://www.example.com/`), lo cual convertiría su URI en `http://www.example.com/CarlosJimenez`. Otras entidades son `Person` perteneciente al espacio de nombre `foaf`, y `type`, perteneciente a `rdf`. Estas sirven para indicar propiedades. Por ejemplo `Person` indica que el recurso `CarlosJimenez` es una persona cuyo nombre (`name`) es ‘‘`Carlos Jimenez`’’. Vemos en este ejemplo que los literales son representados entre comillas dobles. El grafo que representa las tripletas del ejemplo 2.2 puede ser visto en la Figura 2.3.

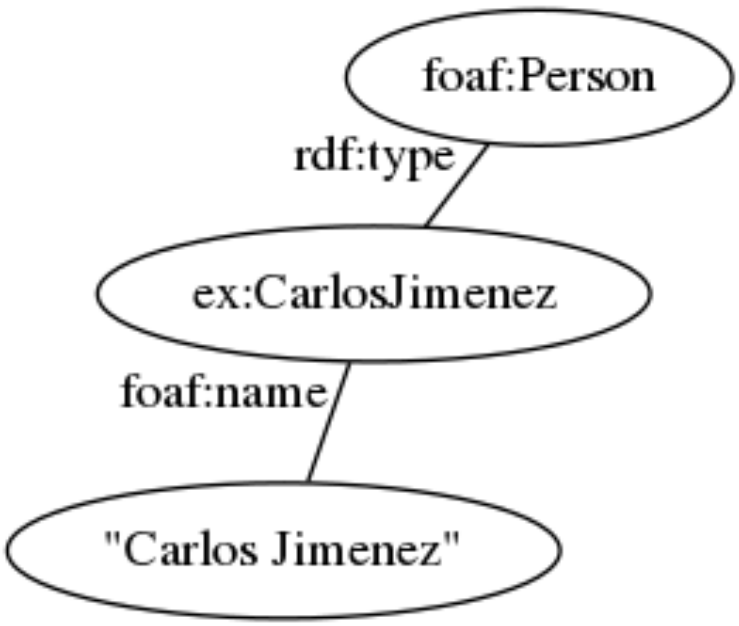

Figura 2.3: Grafo que representa las tripletas del ejemplo de la Figura 2.2

2.3.2 SPARQL

De manera general, las tripletas son las maneras más simples de representar una conexión entre dos recursos, pero de nada nos sirve tener los datos representados si no tenemos una manera de acceder a ellos. Es aquí donde entra en juego SPARQL, el estándar para acceder a datos RDF. ***SPARQL Protocol And RDF Query Language*** (SPARQL, por sus siglas en inglés) es un lenguaje de consultas que trabaja basado en la estructura del RDF.

Al igual que RDF, SPARQL es una recomendación del W3C, por lo que es el estándar de la Web Semántica para hacer consultas.

Las consultas en SPARQL son representadas en un lenguaje inspirado del lenguaje Turtle, pero comparten muchas características con otros lenguajes de consulta, en particular con SQL, por lo que su formato se asemeja mucho.

Un ejemplo sencillo de una consulta en SPARQL, sobre las tripletas del ejemplo de la Figura 2.2, puede ser visto en la Figura 2.4.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
PREFIX foaf: <http://xmlns.com/foaf/0.1/>.
SELECT ?uri ?name
WHERE{
    ?uri rdf:type foaf:Person.
    ?uri foaf:name ?name.
}
```

Figura 2.4: Ejemplo de una consulta en SPARQL

Las consultas en SPARQL tratan de emparejar los patrones descritos en la sección `WHERE` con las variables, las cuales son precedidas por un signo de interrogación (`?`). El ejemplo de la Figura 2.4 trata de encontrar una variable llamada `uri`, que es de tipo persona (indicado por la tripleta `?uri rdf:type foaf:Person.`), que tiene un nombre, el cual estará en la variable `name` (indicado en `?uri foaf:name ?name.`). Después de emparejar los patrones, selecciona las variables `uri` y `name`, en el bloque `SELECT`. El resultado de esta consulta sobre los datos del ejemplo de la Figura 2.2 sería el indicado en la Tabla 2.1.

Tabla 2.1: Resultado de la consulta en el ejemplo de la Figura 2.4

uri	**name**
ex:CarlosJimenez	Carlos Jimenez

2.3.3 Datos Enlazados

En los ejemplos expuestos previamente sobre la Web Semántica ¿Cuál era el problema? Los datos no eran consistentes. En el ejemplo del restaurante se decía que el mismo servía cierto plato, pero al dirigirse a la página web del restaurante este no ofrece ese plato; puede que alguna de las dos páginas esté desactualizada. Lo mismo sucede con el ejemplo del cine. Probablemente al administrador de la página del cine se le olvidó actualizar la información de la página web. En el ejemplo de la tienda, la página web que posee la dirección, está en texto fijo, y no hay una noción explícita de que ese texto representa una dirección.

Los humanos tenemos la gran capacidad de poder leer ese texto, y debido a que tenemos un contexto, sabemos que las direcciones se representan de cierto modo. Por lo tanto, podremos interpretar la información como lo que representa, por ejemplo, una dirección. Sin embargo, para las máquinas, esto no es algo explícito. Debemos indicarle a ellas qué significa el texto que estamos colocando como un texto fijo en la página web.

Si las páginas web de los lugares implicados en los problemas de los ejemplos se construyeran siguiendo la información que contiene una base de datos, probablemente la información podría estar actualizada si la base de datos se actualiza. Pero esto lleva a otro problema, las bases de datos de las web implicadas son distintas, y si se propone una base de datos centralizada para los casos mencionados, por ejemplo, una base de datos conjunta para la página web de opiniones de restaurantes y la información del restaurante como empresa, esta idea podrá, quizás, ser encontrada descabellada o absurda.

Entonces, la solución propuesta por la Web Semántica es que el modelo de los datos no sea centralizado en una base de datos que sea llevada por varias organizaciones, sino un modelo de datos único y consistente, descentralizado, que sea parte de la infraestructura de la web. Cuando un usuario, por ejemplo el administrador de la

página del cine el cual no proyectaba la película, publica información, no lo hace de manera que sea solo información entendible para humanos. Lo hace de manera que sea una descripción de la información, que además sea distribuible y leíble por máquinas. Para saber más sobre el modelo de datos que usa la Web Semántica para este fin, véase 2.3.1.

Los datos entonces son publicados en la web de una manera distribuible, y de una forma para que las máquinas lo entiendan. Estos datos son lo que llamamos datos enriquecidos semánticamente, que por medio del modelo de datos que se usa, están conectados a otros datos, y no solo conectados, sino conectados con un sentido. A su vez, los datos a los que se conectan los datos iniciales, están conectados a más datos, y así sucesivamente (véase la Figura 2.5).

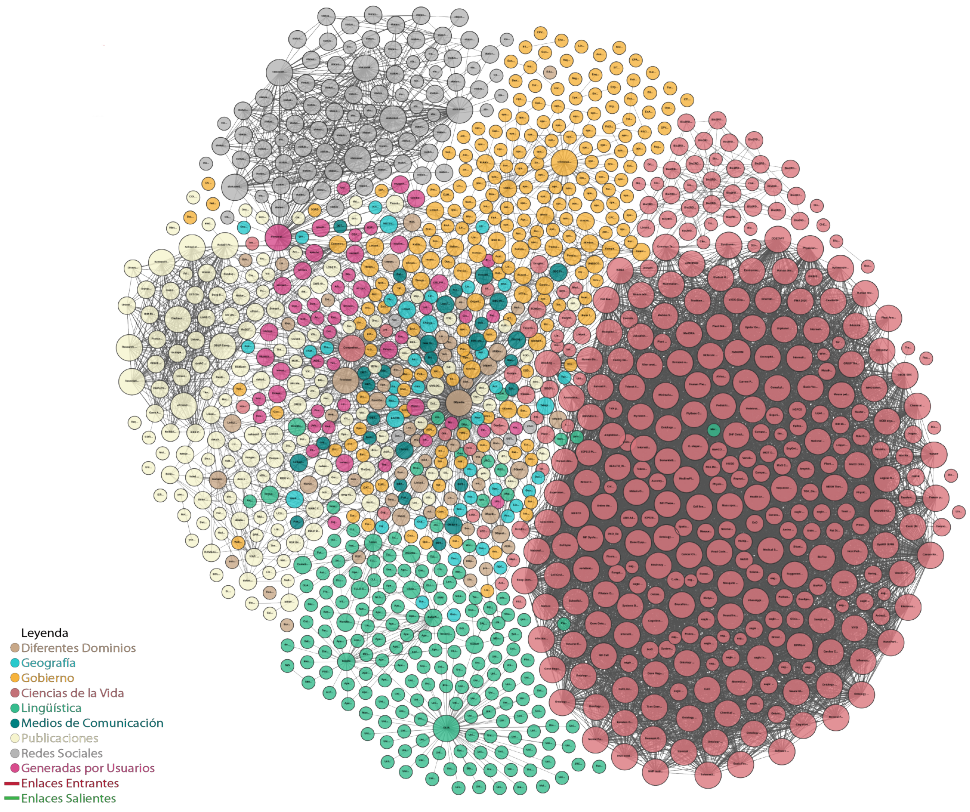

Figura 2.5: Conjuntos de Datos enlazados en la Web en febrero de 2017

El diagrama de la Figura 2.5 muestra los conjuntos de datos enlazados que

estaban conectados para febrero de 2017.

El creador de la WWW, y tambien precursor de la idea de la Web Semantica, Tim Berners-Lee, expone cuatro principios básicos en su nota Berners-Lee (2006):

1. Usar ***Uniform Resource Identifiers*** (URIs, por sus siglas en inglés) como los identificadores de las cosas.

2. Usar URIs HTTP para que las personas puedan buscar esos nombres.

3. Proveer información útil acerca de los URIs usando estándares (RDF ver 2.3.1, SPARQL ver 2.3.2) y vocabularios, para cuando las personas los busquen.

4. Incluir enlaces a otros URIs, para que se puedan descubrir más cosas.

Para ilustrar el enlazado de datos, consideremos el siguiente ejemplo de la Figura 2.6, en formato Notation3, el cual representa el enlazado de datos del ejemplo de la Figura 2.2.

```
@prefix ex: <http://www.example.com/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@prefix dc: <http://dublincore.org/documents/2012/06/14/dcmi-terms/>.
ex:CarlosJimenez rdf:type foaf:Person;
        foaf:name "Carlos Jimenez";
        dc:creator ex:ProyectoDeGrado.
ex:ProyectoDeGrado rdf:type foaf:Document.
        rdfs:label "Modelo de Conocimiento Basado en Datos Enlazados
                                para Analizar el Contexto en Aplicaciones
                                Conscientes de la Localización"@es.
```

Figura 2.6: Ejemplo de Datos Enlazados escritos en formato Notation3.

En la Figura 2.6 se extiende el ejemplo de la Figura 2.2 agregando más vocabularios, como lo son `rdfs` (un vocabulario para describir propiedades de los recursos) y `dc` (un vocabulario para proporcionar información descriptiva para cualquier recurso en la Web Semántica). Además del recurso `ex:CarlosJimenez`, ahora también

existe el recurso `ex:ProyectoDeGrado`. Se indica que `ex:CarlosJimenez` es el creador de `ex:ProyectoDeGrado`. Adicionalmente, del último se indica que es un documento cuyo nombre se define mediante la propiedad `label` del vocabulario `rdfs`.

El enlazado de datos ocurre cuando conectamos mediante relaciones a recursos, como esta sucediendo en el ejemplo de la Figura 2.6. El grafo correspondiente a los datos del ejemplo de la Figura 2.6 se puede ver en la Figura 2.7.

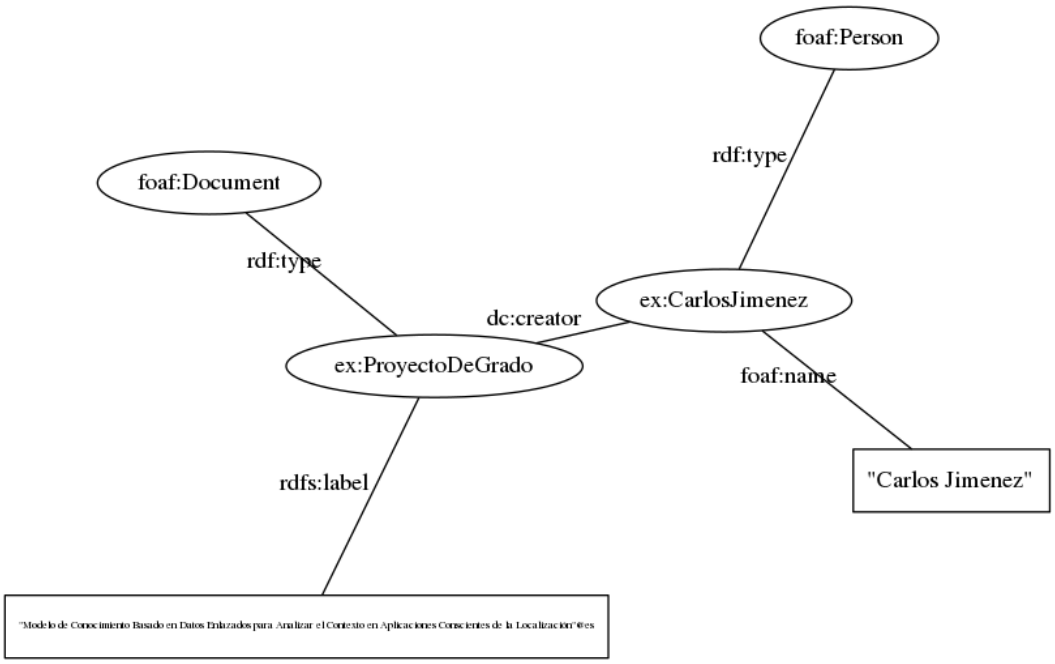

Figura 2.7: Grafo que representa las tripletas del ejemplo del la Figura 2.6

Para consultar este grafo, podríamos hacer la consulta en SPARQL expuesta en la Figura 2.8. La salida de la consulta de la Figura 2.8 es expuesta en la Tabla 2.2.

Tabla 2.2: Resultado de la consulta en el ejemplo de la Figura 2.8

author	**title**
Carlos Jimenez	Modelo de Conocimiento Basado en Datos Enlazados para Analizar el Contexto en Aplicaciones Conscientes de la Localización

Mediante el enlazado de datos, ahora podemos descubrir nuevos recursos, que están conectados a través de las relaciones de la Web Semántica, y que previamente no sabíamos que estaban.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
PREFIX foaf: <http://xmlns.com/foaf/0.1/>.
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dc: <http://dublincore.org/documents/2012/06/14/dcmi-terms/>.
SELECT ?name ?title
WHERE{
    ?author rdf:type foaf:Person.
    ?author foaf:name ?name.
    ?author dc:creator ?publication.
    ?publication rdfs:label ?title.
}
```

Figura 2.8: Ejemplo de consulta en SPARQL usando datos enlazados.

Con la misma consulta de la Figura 2.8 podemos conseguir todos los autores y sus respectivas publicaciones, siempre y cuando estén publicados de la manera correspondiente en la web, mostrando así el gran potencial a la Web Semántica.

