UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 5

Trabajo de Diploma para optar por el título de Ingeniería en Ciencias Informáticas

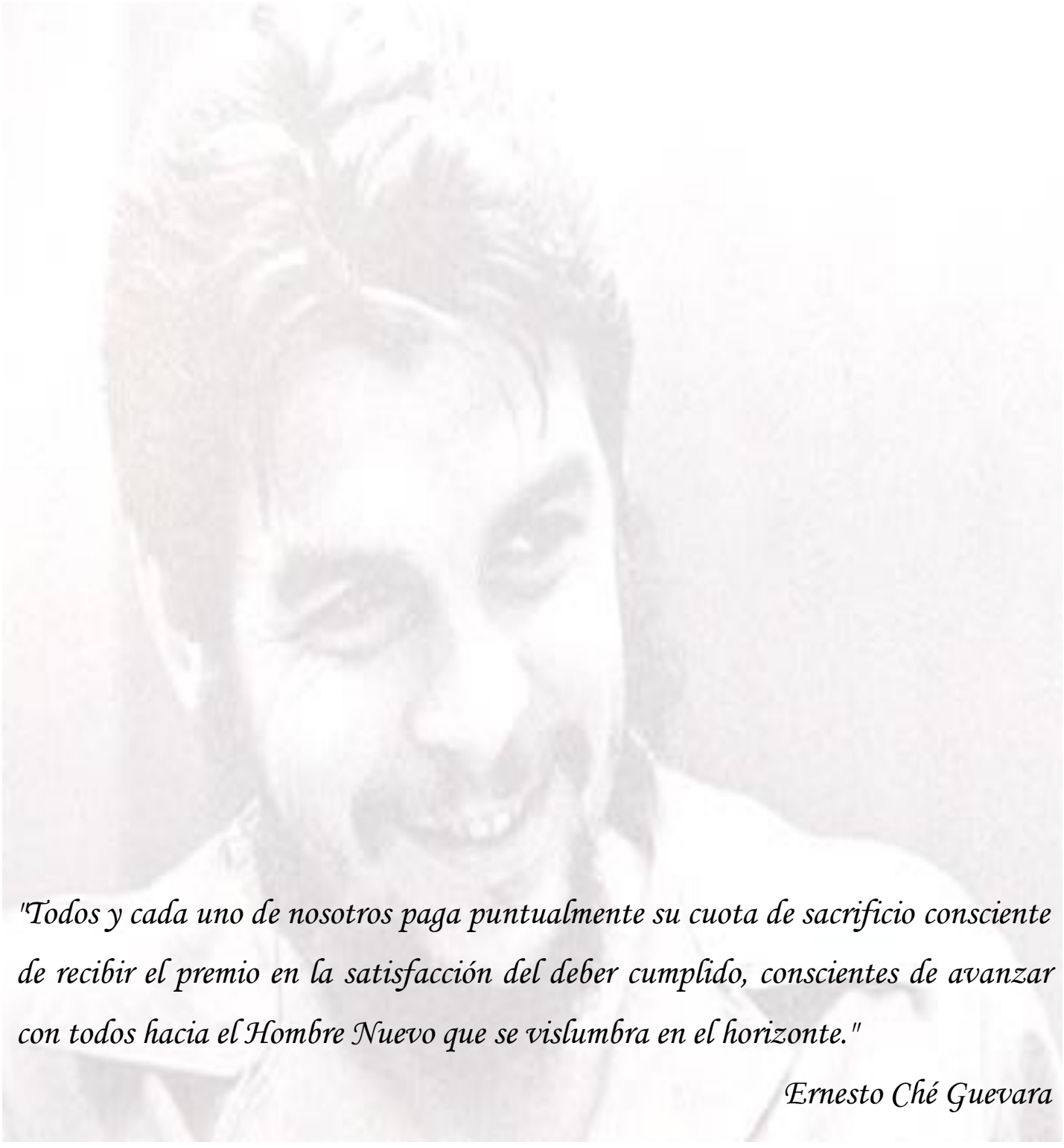
Título: *Stream Reasoning* utilizando Lógicas Descriptivas

Autora:
Patricia Airela Abreu Fong

Tutores:
MSc. Liudmila Reyes Álvarez
MSc. José Manuel Fernández Hechavarría

La Habana, 2012

"Año 54 de la Revolución"



"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

Ernesto Ché Guevara

Declaración de autoría

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de _____ del año_____.

_____

Firma de la autora

Patricia Airela Abreu Fong

_____ _____

Firma de la Tutora Firma del Tutor

MSc. Liudmila Reyes Álvarez MSc. José Manuel Hernández

Datos de contactos

Tutora:

Nombre y Apellidos: MSc. Liudmila Reyes Álvarez.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: lreyes@uci.cu

Graduada de la UCI en el año 2007. Profesor asistente con 5 años de experiencia docente y 8 años de experiencia en la producción. Pertenece al Departamento Integración y Despliegue del Centro de Desarrollo de Informática Industrial. Además es Máster en Ingeniería de Software e Inteligencia Artificial de la Universidad de Málaga, España.

Tutor:

Nombre y Apellidos: MSc. José Manuel Fernández Hechavarría.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: argos@instec.cu, argos@uci.cu

Profesor instructor con 10 años de experiencia docente. Pertenece al Departamento de Ciencias Básicas de la Facultad 5. Además es Máster en Bioinformática.

Agradecimientos

A mi madre por tantas noches de desvelo y por su amor sin límites.

A mi abuela María por darme una lección de amor y generosidad cada día.

A mi abuelo Giraldo por su cariño incondicional de padre y abuelo.

A mi tía Betty por su amor de madre y sus gestos de cariño.

A mi novio Liosbel por convertirme en el centro de su vida y ser cómplice en mis sueños.

A mi madrina Airela por vencer los obstáculos de la distancia y estar presente en mi vida.

A Iraida, que supera los lazos de amiga para convertirse en mi hermana.

A mi padrastro Gerar, porque sin su presencia nuestras vidas no hubiesen sido iguales.

A mis amigos en general, que no se distinguen por la cantidad pero sí por la calidad.

A mis tutores, en especial a Liudmila, por su apoyo a pesar de tantos y tantos compromisos.

Dedicatoria

A mi mamá por centrar su vida en mi felicidad y en hacerme una mujer de bien. Es por eso que cada logro de mi vida estará siempre dedicado a ella.

A mi familia, que aunque pequeña me ha entregado tanto amor que no me alcanzará la vida para retribuírselo.

Resumen

Un significativo número de proveedores de datos, motivados por Tim Berners-Lee en años recientes han adoptado los principios de datos vinculados para la publicación y conexión de datos estructurados sobre la web, creando un espacio de datos conocido como: Web Semántica. Recientemente con la aparición de la Web Semántica se hace necesario una eficiente representación del conocimiento y un razonamiento escalable sobre todo este conocimiento/datos representados en la web actual. De ahí la importancia del estudio de las Lógicas Descriptivas que son una familia de lenguajes muy usadas para la representación y razonamiento del conocimiento.

Teniendo en cuenta lo anterior se traza como objetivo de la investigación: desarrollar una propuesta de algoritmo que utilice las técnicas de razonamiento que presentan las *DLs* para aumentar la escalabilidad y la eficiencia en el proceso de razonamiento sobre grandes corrientes de datos (*stream reasoning*).

Con el fin de darle solución al objetivo trazado se decide dividir el trabajo presentado en tres capítulos. El primero realiza un estudio del estado del arte respecto a las Lógicas Descriptivas y el *stream reasoning*. En el segundo capítulo se abordan las herramientas utilizadas para razonar siguiendo el uso de los formalismos lógicos presentes en las Lógicas Descriptivas. En el tercero se selecciona el razonador de Lógicas Descriptivas y se propone la implementación de un pseudocódigo utilizando la técnica de razonamiento incremental basada en la clasificación incremental para que se resuelva el problema del *stream reasoning*.

Palabras Clave: *data stream*, Lógicas Descriptivas, *Pellet*, razonamiento incremental, *stream reasoning*.

Índice

Índice

Índice de Figuras

Índice de tablas

Introducción

La era digital ha involucrado a los profesionales del mundo de hoy y demanda inexorablemente la aplicación de las Nuevas Tecnologías de la Información y de las Comunicaciones (NTICs) en las diversas esferas de la sociedad. De esta manera a nivel mundial se impone la implementación de nuevas tecnologías, haciendo valederas las palabras del prestigioso empresario estadounidense Bill Gates ¨La tecnología no es en sí el fin sino el medio entre la sociedad del conocimiento y el desarrollo mundial¨.

Las investigaciones realizadas a escala mundial, que giran en torno a la Representación del Conocimiento (RC), se centran por lo general en la búsqueda de métodos que proporcionen descripciones del mundo de alto nivel. Para la RC se utiliza una familia de formalismos basados en la Lógica de Primer Orden (LPO) denominados: Lógicas Descriptivas o Lógicas de Descripción conocido en inglés como *Description Logics* (*DLs*). Estos formalismos se distinguen por contar con una semántica formal y por proveer servicios de inferencia y razonamiento al conocimiento que ha sido representado.

En Cuba se ha profundizado en las investigaciones relacionadas con el tema de la RC, enfocadas al estudio de lenguajes de RC como *OWL*[1] (*Ontology Web Language*) y *RDFS*[2] (*Resource Descripcion Framework Schema*) y en la mejora de la RC utilizando estos lenguajes; por medio del desarrollo de ontologías y del proceso de razonar sobre estas. La Universidad de las Ciencias Informáticas (UCI) ha promovido investigaciones relevantes que se adentran en el tema y proporcionan soluciones cada vez más eficientes. En este sentido se destacan los proyectos desarrollados en la facultad 6 en el centro de Geoinformática y Señales Digitales (GEySED) y además las tesis doctorales de los profesores Liudmila Reyes, Yuniel E. Proenza y Dayany Díaz; todas se enmarcan en investigaciones que responden a novedosos métodos de RC, Razonamiento y Datos Vinculados (*Linked Data* – como se le conoce en inglés) para la Web Semántica.

El uso de Internet como una fuente importante de información y el surgimiento de las redes sociales; ha creado nuevos desafíos para la informática y ha conducido a una importante innovación en áreas como las

[1] *OWL: http:// http://www.w3.org/TR/owl-features*
[2] *RDFS: http://www.w3.org/standards/techs/rdf#w3c*

bases de datos, recuperación de información y las tecnologías semánticas. En la actualidad, nos enfrentamos con un cambio importante. Tradicionalmente la información que se procesaba, se almacenaba de manera estática y poco variable. Hoy en día, la información es cada vez más dinámica y en ocasiones es adquirida a partir de sensores y en tiempo real. Razón por la cual el procesamiento continuo de corrientes de datos se ha convertido en un tema popular en las investigaciones relacionadas con bases de datos. Afrontar el desafío de responder las preguntas de manera cada vez más eficiente, implica que continúe la búsqueda de métodos de razonamiento más potentes. Los mismos constituyen la base para procesos que están a niveles superiores de toma de decisiones y que requieren un razonamiento complejo sobre corrientes de datos.