2.4 Consciencia de Contexto

De acuerdo con Dey (2001), para la computación, "el contexto es cualquier información que pueda ser usada para caracterizar el entorno de una entidad. Una entidad es una persona o un objeto que es considerado importante para la interacción entre el usuario y la aplicación". El mismo autor describe los sistemas conscientes de contexto como los que proveen información importante que caracteriza su entorno a sus usuarios, la cual les es útil en la situación en la que se encuentran.

La interacción con los sistemas conscientes de contexto viene dada en dos puntos, la ejecución y la configuración. La ejecución se refiere a las acciones y comportamientos del sistema en una situación especifica, por ejemplo, silenciar el teléfono mientras el usuario duerme. La configuración son los ajustes hechos a las acciones y comportamientos del sistema.

El contexto posee un ciclo de vida, el cual define cuándo la información es generada y cuándo es consumida (ver Figura 2.9). En este ciclo de vida, la consciencia de contexto puede verse como un servicio. Sus cuatro fases son:

- Adquisición de Contexto: Las técnicas para adquirir el contexto pueden variar, basadas en quién es el responsable, la frecuencia, la fuente, el proceso de adquisición y los tipos de sensores, entre otros.
- Modelado de Contexto: También llamado la representación del contexto, puede ser estática o dinámica. Además, existen muchas técnicas para modelarlos, tales como las basadas en ontologías, en lógica, los esquemas Clave - Valor, entre otros.
- Razonamiento de Contexto: El razonamiento de contexto puede ser definido como un método para deducir nuevo conocimiento a partir del contexto. También puede ser definido como un proceso para derivar contextos de alto nivel a partir de un conjunto de contextos. Antes de razonar, se deben resolver dos problemas sobre la información del contexto: la imperfección y la incertidumbre.
- Distribución de Contexto: Se usan dos métodos: por petición, donde el usuario solicita el contexto y el sistema responde; y por subscripción, o también llamado publicación.

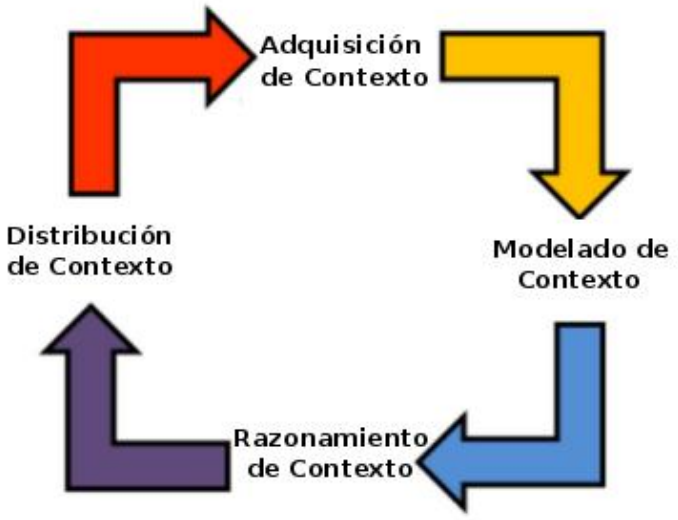

Figura 2.9: Ciclo de vida del contexto (Fuente Aguilar et al. (2017))

La forma común de modelar un sistema consciente de contexto es basada en el modelo clásico de computación autonómica. Este modelo consiste en un conjunto

de servicios que monitorean el contexto, otro conjunto de servicios que analizan el contexto, un conjunto de servicios que planean las acciones que deberá realizar el sistema consciente del contexto para suplir las necesidades en el sistema, y por último, un conjunto de servicios para ejecutar las acciones planeadas por los servicios anteriormente mencionados. Para funcionar, todos estos servicios requieren de una base de conocimientos. En Aguilar et al. (2015) se encuentra una propuesta detallada basada en estas ideas, de un Middleware Autonómico de Servicios de Conciencia de Contexto.

Así, un sistema de consciencia de contexto recolecta información sobre el entorno y presta servicios de información que pueden ser importantes para sus usuarios (aplicaciones, personas, etc.).

Capítulo 3

Contexto del Proyecto de Grado

Este trabajo está basado en dos trabajos previos. El primero, un middleware reflexivo para aplicaciones conscientes del contexto llamado CARMICLOC, descrito en Aguilar et al. (2015). Este sirve como base teórica para el segundo trabajo, que es presentado en Aguilar et al. (2017), el cual introduce una ontología para modelar contextos. Debido a que este trabajo se hizo en el contexto de dichos trabajos, en esta sección se presentan los mismos.

3.1 CarmiCLOC

Los avances tecnológicos en las áreas de sensores y dispositivos móviles, entre otras, han permitido el surgimiento de una nueva generación de aplicaciones, llamadas Aplicaciones Conscientes de Contexto, las cuales están en constante monitoreo del ambiente y cambian según sea necesario.

Debido al constante monitoreo que ellas requieren, las aplicaciones necesitan ciertos servicios y funcionalidades que les permita gestionar sus contextos, los cuales deben ser dinámicos debido a los constantes cambios en los contextos. Además, estos servicios pueden involucrar una gran cantidad de datos y capacidades de procesamiento, los cuales no pueden ser ofrecidos por los equipos móviles, por lo que suele requerirse el uso de la nube.

Para resolver estos problemas, Aguilar et al. (2015) propone ***Context***

Awaresness Reflective Middleware in Cloud Computing (CARMiCLOC, por sus sigas en inglés), el cual esta basado en servicios web que pueden ser consumidos por aplicaciones conscientes del contexto o no. En particular, **CARMiCLOC** es un Middleware Reflexivo que incorpora la computación autonómica para la auto-adaptación, e incluye el paradigma de computación en la nube para permitir comportarse como un SaaS (Software as a Service) o como un PaaS (Platform as a Service).

Este middleware esta dividido en dos niveles, como es normal en los middlewares reflexivos. El nivel base, referido a los usuarios y aplicaciones que se encuentras en la zona de contexto compartido. En este nivel, el middleware debe conocer y monitorear las interacciones en el contexto y sus cambios. El segundo nivel, o nivel meta, ofrece la capacidad de reflexión, a partir de la observación del nivel base. En este nivel se encuentran los servicios que ofrece **CARMiCLOC**, el gestor de servicios de contexto, y un registro de todos los servicios en uso en el nivel base.

La arquitectura, además, se conforma de las aplicaciones manejadas, los puntos de enlace con ellas para monitorearlas o actuar sobre ellas, un gerente autonómico que implementa los lazos inteligentes de gestión del contexto que automatizan la auto-regulación de las aplicaciones, el orquestador de los gerentes autonómicos que los coordina, un gerente manual que crea la interfaz hombre-máquina, y por último, las fuentes de conocimiento. La fuente de conocimiento puede hacer uso de la nube para guardar y procesar data cuando los dispositivos no tienen suficiente capacidad de cómputo. La arquitectura de **CARMiCLOC** se puede apreciar en la figura 3.1.

3.1.1 Servicios de CARMiCLOC

CARMiCLOC tiene siete servicios para realizar la gestión de contexto, y pueden ser consumidos por aplicaciones conscientes de contexto o no. Estos servicios son necesarios para la gestión del ciclo de vida del contexto (véase la Figura 2.9). Los siete servicios son:

1. Adquisición del Contexto/Pre-configuración (Se1): En este servicio se encuentran las actividades de descubrimiento del contexto y anotación semántica del

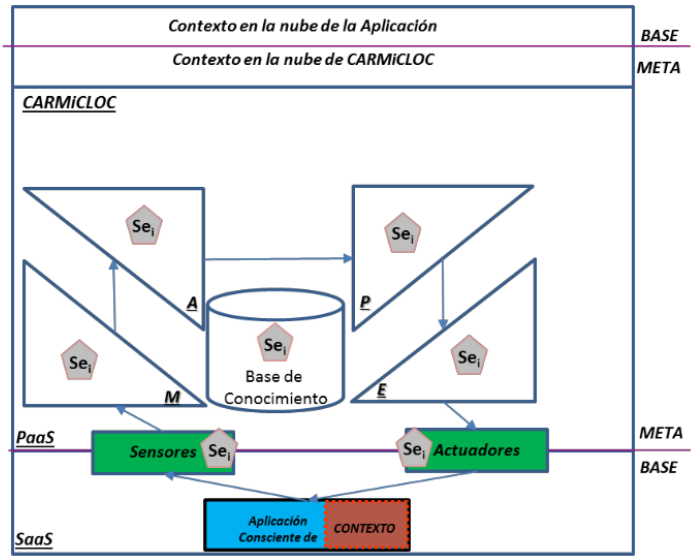

Figura 3.1: Arquitectura de **CARMiCLOC** (Fuente Aguilar et al. (2015))

contexto. Se basa en la extracción de datos desde los sensores y el pre-procesamiento de los mismos. Además, configura de manera automática los sensores y donde se registran sus datos.

2. Modelado del Contexto (Se2): Es uno de los servicios más importantes, ya que es donde se representa el contexto en forma de conocimiento para después poder ser usado para razonar, e inferir información, entre otras cosas. CARMiCLOC usa ontologías para el modelado del conocimiento, en particular CameOn (ver sección 3.2). Por último, también tiene tareas de almacenamiento de data y la gestión de herramientas para su explotación.

3. Razonamiento de Contexto (Se3): Este servicio es el responsable de aplicar las técnicas de razonamiento. Eventualmente, también necesita analizar la incertidumbre en el contexto, y en general, infiere posibles conclusiones del contexto a partir del conocimiento que se tenga del mismo.

4. Distribución de Contexto (Se4): Es responsable de hacer la entrega del contexto a los usuarios. Da formato a los datos del contexto, facilita la accesibilidad de los mismos y maneja requerimientos específicos de los usuarios, entre otras cosas.

5. Verificación y resolución de inconsistencia del Contexto (Se5): Es el responsable de detectar inconsistencias en el contexto y tomar acciones para resolverlas.

6. Seguridad del Contexto (Se6): Es el responsable de la seguridad (privacidad e integridad de los datos) en todo el ciclo de vida del contexto.

7. Gestor de Servicios del Contexto (Se7): Es responsable de la gestión de los servicios dentro del contexto. Estos incluyen los servicios que puedan estar en la nube. Este gestor es un orquestador de los servicios de **CARMiCLOC**.

En la Figura 3.2 podemos ver cómo se incorporan los servicios de **CARMiCLOC** en la arquitectura MAPE en la que el middleware se basa. Por ejemplo, el servicio Se1 pertenece a la fase de Monitoreo de **CARMiCLOC**.

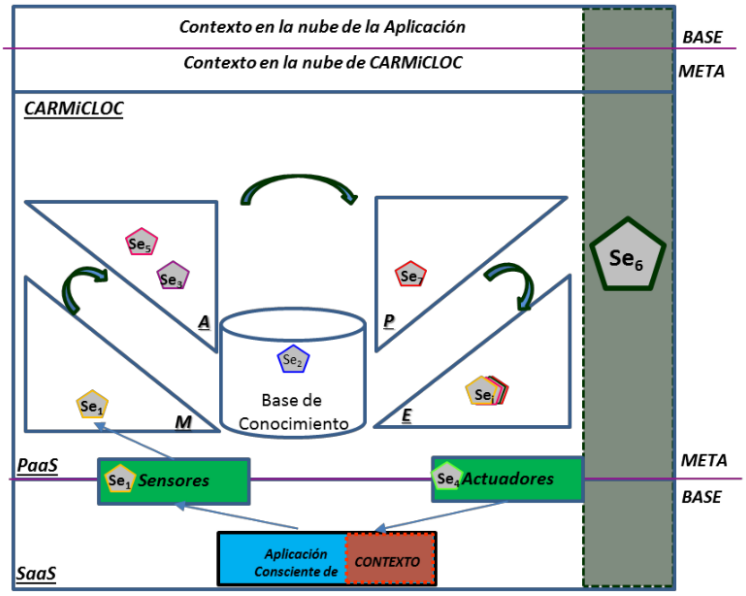

Figura 3.2: Servicios de **CARMiCLOC** incorporados en su arquitectura (Fuente Aguilar et al. (2015))

3.2 CameOn

En un ambiente inteligente, los usuarios quieren ser automáticamente proveídos de servicios pertinentes. Para esto, es necesario poseer información contextual apropiada.

Por ejemplo, la situación del usuario, su localización, el tiempo, los dispositivos, el ambiente, entre otros. Actualmente, existen muchos dominios que necesitan el modelado del contexto y hacen uso de él, como lo son la Realidad Virtual, el Internet de las Cosas (IoT, por sus siglas en inglés *Internet of Things*), el Internet de Todo (IoE, por sus siglas en inglés *Internet of Everything*), entre otros.

Para el modelado de contexto existen muchas técnicas, pero el uso de ontologías para expresar el contexto provee una ventaja, puesto que estas pueden expresar las distintas características del contexto, por lo que esta técnica ha ganado fuerza y se ha establecido como un estándar gracias a su interoperabilidad y su expresividad semántica.

En específico, para poder modelar el contexto de cualquier dominio, es necesaria una ontología que capture de manera genérica conceptos de alto nivel del contexto. Además, el modelo de contexto debe proveer mecanismos para extender la información específica de un contexto. **CameOn** es propuesto por Aguilar et al. (2017) como una ontología con estas características, basado en el principio de *las 5W*: quién, cuándo, qué, dónde y por qué (en inglés, *who, when, what, where and why*). **CameOn** es usado por **CARMICLOC** en sus servicios de representación de contexto, razonamiento sobre el contexto, compartición del contexto, entre otros servicios, ya que **CARMICLOC**, como un middleware para aplicaciones conscientes del contexto, necesita una base de conocimientos que sea usada por sus servicios, siendo **CameOn** parte de ella.

Particularmente, la información en un modelo de contexto debe estar definida en términos de atributos, características, relaciones contextuales, entre otros. Un enfoque simple y completo incorpora solo 4W, quién (who?), qué (what?), dónde (where?) y cuándo (when?). Debido a que CameOn busca usarse en aplicaciones móviles, es importante conocer quién ejecuta las acciones y con quién (who?). Además, se requiere establecer las relaciones entre el ambiente y los servicios que son ejecutados, o podrían ejecutarse. Todo lo anterior define el por qué (why?) en CameOn, que ayuda a responder la motivación del para qué (what?). Además, **CameOn** incorpora clases para responder las preguntas de cuándo (when?) y dónde (where?).

Por otro lado, la ontología **CameOn** está caracterizada por dos niveles jerárquicos; el primero, una ontología general independiente del dominio; y el segundo,

permite incorporar ontologías de dominios específicos (ver Figura 3.3). Además, el primer nivel de **CameOn** categoriza el contexto en tres categorías contextuales: contexto interno, contexto externo y contexto de frontera.

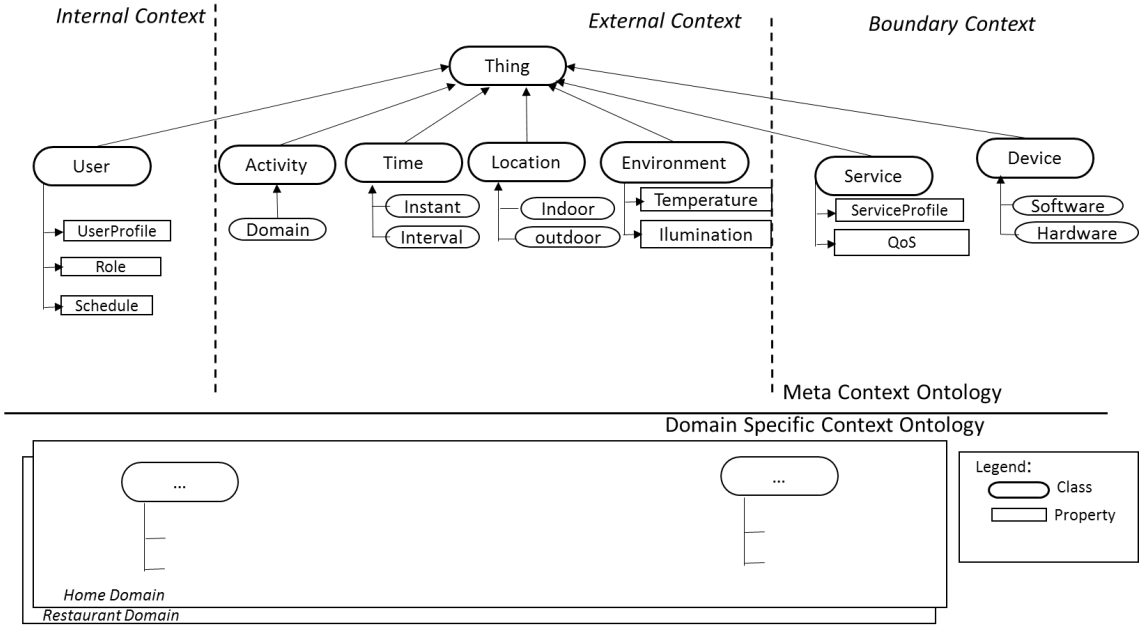

Figura 3.3: Modelo jerárquico de **CameOn** (Fuente Aguilar et al. (2017))

- Contexto Interno: describe al usuario y elementos personales como su perfil y su rol. Es descrito por la siguiente categoría contextual:
  - Usuario: Contiene información sobre el perfil del usuario, situación y preferencias.
- Contexto externo: describe el entorno que rodea al usuario, y es descrito por las siguientes categorías contextuales:
  - Actividad: Describe las distintas actividades que pueden ser desarrolladas en el contexto.
  - Tiempo: Describe la noción de tiempo en el contexto, la cual puede ser usada para definir la cronología de las situaciones.

  - Ubicación: Describe la ubicación del contexto, tanto su espacio interior como exterior.
  - Ambiente: Describe las condiciones y propiedades de la ubicación del usuario.

- Contexto de Frontera: describe las entidades que realizan la interacción entre el usuario y el entorno, y es descrito por las siguientes categorías contextuales:
  - Dispositivo: Describe los dispositivos de software y hardware en el contexto.
  - Servicios: Describe los servicios existentes sobre el contexto (en este caso, los definidos en la sección 3.1.1).

Además de estas clases principales, **CameOn** incorpora otras clases que ayudan al modelado del contexto. Ejemplos de estas clases se pueden ver en la Figura 3.4, como lo son las clases Rol, Perfil del Servicio y Dominio de la Actividad. Estas clases se relacionan con las demás clases previamente definidas. El Perfil del Servicio describe el Servicio en más detalle, y le asocia un Rol a ese perfil de servicio. El rol también se asocia a un dominio de actividades, que también se describe en más detalle.

Así, las preguntas formuladas por *las 5W* son contestadas mediante estas clases. El ¿quién? (*who?*) por la clase usuario; el ¿cuándo? (*when?*) por la clase tiempo; el ¿dónde? (*where?*) por la clase Ubicación; el ¿qué? (*what?*) por la clase Servicios, y el ¿por qué? (*why?*) por la todas las clases de **CameOn** actuando juntas.

Las relaciones entre las clases, se muestran en la Figura 3.4. Dichas relaciones permiten describir el contexto, por ejemplo, la relación entre usuario y su rol en un dominio hace posible encontrar el por qué de una actividad, la cual es completada con las relaciones entre el usuario, la actividad y el dispositivo. Por otro lado, un ambiente tiene una ubicación, y un dispositivo provee un servicio de acuerdo a sus capacidades. También, los usuarios usan los dispositivos en una ubicación, ellos tienen roles en sus actividades, hechas en un ambiente y un tiempo especifico.

Podemos ver así, que **CameOn** es un ontología general, que no es particular a un dominio especifico de aplicaciones, por lo que es capaz de modelar contextos en distintos dominios. **CameOn** describe los componentes principales considerados en

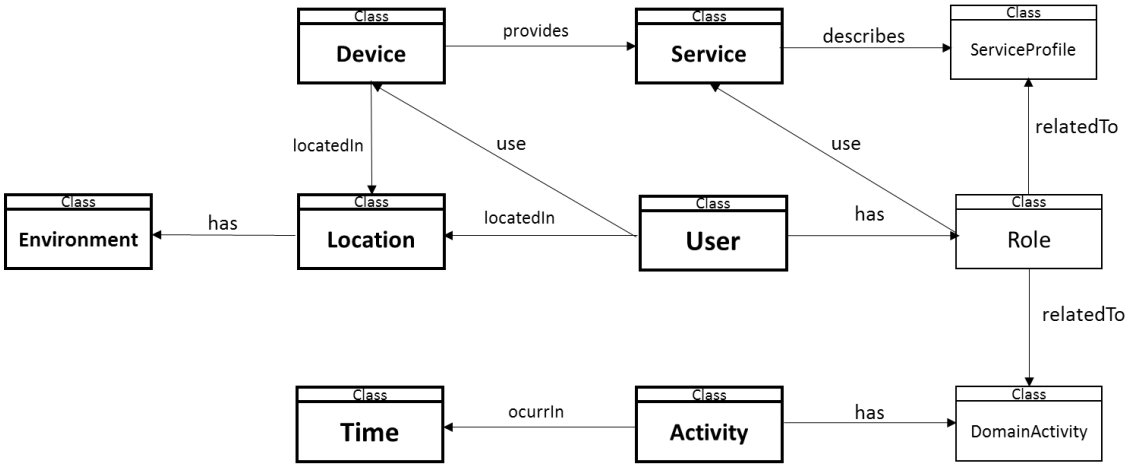

Figura 3.4: Relaciones entre las clases de **CameOn** (Fuente Aguilar et al. (2017))

las aplicaciones conscientes del contexto, y puede ser extendida, mediante su segundo nivel, con la información específica del dominio de la aplicación. Mediante el uso de *las 5W*, crea modelos para cada dimensión, y así, describe los detalles más importantes del contexto.

Capítulo 4

Diseño del Sistema

4.1 Modelo Arquitectónico del Sistema

4.1.1 Arquitectura del Sistema

El sistema diseñado debe poder realizar varias acciones, entre las cuales se encuentran:

- Obtener el contexto (la ontología).
- Realizar inferencias sobre los datos que posee la ontología, usando lógica de primer orden.
- Consultar la información enriquecida semánticamente.
- Manejar ambigüedades en los datos que posee la ontología, usando lógica dialéctica.

Todo lo anterior, lo debe ofrecer como servicios del middleware CARMiCLOC. En ese sentido, la arquitectura del sistema puede verse en la Figura 4.1. Es una arquitectura con tres componentes principales, el *Enriquecedor Semántico*, el *Razonador Semántico*, y el *Generador de Salida*. A su vez, estos componentes hacen uso de otros componentes para poder cumplir sus funciones.

El componente que enlaza los datos es el *Enriquecedor Semántico*, y esta conectado a los servicios que obtienen el contexto en CARMiCLOC (servicios de

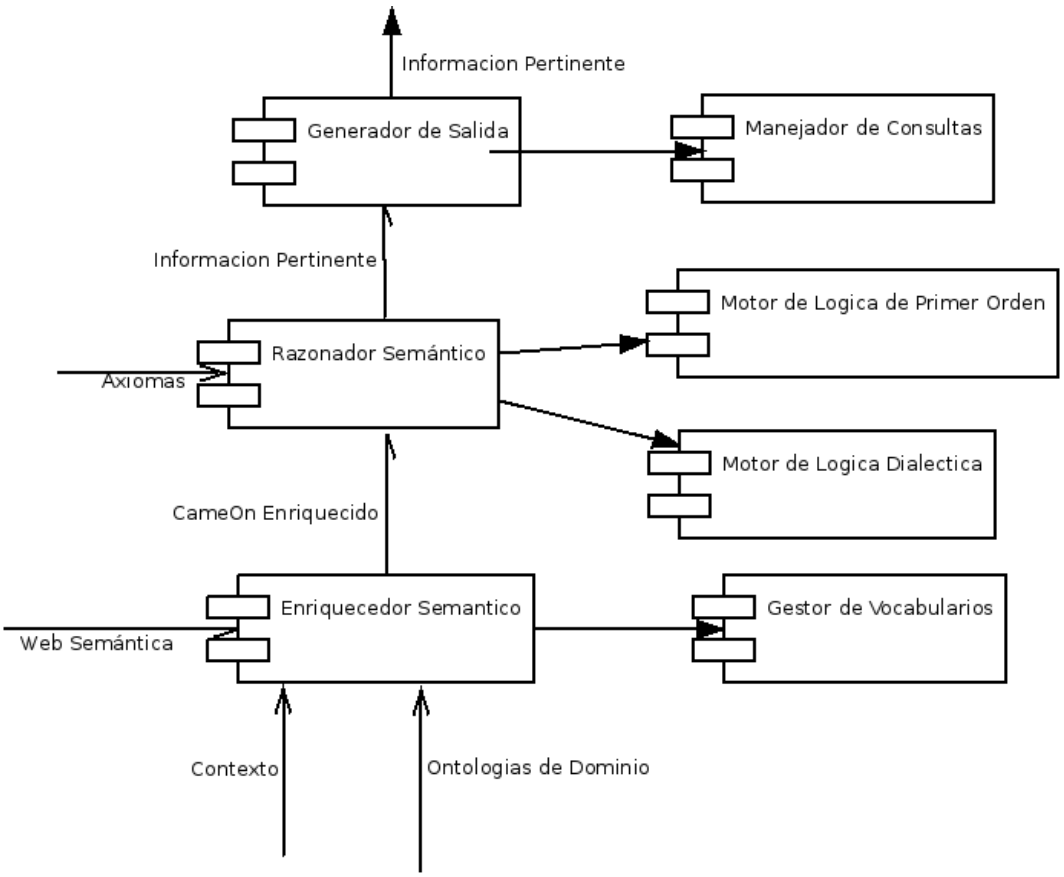

Figura 4.1: Arquitectura propuesta para el Sistema

Descubrimiento, Pre-Configuración y Modelado del Contexto). Algunas de las fuentes de datos que usan los servicios anteriores pueden ser sensores, el dispositivo del usuario, Internet, y cualquier otro medio con datos del contexto. También, otro insumo clave son las ontologías de dominio, y del contexto, la Web Semántica. Con toda esa información, el *Enriquecedor Semántico* enriquece las ontologías mediante el enlazado de datos.

Además, para poder usar esa información extendida haciendo uso de la Web Semántica, el *Enriquecedor Semántico* necesita un componente para poder agregar los vocabularios conocidos de la web a la ontología CameOn, y así poder explotar los datos enlazados. En la Figura 4.1 es el *Gestor de Vocabularios*.

La salida del *Enriquecedor Semántico* es enviada al segundo componente, el *Razonador Semántico*, que es el componente de razonamiento. El *Razonador Semántico* recibe la ontología CameOn enriquecida mediante los datos enlazados, y sobre ella,

usando axiomas, realiza inferencia sobre los datos recopilados. El *Razonador Semántico* necesita de dos componentes más para poder realizar las inferencias necesarias. El primero es un componente para las inferencias mediante el uso de *Lógica de Primer Orden*. Ahora bien, como existen ambigüedades en el contexto, el *Razonador Semántico* hace uso de un segundo componente que maneja la *Lógica Dialéctica*, el cual resuelve dichas ambigüedades.

La salida del *Razonador Semántico* es luego enviada al tercer componente, el *Generador de Salida*, quien mediante el uso del *Manejador de Consultas*, entrega la información pertinente a quien invoco el servicio. Este componente toma los datos que el motor de inferencia considero pertinentes, y les da formato para entregarlos a las aplicaciones que los hayan invocado.

4.1.2 Axiomas del Sistema

El segundo componente, el *Razonador Semántico*, esta compuesto por un grupo de axiomas. Estos axiomas son los que usa el componente para analizar los datos capturados por el sistema. El conjunto de axiomas propuesto es genérico, basado en las clases que componen a CameOn. En la Tabla 4.1 se pueden ver los axiomas, con un ejemplo correspondiente y el tipo de lógica al que corresponden.

Tabla 4.1: Axiomas del *Razonador Semántico*

N	**Axioma**	**Ejemplo**	**Tipo de Lógica**
1	Si un rol se desarrolla en una ubicación, entonces las actividades con dominio de actividad relacionado al rol se pueden hacer en la ubicación.	Si se es estudiante y se esta en la biblioteca, entonces se puede leer, investigar, exponer, estudiar, en la biblioteca.	Lógica de Primer Orden.

2	Si un usuario usa un dispositivo en una ubicación, y el dispositivo provee un servicio, entonces la ubicación tiene el dispositivo.	Si Carlos investiga usando un computador en una biblioteca, entonces la biblioteca tiene un computador.	Lógica de Primer Orden.
3	Si un usuario esta en una ubicación, y usa un dispositivo que provee un servicio, entonces el servicio es proveído por la ubicación.	Si Maria navega en Internet en una biblioteca, entonces la biblioteca provee Internet.	Lógica de Primer Orden.
4	Si localización A tiene actividad en tiempo B y usuario se encuentra en localización C en tiempo cercano a B, y A es cerca de B, el usuario puede ir a la actividad.	**Ejemplo 1:** Si hay un concierto en un auditorio y es a las 5:00pm y Juan está en una plaza a las 4:45pm, entonces Juan puede ir al concierto en el auditorio. **Ejemplo 2:** Si Carlos está en su casa a las 3:00pm y hay una obra de teatro a las 4:15pm en un teatro, entonces Carlos puede ir a la obra de Teatro.	Lógica Dialéctica. Se usa porque hay un caso de ambigüedad: Vaguedad que existe debido al lenguaje natural en los términos de distancia y tiempo.
5	Si un usuario posee un rol y esta en una ubicación, entonces el rol se desarrolla en la ubicación.	Si Andrea es estudiante en la Facultad de Ingeniería, entonces en la Facultad de Ingeniería se puede ser estudiante.	Lógica de Primer Orden.

6	Si un usuario usa un dispositivo que provee un servicio, entonces el usuario puede consumir el servicio.	Si Pedro usa un computador que provee la información de los libros en una biblioteca, entonces Pedro puede usar la información de los libros de la biblioteca.	Lógica de Primer Orden.
7	Si un usuario realiza una actividad en una ubicación en tiempo A, entonces la ubicación está abierta en tiempo A.	**Ejemplo 1:** Si Oscar está en un concierto en un auditorio, entonces el auditorio está abierto. **Ejemplo 2:** Si Gerardo está cenando en el comedor, entonces el comedor está abierto.	Lógica Dialéctica. Se usa porque hay dos casos de ambigüedad: Vaguedad que existe debido al lenguaje natural. Declaraciones contingentes sobre el futuro debido a la situación en el sitio (abierto o cerrado).
8	Si un usuario hace una actividad en un tiempo A, entonces el usuario no puede hacer más actividades en el tiempo A.	Si Valentina tiene un examen a las 10:00AM, entonces no puede tener más exámenes a esa hora.	Lógica de Primer Orden.