Teniendo en cuenta la dinámica de la web actual surge la siguiente situación problémica:

- ✓ Insuficiente procesamiento continuo del gran número de datos con que cuenta la web actual en tiempo real.
- ✓ Apoyo restringido a los procesos de toma de decisión en un número elevado de usuarios concurrentes en la web.
- ✓ Razonamiento desde cero sin necesidad del mismo, cuando se realizan cambios constantes en las bases de datos o en las bases de conocimiento de las ontologías.
- ✓ Las ontologías desarrolladas actualmente presentan grandes bases de conocimiento, lo que enlentece el proceso de razonamiento de las mismas.

Todos estos problemas se han intentado resolver con el empleo de las *DLs*. Las que pueden definirse como un conjunto de lenguajes que pueden ser usados para RC terminológico de un dominio de aplicación de una forma estructurada y con una semántica formal bien definida, de ahí que éstas se conozcan como un lenguaje formal para la RC y para razonar sobre él. Por tanto surge la necesidad de analizar y resolver el problema de la mejora de la eficiencia y la escalabilidad en el proceso de razonamiento sobre grandes corrientes de datos (*stream reasoning*).

Por lo que el **problema científico** de esta investigación se basa en: Mejorar el proceso de razonamiento sobre grandes corrientes de datos.

Centrando el **objeto de estudio** a: las técnicas de razonamiento de las *DLs*.

El **objetivo general** de esta investigación se enfoca en: Desarrollar una propuesta de algoritmo que utilice las técnicas de razonamiento que presentan las *DLs* para aumentar la escalabilidad y la eficiencia en el proceso de razonamiento sobre grandes corrientes de datos (*stream reasoning*).
Teniendo como **campo de acción**: el *stream reasoning*.

Para darle cumplimiento al objetivo se establecen una secuencia de **tareas de investigación**, tales como:

- Elaboración del marco teórico para fundamentar el estudio del estado del arte de la investigación, asociado a los temas de razonamiento con *DLs* y *stream reasoning*.
- Selección de las herramientas existentes que utilizan las *DLs* para el razonamiento de conjuntos de datos.
- Selección de las técnicas de las *DLs* y el *stream reasoning* que permiten aumentar la escalabilidad y eficiencia en el proceso de razonamiento.
- Elaboración de la propuesta de pseudocódigo de *stream reasoning* utilizando *DLs* para garantizar el control de los constantes cambios realizados en los componentes de la ontología de gran tamaño.
- Elementos teóricos que validan la utilidad del pseudocódigo propuesto.

La **idea a defender** persigue que: Con el desarrollo de una propuesta para el *stream reasoning* que utilice las *DLs*, se sentarán las bases para que durante el proceso de *stream reasoning* se aumente en escalabilidad y eficiencia.

Para la recopilación de información y el desarrollo de la investigación se emplearon una serie de **Métodos Científicos**, entre los que se encuentran:

- Los métodos teóricos, entre los que figuran:

  ✓ **El Analítico –Sintético:** Se utilizará en la investigación para extraer y analizar la información, al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta. Para aplicar este método científico primeramente se profundizará en los elementos relacionados con las *DLs* y el

proceso de *stream reasoning*; para luego definir las técnicas de razonamiento que mejoren la eficiencia y la escalabilidad en el proceso *stream reasoning*.

✓ **Análisis Histórico–Lógico:** Este método será empleado para el análisis del estado del arte de la investigación, su evolución y las tendencias actuales relacionadas con el tema a investigar.

- Entre los Métodos Empíricos:

  ✓ **Análisis de fuentes de información:** Serán empleadas distintas fuentes de información tales como: Internet, revistas, publicaciones, tesis anteriores.

  ✓ **Consulta a especialista:** Serán consultados algunos profesores del centro con conocimientos sobre el tema.

  ✓ **Observación:** El empleo de este método avala una investigación enmarcada en el verdadero problema a resolver. Permitiendo percibir la realidad objetiva de la situación y enfocarse en este aspecto para poder construir soluciones más eficientes.

La investigación se ha desglosado de la manera siguiente:

Capítulo #1: ¨Fundamentación Teórica¨:

En este capítulo se abordan dos temas que resulta el enfoque principal de la investigación. Por un lado se profundiza en las *DLs* como una herramienta potente para la representación y razonamiento sobre el conocimiento y por otro se analiza el *stream reasoning*.

Capítulo #2: ¨Razonadores de las *DLs*¨:

En este capítulo se caracterizan los razonadores de las *DLs*. Para este fin se tienen en cuenta una serie de elementos que resulten relevantes y que permiten realizar una selección del más factible para su aplicación en el *stream reasoning*.

Capítulo #3: ¨Propuesta para el *stream reasoning*¨:

En este capítulo se propone un pseudocódigo para llevar a cabo el *stream reasoning* empleando el razonador que sea seleccionado. Se ilustra una ontología que resulta eficaz para posteriores validaciones del algoritmo de *stream reasoning* con el empleo del razonador de la *DLs*.

Capítulo 1

Fundamentación Teórica

INTRODUCCIÓN

La necesidad de apoyar los numerosos procesos concurrentes, que se generan en la actualidad y que forman parte de la vida diaria, exige de aplicaciones eficientes y capaces de razonar sobre la información almacenada. Las *DLs* abarcan una amplia gama de lenguajes y ofrecen numerosas técnicas para la mejora de los servicios de inferencia. Mediante la utilización de constructores y con una semántica formal bien definida, viabilizan el manejo de la información y ayudan a la toma de decisiones de los usuarios concurrentes en la Web. Las mismas resultan una herramienta eficaz para ser aplicada en los procesos de *stream reasoning*. Por tanto en este capítulo se realiza un estudio de las *DLs* y de los conceptos asociados al *stream reasoning*.

1.1 Lógicas Descriptivas

Las *DLs* son una evolución natural de las redes semánticas o los *frames*[3]. Su surgimiento se debe a que sus antecesores no estaban equipados de una semántica basada en la lógica. Tradicionalmente son usados para la RC *taxonómico*[4] en aplicaciones de Inteligencia Artificial y conforman, por lo general, subconjuntos de LPO. Definen la descripción de conceptos relevantes de un dominio particular y la utilización de estos para identificar las propiedades de los objetos e individuos que conforman el dominio. [1]

Luego del estudio de disímiles bibliografías se llega a la conclusión que las *DLs* son una familia de lenguajes de Representación del Conocimiento. Pueden ser usados para representar el conocimiento

[3] Frames (Marcos): Son redes semánticas estructuradas. Conforman una colección de atributos y la descripción de sus características.

[4] Conocimiento Taxonómico: Benjamín Bloons creó esta taxonomía para categorizar el conocimiento. Tiene un orden jerárquico, es decir, va desde el conocimiento más simple al más elaborado o complejo.

terminológico de un dominio de aplicación de una forma estructurada y con una semántica formal bien definida.

En esta investigación es asumida la definición de las *DLs* como una familia de formalismos basados en la lógica para la Representación del Conocimiento. Describen al dominio en función de conceptos, roles e individuos. Proveen servicios de inferencia y cuentan con una semántica formal basada en la lógica. Este concepto abarca las características esenciales de las *DLs* y permite definir de manera detallada sus elementos distintivos.

Las *DLs* se enfocan en representar un dominio determinado a partir de la utilización de conceptos. El elemento que las distingue es que esta descripción la realiza apoyada en una semántica formal basada en la lógica. Estos elementos son ilustrados a continuación:

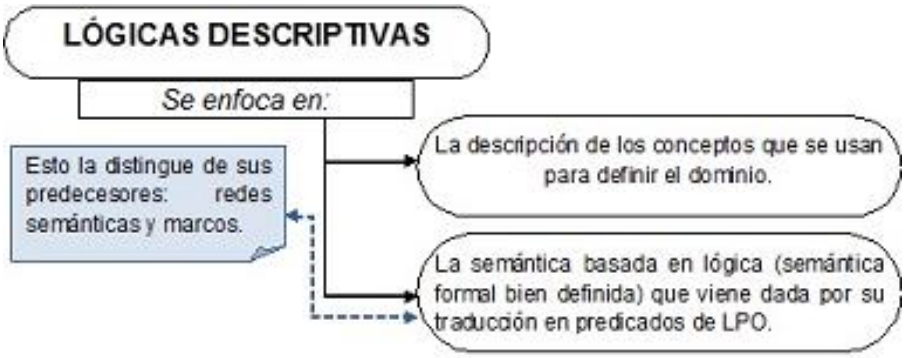


Fig.1: Definición de las DLs.

En la investigación serán empleadas las *DLs* debido a que abarcan un conjunto de lenguajes para la RC. Los mismos resultan una herramienta poderosa para el razonamiento sobre el conocimiento representado porque contienen propiedades que determinan su nivel de expresividad y el propósito con que serán usados. Fundamentalmente se profundizará en los temas asociados a las *DLs* debido a que posibilitan el proceso de RC de una manera clara y alcanzando altos niveles de expresividad. Por otro lado cabe destacar que proveen un conjunto de técnicas para llevar a cabo el razonamiento.

1.2 Base de conocimiento o *Knowledge Base* (*KB*)

Las *KBs*, están diseñadas para almacenar información de una forma legible; usualmente con el fin de

obtener razonamiento deductivo. Contienen una serie de datos, por lo general en forma de reglas que describen el conocimiento de manera lógicamente consistente y pueden ser usadas para razonar sobre él. La investigación está centrada en cómo obtener técnicas de razonamiento más eficientes que logren razonar sobre la información que circula constantemente en tiempo real.

Una *KB* se compone de dos elementos principales: [2]

- Una parte terminológica (*TBox*: *Terminological Component*) donde se encuentran los conceptos; además de los roles y construcciones complejas a partir de éstos. Esta se estructura en forma de jerarquía de conceptos, definiéndose qué conceptos subsumen o son subsumidos por otros.
- Una parte extensional (*ABox*: *Assertions on Concepts*) donde se encuentran las afirmaciones sobre individuos concretos.