Como se puede ver en la Tabla 4.1, hay axiomas tanto en Lógica de Primer Orden, como en Lógica Dialéctica.

El axioma numero 4 involucra dos cuantificadores que generan ambigüedad dentro del axioma. El primero involucra al tiempo, depende del usuario; mientras

a un usuario le puede parecer que una hora es un largo tiempo de espera, a otro usuario le puede parecer que es corto. También puede depender de la situación en la que se encuentra el usuario, ya que, en un aeropuerto una escala de una hora es corta, mientras que en una cola de banco una espera de una hora es larga. El segundo cuantificador involucrado es la cercanía en distancia, de la misma manera que en el caso anterior, este cuantificador depende del usuario, y su situación, mientras a unos usuarios desplazarse unas cantidad X de metros puede parecerles mucho, a otros puede parecerles poco.

El siguiente axioma que involucra Lógica Dialéctica es el axioma numero 7, el cual cae en dos casos de la Lógica Dialéctica, el primero de vaguedad del lenguaje natural. En el ejemplo 1 del axioma, se expone que Oscar esta en un auditorio, por lo que este esta abierto, sin embargo, no necesariamente es el caso, un posible caso seria que haya un evento especial y que no hayan entradas libres. El segundo caso son las declaraciones contingentes sobre el futuro, al ver el ejemplo 2, un posible caso ~~para este~~ es que justo en ese momento esta abierto, sin embargo, están cerrando, por lo que en cuestión de minutos ya no lo estará.

Los demás axiomas involucran Lógica de Primer Orden, sin ambigüedades, por lo que son claros y concretos.

4.2 Modelo Tecnológico del Sistema

Una vez definida la arquitectura del sistema, es necesario pasar al diseño tecnológico del mismo. Para el desarrollo del sistema se consideraron distintas tecnologías las cuales pueden ser vistas en la Figura 4.2. A continuación, describimos las tecnologías que fueron usadas para desarrollar los diferentes componentes del sistema.

4.2.1 Implementación de CameOn

En primera instancia, fue necesario tener la ontología CameOn, por lo que fue usado el programa *Protégé* en su version 5.2.0 (Musen (2015)). *Protégé* es una plataforma gratis y de código abierto, con un gran conjunto de herramientas para construir ontologías, también sirve como framework para editar dichas ontologías.

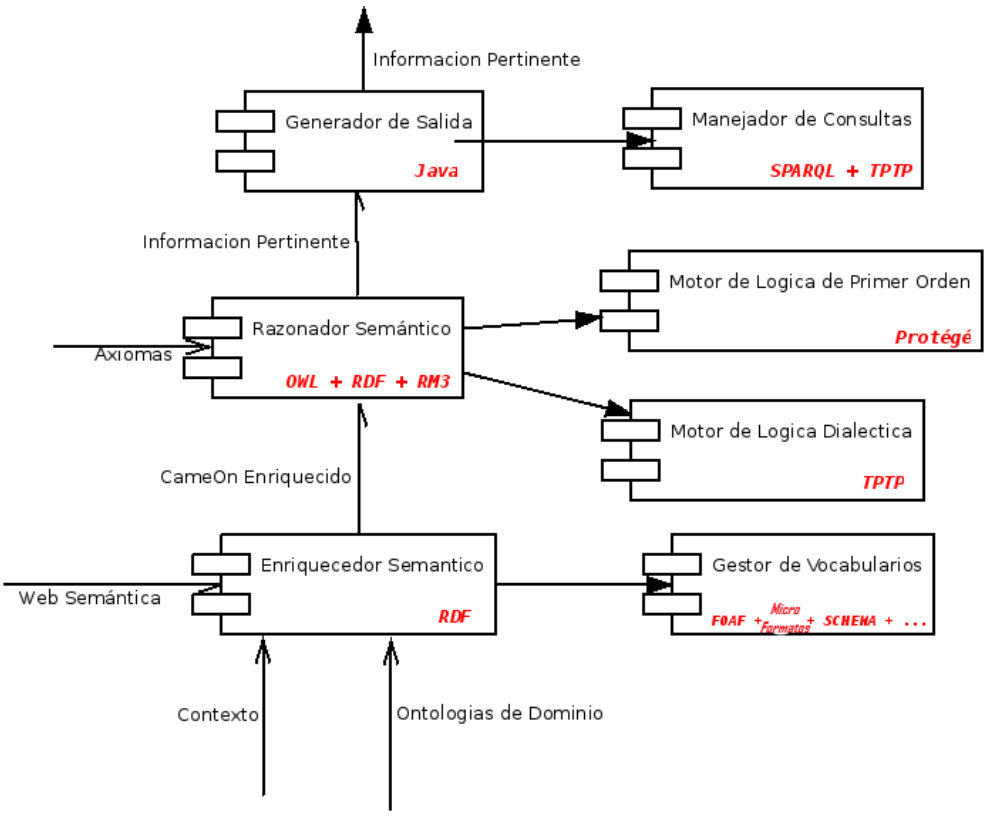

Figura 4.2: Arquitectura propuesta del Sistema con las tecnologías usadas

Usando *Protégé* se procedio a crear las clases que usa CameOn, las cuales pueden apreciarse ya creadas en la Figura 4.3. Además de las clases de CameOn, se agregaron clases de vocabularios conocidos, como *Friend of a Friend* (conocida como **FOAF** generalmente, Brickley y Miller (2000)) y *Schema*(Google et al. (2011)).

Luego de definir las clases, se procedió a definir las propiedades de las clases. En *Protégé*, las propiedades de los objetos se dividen en dos tipos. El primero, las propiedades de datos, que definen propiedades que tienen un literal como objeto de la sentencia; y las propiedades de objetos, las cuales definen relaciones entre las clases. En las Figuras 4.4 y 4.5 se pueden ver las propiedades de datos y de objetos, respectivamente, que fueron implementadas en CameOn.

Para las propiedades de datos (véase Figura 4.4), se incluyeron algunas propiedades definidas por la ontología *FOAF* con el fin de demostrar su uso, sin embargo, son solo algunas y podrían agregarse muchas más en el futuro.

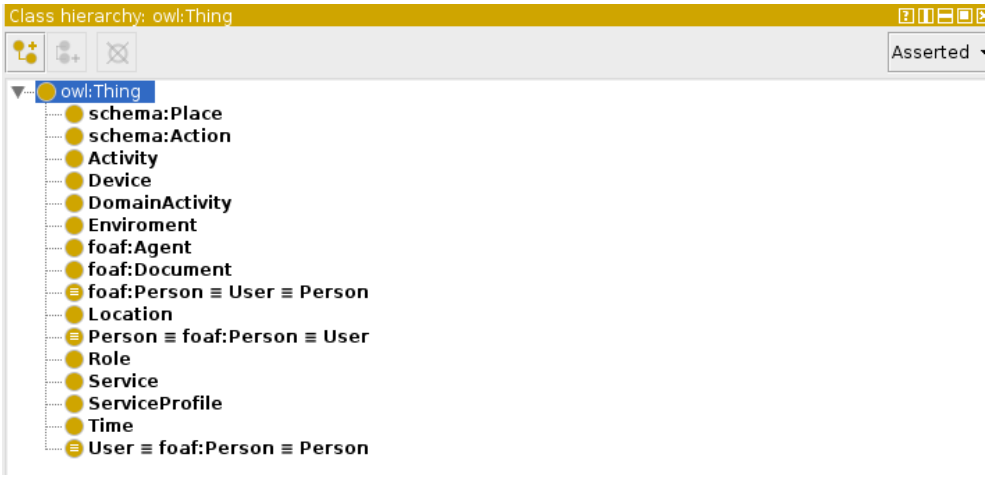

Figura 4.3: Clases de CameOn en *Protégé*

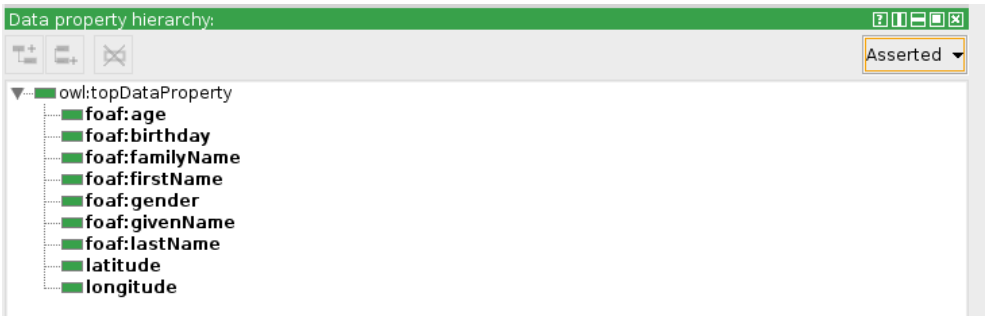

Figura 4.4: Propiedades de datos de CameOn en *Protégé*

Por otro lado, las propiedades de objetos (véase Figura 4.5) que se definieron incluyen algunas definidas en el trabajo Aguilar et al. (2017), y otras agregadas para poder suplir las necesidades del sistema que se esta implementando.

Estos son los elementos de base del componente *Enriquecedor Semántico*, que obtiene la ontología y la extiende mediante el uso de la Web Semántica.

Posteriormente, usando *Protégé* de nuevo, se definieron los axiomas basados en Lógica de Primer Orden (véase la Tabla 4.1), usando la herramienta de *SWRL* que provee dicho sistema. El *Semantic Web Rule Language* (*SWRL*) es un lenguaje basado en *OWL* y *Rule ML*, que permite definir reglas en la forma: un conjunto de 0 o más antecedentes, llamados el cuerpo de la regla, y un conjunto de 0 o más consecuentes, llamados la cabeza de la regla.

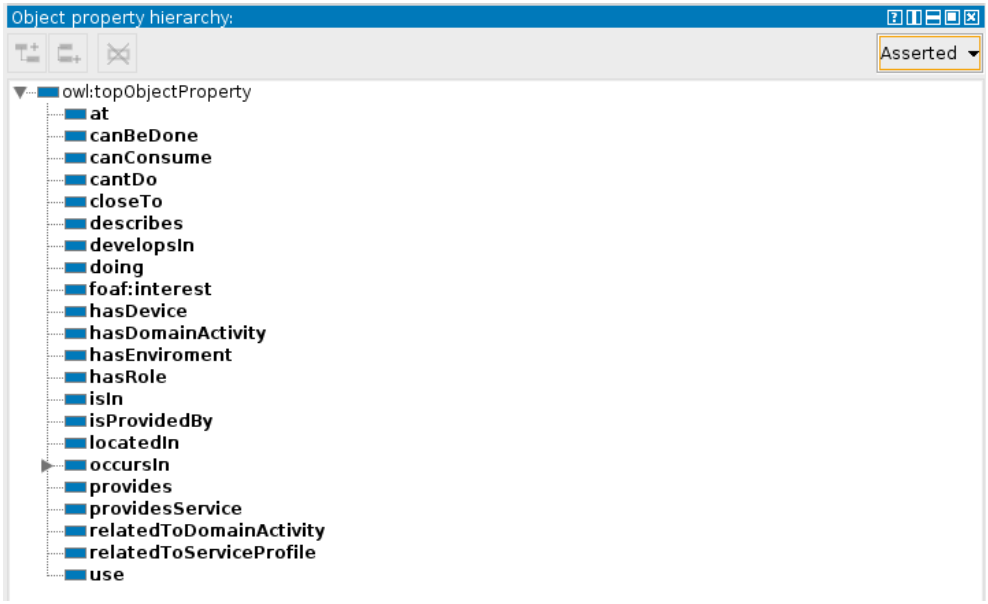

Figura 4.5: Propiedades de Objetos de CameOn en *Protégé*

Los antecedentes y consecuentes son conjuntos de átomos unidos por conjunciones, donde los átomos pueden indicar: una variable de una clase; una relación entre variables; equivalencia de variables; diferencia de variables.

Las reglas definidas en la Tabla 4.1, o axiomas, pueden ser apreciados en la Figura 4.6.

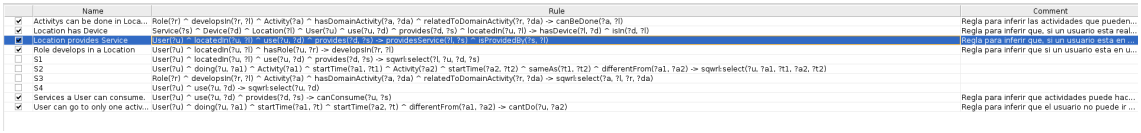

Figura 4.6: Axiomas definidos en *Protégé* usando *SWRL*, explicados más adelante

Los axiomas basados en Lógica de Primer Orden en *SWRL*, se muestran a continuación:

1. Si un rol se desarrolla en una ubicación, entonces las actividades con dominio de actividad relacionado al rol se pueden hacer en la ubicación.

   $Role(?r) \wedge developsIn(?r, ?l) \wedge Activity(?a) \wedge hasDomainActivity(?a, ?da) \wedge relatedToDomainActivity(?r, ?da) \rightarrow canBeDone(?a, ?l)$

2. Si un usuario usa un dispositivo en una ubicación, y el dispositivo provee un servicio, entonces la ubicación tiene el dispositivo.

   $Service(?s) \land Device(?d) \land Location(?l) \land User(?u) \land use(?u, ?d) \land locatedIn(?u, ?l) \rightarrow hasDevice(?l, ?d) \land isIn(?d, ?l)$

3. Si un usuario esta en una ubicación, y usa un dispositivo que provee un servicio, entonces el servicio es proveído por la ubicación.

   $User(?u) \land locatedIn(?u, ?l) \land use(?u, ?d) \land provides(?d, ?s) \rightarrow providesService(?l, ?s) \land isProvidedBy(?s, ?l)$

4. Si un usuario posee un rol y esta en una ubicación, entonces el rol se desarrolla en la ubicación.

   $User(?u) \land locatedIn(?u, ?l) \land hasRole(?u, ?r) \rightarrow developsIn(?r, ?l)$

5. Si un usuario usa un dispositivo que provee un servicio, entonces el usuario puede consumir el servicio.

   $User(?u) \land use(?u, ?d) \land provides(?d, ?s) \rightarrow canConsume(?u, ?s)$

6. Si un usuario hace una actividad en un tiempo A, entonces el usuario no puede hacer más actividades en el tiempo A.

   $User(?u) \land doing(?u, ?a1) \land startTime(?a1, ?t) \land startTime(?a2, ?t) \land differentFrom(?a1, ?a2) \rightarrow cantDo(?u, ?a2)$

Luego de tener estas reglas definidas, es necesario enviar las reglas junto con los datos almacenados en la ontología (hechos, instancias o individuos) al motor de reglas, el cual toma estos datos, infiere sobre ellos, y genera los nuevos predicados a partir de las reglas aplicadas sobre las instancias de la ontología. Luego, los nuevos predicados son añadidos a la ontología. Para la ejecución de todo este proceso nos apoyamos en el uso de *Protégé*, el cual provee una pestaña que incorpora estas funcionalidades a su interfaz de usuario.

4.2.2 Implementación de la Lógica dialéctica

Para implementar la Lógica Dialéctica se uso el Motor de Inferencia *TPTP*, descrito en el trabajo de Sutcliffe (2017). Los axiomas generados que implican Lógica Dialéctica fueron escritos en la sintaxis que acepta el sistema *TPTP*, para poder hacer uso de él.