La siguiente imagen muestra el conocimiento que se almacena en cada uno de los componentes de las *KBs*:

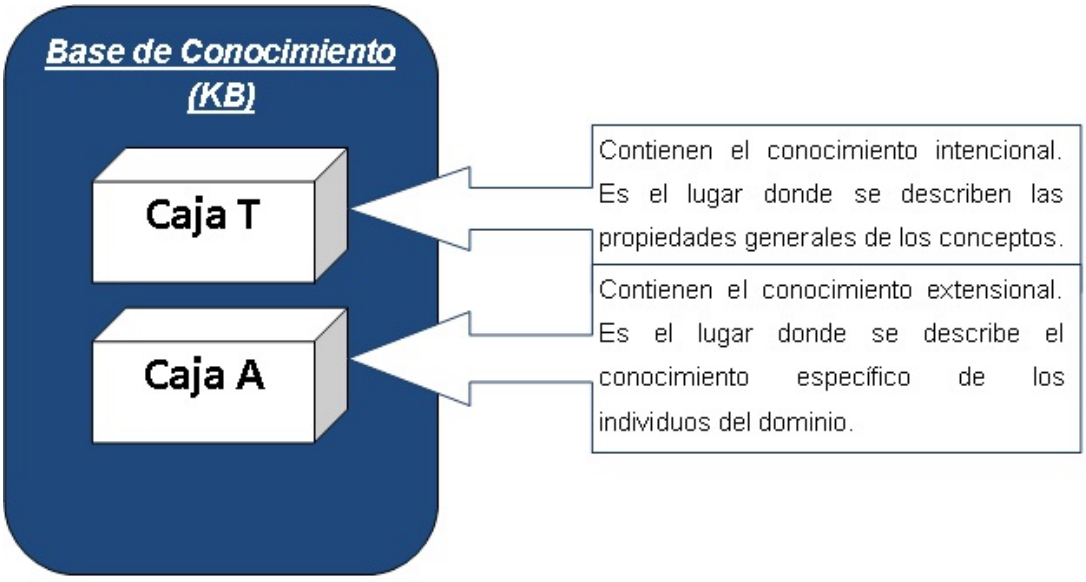


Fig. 2: Estructura de una KB.

Para la construcción de una *KB* se debe tener en cuenta el nivel de expresividad que se quiere lograr y en relación a esto serán determinados los lenguajes que se emplearán. La utilización de las *KBs* reviste una gran importancia y su empleo es vital para las investigaciones asociadas a las *DLs.* Una de sus

aplicaciones fundamentales, y que se relaciona de manera esencial con la investigación, es que constituye la base sobre la cual actúan los razonadores para poder llevar a cabo su proceso de inferencia y por consiguiente lograr deducir el nuevo conocimiento. Razón por la cual las *KBs* deben ser tomadas en cuenta no solo a la hora de la RC sino también en el proceso de razonar sobre él.

1.3 Características de las *DLs*

El empleo de las *DLs* parte tanto desde la definición de las *KBs*; como del razonamiento sobre ellas. Describen diversos niveles de expresividad y tienen cierta complejidad a la hora de inferir nuevo conocimiento. El proceso de razonamiento permite inferir implícitamente sobre conocimiento explícito previamente definido, gracias a la aplicación de estrategias deductivas.

Entre los disímiles componentes asociados a las *DLs*, existen dos características esenciales que las distinguen. Ellas son:

- ✓ Están equipadas de una semántica formal basada en la lógica.
- ✓ Su servicio central es el razonamiento.

Existen otros componentes distintivos que constituyen los elementos fundamentales del lenguaje de las *DLs*, y que permiten el trabajo con el conocimiento:

- ✓ **Un formalismo descriptivo:** Compuesto por conceptos, roles, individuos y constructores.
- ✓ **Un formalismo terminológico:** Compuesto por axiomas terminológicos que introducen descripciones complejas y propiedades de la terminología descriptiva.
- ✓ **Un formalismo asertivo:** Se introducen propiedades de individuos.
- ✓ **Mecanismos de inferencia:** Permiten inferir nuevo conocimiento a partir de un conocimiento dado; incluyendo por tanto, algoritmos de razonamiento que son decidibles.

Esta investigación se centra básicamente en los mecanismos de inferencia que son los orientados al razonamiento sobre el conocimiento. Independientemente de este enfoque, se tendrán en cuenta el resto de los elementos; especialmente en la lógica que sea empleada para representar los conceptos, roles e individuos. El funcionamiento del razonamiento será tan eficaz como expresiva sea la KB sobre la cual actúa.

1.4 Sintaxis de las *DLs*

Las *DLs* cuentan con una sintaxis bien definida que es usada en el momento de la RC. Existen componentes centrales del alfabeto del lenguaje de las *DLs*; los mismos son empleados para estructurar las *KBs* de manera formal. Tanto el conocimiento intencional como el extensional es representado y se desglosa de la manera siguiente: [3]

- ✓ **Conceptos (clases):** Un concepto denota un conjunto de objetos, que corresponden a predicados unarios en LPO. El nombre de concepto (*concept name*), es un nombre que se le asigna a un conjunto de objetos para que puedan ser identificados y que comparten características en común.
- ✓ **Roles (relaciones) o Propiedades:** Los roles se emplean para especificar propiedades o atributos de los objetos. Estos corresponden a los predicados binarios en LPO. El nombre de rol (*rol name*), denota la relación existente entre los objetos.
- ✓ **Objetos, individuos o instancias:** Estos corresponden a las constantes en la LPO. Los individuos son instancias de los conceptos y también se pueden relacionar por medio de un rol.
- ✓ **Constructores (*constructor*):** Permite relacionar nombres de conceptos y nombres de roles. Posibilita la creación de conceptos complejos (*complex concept*) a partir de conceptos atómicos.
- ✓ **Tipos de datos:** Este elemento es opcional y describe el tipo de dato a los cuales pertenecen los objetos.

Cada uno de estos componentes permiten la RC y para este fin pueden ser usados los lenguajes dependiendo del nivel de expresividad que se quiera lograr. Son también conocidos como los 5 componentes básicos de una ontología. Han sido ampliamente utilizados en las investigaciones referentes a las *KBs*, y en general a las que requieran una representación lo más completa posible de un dominio de aplicación determinado.

1.5 Lenguajes de las *DLs*

Los elementos principales que componen las *DLs* son los conceptos y el rol atómico. Es posible la

construcción de conceptos complejos a partir de la utilización de los diferentes constructores que proponen los lenguajes de las *DLs*.

Las *DLs* se basan por lo general en constructores provenientes de las LPO o Lógica de Predicado. Los mismos permiten RC y establecer relaciones entre las entidades; facilitando el proceso de manipulación de la información. Actualmente la búsqueda de nuevos constructores continúa en ascenso, con el objetivo de hacer cada vez más eficiente el trabajo con las *KBs*. Sus aplicaciones son numerosas porque permiten la representación de las más disímiles esferas de la vida y toman cada vez más importancia debido a que constituyen la base de numerosos procesos de razonamiento que se están creando actualmente.

Para poder construir un Sistema basado en *DLs* es posible combinar los distintos lenguajes, lo que determinará la expresividad de la *KB*. Por ejemplo, $\mathcal{SHIQ}$ es un lenguaje que combina $\mathcal{AL}$ y *C* junto con los constructores de Intersección, Restricción de Valores, Concepto Universal, Vacío, Negación, Unión, Restricción Existencial, Herencia de Roles, Rol Inverso y Restricción Numérica Calificada.

La semántica de los constructores en las *DLs* se define mediante una teoría de modelos (*model theory*), basada en la teoría de Zermelo- Frankel. En esta teoría, el dominio de discurso es la parte del mundo que se está modelando, se representa como un conjunto Δ. Los elementos del mundo se interpretan como elementos de Δ, de la siguiente manera:

- ✓ Las instancias son elementos de Δ.
- ✓ Los conceptos son elementos de Δ.
- ✓ Las propiedades son subconjuntos de Δ x Δ.
- ✓ Las relaciones clase-subclase se interpretan como inclusión de conjuntos.

La siguiente tabla resume los distintos lenguajes de los *DLs* con sus constructores, sintaxis y sus nombres correspondientes [4]:

CONSTRUCTOR	SINTAXIS	LENGUAJE			
Concepto	A	FL_0	FL^-	AL	S
Rol	R				
Intersección	C ∩ D				
Valor de Restricción	∀ R.C				
Cuantificador Existencial	∃ R				
Universo o Tope	T				
Fondo o Vacío	⊥				
Negación Atómica	¬ A				
Negación	¬ C			C	
Unión	C ∪ D			U	
Restricción Existencial	∃ R.C			E	
Restricción Numérica	(≥ n R) (≤ n R)			N	
Nominales	{a1,....,an}			O	
Herencia de Roles	R ⊆ S			H	
Rol Inverso	R-			I	
Restricción Numérica Calificada	(≥ n R.C) (≤ n R.C)			Q	

Tabla 1: Constructores de las DLs.

Entre los constructores anteriormente enunciados se pueden seleccionar algunos de los que más frecuentemente son empleados en la actualidad para el proceso de RC. La siguiente tabla recoge los constructores más aplicados actualmente según Ian Horrocks, que es el investigador principal de los temas relacionados a las *DLs*, en la que se ha reflejado la semántica y algunos ejemplos.

Para la semántica propuesta en la tabla, se determina el siguiente dominio de aplicación: $I = (\Delta^I, \cdot^I)$.

CONSTRUCTORES	SINTAXIS	EJEMPLOS	SEMÁNTICA
Concepto Atómico	A	Humano	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Rol Atómico	$\mathcal{R}$	le gusta	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Para C, D *conceptos* y R el nombre del *rol*			
Conjunción	$C \sqcap D$	Humano ⊓ Masculino	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disyunción	$C \sqcup D$	Agradable ⊔ rico	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negación	$\neg C$	¬Came	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Restricción de Existencia	$\exists \mathcal{R}.C$	∃ tieneHijos.Humano	$\{x \mid \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Restricción de Valor	$\forall \mathcal{R}.C$	∀ tieneHijos.Rubio	$\{x \mid \forall y.\langle x,y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
Restricción de Cantidad	$(n \geq \mathcal{R})$	$\geq$ 7 tieneHijo	$\{x \mid \lvert\{y.\langle x,y\rangle\} R^{\mathcal{I}}\rvert \geq n\}$
(? *ALCN*	$(n \leq \mathcal{R})$	$\leq$ 1 tieneMadre	$\{x \mid \lvert\{y.\langle x,y\rangle\} R^{\mathcal{I}}\rvert < n\}$
Rol Inverso	$\mathcal{R}^{-}$	tieneHijo^{-}	$\{\langle x,y\rangle \mid \langle y,x\rangle \in R^{\mathcal{I}}\}$
Rol Transitivo	$\mathcal{R}^{+}$	tieneHijo^{+}	$(R^{\mathcal{I}})^{+}$
Dominio Concreto	$u_1,\ldots,u_n.P$	tienePadre.edad, edad. >	$\{x \mid \langle u_1^{\mathcal{I}}(x),\ldots,u_n^{\mathcal{I}}(x)\rangle \in P\}$

Tabla 2: Ejemplos de los constructores de las DLs.

El nivel de expresividad que se logre en la unión de los lenguajes de las *DLs* estará determinado por los que se quiera expresar y por las necesidades que existan para construir los conceptos que formarán parte del dominio de aplicación en que se trabaje. Investigaciones de autores reconocidos como Ian Horrocks

describen a los sistemas $\mathcal{SH}$ como uno de los más expresivos y que están siendo implementados actualmente en las ontologías. Dentro de este sistema que logra mayores niveles de expresividad se encuentran:

- ✓ $\mathcal{SHIQ}$: es $\mathcal{ALCQI}$ + roles transitivos + inclusión de roles.
- ✓ $\mathcal{SHOIQ}$: es $\mathcal{SHIQ}$ + nominales.
- ✓ $\mathcal{SHOIN}$ $(\mathcal{D})$: es $\mathcal{ALCIN}$ + nominales + dominios concretos.

Este elemento se tendrá en cuenta a la hora de enfocarse en un razonador con el objetivo de lograr altos niveles de expresividad en el proceso de razonamiento y un número elevado de constructores que faciliten la construcción de conceptos complejos en las ontologías.

1.6 Servicios de Inferencia

Los servicios de inferencia que son empleados por las *DLs* constituyen procedimientos de decisión sólidos y completos para problemas claves. Una de las características de las *DLs* es que su razonamiento es decidible y completo; debido a dos elementos claves: todas las inferencias terminan en un tiempo finito y se obtienen todos los axiomas implícitos.

Siendo $\mathcal{T}$ un *TBox*, e $\mathcal{I}$ una función de interpretación en un conjunto no vacío del domino del discurso, las tareas de inferencia más comunes que se llevarán a cabo son: [5]

- ✓ **La Subsunción:** Esta tarea de inferencia prueba formalmente que un concepto es más específico que otro.

  De manera analítica un concepto $\mathcal{C}$ se subsume a otro concepto $\mathcal{D}$ con respecto a $\mathcal{T}$ si se cumple que $\mathcal{C}^{\mathcal{I}} \subseteq \mathcal{D}^{\mathcal{I}}$ para todo modelo $\mathcal{I}$ de $\mathcal{T}$. En este caso se escribe $\mathcal{T} \models \mathcal{C} \subseteq \mathcal{D}$.

- ✓ **La Satisfactibilidad:** Se enfoca en determinar si una *KB* es satisfactible o no, para esto verifica que no contenga contradicciones.

  De manera analítica un concepto $\mathcal{C}$ es satisfactible respecto a $\mathcal{T}$ si existe un modelo $\mathcal{I}$ de $\mathcal{T}$ tal que $\mathcal{C}^{\mathcal{I}}$ es no vacío. En este caso se dice también que $\mathcal{I}$ es un modelo de $\mathcal{C}$.

Existen otros dos servicios de inferencia que aunque no son los fundamentales, pueden ser usados si resultan necesarios. Ellos son:

- ✓ **Equivalencia:** Este servicio de inferencia determina si dos conceptos C y D son equivalentes respecto a T si $C^I = D^I$ para todo modelo I de T. En este caso se escribe $T \models C \equiv D$.
- ✓ **Disyunción:** Este servicio de inferencia determina si dos conceptos C y D son disjuntos respecto a T si $C^I \sqcap D^I = \emptyset$ para todo modelo I de T.

Es válido aclarar que si la *DL* permite negación e intersección de conceptos, las tareas de subsunción, equivalencia y disyunción pueden reducirse a satisfactibilidad. A la hora de realizar el razonamiento es posible calcular la complejidad del razonamiento, el cual sería equivalente a la cantidad de operaciones requeridas para resolver un problema determinado.

Estos servicios de inferencia constituyen elementos claves para el trabajo con el conocimiento y permiten comprobar la consistencia de las *KBs*. Su importancia es fundamental porque evitan las inconsistencias del conocimiento representado y por consiguiente validan que los procesos de razonamiento que se lleven a cabo obtengan soluciones óptimas.

1.7 Arquitectura de los Sistemas basados en *DLs*

En la actualidad se han implementado razonadores semánticos basados en las *DLs*. El mayor problema de los mismos es que si el lenguaje es muy expresivo, su tratamiento puede tener una complejidad computacional muy alta e incluso puede ser no decidible.

Las Sistemas Basados en Lógica Descriptiva (SBLD) se caracterizan por tener una *KB*. En el componente *TBox* se indicará la terminología del dominio; mientras la *ABox* contendrá las aserciones acerca de los individuos particulares del dominio con base en el vocabulario que ha sido empleado de acuerdo a la terminología.

Para la construcción de estos sistemas se emplea un lenguaje descriptivo específico, elemento que caracteriza un SBDL de cualquier otro. La unión de los lenguajes que sean empleados determinará el nivel de expresividad de su *KB*. Además estos sistemas ofrecen tareas de razonamiento, que indican si una

aserción se satisface (no es contradictoria) o una es más general (si se subsume) que otra.

El objetivo fundamental de esta investigación es la mejora de las herramientas empleadas para realizar las tareas de razonamiento sobre las *KBs*. Específicamente en las que forman parte de las *DLs* para llevar a cabo los procesos de inferencia. A continuación se representan los elementos por los que está compuesto un SBLD:

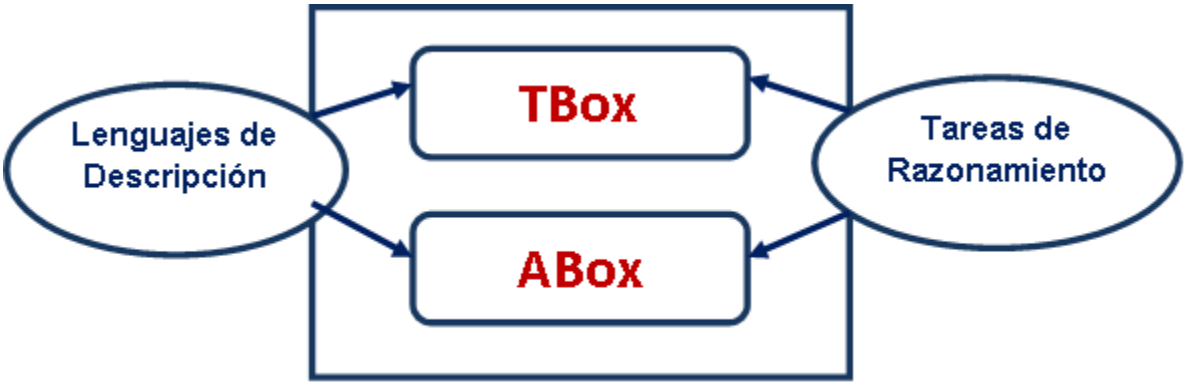


Fig. 3: Componentes de un SBDL.

Los sistemas de razonamiento son un conjunto de axiomas y reglas de inferencia, unido a una serie de propiedades como la:

- **Expresividad:** Capacidad de representar un problema.
- **Consistencia:** Todo lo que se deduce es correcto.
- **Completitud:** Todo lo que es correcto puede deducirse.
- **Decidibilidad:** Existe un algoritmo para decidir si se cumple una conclusión.
- **Tratabilidad:** El algoritmo de decisión tiene una complejidad razonable.

El enfoque principal está dirigido a las tareas de razonamiento que aplica un SBLD. Hasta el momento se ha alcanzado un desarrollo medianamente eficiente en el proceso de inferencia sobre la información almacenada de manera estática. Es por ello que la investigación está orientada a lograr la aplicación de las tareas de inferencias en el proceso de *stream reasoning*.

1.8 Proceso de *stream reasoning*

El proceso de *stream reasoning* se ha convertido en un tema de interés en las investigaciones referentes a las base de datos. Por otra parte las corrientes de datos adquieren una importancia creciente, siendo la base para los procesos que están a un nivel superior para la toma de decisiones y que requieren un razonamiento complejo. Esto implica que sea necesario inferir sobre enormes cantidades de datos en tiempo real con el objetivo de apoyar los numerosos procesos de decisión concurrentes que a diario se generan.

En la actualidad la información está cada vez más accesible en la Web Semántica, brindando la posibilidad de ser empleada para darle respuesta a disímiles problemas de la vida cotidiana; aunque resultando insuficientes ya que no existen sistemas de software que tengan la capacidad de computar las respuestas de una manera eficiente y en tiempo real. La razón de esto es muy sencilla: lograr responder consultas de este tipo requiere de sistemas capaces de manejar rápidamente el plano semántico de un mundo tan cambiante. Hasta hoy estos datos pueden ser analizados sobre sistemas especializados en la gestión de flujos de datos, pero no pueden hacer tareas complejas de razonamiento y carecen de un protocolo para publicar totalmente la información y facilitar el acceso a los datos que cambian vertiginosamente.

1.9 *Stream reasoning*

Por la imposibilidad de establecer un concepto exacto en español de *stream reasoning*, se ofrece el siguiente concepto en inglés:

Stream reasoning: *Logical reasoning in real time on gigantic and inevitably noisy data streams in order to support the decision process of extremely large numbers of concurrent users.* [6]

De manera general puede ser definido como un razonamiento lógico sobre grandes corrientes de datos que circulan en tiempo real. Su objetivo cardinal es apoyar los procesos de decisión del número elevado de usuarios concurrentes en la web.

Este tema resulta vital en el desarrollo de la investigación y de manera general en la vida diaria. El manejo de corrientes de datos en tiempo real es un tema que está en ascenso y que forma parte de la mayoría de los procesos que surgen en la actualidad. La necesidad de lograr la aplicación de mecanismos que sean

capaces de procesar toda esta información, es el objetivo fundamental de la presente investigación.

1.10 *Estado del arte en el stream reasoning*

Las investigaciones que se han efectuado de los temas referentes al *stream reasoning*, se orientan a la lógica temporal y a la revisión de afirmaciones. Estas herramientas son adecuadas cuando los datos tienen un bajo índice de cambios de volumen y frecuencia.

De igual manera, el problema de modificar vocabularios y la evolución de ontologías ha sido objeto de exhaustiva búsqueda. Pero en este caso la práctica estándar se basa en técnicas de administración de la configuración tomada de la Ingeniería de Software, tales como el vocabulario y el control de versiones de la *ontología*[5].

Por otra parte la arquitectura típica de la Web Semántica, que almacena la información, no puede ser aplicada a los datos que cambia tan velozmente; debido a que serían obsoletos en el momento de la consulta. Haciendo necesario un enfoque totalmente diferente.

El *stream reasoning* es un área de investigación inexplorada aún pero con un alto impacto. Es un nuevo enfoque multidisciplinario que puede proporcionar las abstracciones, métodos y herramientas necesarias para integrar los flujos de datos, la Web Semántica y los sistemas de razonamiento; proporcionando así una manera de responder preguntas de manera eficiente.

No existen investigaciones que aporten muchos más elemento en el trabajo con el *stream reasoning*, esta razón convierte este tema en novedoso y poco desarrollado. Por otro lado, su gran impacto hace que cada día aumenten los interesados en adentrarse en este contenido y aportar nuevas soluciones.

1.11 *Data streams*

Por la imposibilidad de establecer un concepto exacto en español de las corrientes de datos, es ofrecido a continuación en inglés:

Data streams: *Is a continuous flow of large data sequences that go from one place to another in short periods of time. Furthermore, these data sequences are used to transmit or receive data and information*

[5]Ontología: “Una ontología es una especificación explícita de una conceptualización”. Gruber (1993)

found on streaming. [7]

La traducción más aceptada de *data streams* es: corrientes de datos en tiempo real; aunque en la investigación será tratado el término en inglés debido a que no se logra la exactitud a la hora de llevarlo al español. Para definir un concepto aproximado en español de este término se puede decir que las *data streams* son un flujo constante de grandes cantidades de datos transferidos a una alta velocidad y que viajan de un lado a otro en períodos cortos de tiempo. Son capaces de soportar aplicaciones como televisión de alta definición o la copia de seguridad continua a un medio de almacenamiento.