El archivo generado como entrada a *TPTP*, correspondiente al axioma 4 (véase Tabla 4.1), se puede observar en la Figura 4.7.

```
fof(userCanGoToActivityNearHisLocation, axiom,(
    ?[User, Location2, Time1, Activity, Location1, Time2]:
    (( user(User)
    & location(Location2)
    & time(Time1)
    & activity(Activity)
    & location(Location1)
    & time(Time2)
    & locatedIn(User, Location2)
    & starts(Activity, Time1)
    & isIn(Activity, Location1)
    & soon(Time2, Time1)
    & close(Location2, Location1)
    & isTime(User, Time2) )
    => goes(User, Activity) ) )).
```

Figura 4.7: Archivo con el axioma 4 (véase la Tabla 4.1)

Lo mismo se hizo con el axioma 7 (véase Tabla 4.1), que es el otro axioma basado en Lógica Dialéctica. El axioma puede apreciarse en la figura 4.8.

```
fof(locationIsOpen, axiom,(
    ? [User, Activity, Location, Time]:
    ((user(User)
    & activity(Activity)
    & location(Location)
    & time(Time)
    & locatedIn(User, Location)
    & doing(User, Activity)
    & isTime(User, Time) )
    => open(Location, true) ) )).
```

Figura 4.8: Archivo con el axioma 7(véase la Tabla 4.1)

Luego de tener los axiomas que hacen uso de la Lógica Dialéctica escritos en el formato aceptado por el sistema *TPTP*, se procedió a hacer uso del sistema, el cual

razona según los individuos que se instancian en el archivo que describe el axioma. El resultado de este proceso es mostrado en el próximo capítulo.

4.2.3 Interfaz del Sistema

Para poder mostrar las funcionalidades del sistema, se decidió desarrollar una interfaz usando el lenguaje de programación *Java*, debido a su portabilidad y facilidad de ejecución. También influyó en la decisión, el hecho de, que existe una interfaz de programación de aplicaciones (API por sus siglas en inglés, *Application Programming Interface*) desarrollada en este lenguaje, llamada *Apache Jena* (The Apache Software Foundation (2011)). Esta permite la programación de aplicaciones que usen ontologías y la Web Semántica, proveyendo métodos y clases que facilitan su creación.

Se creo una clase `Prototype` para la implementación de la interfaz, la cual puede apreciarse en la Figura 4.9. La clase `Prototype` contiene un conjunto de métodos para mostrar las funcionalidades del sistema. La clase tiene dos constructores, uno por defecto y otro parametrizado, para poder indicar la ubicación del archivo donde se encuentra el contexto (es decir, la ubicación de la ontología). También posee unos métodos privados que facilitan las tareas que la clase realiza y evitan la repetición de código. Estos métodos son: `readContext(String):void`, `replacePrefix(String):String`, `executeQuery(String):Model` y `getDialecticLogicAnswer(BufferedReader, BufferedReader): String`. El primero, lee el archivo en la dirección indicada por el parámetro, y asigna el contenido de dicha ontología al modelo que es atributo de la clase `Prototype`. El segundo, por su parte, facilita la salida de los modelos remplazando los espacios de nombre que existen en los URIs por su prefijo, este método es de gran utilidad en nuestro trabajo. El tercero ejecuta una consulta descrita por su parámetro, sobre el contexto, y retorna el respectivo modelo. Por último, `getDialecticLogicAnswer` invoca al sistema *TPTP*.

Además, la clase `Prototype` posee una gran cantidad de métodos públicos, los cuales en nuestro caso son usados para especificar las consultas detrás de nuestros axiomas, los cuales son provistos como opciones por nuestra interfaz (ver Figura 4.9):

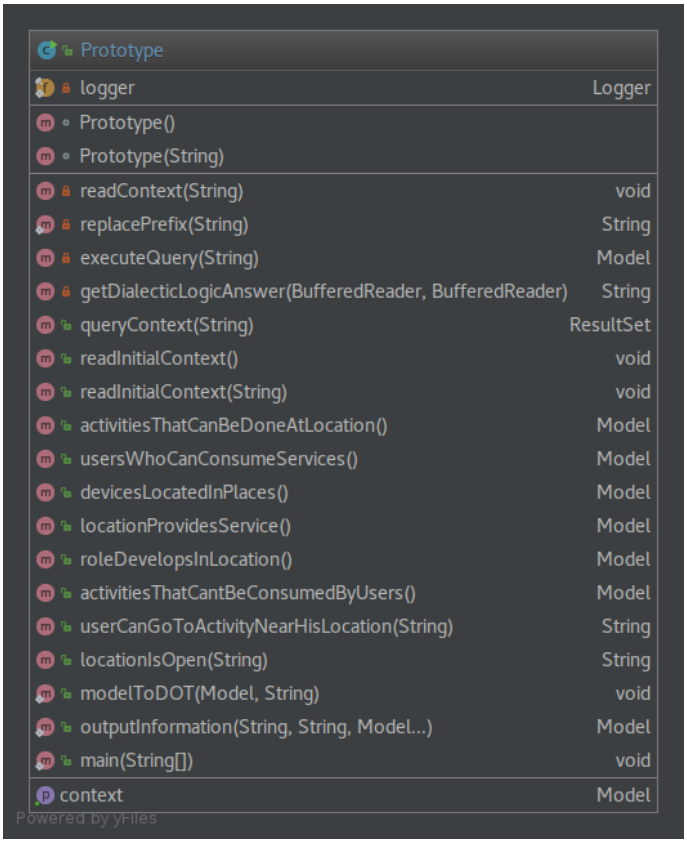

Figura 4.9: Diagrama de la clase `Prototype`.

- `queryContext(String):ResultSet`: Este método permite consultar un predicado en el contexto.
- `readInitialContext():void`: Permite leer el contexto a partir de una dirección por defecto. Hace uso del método `readContext(String)`.
- `readInitialContext(String):void`: Al igual que el método anterior, permite leer el contexto de una dirección, pero esta vez indicada por el parámetro.
- `activitiesThatCanBeDoneAtLocation():Model`: Usa la consulta escrita en SPARQL de la Figura 4.10, para consultar el contexto. Esta consulta invoca al axioma 1 (véase la Tabla 4.1). Hace uso del método `executeQuery(String)`.
- `usersWhoCanConsumeServices():Model`: Consulta el contexto mediante la consulta de SPARQL mostrada en la Figura 4.11. Esta consulta invoca al axioma

```
PREFIX : <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#>
DESCRIBE ?activity ?location
WHERE {
        ?activity :canBeDone ?location
}
```

Figura 4.10: Consulta para el axioma número 1

6 (véase la Tabla 4.1). Hace uso del método `executeQuery(String)`.

```
PREFIX : <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#>
DESCRIBE ?person ?service
WHERE {
        ?person :canConsume ?service
}
```

Figura 4.11: Consulta para el axioma número 6

- `devicesLocatedInPlaces():Model`: Consulta el contexto mediante la consulta de SPARQL de la Figura 4.12. Esta consulta invoca al axioma 2 (véase la Tabla 4.1). Hace uso del método `executeQuery(String)`.

```
PREFIX : <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#>
DESCRIBE ?location ?device
WHERE {
        ?location :hasDevice ?device
}
```

Figura 4.12: Consulta para el axioma número 2

- `locationProvidesService():Model`: Consulta el contexto mediante la consulta de la Figura 4.13. La consulta de la Figura 4.13 invoca al axioma 3 (véase la Tabla 4.1). Hace uso del método `executeQuery(String)`.

- `roleDevelopsInLocation():Model`: Consulta el contexto mediante la consulta de la Figura 4.14. La consulta de la Figura 4.14 da respuesta al axioma 5 (véase la Tabla 4.1). Hace uso del método `executeQuery(String)`.

```
PREFIX : <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#>
DESCRIBE ?location ?service
WHERE {
        ?location :providesService ?service
}
```

Figura 4.13: Consulta para el axioma número 3

```
PREFIX : <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#>
DESCRIBE ?role ?location
WHERE {
        ?role :developsIn ?location
}
```

Figura 4.14: Consulta para el axioma número 5

- `activitiesThatCantBeConsumedByUsers():Model`: Consulta el contexto mediante la consulta de la Figura 4.15. La consulta de la Figura 4.15 invoca al axioma 8 (véase la Tabla 4.1). Hace uso del método `executeQuery(String)`.

```
PREFIX : <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#>
DESCRIBE ?user ?activity
WHERE {
        ?user :cantDo ?activity
}
```

Figura 4.15: Consulta para el axioma número 8

- `userCanGoToActivityNearHisLocation(String): String`: Haciendo uso del sistema *TPTP*, realiza una inferencia sobre el archivo indicado como parámetro, y retorna la respuesta del sistema *TPTP*. La consulta de la Figura 4.16 invoca al axioma 4 (ver la Tabla 4.1 y la Figura 4.7). El código del método puede ser visto en la Figura 4.16.

- `locationIsOpen(String): String`: Hace uso del sistema *TPTP* para realizar inferencia sobre el archivo indicado como parámetro (en nuestro caso, el archivo de la Figura 4.8). Retorna la respuesta del sistema *TPTP*. Esta consulta invoca al axioma 7 (véase Tabla 4.1).

```
public String userCanGoToActivityNearHisLocation(String filename) {
        try {
            String s;
            Process rm3 = Runtime.getRuntime().exec("./resources/ld/jgxyz "
                    + filename + " 1000 rm3 jeffs_way xyz_true vampire 80 vampirefmo 20");
            BufferedReader stdInput = new BufferedReader(
                    new InputStreamReader(rm3.getInputStream()));
            BufferedReader stdError = new BufferedReader(
                    new InputStreamReader(rm3.getErrorStream()));
            return getDialecticLogicAnswer(stdInput, stdError);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return "";
    }
```

Figura 4.16: Código del método `userCanGoToActivityNearHisLocation`

- `modelToDOT(Model, String):void`: Este método guarda un archivo DOT con el modelo de datos suministrado como parámetro que describe el grafo del modelo. El código se muestra en la Figura 4.17, y el resultado de aplicarlo en la Figura 4.18.

- `outputInformation(String, String, Model...):Model`: Este método toma los modelos de datos que sean enviados como parámetros y los une en uno solo, que luego guarda en el formato indicado, y en el archivo indicado, facilitando así el compartir la información explorada por el sistema. Podemos apreciar el método en la Figura 4.19, y el resultado de aplicarlo en la Figura 4.18 se puede apreciar en la Figura 4.20.

```
public static void modelToDOT(Model m, String filename) {
        try(OutputStream out = new FileOutputStream(filename + ".dot")) {
            out.write("graph SimpleGraph{\n".getBytes());
            out.write("overlap = false;\n".getBytes());
            out.write("splines = true;\n".getBytes());
            StmtIterator iter = m.listStatements();
            while(iter.hasNext()){
                Statement stmt = iter.nextStatement();
                Resource subject = stmt.getSubject();
                Property predicate = stmt.getPredicate();
                RDFNode object = stmt.getObject();
                String statementString = "\"" + replacePrefix(subject.toString())
                        + "\" -- \""
                        + replacePrefix(object.toString())
                        + "\" [label=\""
                        + replacePrefix(predicate.toString())
                        + "\"]\n";
                out.write(statementString.getBytes());
            }
            out.write("}".getBytes());
            Runtime.getRuntime().exec("neato -Tpng -O" + filename
                    + ".png " + filename + ".dot");
        }
        catch (IOException e){
            e.printStackTrace();
        }
    }
```

Figura 4.17: Metodo `modelToDOT(Model, String):void`

Figura 4.18: Ejemplo de imagen generada por el método `modelToDOT(Model, String):void`

```
public static Model outputInformation(String filename, String lang, Model... models) {
        Model union = ModelFactory.createDefaultModel();
        for(Model m : models) {
            union.add(m);
        }
        try(OutputStream out = new FileOutputStream(filename)) {
            union.write(out, lang);
        }
        catch(IOException e) {
            e.printStackTrace();
        }
        return union;
    }
```

Figura 4.19: Método `outputInformation(String, String, Model...):Model`

```
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://xmlns.com/foaf/0.1/firstName> "Maria"^^<http://www.w3.
      org/2000/01/rdf-schema#Literal> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#
      cantDo> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#actividad7> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3
      .org/2000/10/swap/pim/contact#Person> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#
      doing> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#actividad6> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.
      semanticweb.org/carlos/ontologies/2018/4/cameon.owl#User> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://xmlns.com/foaf/0.1/lastName> "Rojas"^^<http://www.w3.org
      /2000/01/rdf-schema#Literal> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.
      com/foaf/0.1/Person> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3
      .org/2002/07/owl#NamedIndividual> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.
      com/foaf/0.1/Agent> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#usuario2> <http://xmlns.com/foaf/0.1/gender> "female"^^<http://www.w3.org
      /2000/01/rdf-schema#Literal> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#actividad7> <http://www.w3.org/2000/01/rdf-schema#label> "Concierto de
      Orquesta Sinfonica de Merida"@es .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#actividad7> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.
      owl#startTime> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#startTime1> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#actividad7> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.
      owl#occursIn> <http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#startTime1> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#actividad7> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.
      semanticweb.org/carlos/ontologies/2018/4/cameon.owl#Activity> .
<http://www.semanticweb.org/carlos/ontologies/2018/4/cameon.owl#actividad7> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.
      w3.org/2002/07/owl#NamedIndividual> .
```

Figura 4.20: Texto generado por el metodo `outputInformation(String, String, Model...):Model` del grafo de la Figura 4.18

Capítulo 5

Experimentación

5.1 Definición de la Métrica de Calidad

Como métrica de la calidad de los modelos que se obtienen, usaremos la cantidad de nodos y la cantidad de aristas que se obtienen al usar sólo Lógica de Primer Orden, para compararlo contra cuando se usa Lógica de Primer Orden con datos enlazados.

La cantidad de nodos y cantidad de aristas representan conocimiento, por lo que mientras mayor es la cantidad de aristas y nodos agregados, podría decirse que se agrega más conocimiento, siempre y cuando la información en los nodos y las aristas sea pertinente y no redundante.

5.2 Descripción de los Escenarios de Prueba

5.2.1 Descripción del Escenario de Prueba usando Lógica de Primer Orden

Para probar el sistema desarrollado, necesitamos definir un escenario que implique diferentes contextos, y tal que se requieran utilizar las diferentes funcionalidades del sistema. Para ello, supongamos la siguiente situación, dividida en 2 partes:

1. Primera Parte:

   Un sujeto (`usuario1`) invita a sus amigos (`usuario2` y `usuario3`) a un

Restaurante, para ver el partido de ese día (`partido`) y comer (su rol `rol` es ir a comer). Los usuarios invitados revisan si todos pueden ir, usando la funcionalidad que provee el sistema. Los usuarios que les es posible asistir no saben donde ir, por lo que el sistema puede mostrar Restaurantes donde se puede ir a comer. Pero además, como ellos quieren ver el juego, usan la funcionalidad del sistema para comprobar en cuales de esas ubicaciones pueden ver el partido. Una vez en el sitio, los asistentes quieren ver si la ubicación en la que están provee servicio para hacer apuestas, usando la funcionalidad del sistema.