En el proceso de *stream reasoning* este es un elemento fundamental ya que sobre él se efectúa el procesamiento. Más adelante serán tratadas estas corrientes para conformar el pseudocódigo, que responderá al proceso de *stream reasoning*. En la siguiente imagen se ilustra una *data streams*:

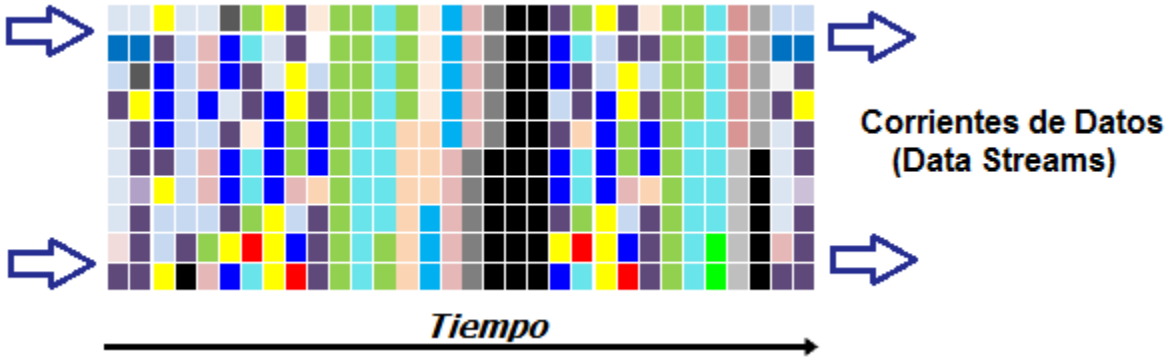


Fig. 4: Data Streams.

1.12 Características del procesamiento

El proceso de *stream reasoning* puede caracterizarse de acuerdo a tres conceptos claves en cuanto a su procesamiento:

- *Streams* (corrientes): Las corrientes de datos son secuencias ilimitadas de datos. Sus elementos van variando, para formar una corriente continua de información. Dado que *RDF* es el formato de intercambio de datos para razonadores, estos datos se definen como tripletas *RDF* y sus marcas de tiempo T_i. Quedando representado de la siguiente manera:

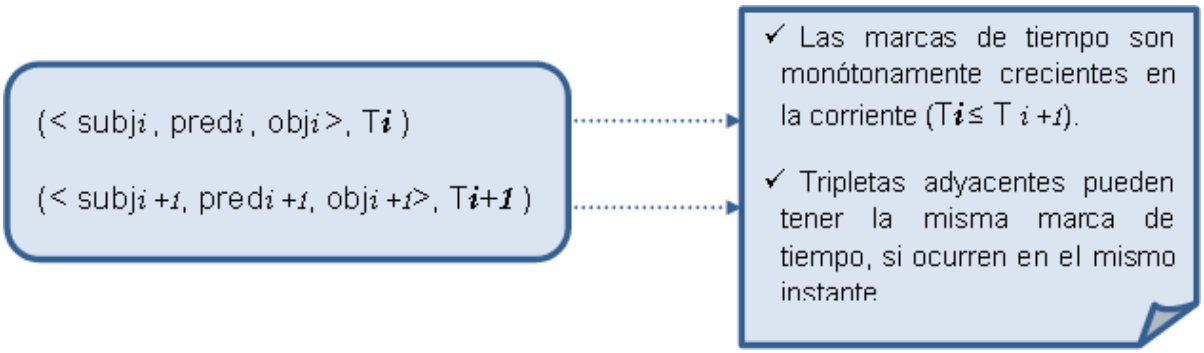


Fig. 5: Tripletas RDF.

- *Windows* (Ventanas de tiempo): En los problemas tradicionales de razonamiento, debemos tener en cuenta toda la información disponible. Sin embargo en el razonamiento sobre corrientes se restringe el proceso a la elaboración de una cierta ventana de tiempo. Es decir, se centra en el subconjunto de las recientes declaraciones de la corriente y haciendo caso omiso a las declaraciones anteriores. Aunque de cualquier manera, el efecto acumulativo de las últimas ventanas (procesadas en el pasado) y de las ventanas presentes, pueden ser tenidos en cuenta. Esta característica es notablemente importante ya que permite ahorrar recursos de cómputo en términos de memoria y realizar el razonamiento en muy cortos períodos de tiempo.
- *Continuous Processing* (Procesamiento continuo): Los enfoques tradicionales de razonamiento tiene bien definidos los comienzos y finales de las tareas de inferencia. Indicando cuándo las tareas se presentan al razonador y cuándo los resultados se obtienen. Las corrientes de razonamiento de este modelo de procesamiento son movidas a un modelo continuo, donde las tareas son registradas y evaluadas de forma continua con los datos que fluyen.

Estos tres conceptos enfocan el manejo del conocimiento a la hora de procesarlo. Para esto habrá que enmarcar bien cuál es la información que es realmente importante para ser procesada, definiendo el inicio y el final de las tareas. Las tripletas *RDF* serán empleadas a la hora del procesamiento, aunque el trabajo se va a centrar en el formato *OWL*. Para elaborar un pseudocódigo que responda al proceso de *stream reasoning* resulta vital tener estos conceptos bien definidos para poder ser aplicados de manera eficaz.

1.13 Ejemplos concretos del *stream reasoning*

El *stream reasoning* puede beneficiar numerosas áreas de la vida común. La monitorización del tráfico y la detección de patrón de tráfico son zonas de aplicación. Este tema está tomando cada vez más fuerza y numerosos investigadores se están implicando en investigaciones, condicionando que la comunidad de la Web Semántica desarrolle nuevos proyectos que se enfoquen en estas áreas. Las auditorias a las transacciones financieras, la monitorización de las plantas de energía eólica y los sistemas de monitoreo de pacientes; son sólo algunos ejemplos que pudieran enunciarse en este sentido.

Si los investigadores dirigieran más su atención al *stream reasoning* serían capaces de crear métodos y herramientas aptas de darle una respuesta rápida y eficiente a las consultas realizadas. Propiciando el desarrollo de razonadores y por consiguiente la mejora de los procesos de razonamiento.

Como aplicaciones concretas para el proceso de *stream reasoning*, se pueden mencionar:

- **Razonamiento para aplicaciones móviles:** La movilidad es una de las características definitorias de la vida moderna. Esta tecnología puede acompañarnos a cualquier lugar y apoyar tanto un asunto de negocio como un tema de entretenimiento.

  Los teléfonos móviles son profusamente empleados, proporcionando un amplio espectro de aplicación al concepto de *stream reasoning*. Para poder ser introducidos en nuestra vida cotidiana, sus aplicaciones deben cumplir con requisitos en tiempo real sobre todo si vamos a utilizarlos para realizar decisiones inmediatas.

  Manejando los datos procedentes de los sensores, que muy probablemente provienen en secuencias, las aplicaciones móviles deben ser capaces de resolver problemas *stream reasoning*; haciendo frente a datos imprecisos, ocupándose de posibles errores y moviendo numeroso cálculos de razonamiento para un servidor por la imposibilidad de realizarlos desde el dispositivo.

  Primeramente estas aplicaciones deben conocer qué parte de la transmisión es relevante y cuál es su significado. Por ejemplo tienen que extraer la información cuantitativa como la latitud y la longitud del hogar, del gimnasio o de la oficina. Luego deben elevar el nivel de abstracción para razonar sobre problemas concretos como la forma de llegar a esos lugares, los medios de transporte, la ruta que evite atascos de tráfico. Si se lograra implementar métodos de razonamiento eficientes, las aplicaciones móviles podrían proporcionar una comprensión real del mundo y su

estructura.

- **Control de riesgo de Salud Pública:** La detección temprana de una amenaza potencial para eventos de salud pública, tales como brotes y epidemias, es una de los principales objetivos para los organismos nacionales e internacionales relacionados con la salud. Tratar este tema es de prioridad principal, para la detección de epidemias en tiempo real. Algunos ejemplos recientes son las infecciones como el *SRAS*, la gripe aviar *H5N1*, y el virus *H1N1*.

  Esto requeriría de un enfoque integrado en la salud pública, con una plataforma de detección de eventos que supervisarían una gran cantidad de datos para detectar eventos que constituyan una amenaza potencial para la salud. Esta plataforma dinámica tiene que identificar, integrar e interpretar una secuencia de datos heterogéneos, distribuidos, con la información que fluye de estas fuentes de datos. Uno de los ejemplos de los sistemas existentes es *FluTrends* de *Google*. El reto es cambiar de los sistemas hechos a mano al razonamiento automático, de manera que se pueda agilizar y perfeccionar el proceso con mecanismos eficientes para la inferencia.

CONCLUSIONES DEL CAPÍTULO

Luego de realizar un estudio detallado del estado del arte de los temas relacionados a las *DLs* y al *stream reasoning;* se derivan las conclusiones siguientes:

- Las *DLs* cuentan con razonadores capaces de inferir nuevo conocimiento a partir del conocimiento representado.
- Los razonadores de las *DLs* deben ser estudiados y analizados para seleccionar el más factible para ser aplicado en el proceso de *stream reasoning*.
- Los conceptos de *windows, continuous processing* y *streams*; deben tenerse en cuenta a la hora de llevar a cabo el proceso de *stream reasoning*.

Capítulo 2

Razonadores de las DLs

INTRODUCCIÓN

Empleando las *KB* y los motores de inferencia; los razonadores semánticos hacen posible la tarea de inferir el nuevo conocimiento. La necesidad de implementar mejoras en los razonadores crece ascendentemente debido al desarrollo y el nivel de expresividad que han alcanzado los lenguajes empleados para la RC en la Web. En este capítulo son estudiados y analizados definiendo criterios importantes que los identifiquen y que permitan realizar una comparación desde diferentes puntos de vista; para luego proponer mejoras en el proceso de *stream reasoning*.

2.1 Razonadores Semánticos

Un razonador semántico pude ser definido como: “Un comprobador de coherencia *OWL*, tiene un documento como entrada y devuelve una palabra: coherente, inconsistente, o desconocido”. [8]

Un razonador es también conocido como motor de razonamiento o motor de reglas; algunos investigadores lo describen como una pieza de software capaz de deducir las consecuencias lógicas de un conjunto de axiomas. Realiza su funcionamiento como un motor de inferencia, proporcionando un conjunto de mecanismos para trabajar con reglas de inferencia que son definidos por medio de lenguajes de ontologías y con frecuencia en lenguajes de descripción.

De manera general son aplicaciones informáticas que permiten generar conocimiento y hacer inferencias a partir de un conjunto de axiomas y hechos, permitiendo comprobar la coherencia de una ontología. Emplean un motor de inferencia y un conjunto de reglas expresadas en lenguajes semánticos como *OWL*.

La expresividad alcanzada por los lenguajes como *RDFS*, *RDF* y *OWL* ha limitado en gran medida el nivel de expresividad de los razonadores semánticos.