2. Segunda Parte:
   Luego del partido, los usuarios quieren jugar Fútbol, por lo que invitan a otros amigos (`usuario10`, `usuario11`, `usuario12` y `usuario13`), y usando el sistema se aseguran que los invitados tienen la indumentaria (es decir, tengan la ropa adecuada para jugar). Los usuarios invitados revisan si pueden ir, usando el sistema. También, usando el sistema revisan que lugares proveen un balón de fútbol para jugar, y con otra consulta al sistema, revisan los lugares que proveen el servicio de alquiler de canchas.

Con esta situación, se pueden instanciar todos los axiomas que implementa el sistema en Lógica de Primer Orden, y en 2 contextos distintos, para probar su generalidad y su utilidad.

5.2.2 Descripción del Escenario de Prueba usando Lógica Dialéctica

Para probar los axiomas que hacen uso de la Lógica Dialéctica, se decidió instanciar algunos recursos y realizar inferencia sobre ellos usando los dos axiomas en Lógica Dialéctica y el sistema *TPTP*.

Para el axioma número 4 (véase Tabla 4.1), el caso que usaremos sera el siguiente ejemplo: hay un concierto en un tiempo, un usuario esta en una ubicación cercana, y falta poco para la hora del concierto.

Por otro lado, para el axioma 7 (véase Tabla 4.1), el caso de estudio sera el siguiente: un usuario esta en un Restaurante en una cena de la compañía a una cierta

hora.

5.3 Ejecución de los Escenarios de Prueba

5.3.1 Primera Parte del Escenario de Prueba usando Lógica de Primer Orden

Para corroborar el sistema con el caso de prueba definido en la sección 5.2.1, primero se procedió a instanciar a CameOn en el escenario de estudio. En la Figura 5.1 se pueden apreciar las instancias creadas.

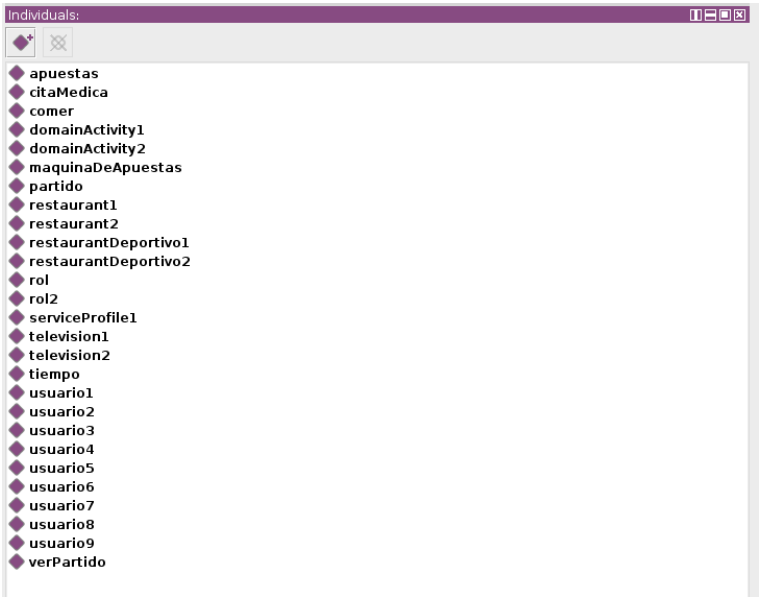

Figura 5.1: Instancias creadas para realizar la primera parte de la prueba

En la Figura 5.1 se muestran las distintas instancias creadas para la primera parte de la prueba. Unos usuarios(`usuario1`, ... `usuario 9`), varios dispositivos de distintos tipos (`televisor1`, `televisor2`, `maquinaDeApuestas`), varias ubicaciones (`restaurant1`, `restaurant2`, `restaurantDeportivo1` y `restaurantDeportivo2`), algunos roles (`rol` que se refiere a ir a comer, y `rol2` que es un rol de ver deportes), una instancia de tiempo y algunas actividades (`citaMedica`, `comer`, `partido`).

Para probar el sistema, se programó una clase que utilizara el sistema para

realizar las consultas pertinentes al sistema, cuyo código puede ser visto en la Figura 5.2. Un código similar a este es el que debería usar cualquier aplicación que desee usar nuestro sistema.

```
public class Case1 {
    public static void main(String[] args){
        org.apache.log4j.BasicConfigurator.configure(new NullAppender());
        System.out.println("Primera parte del caso de estudio");
        Prototype caso1 = new Prototype("../ontology/pruebas/restaurant.owl");
        Model whereToEat = caso1.roleDevelopsInLocation();
        Prototype.modelToDOT(whereToEat, "./resources/pruebas/caso1/whereToEat");
        Model whereIsTheGame = caso1.activitiesThatCanBeDoneAtLocation();
        Prototype.modelToDOT(whereIsTheGame, "./resources/pruebas/caso1/whereIsTheGame");
        Model whoCantGo = caso1.activitiesThatCantBeConsumedByUsers();
        Prototype.modelToDOT(whoCantGo, "./resources/pruebas/caso1/whoCantGo");
        Model devicesInLocations = caso1.devicesLocatedInPlaces();
        Prototype.modelToDOT(devicesInLocations, "./resources/pruebas/caso1/devicesInLocations");
        Model information = Prototype.outputInformation("./resources/pruebas/caso1/information.owl",
                "N-TRIPLES", whereToEat, whereIsTheGame, whoCantGo, devicesInLocations);
    }
}
```

Figura 5.2: Clase programada para la primera parte de la prueba

La primera consulta que se necesita realizar a través del sistema es para asegurarse de quienes pueden ir. Usando la clase `Prototype`, los usuarios pueden usar el método `activitiesThatCantBeConsumedByUsers`, el cual al ser usado genera la imagen de la Figura 5.3, a partir de la información de CameOn. La Figura 5.3 muestra en el área amarilla que para este caso, el usuario `usuario3` tiene otra actividad (`citaMedica`), por lo que no puede ir a ver el partido con sus amigos.

Luego, les interesa saber los lugares en donde se puede ir a comer. Usando el método `roleDevelopsInLocation`, se obtiene la imagen mostrada en la Figura 5.4, la cual es generada con los datos procedentes de CameOn. En la Figura 5.4 se puede apreciar en el área amarilla que `rol` (referido a ir a comer), se puede hacer en `restaurant1`, `restaurant2`, `restaurantDeportivo1` y `restaurantDeportivo1`.

La siguiente consulta realizada a través del sistema es dónde se puede ver el juego, esta consulta es realizada mediante el método `activitiesThatCanBeDoneAtLocation`, el cual toma la información de CameOn y genera la imagen de la Figura 5.5. En la Figura 5.5, podemos apreciar

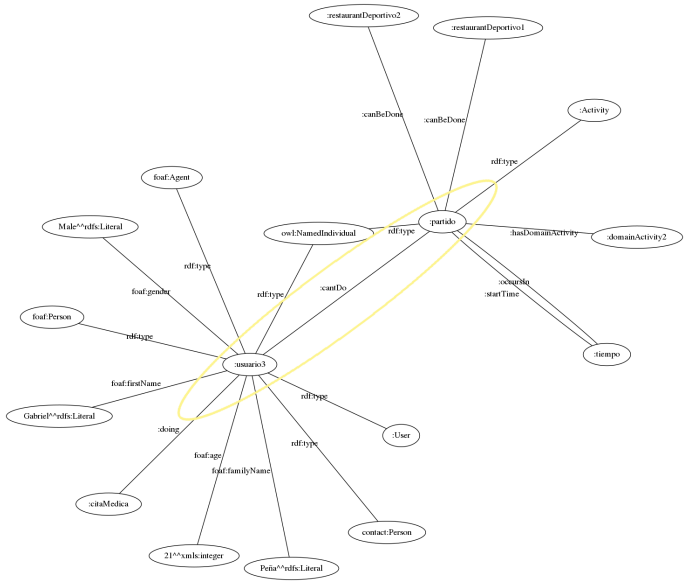

Figura 5.3: Imagen obtenida del sistema usando el método `activitiesThatCantBeConsumedByUsers`

en el área amarilla que el partido puede verse en `restaurantDeportivo1` y `restaurantDeportivo1`. Esto reduce las opciones de los usuarios en cuanto a donde ir.

Por último, para este caso, los usuarios quieren saber si pueden apostar en la ubicación que eligieron para ver el partido. Esto se realiza usando el método `devicesLocatedInPlaces` del sistema, el cual genera la imagen expuesta en la Figura 5.6 a partir de CameOn. En el área de color amarillo de la Figura 5.6, podemos ver que ambos restaurantes deportivos, `restaurantDeportivo1` y `restaurantDeportivo1`, poseen el dispositivo para realizar apuestas.

5.3.2 Segunda Parte del Escenario de Prueba usando Lógica de Primer Orden

Para la segunda parte del escenario de prueba, los usuarios cambian de contexto, por lo que se crearon nuevas instancias que pueden ser vistas en la Figura 5.7. En la segunda parte, al igual que en la primera, se creo una clase que usará el sistema y realizará las consultas. Esta clase es un ejemplo de como un algoritmo que quisiera

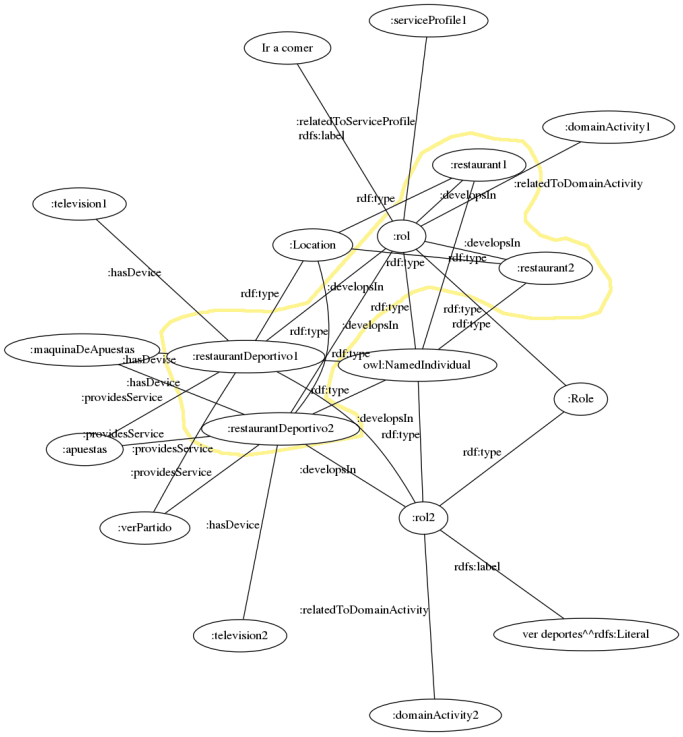

Figura 5.4: Imagen obtenida con el sistema usando el método `roleDevelopsInLocation`

usar los servicios de nuestro sistema debería invocarlo, y se puede apreciar en la Figura 5.8. Se crearon las instancias de usuarios (`usuario1`, `usuario2`, `usuario3`, `usuario10`, ..., `usuario14`), algunas actividades (`cenaFamiliar` y `juegoDeFutbol`), unos dispositivos (`balonDeFutbol`, `canchaFutbolSala` y `ropaDeportiva`), una ubicación (`canchasDeFutbol`), un servicio (`hacerDeporte`) y, por último, una instancia de tiempo.

En este caso, los usuarios primero desean asegurarse de que sus invitados posean la indumentaria para poder jugar. Esto se realiza usando el método `usersWhoCanConsumeServices`, y genera la imagen de la Figura 5.9, donde se puede apreciar dentro del círculo amarillo los usuarios que poseen un arco titulado `:canConsume` con el nodo `:hacerDeporte`.

Luego, revisan si alguno de los usuarios no puede ir al juego. Usando el método `activitiesThatCantBeConsumedByUsers` se obtiene la imagen Figura 5.10 usando los datos de CameOn, donde se ve claramente que el `usuario12` no puede ir debido a otra

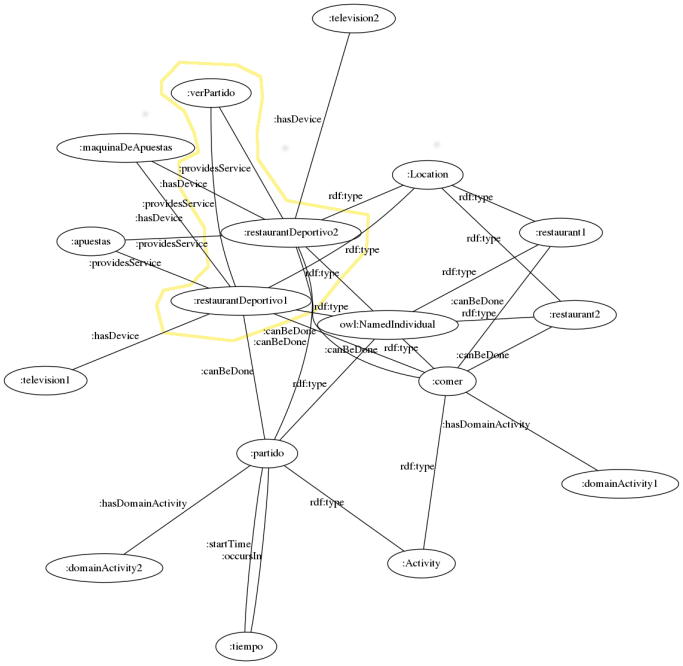

Figura 5.5: Imagen obtenida con el sistema usando el método `activitiesThatCanBeDoneAtLocation`

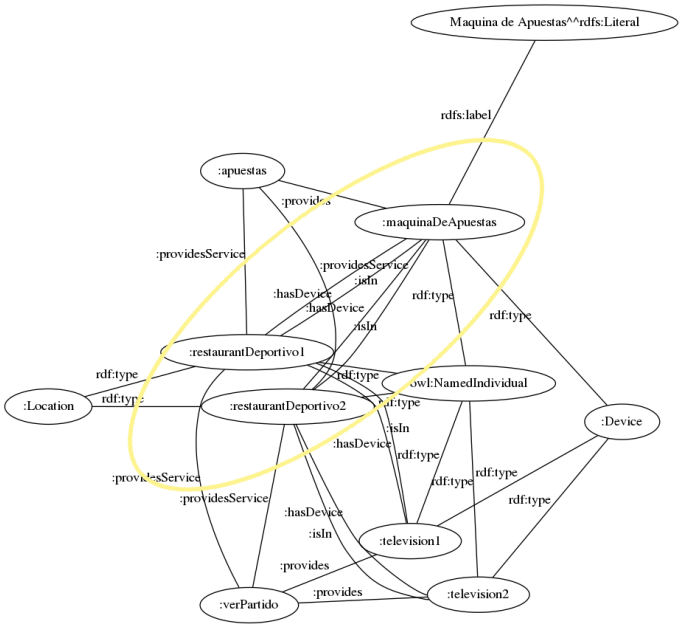

Figura 5.6: Imagen obtenida del sistema usando el método `devicesLocatedInPlaces`

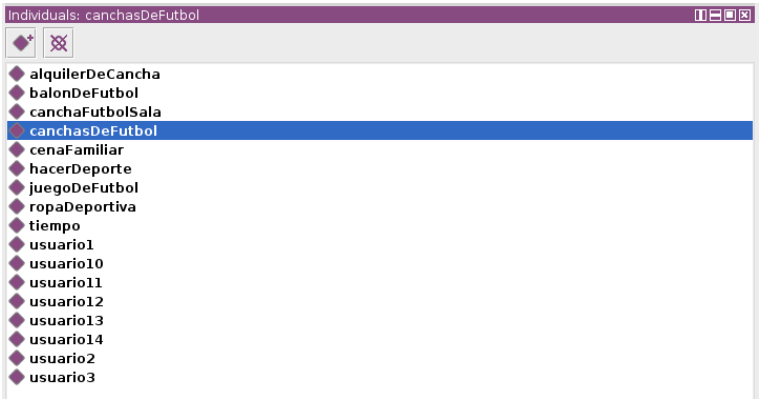

Figura 5.7: Instancias creadas para realizar la segunda parte de la prueba

```
public class Case2 {
    public static void main(String[] args) {
        org.apache.log4j.BasicConfigurator.configure(new NullAppender());
        System.out.println("Segunda parte del caso de estudio");
        Prototype caso2 = new Prototype("../ontology/pruebas/futbol.owl");
        Model canPlay = caso2.usersWhoCanConsumeServices();
        Prototype.modelToDOT(canPlay, "./resources/pruebas/caso2/canPlay");
        Model whoCantPlay = caso2.activitiesThatCantBeConsumedByUsers();
        Prototype.modelToDOT(whoCantPlay, "./resources/pruebas/caso2/whoCantPlay");
        Model whereIsABall = caso2.devicesLocatedInPlaces();
        Prototype.modelToDOT(whereIsABall, "./resources/pruebas/caso2/whereIsABall");
        Model whoLendsFields = caso2.locationProvidesService();
        Prototype.modelToDOT(whoLendsFields, "./resources/pruebas/caso2/whoLendsFields");
        Model information = Prototype.outputInformation("./resources/pruebas/caso2/information.owl",
                "N-TRIPLES", canPlay, whoCantPlay, whereIsABall);
    }
}
```

Figura 5.8: Clase programada para la segunda parte de la prueba

actividad (`cenaFamiliar`) que puede ser apreciada en el área amarilla.

Después, los usuarios desean poder alquilar un balón para poder jugar, por lo que usando el método `devicesLocatedInPlaces`, se obtiene la información desplegada en la imagen de la Figura 5.11, que indica quienes alquilan balón (ver área amarilla) .

Por último, los usuarios necesitan saber donde se alquilan canchas, lo que hacen mediante el método `locationProvidesService`, que genera la imagen de la Figura 5.12 (ver área amarilla, que muestra quienes alquilan canchas).

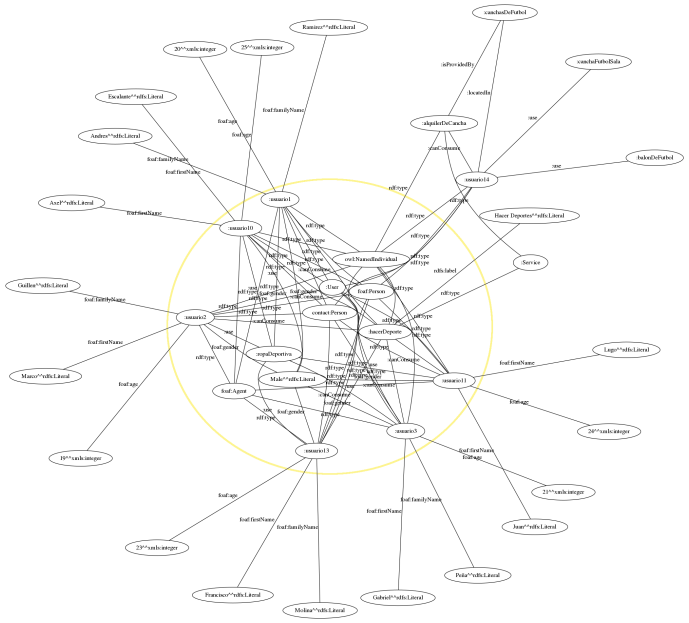

Figura 5.9: Imagen obtenida del sistema usando el método `usersWhoCanConsumeServices`

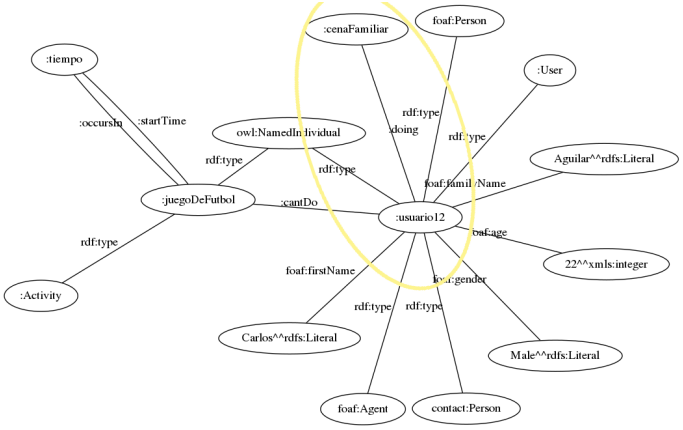

Figura 5.10: Imagen obtenida a partir del sistema usando el método `activitiesThatCantBeConsumedByUsers`

5.3.3 Ejecución los Escenarios de Prueba usando Lógica Dialéctica

Para el caso de estudio del axioma número 4 (véase Tabla 4.1), se desarrollo una clase sencilla que haciendo uso de la clase `Prototype` y del método

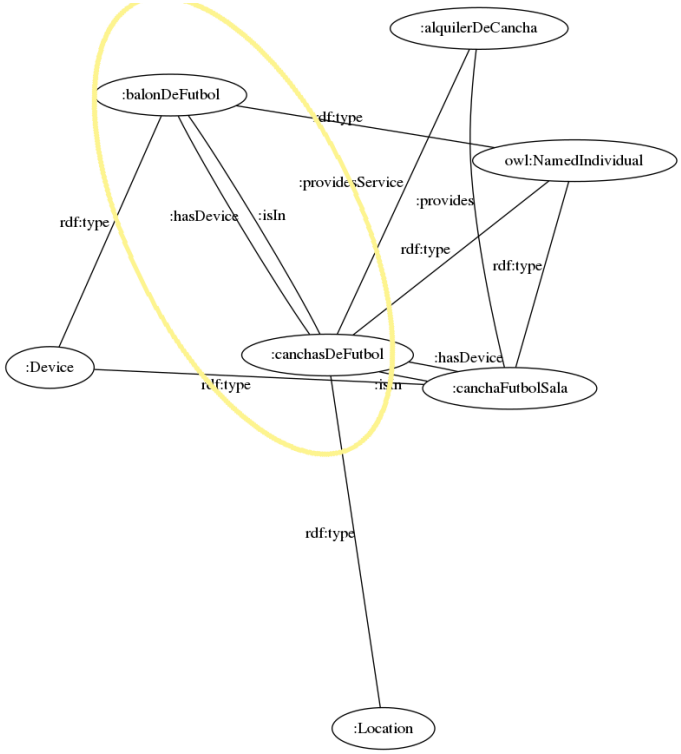

Figura 5.11: Imagen obtenida a partir del sistema usando el método `devicesLocatedInPlaces`

`userCanGoToActivityNearHisLocation`, consultará el archivo donde se instanció el contexto. El código de la clase puede apreciarse en la Figura 5.13. El archivo donde se describe el contexto en que será evaluado el axioma 4, en el formato aceptado por el sistema *TPTP*, se puede ver en la Figura 5.14.

Antes que nada, en el contexto de *TPTP*, el término "axioma" sirve para hacer declaraciones. El archivo que se muestra en la Figura 5.14 contiene en los axiomas `1`, `2`, `3`, `4`, `5`, y `6` declaraciones de instancias de las clases de CameOn, que representan los hechos (individuos) que describen el contexto en que sera evaluado (instanciado) el axioma 4. . Seguidamente, en los axiomas `7`, `8`, `9`, `10`, `11`, y `12` están las relaciones entre las instancias previamente declaradas , que son también requeridas por el axioma 4 para describir el contexto. Luego viene la descripción del axioma 4 que construimos para el desarrollo de este trabajo (véase Tabla 4.1). Por último, se encuentra la conjetura que queremos evaluar, es decir, que es lo que queremos resolver: ¿irá el usuario al

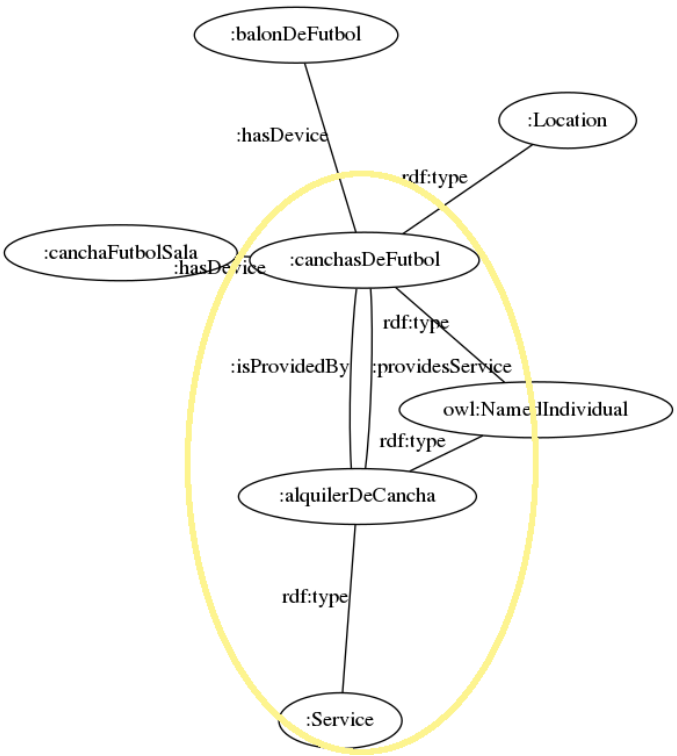

Figura 5.12: Imagen obtenida a partir del sistema usando el método `devicesLocatedInPlaces`

```
public class Case3 {

    public static void main(String[] args) {
             org.apache.log4j.BasicConfigurator.configure(new NullAppender());
        Prototype caso3 = new Prototype();
        System.out.println("Resultado de evaluar el axioma 4:");
        System.out.println(caso3.
                userCanGoToActivityNearHisLocation("./resources/ld/problemas/pruebaAxioma4.p"));
    }

}
```

Figura 5.13: Clase programada para la primera prueba usando Lógica Dialéctica

concierto?

Al ejecutar el método, la respuesta obtenida es `CounterSatisfiable`, lo cual puede apreciarse en la Figura 5.15. Analizaremos este resultado mas adelante.

Por otro lado, para el axioma número 7, también se creo una clase sencilla que

```
fof(1, axiom, activity(concierto) ).
fof(2, axiom, location(ubicacionDelConcierto) ).
fof(3, axiom, location(ubicacionCercana) ).
fof(4, axiom, time(horaDelConcierto) ).
fof(5, axiom, time(horaActual) ).
fof(6, axiom, user(usuario) ).
fof(7, axiom, locatedIn(usuario, ubicacionCercana) ).
fof(8, axiom, starts(concierto, horaDelConcierto) ).
fof(9, axiom, isIn(concierto, ubicacionDelConcierto) ).
fof(10, axiom, soon(horaActual, horaDelConcierto) ).
fof(11, axiom, close(ubicacionCercana, ubicacionDelConcierto) ).
fof(12, axiom, isTime(usuario, horaActual) ).
fof(userCanGoToActivityNearHisLocation, axiom,(
    ?[User, Location2, Time1, Activity, Location1, Time2]:
    (( user(User)
    & location(Location2)
    & time(Time1)
    & activity(Activity)
    & location(Location1)
    & time(Time2)
    & locatedIn(User, Location2)
    & starts(Activity, Time1)
    & isIn(Activity, Location1)
    & soon(Time2, Time1)
    & close(Location2, Location1)
    & isTime(User, Time2) )
    => goes(User, Activity) ) )).
fof(conjecture, conjecture, goes(usuario, concierto)).
```

Figura 5.14: Instancia del axioma número 4 (véase Tabla 4.1)

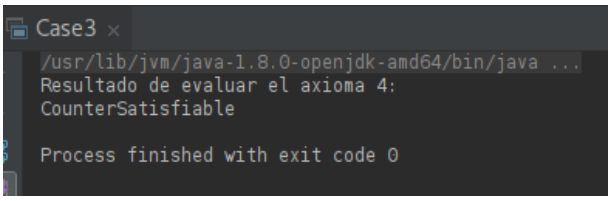

Figura 5.15: Resultado obtenido del sistema al probar el axioma 4.

use la clase `Prototype` y el método `locationIsOpen`. El código puede ser visto en la Figura 5.16. El contexto instanciado puede apreciarse en la Figura 5.17.

El archivo de la Figura 5.17, contiene en primer lugar, en los axiomas `1`, `2`, `3`, y `4` instanciaciones de las clases (individuos) de CameOn según el contexto a analizar con el axioma `7`. Luego, en los axiomas `5`, `6`, y `7` están las relaciones entre las instancias

```
public class Caso4 {

    public static void main(String[] args) {
        org.apache.log4j.BasicConfigurator.configure(new NullAppender());
        Prototype caso4 = new Prototype();
        System.out.println("Resultado de evaluar el axioma 7:");
        System.out.println(
                caso4.locationIsOpen("./resources/ld/problemas/pruebaAxioma7.p"));
    }

}
```

Figura 5.16: Clase programada para la segunda prueba usando Lógica Dialéctica

que se crearon previamente, requeridas por el axioma 7 para completar de describir el contexto a evaluar. Después, esta el axioma numero 7 (véase Tabla 4.1), y por último la conjetura a evaluar: ¿está el restaurante abierto?

```
fof(1, axiom, user(usuario) ).
fof(2, axiom, location(restaurant) ).
fof(3, axiom, activity(cenaCompania) ).
fof(4, axiom, time(horaActual) ).
fof(5, axiom, locatedIn(usuario, restaurant) ).
fof(6, axiom, doing(usuario, cenaCompania) ).
fof(7, axiom, isTime(usuario, horaActual) ).
fof(locationIsOpen, axiom,(
    ? [User, Activity, Location, Time]:
    ((user(User)
    & activity(Activity)
    & location(Location)
    & time(Time)
    & locatedIn(User, Location)
    & doing(User, Activity)
    & isTime(User, Time) )
    => open(Location, true) ) )).
fof(conjetura, conjecture, open(restaurant, true) ).
```

Figura 5.17: Instancia del axioma número 7 (véase Tabla 4.1)

Al ejecutar el método, la respuesta obtenida es `CounterSatisfiable`, lo cual se puede ver en la figura 5.18. Esta respuesta sera analizada mas adelante.

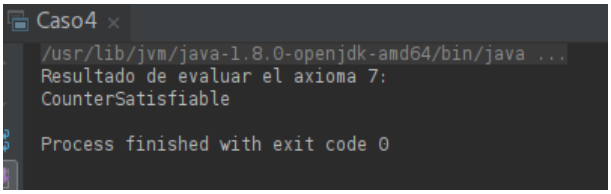

Figura 5.18: Resultado obtenido del sistema al probar el axioma 7.

5.4 Resultados Con Lógica de Primer Orden y Datos Enlazados

Los resultados de las métricas en ambas partes del caso de estudio son expuestos en las Tablas 5.1 y 5.2.