En el proceso de inferencia de la Web Semántica; la definición de reglas desempeña un papel importante, debido a que a través de ellas se expresa el comportamiento de un individuo dentro de un dominio. La buena definición de estas reglas, o axiomas como también se nombran, va a definir el éxito de la generación de nuevo conocimiento.

Los razonadores semánticos emplean dos elementos fundamentales para inferir el nuevo conocimiento:

- ✓ ***KB*:** Es similar a una base de datos donde se gestiona conocimiento, y tiene los medios necesarios para que esta información sea computarizada, mediante un conjunto de aserciones y reglas. Su función es dar al motor de inferencia información sobre un problema determinado.
- ✓ **Un motor de inferencia:** Permite al agente obtener conclusiones de la *KB* y resolver un problema concreto, de la misma forma se obtienen nuevas sentencias a partir de sentencias ya existentes, así por ejemplo; es más factible determinar si una sentencia es deducible a partir de sentencias de su *KB*. Es importante recordar que un lenguaje lógico es un lenguaje formal para representar información de la que se deduce conclusiones y dispone de una sintaxis y una semántica.

Analizando el concepto dado previamente; para que un motor de inferencia funcione necesita obtener información de la *KB*; y parte de sentencias nuevas, basadas en el conocimiento ya existente, para hacer más fácil la deducción. Como parte de los razonadores semánticos se encuentran los razonadores lógicos. Estos pueden clasificarse en dos grupos, como se muestra en la siguiente imagen:

Fig. 6: Clasificaciones de los Razonadores Lógicos.

La presente investigación está enfocada en los razonadores de las *DLs* y es por ellos que sólo será analizado este grupo en particular. Los cuales han sido aplicados a numerosos problemas de razonamiento pero no existe ninguno que sea capaz de llevar a cabo el proceso de *stream reasoning*.

2.2 Razonadores de la *DLs*

Como anteriormente fue abordado; entre los razonadores lógicos que actualmente son empleados, se encuentran los razonadores de la *DLs*. Estos se distinguen de otros razonadores por el hecho de emplear los lenguajes de las *DLs* en sus tareas de razonamiento.

Existen dos tipos de razonamiento de las *DLs*: *TBox* y *ABox*. Cada uno está orientado a un fin específico, aunque el razonamiento sobre la *ABox* es el método más completo de recuperación de instancias y consultas: [9]

En la siguiente imagen se define el fin de cada tipo de razonamiento:

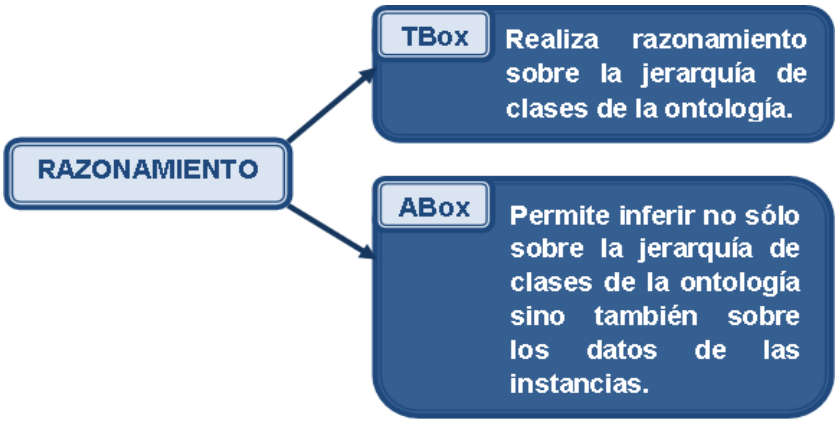


Fig. 7: Tipos de razonamiento que llevan a cabo las DLs.

Los razonadores de las *DLs* deben ser capaces de brindar los servicios de inferencia que a continuación son enunciados:

- ✓ **Chequear la consistencia de una ontología:** El razonador debe ser capaz de asegurar que la ontología en cuestión no contiene hechos contradictorios.

- ✓ **Satisfactibilidad de los conceptos de la ontología:** El razonador determina si es posible que una clase tenga instancias. En el caso de que un concepto sea insatisfactible, si un objeto se define implícita o explícitamente instancia de esta clase, entonces la ontología será inconsistente.
- ✓ **Clasificación de la ontología:** El razonador computa a partir de los axiomas declarados en la *TBox* de la ontología las relaciones de subclase entre todos los conceptos declarados explícitamente (tienen una *URI*[6]) para construir la jerarquía de clases. Esta jerarquía se puede utilizar para formular *queries* como: todas las subclases de un concepto, inferir nuevas subclases de un concepto, las superclases directas.
- ✓ **Instanciación de los conceptos de la jerarquía:** Un razonador *DL* puede inferir cuáles son las clases a las que directamente pertenece. Si además se utiliza la jerarquía inferida mediante el servicio de clasificación anterior, es posible obtener todas las clases a las que indirectamente pertenece una instancia dentro de la ontología.

2.3 Técnicas de razonamiento

Actualmente existen lenguajes ontológicos para la Web, destacándose *OWL*; y los razonadores semánticos como: *FaCT++, Pellet, RacerPro*. Surgido todos por la necesidad de introducir información para ser interpretada por agentes inteligentes en la Web Semántica, con el objetivo de incrementar su *KB* y realizar inferencias que permitan apoyar los procesos que actualmente los usuarios efectúan de forma manual. Como parte de la teoría de los razonadores, se encuentran los algoritmos razonamiento; caracterizados a continuación:

- **Basado en reglas (*Rule-based*)**

Por lo general el conocimiento es representado en forma declarativa y estática (como un conjunto de cosas que son verdaderas). Sin embrago estos algoritmos de razonamiento emplean un sistema basado en reglas que es el que representa el conocimiento en términos de un conjunto de reglas que indican lo que debe hacer o lo que podría concluirse en diferentes ocasiones. Estos sistemas se distinguen por

[6] *URI* (*Uniform Resource Identifier*): Sirve para identificar los recursos de *Internet*. Se diferencia de un *URL* en que permite incluir en la dirección una subdirección, determinada por el "fragmento".

contar con un conjunto de reglas *if-then*, de hechos y de un intérprete que controla las aplicaciones de estas reglas, dados los hechos.

Los sistemas basados en reglas pueden ser clasificados en dos grandes grupos:

- ✓ **Sistemas de encadenamiento hacia adelante:** Estos sistemas están dirigidos por datos. El primer paso es comenzar con los hechos y luego ir utilizando las reglas para sacar nuevas conclusiones, dados los hechos.
- ✓ **Sistemas de encadenamiento hacia atrás:** Están dirigidos por objetivos y comienzan con las hipótesis (o metas) que se está tratando de probar. Luego se van buscando las reglas que permitan concluir la hipótesis y establecer nuevos sub-objetivos, para poder ir realizando la demostración de que se cumple la hipótesis.

- ***First-order prover***

Este algoritmo es soportado sólo por algunos razonadores y se basa en *FOL* (*First Lenguage Order*- Lenguajes de Primer Orden). La LPO permite hacer cuantificación sobre los objetos de un dominio y representar propiedades a través de relaciones y funciones. Los *FOL* extienden la lógica de predicados o lógica proposicional y están compuestos por fórmulas que pueden contener los siguientes elementos:

1).¬, **∧, ∨, →, ↔:** conectivos lógicos usuales.

2).=: un predicado especial para la igualdad.

3).**∃, ∀:** cuantificadores existencial y universal.

4).x1, x2,...: conjunto enumerable de variables, que informalmente puede llamarse x, z, u y son llamados términos.

La mayor debilidad en *FOL* es resolver si es resoluble un problema determinado. Esto se puede demostrar si hay algún proceso computacional que resuelve el problema en un número finito de pasos. En el caso de *FOL*, está demostrado que el problema de averiguar si una fórmula se deduce de una *KB* es irresoluble. Sin embargo, existen razonadores para *FOL* como *Vampiro* o *E-SETHEO* que se basan en la búsqueda de incongruencias en una *KB*, utilizando procedimientos no completos; por lo que es imposible asegurar

que si el razonador no encuentra incongruencias no las haya.

- **Tablas semánticas (*Tableaux*)**

Los razonadores soportan métodos automáticos de deducción como *Tableaux*. Las tablas semánticas son un método de demostración por refutación.

El principio de refutación enuncia lo siguiente: Dado un conjunto de premisas Φ y una conclusión φ, mostrar que el conjunto Φ U {¬ φ} no tiene un modelo que lo satisfaga, demostrando que φ es consecuencia lógica de Φ.
Este método se basa más en la semántica que en la sintaxis de las fórmulas, aunque se requiere de una clasificación de las fórmulas basada en la sintaxis. De acuerdo con su tipo, una fórmula genera una extensión en el *Tableuax*. Las clasificaciones para las fórmulas son:

- ✓ **Tipo α:** las fórmulas conjuntivas.
- ✓ **Tipo β:** las fórmulas disyuntivas.
- ✓ **Tipo γ:** las fórmulas universales.
- ✓ **Tipo δ:** las fórmulas existenciales.

- **Métodos de resolución (*Resolution*)**

El método de resolución es una regla de inferencia que toma dos cláusulas y produce una tercera que es consecuencia de ellas. Identifica y elimina la cláusula complementaria de esas dos cláusulas y combina las otras literales para formar una cláusula nueva. Se requiere que una fórmula esté en su forma clausular equivalente.

En *Resolution*, una lista de cláusulas es inconsistente si se puede derivar la cláusula vacía de la misma. Para demostrar que una fórmula se puede derivar de un conjunto de sentencias, se niega la conclusión y se agrega al conjunto de sentencias que conforman la premisa; y, si a partir de estos argumentos es posible derivar la cláusula vacía, se dice que la premisa deriva a la sentencia de la conclusión.

- **Lógica para los datos (*Datalog*)**

Los razonadores semánticos también dan soporte a la lógica para los datos. *Datalog* se deriva de las LPO y fue inicialmente desarrollado para bases de datos deductivas. *Datalog* significa "lógica para los datos" y puede ser considerado como un lenguaje de descripción y de manipulación de bases de datos. El modelo de descripción de datos sostenido por *Datalog* es básicamente relacional, una relación se ve como un predicado de la lógica. El lenguaje de manipulación es un lenguaje de reglas construido a partir de las *Cláusulas de Horn*[7].

2.4 Historia de los razonadores lógicos

El nombre de *DLs* se enuncia por primera vez en la década de 1980. Anteriormente se nombraban "sistemas terminológicos" y luego "lenguaje de conceptos". En la actualidad las *DLs* se han convertido en un componente fundamental para la Web Semántica en el diseño de ontologías.