Tabla 5.1: Comparación de Resultados para la primera parte del caso de estudio

Métodos	Lógica de Primer Orden		Datos Enlazados	
	Nodos Obtenidos	Aristas Obtenidas	Nodos Obtenidos	Aristas Obtenidas
activitiesThatCanBeDone AtLocation	6	6	17	29
devicesLocatedInPlaces	5	4	11	26
roleDevelopsInLocation	6	6	19	31
activitiesThatCantBe ConsumedByUsers	2	1	17	18

Para generar los datos de Lógica de Primer Orden en los casos de estudio, se usaron las reglas escritas en *SWRL* que se describieron en 4.2.1, y se contaron tanto las relaciones (aristas) como los recursos (nodos) generados.

Como podemos ver, la cantidad de conocimiento (nodos y aristas) es bastante mayor siempre en los casos en los que se usan los datos enlazados. En la sección 5.1 mencionamos que la cantidad de nodos y aristas generados representan conocimiento, ya que, los nodos y las aristas por si solos son sólo datos, pero al unir un par de nodos con una arista se genera información, la cual nuestro sistema transforma en conocimiento al darle un uso para mejor caracterizar el contexto.

Tabla 5.2: Comparación de Resultados para la segunda parte del caso de estudio

Métodos	Lógica de Primer Orden		Datos Enlazados	
	Nodos Obtenidos	Aristas Obtenidas	Nodos Obtenidos	Aristas Obtenidas
usersWhoCanConsume Services	9	7	39	72
devicesLocatedInPlaces	3	2	7	12
locationProvidesService	2	1	7	8
activitiesThatCantBe ConsumedByUsers	2	1	14	15

En las Tablas 5.1 y 5.2 podemos ver que el principio de la Web Semántica, de enriquecer la información usando conexiones semánticas, permite obtener mas información sobre los recursos que son de interés.

Tomemos como ejemplo la Figura 5.3. Al usar sólo Lógica de Primer Orden, en este caso, la información obtenida es la que puede ser apreciada en la figura 5.19.

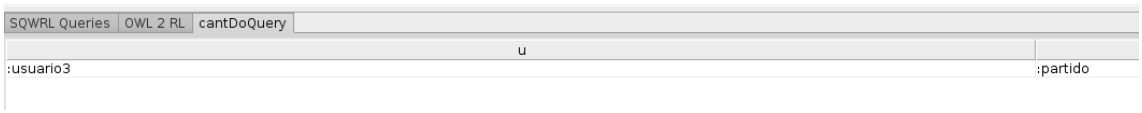

Figura 5.19: Información obtenida del método `activitiesThatCantBeConsumedByUsers` si solo usa Lógica de Primer Orden

Viendo la Figura 5.19, podemos rápidamente apreciar que la información es muy pobre. Sabemos que el usuario `usuario3` no puede asistir a la actividad `partido`, pero no sabemos nada acerca del usuario, ni de la actividad, si no conocemos el contexto. Lo anterior no lo podemos considerar como información útil. En cambio, al observar la imagen 5.3 podemos ver información adicional acerca del contexto en el cual se esta realizando la aseveración "El `usuario3` no puede asistir a la actividad `partido`", porque el usuario está en otra actividad (`citaMedica`), que el usuario es un hombre, de nombre "Gabriel Peña", así como también, que el partido tiene hora de comienzo determinada por `tiempo`.

Otro ejemplo de esto, puede apreciarse tomando como ejemplo la Figura 5.11. Si solo tomamos la información obtenida con Lógica de Primer Orden, obtenemos la información mostrada en la Figura 5.20. Viendo la Figura 5.20, podemos ver que la información obtenida es muy simple, la ubicación `canchasDeFutbol` tiene `balonDeFutbol` y `canchaFutbolSala`. Por otro lado, en la Figura 5.11, con los datos enriquecidos, podemos ver que además esta ubicación ofrece el servicio de alquiler de cancha y de balón, servicios requeridos por los usuarios.

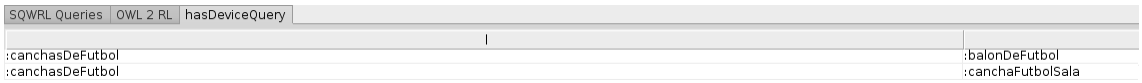

Figura 5.20: Información obtenida del método `devicesLocatedInPlaces` si solo usa Lógica de Primer Orden

5.5 Resultados Con Lógica Dialéctica

Para ambos axiomas, la respuesta generada por nuestro sistema fue `CounterSatisfiable` (ver Figura 5.15, para el caso de del axioma 4, y Figura 5.18 para el caso del axioma 7).

La documentación del sistema *TPTP* indica que el sistema recibe como entrada un problema F de la forma $Ax \implies C$, donde Ax son un conjunto de fórmulas lógicas que se deben cumplir, para que se infiera C, y C es una fórmula, de salida. *TPTP* evalúa esa entrada Ax en un conjunto de hechos (instancias) para determinar la veracidad del problema F.

Mas adelante, la documentación del sistema *TPTP*, indica que dicho estatus (`CounterSatisfiable`, o incontestable en español) es generado cuando "Alguno de los axiomas Ax, permiten corroborar o no la conjetura C.". En el marco de *TPTP*, F puede ser no valida, ó satisfacible en el contexto donde esta siendo instanciado. En nuestro caso, para los dos ejemplos de instanciación de cada axioma dio incontestable.

Este resultado nos indica que las conjeturas de la Lógica Dialéctica son verificables en el sistema *TPTP*, es decir, que puede verificarse las instanciaciones del contexto válidas para los axiomas. En nuestro caso, las afirmaciones que se

validaron para ese contexto fueron, en el caso del axioma 4 la conjetura `goes(usuario, concierto)`, y, en el caso del axioma 7 `open(restaurant, true)`, (es lo que se muestra en las Figuras 5.14 y 5.17, respectivamente).

Capítulo 6

Conclusiones

La Web Semántica es una poderosa herramienta, la cual posee increíbles cantidades de información. Los Datos Enlazados, por su parte, ayudan a explotar las enormes cantidades de información que se encuentran en la Web Semántica, y a su vez, ayudan a crear más información, crear conexiones semánticas entre datos, para extender la información existente.

En un mundo donde cada vez se generan mas datos y mas información, la tecnología necesita hacer uso de los datos y de la información generada para crear conocimiento, y permitir mejores experiencias para sus usuarios. Las Aplicaciones Conscientes de Contexto se basan en este principio, para brindar información del contexto que permita mejorar la calidad de la experiencia.

Al unir los Datos Enlazados, junto con las Aplicaciones Conscientes de Contexto, se explotan los datos y la información que existe en la Web Semántica, para ofrecer una mejor experiencia de usuario. Esto permite crear conocimiento sobre el contexto del usuario, el cual cambia constantemente, que luego son consumidos de nuevo para crear nuevo conocimiento, y así sucesivamente.

Por otro lado, es común que existan ambigüedades en el mundo real, donde en muchos casos la verdad solo es absoluta para cada individuo. En estos casos no existe una respuesta correcta, y esto lo busca atacar el enfoque de la Lógica Dialéctica.

Durante el desarrollo de este trabajo, se tomo la ontología CameOn, cuyo propósito es modelar el contexto de los usuarios. Como primera parte del trabajo,

esta ontología fue extendida con algunos vocabularios conocidos de la Web Semántica, permitiendo así la asociación de las clases de la ontología con las clases existentes en la Web Semántica. Por este lado, se obtiene ya el beneficio previamente expuesto de los datos enlazados, permitiendo la creación de más información con significado semántico legible por las máquinas.

Por otro lado, se generaron axiomas que hacen uso de las clases previamente definidas en la ontología CameOn, y generan información acerca de las instancias que se encuentren en ella, pudiendo así generar más conocimiento que al ser entregado como servicio, será de utilidad para las aplicaciones. Estos axiomas se generaron usando Lógica de Primer Orden, cuando se podía afirmar que dados los hechos, la conclusión de los axiomas es verdadera.

Sin embargo, por las razones explicadas previamente, hay casos en los que la Lógica de Primer Orden no es suficiente para describir con certeza la situación, y es necesario otro enfoque que tome en cuenta las ambigüedades existentes en la situación. De esta manera, este trabajo abre camino en este área, usando el enfoque de la Lógica Dialéctica en algunos de los axiomas generados, para poder producir información que ya no es verdadera en todos los casos, sino solo en el de algunos de los usuarios.

De esta manera, el sistema desarrollado en el marco de este trabajo explota todo este potencial, basándose en una ontología que es capaz de describir cualquier contexto en el que se encuentra el usuario, extendiéndolo semanticamente con vocabularios comúnmente usados en la Web Semántica, para posteriormente razonar usando Lógica de Primer Orden y Lógica Dialéctica, basado en axiomas que usan los datos extraídos, tanto de los instanciados en el contexto del usuario, como de los disponibles en la web que fueron enlazados al contexto del usuario, y a partir de ello crear conocimiento.

También, el sistema hace disponible el conocimiento extraído como un servicio, a cualquier aplicación que desee hacer uso de ella. Mediante esto, las aplicaciones que consuman el servicio que es prestado por el sistema, pueden hacer uso del conocimiento generado, para crear más conocimiento y mejorar la calidad de los servicios que ellas brindan a sus usuarios.

Debido a los resultados obtenidos, fácilmente es posible notar que el uso de los Datos Enlazados, y la posterior inferencia usando cualquiera de los enfoques de Lógica

propuestos en este trabajo, puede solucionar muchos problemas cotidianos. Existen muchos trabajos que innovan en el área de los datos enlazados y la consciencia de contexto, algunos de ellos fueron expuestos como antecedentes de este trabajo. A su vez, también existen trabajos que buscan explotar otros enfoques a la Lógica; sin embargo, no conseguimos trabajos que incluyan todas estas áreas y exploten los grandes potenciales de cada área.

Este trabajo fue desarrollado sobre dos trabajos más, como lo son el Middlware CARMiCLOC y la ontología CameOn. Sobre ellos, el sistema propuesto agrega una tecnología en auge, como lo son los datos enlazados, y el manejo de ambigüedades, el cual le da nuevas maneras de gestión de los datos, para luego entregar su producto a aplicaciones Conscientes de Contexto que hagan uso de él.

6.1 Recomendaciones

En futuros trabajos se recomienda, en primer lugar, agregar más axiomas explotando todas las posibilidades que dan las clases y propiedades de CameOn, de manera que la cantidad de conocimiento que sea generada sea mayor, lo cual mejorará la funcionalidad del sistema desarrollado en este trabajo.

Por otro lado, también se recomienda continuar la extensión semántica de CameOn, usando la mayor cantidad posible de vocabularios que puedan ser encontrados públicos en la Web Semántica; de esta manera, la cantidad de datos obtenidos de los datos enlazados será mayor y podrá generarse mas conocimiento.

Otro trabajo futuro interesante es la consideración de más casos (escenarios). Todas las recomendaciones anteriores, posibilitaran la idea de ir generando futuras versiones de este trabajo, como también futuras versiones de la ontología usada, a partir de la incorporación de más propiedades y clases, según sea la necesidad generada en los distintos casos estudiados.

A su vez, también se recomienda extender la funcionalidad del sistema, automatizando la generación de los archivos necesarios por el sistema *TPTP*, basándose en los datos que se encuentran en la instancia de CameOn.

Finalmente, este trabajo se basa en la instanciación de una ontología para poder

funcionar, por lo que un trabajo interesante y necesario seria la automatización de la instanciación del contexto de cada usuario; y debido a la cantidad de datos que se manejarían, otros trabajos podrían ser el manejo de la seguridad y la privacidad de los datos del sistema como el uso de técnicas de datos masivos (*Big Data*) para su tratamiento.

Bibliografía

Aguilar, J., Cerrada, M., Altamiranda, J., Pacheco, F., y Rangel, C. (2013). Methodology for detecting the feasibility of using data mining in an organization. *XXXIX Conferencia Latinoamericana en Informática (CLEI 2013)*, 1:502–513.

Aguilar, J., Jerez, M., Exposito, E., y Villemur, T. (2015). CARMiCLOC: Context Awareness Middleware in CLOud Computing. *2015 Latin American Computing Conference*.

Aguilar, J., Jerez, M., Rodriguez, T., y Exposito, E. (2017). CAMeOn: Context Awareness Meta Ontology Modeling. *Applied Computing and Informatics*.

Berners-Lee, T. (2006). Linked data. `https://www.w3.org/DesignIssues/LinkedData.html`. Visitado el 26-04-2018.

Brickley, D. y Miller, L. (2000). Friend of a friend. `http://xmlns.com/foaf/spec/`. Visitado el 06-06-2018.

Costabello, L. (2013). *Context-aware access control and presentation of linked data*. Tesis de Doctorado, Université Nice Sophia Antipolis.

de Freitas, A. A., Nebeling, M., Ranithangam, A. S. K. K., Yang, J., y Dey, A. K. (2016). Bluewave: Enabling Opportunistic Context Sharing via Bluetooth Device Names. En *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '16, p. 38–49, New York, NY, USA. ACM.

Dean Allemang, J. H. (2011). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, 2 edición.

Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7.

Google, Bing, y Yahoo! (2011). Schema vocabulary. `http://www.schema.org`. Visitado el 06-06-2018.

Manzano, M. y Huertas, A. (2000). *Lógica para principiantes*. Proyecto ARACNE.

Musen, M. A. (2015). The protégé project: A look back and a look forward. *AI Matters*, 1(4):4–12.

Pacheco, F., Rangel, C., Aguilar, J., Cerrada, M., y Altamiranda, J. (2014). Methodological Framework for Data Processing based on the Data Science Paradigm. *XL Conferencia Latinoamericana en Informática (CLEI 2014)*.

Pelletier, F., Sutcliffe, G., y Hazen, A. (2017). Automated Reasoning for the Dialetheic Logic RM3. *Thirtieth International Florida Artificial Intelligence Research Society Conference*, p. 110–115.

Phalle, P. y Salunkhe, S. (2009). Using Linked Data to Context-Aware Annotate and Search Educational Video Resources. *IOSR Journal of Computer Engineering (IOSR-JCE)*, p. 61–63.

Posdorfer, W. y Maalej, W. (2016). Towards Context-aware Surveys Using Bluetooth Beacons. *Procedia Computer Science*, 83:42–49. The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops.

Real Academia Española (2014). *Diccionario de la Real Academia Española*. Real Academia Española, 23va edición. Consultado en linea en http://dle.rae.es/.

Rodriguez, T., Dos Santos, R., Aguilar, J., y Gonzalez, A. (2017). Metodología para el desarrollo de Aplicaciones Web utilizando Datos Enlazados. *XLIII Conferencia Latinoamericana en Informática*.

Segaran, T., Evans, C., Taylor, J., Toby, S., Colin, E., y Jamie, T. (2009). *Programming the Semantic Web*. O'Reilly Media, Inc., 1st edición.

Sutcliffe, G. (2017). The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502.

The Apache Software Foundation (2011). Apache jena. `http://jena.apache.org`. Visitado el 06-06-2018.

Wang, X. H., Zhang, D. Q., Gu, T., y Pung, H. K. (2004). Ontology Based Context Modeling and Reasoning using OWL. *Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW 04)*.

Zhong-Jun, L., Guan-yu, L., y Ying, P. (2016). A Method of Meta-Context Ontology Modeling and Uncertainty Reasoning in SWoT. *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*.