A continuación se ofrece una cronología de los diferentes sistemas basados en las *DLs* que se han creado. Los que se mostrarán en esta investigación no son los únicos existentes, actualmente se continúan desarrollando nuevos razonadores, pero sí resultan los más desarrollados hasta el momento:

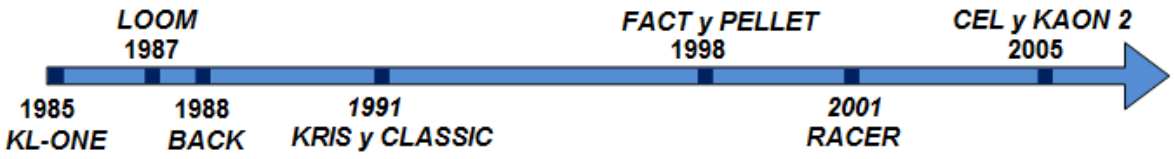


Fig. 8: Evolución de los razonadores de las DLs.

La necesidad de desarrollar sistemas cada vez más capaces de realizar un mejor razonamiento sobre el conocimiento representado, ha exigido que numerosos investigadores dirijan sus esfuerzos en este sentido. Sistemas como *MSPASS*, *Hermit*, *QuOnto*; son otros mecanismos de razonamiento que se han implementado para realizar razonamiento basándose en las *DLs* pero no cuentan con las suficiente

[7] Cláusulas de *Horn*: Es una regla de inferencia lógica con una serie de premisas (cero, una o más), y un único consecuente. Es aquella que tiene como máximo un literal positivo.

eficiencia en el proceso de inferir. Para esta investigación serán tomados en cuenta los razonadores de las *DLs* que más se han desarrollados, con el fin de aplicar uno de ellos al proceso de *stream reasoning* y lograr eficiencia y escalabilidad.

2.5 Razonadores Lógicos

A continuación serán abordados algunos de los razonadores que forman parte de las *DLs*. Actualmente se está trabajando en la creación de nuevos razonadores cada vez más potentes y que sean capaces de responder las preguntas realizadas a las *KBs*; de una manera eficiente y en períodos cortos de tiempo. Por su importancia serán abordados razonadores como: *Kaon2*, *Pellet*, *RacerPro*, *FACT* ++, que figuran entre los más empleados a nivel mundial. También se enunciarán características relacionadas a los menos desarrollados como: *CEL*, *MSPASS*, *IRIS*; aunque los mismos no serán tomados en cuenta a la hora del análisis y selección del razonador para ser aplicado al proceso de *stream reasoning*.

- ***Pellet***

Es un razonador *open source* para *OWL-DL* (*Ontology Web Language-Description Logic*) construido en *Java*. *Pellet* soporta toda la expresividad de las *DLs*, incluyendo el razonamiento con nominales que son las clases definidas por extensión. De igual manera soporta el razonamiento sobre las clases enumeradas. Implementa estrategias de *TBox*, absorción y ramificación semántica.

El núcleo de este razonador está basado en el algoritmo de *Tableaux,* desarrollado para lógicas expresivas potentes, cuyo objetivo es validar las ontologías. Recientemente, *Pellet* ha sido extendido para soportar las nuevas características de un lenguaje web basado en la lógica $\mathcal{SROIQ}(\mathcal{D})$, *que* incrementa la expresividad *OWL-DL*.

Su razonamiento individual, de tipos de datos (*datatype*) y la optimización de las respuestas a las consultas de *ABox*, hacen que sea adecuado para aplicaciones web semánticas de sonido. Pellet es el razonador predeterminado de *Swoop*[8]. Este razonador implementa las mejores técnicas de optimización,

[8] *Swoop*: Es un navegador y editor ontológico.

lo que hace que su desempeño sea bueno, en especial cuando debe evaluar ontologías con mayor complejidad y expresividad.

- ✓ **Características:**
  - Es capaz de analizar y reparar ontologías incorporando un conjunto de heurísticas para detectar y corregir automáticamente las ontologías *OWL*, para asegurarse que son OWL-*DL*.
  - Razona con tipos de datos definidos en el estándar *XML Shema*[9].
  - Valida "especies" para garantizar que los individuos se clasifiquen de modo correcto en la taxonomía de conceptos de la ontología.
  - Tiene un razonamiento con múltiples ontologías.
- ✓ **Arquitectura:** [10]

La parte más importante del sistema es el razonador *Tableaux* que comprueba la consistencia de la *KB*. Está acoplado con *Oracle datatype* que comprueba la consistencia de conjunciones para *XML Shema*.

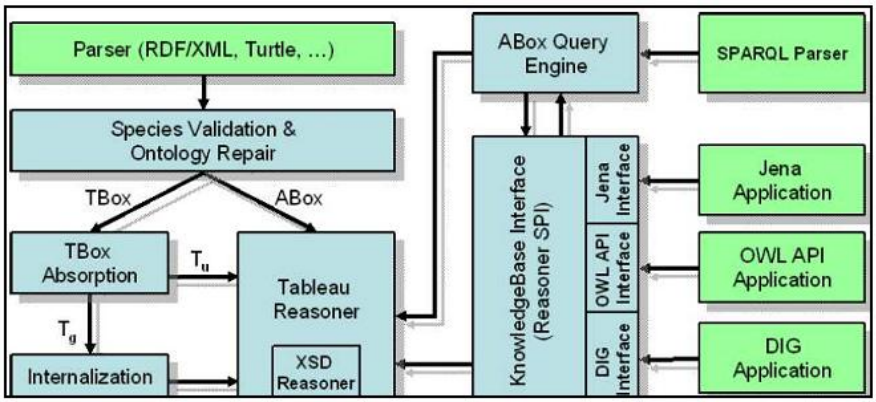


Fig. 9: Arquitectura de Pellet. [10]

[9] *XML Shema*: Es un lenguaje para proporcionar y restringir la estructura y el contenido de los elementos contenidos dentro de documentos *XML*.

- ***Kaon2***

El *Kaon2* es una infraestructura para manejar a *OWL-DL*, *SWRL (Semantic Web Rule Language)*, y ontologías lógicas. Es un razonador semántico, que permite realizar la manipulación y ejecución de inferencias con ontologías representadas en *OWL-DL*, *SWRL* y *F-logic (marco lógico)*; a diferencia de otros razonadores que trabajan con las *DLs* no implementa cálculo de tablas semánticas como base para su proceso de razonamiento; sino que utiliza una serie de algoritmos que reducen una *KB* $\mathcal{SHIQ}(\mathcal{D})$ a un programa *Datalog* disjunto lo que aumenta de forma considerable el rendimiento del razonador a la hora de ejecutar los procesos de inferencia.

✓ **Características:**

- Ofrece una librería de programación para la manipulación ontología en *OWL-DL* y *F-logic*.
- Resuelve consultas conjuntivas que están expresadas en el lenguaje *SPARQL*[10].[12]
- Posee una interfaz *DIG*[11] que permite el acceso a las funcionalidades de *Kaon2* y de otras herramientas, tales como *Protegé*[12]. [13]
- Posee un módulo para extraer datos de base de datos relacionales, un acceso que provee servidor autónomo para las ontologías de manera distribuida y un motor de inferencia para las averiguaciones conjuntivas de contestación.

✓ **Arquitectura:**

Toda la suite de herramientas de *Kaon2* está implementada en *Java*. La arquitectura de este razonador es flexible y escalable; organizándose en estratos de la manera siguiente: [14]

1. Los adaptadores para almacenamiento de la ontología.
2. El software personalizado para los servicios ofrecidos por la suite de herramientas.

[10] *SPARQL*: Lenguaje de recuperación basado en *RDF*.
[11] *DIG*: Estándar o protocolo que se introduce para proporcionar una interfaz común para los razonadores *DL*.
[12] *Protegé*: Es una herramienta de sistemas basados en el conocimiento y para el desarrollo de ontologías, para la solución de problemas y toma de decisiones en dominios particulares.

3. Las aplicaciones del cliente que acceden al software personalizado de la ontología y ofrecen aplicaciones a los usuarios finales.

De esta manera la arquitectura de este sistema *DL* puede representarse de la manera siguiente:

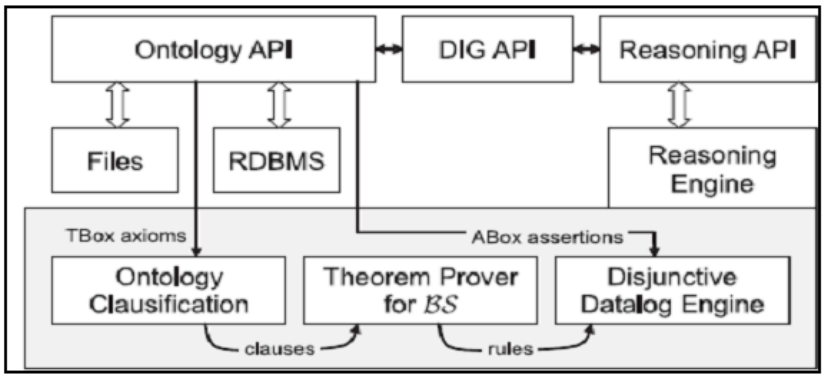


Fig. 10: Arquitectura de Kaon2. [15]

- ✓ **Modelo de Conocimiento:**

El modelo de conocimiento trabaja con la suite de la herramienta *Kaon* (Lenguaje *Kaon*) basado en una extensión de *RDF(S)*. Con *Kaon2* se modela ontologías con los conceptos, las propiedades (para expresar relaciones y atributos), las instancias de conceptos y de propiedades.

Los conceptos son organizados en taxonomías de concepto (con la relación del *subclassOf*) y las propiedades son organizadas en jerarquías de la propiedad (con la relación *PropertyOf*).

Las extensiones para *RDF(S)* son relatadas para propiedades, donde:

- Se definen las cardinalidades de la propiedad.
- Se distinguen entre relaciones y atributos, según el rango, si es un concepto o un tipo de dato (*datatype*).
- Se definen las relaciones simétricas o asimétricas.
- Se definen las relaciones inversas.

El modelo de conocimiento *Kaon2* también coloca etiquetas, documentación y sinónimos. Los

componentes más complicados, como los axiomas formales, no son expresados en *Koan2*.

- ✓ **Ventajas:**

- Actualmente, puede trabajar con sintaxis *XML (Extensible Markup Language)*, *RDF* y *OWL*.
- Soporta el subconjunto $\mathcal{SHIQ}(\mathcal{D})$de *OWL-DL*. Esto incluye todas las características de *OWL-DL* con la excepción de las clases enumeradas.
- Los algoritmos dan contestación a las consultas en *Kaon2* en uno o más órdenes de magnitud más rápido que en los sistemas existentes.
- Facilita servicios de manipulación de una ontología, como la adición y recuperación de axiomas de la ontología, es completamente compatible con *OWL* y *SWRL* en el nivel sintáctico.

- ***RACER (Renamed Abox and Concept Expression Reasoner)***

Es un razonador *DL* para la lógica $\mathcal{SHIQ}$, su nombre comercial es *RacerPro*. *Racer* no tiene licencias libres que permitan su utilización, lo que lo descarta como posible motor de inferencia para algunos sistemas. Soporta *OWL-DL* excepto para los nominales (clases definidas por una enumeración de sus miembros, que implementa como definiciones parciales) y para tipos de datos no estándar. Maneja largas *ABox* en combinación con largas y expresivas *TBox* y provee servicios de inferencia sofisticados.

Tiene sus orígenes dentro de las *DLs*; proporciona una base de criterios para normalizar los lenguajes de las ontologías en el contexto de la Web Semántica. Este razonador gestiona ontologías basadas en *OWL*, sin embargo, también es visto como un repositorio web de información semántica con una optimización en su motor para la recuperación de información. Es conocido como: ¨Razonador para *ABoxes* y expresiones de conceptos renombrados¨. Fue uno de los primeros razonadores para *ABoxes*, el sistema también utiliza lógica modal.

RacerPro es un sistema de RC, utiliza el método *Tableaux* altamente optimizado para las *DLs*. Con los avances que se realizan a diario; ahora es posible ofrecer servicios de razonamiento para múltiples *TBoxes* y *ABoxes*.

✓ **Características:**

- Permite comprobar la coherencia de una ontología *OWL* y un conjunto de descripciones de datos.
- Busca relaciones implícitas, sub-clases incluidas, en la ontología.
- Encuentra sinónimos de los recursos (ya sean clases o los nombres de instancias).
- Apoya el uso de recursos computacionales: Las respuestas que necesitan menos recursos computacionales tienen prioridad, las otras aplicaciones se debe decidir si todas las respuestas valen o no tal esfuerzo.

- ***FaCT++***

FaCT++ es un razonador *DL* que soporta la lógica $\mathcal{SHOIQ}(\mathcal{D})$, desarrollado bajo el proyecto europeo *WonderWeb*. Es una nueva implementación en *C++* del razonador *FaCT*. Los dos razonadores utilizan los algoritmos de *Tableaux*, pero se diferencian en la implementación de nuevas características y optimizaciones, una nueva arquitectura que se puede personalizar para añadir nuevas tácticas de razonamiento y la capacidad de razonar con *DLs* más potentes y cercanas a la expresividad de *OWL-DL*.

Está diseñado como una plataforma para experimentar con nuevos algoritmos del *Tableaux*. También incluye una nueva arquitectura nombrada: “Lista de tareas pendientes”. Se ha demostrado que dicha arquitectura es la más adecuada para los algoritmos *Tableaux* (como los usados para razonar con ontologías *OWL*), estos cuentan con un rango más amplio de optimizaciones heurísticas.

Generalmente este razonador cumple algunas funciones: el chequeo de consistencia de una ontología, la relación que existe entre las clases, clasificar toda la ontología para crear y corregir taxonomías, y comprobar la factibilidad de los conceptos de la ontología. [16]

✓ **Características:**

- Su lógica es expresiva; en particular el razonador $\mathcal{SHIQ}$, el cual es expresivo para ser usado como razonador para las *DLs*, porque razona con esquemas de base de datos.
- Tiene soporte para razonar con *KB* arbitrarias.

- Su implementación de tablas semánticas es muy optimizada. Convirtiéndose en la actualidad en un estándar para los sistemas *DLs*.

✓ **Ventajas:**

Entre las ventajas de *FaCT*++ se encuentran:

- Tiene licencia *GPL* (*General Public License*- Licencia Pública General).
- Tiene mantenimiento y sigue en desarrollo. En la actualidad se está investigando cómo aumentar la capacidad de razonamiento del motor de inferencia para que tenga soporte para *OWL*.
- Trabaja de forma eficiente con la *TBox* de ontologías de tamaño grande y mediano.

✓ **Desventajas:**

FaCT++ es un buen razonador para la *TBox* de una ontología. Sin embargo, hay dos desventajas importantes que lo aplazan como alternativa para utilizar en *EzWeb*[13]:

- *FaCT*++ no tiene soporte para otros tipos de datos que no sean *String* o *Integer*, como sí ocurre, por ejemplo, con Pellet.
- No cuenta con soporte para el razonamiento con la *ABox* de la ontología.

En cualquier caso, ambas carencias no son del todo definitivas, ya que podemos utilizar el lenguaje de consultas para *RDF*: *SPARQL*, para consultar el modelo inferido de una ontología *OWL-DL*. En el caso de los tipos de datos en un entorno web, es preferible que exista soporte para los tipos de datos de *XML Schema*.

- ***CEL***

Es el primer razonador para la *DL-*$\mathcal{EL}+$. El apoyo a la *DL-*$\mathcal{EL}+$, le brinda los medios expresivos para realizar la formulación de ontologías médicas y biológicas. Tiene una interfaz muy interactiva, simple, que provee a los usuarios de todas las funcionalidades esenciales, incluyendo un sistema de ayuda. El desarrollo de este razonador es un trabajo constante, se propone impulsar su poder expresivo lógico,

[13] Proyecto de Software Libre que permite a los usuario combinar varias aplicaciones y servicios de la red unificándolos en una sola aplicación individualizada.

descomponer conceptos, dominios concretos, y nominales. Ayuda como razonamiento práctico en las ontologías de gran extensión de la ciencia de la vida. [17]

✓ **Características:**

- Implementa un algoritmo polinomio-tiempo.
- Se basa en los avances más recientes de las *DLs*; tomando en cuenta las restricciones existenciales de conjunción.

- ***MSPASS***

Son un conjunto de varios teoremas con características automatizadas para numerosas lógicas, es una extensión de *SPASS*, ha sido denominado como un cuentahílos para la LPO y las *DLs*.

El razonador es una extensión de *SPASS* que ayuda a manejar fórmulas de la lógica modal, *DLs* y de cálculo relacional. Las fórmulas de primer orden se hacen de un vocabulario de símbolos de la proposición con dos tipos disjuntos, uno tipo *booleano* (concepto) y otro tipo relacional (rol) específico que integra el repertorio de construcciones lógicas:

- Los operadores estándar booleanos; el tipo booleano y las fórmulas de tipo de relaciones, tienen estos valores: Verdadero, Falso, no, y, o, etc.
- Los operadores relacionales adicionales: la composición, la suma relativa, el reverso, la relación de identidad, la relación de diversidad, la prueba, la restricción de dominio y alineación de restricción.
- Los símbolos verdaderos y falsos tienen diversas interpretaciones, estas sirven también como *booleano* o fórmula de relación. El verdadero es usado como tipo *booleano*, representa un conjunto universal, y es usado como relación universal. De manera similar, el falso es usado como el tipo de relación lo que representa al conjunto vacío o una relación vacía.

- ***IRIS***

Iris es un razonador *open source* licenciado bajo *GNU*. Desarrollado en el lenguaje *Java*. En él se evalúa el registro de datos seguros o inseguros, tipo de esquema de datos *XML*, función de predicados. Su

paquete de instalación contiene tres componentes: motor de razonamiento, analizador, y otros programas de utilidad, incluidos aplicaciones que dan una interfaz de usuario para el motor *IRIS*.

- ✓ **Características:**

- Utiliza el método *Datalog* para su razonamiento.
- Conjunto completo y extensible de construcción en los predicados.
- Soporte para todos los tipos primitivos de datos de esquema *XML*.

- ***BOR***

Razonador desarrollado por el Laboratorio *Sirma* del proyecto *On-To-Knowledge*. Tiene soporte para inferencias sobre instancias y conceptos: chequeo de la consistencia y modelo de la ontología, construcción de la jerarquía de conceptos y clasificación de conceptos definidos. El razonador puede ser usado con ontologías escritas en *OWL-Lite*, con algunas restricciones. Tiene implementados los servicios de raciocinio de validación de la instancia, realización, recuperación, recuperación de componentes, validación del modelo y extracción mínima.

2.6 Comparación de los razonadores de las *DLs*

Los razonadores de las *DLs* pueden ser caracterizados apoyándose en disímiles características que los distinguen a unos de otros. A continuación serán explicadas las características que se han tomado en cuenta para identificar cada uno de los razonadores que son abordados en esta investigación:

- ✓ **Lenguaje:** Un lenguaje de programación no es más que el que facilita el control del rendimiento del software, de un sistema o una máquina. Entre los lenguajes de programación más empleados en la actualidad se encuentran: *Php, Prolog, Python, Pascal, C++, C, Basic, Java.* Los lenguajes de programación pueden clasificarse como compilados y no compilados. Esto afecta la velocidad de procesamiento de una herramienta debido a que los lenguajes compilados son tan rápidos como los no compilados.

- ✓ **Soporte DIG:** La interfaz *DIG* permite el acceso a los razonadores de las *DLs* a través de los editores de ontologías. Para poder lograr esto se utiliza el protocolo *HTTP* para describir los conceptos y las relaciones en la ontología. Esta interfaz utiliza como técnica de búsqueda la optimización de motores de ramificación semántica y sintáctica, la búsqueda hacia atrás (retroceso) y la profundidad. Cuenta además con las técnicas de clasificación e inferencia. En la investigación sólo de definirá si el sistema cuenta con la interfaz DIG o no.
- ✓ **Tipo de Licencia:** La licencia es un contrato que especifica los deberes y derechos de las partes involucradas en el desarrollo del software bajo derechos de autor. En esta caracterización nos referiremos únicamente a los tipos de licencia: libre y no libre. La licencia libre es cuando se trata de software que proporciona la libertad para: ejecutar el programa para cualquier propósito, estudiar cómo funciona el programa y adaptarlo a nuestras necesidades. Luego se podrán distribuir copias y publicar las mejoras, en beneficio de toda la comunidad. El tipo de licencia no libre es aquella que no posee por lo menos alguna de las libertades anteriormente expuestas.
- ✓ **Disponibilidad de API:** Una *API* (Interfaz de Programación de Aplicaciones) es un conjunto de funciones que proporcionan el intercambio de datos entre varios sistemas. La misma facilita diversos tipos de diálogos con el proveedor para obtener o actualizar la información de la aplicación.
- ✓ **Tipo de Consultas:** El tipo de consultas, también puede ser denominado: Mecanismos de Razonamiento. Estos se pueden dividir en dos grupos; los mecanismos de razonamiento en la *TBox* y en la *ABox*.

  El primero de los mecanismos nos permite examinar la estructura del conocimiento representado y realizar la inferencia a partir de esta información. También se clasifican como mecanismos estructurales de razonamiento (subsunción de conceptos) o intencional. Entre los mecanismos de razonamiento habituales de la *TBox*, se encuentran: el chequeo de consistencia, la subsunción y la equivalencia.

  Los mecanismos de razonamiento en la *ABox* es lo que nos permite derivar nuevas aplicaciones y son conocidos igualmente como razonamiento extensional. Entre sus módulos de razonamiento

típicos se conocen: la verificación de instancias, la realización de que es especificado un individuo como una instancia, y la consistencia de la *KB*.

- **Características esenciales de los Razonadores**

El siguiente es un resumen de las características relacionadas con los razonadores más desarrollados que anteriormente fueron caracterizados:

CARACTERÍSTICAS	RAZONADORES			
	Pellet	RacerPro	FACT ++	Kaon2
Algoritmos de Razonamiento	Tableaux	Tableaux	Tableaux	Datalog
Expresividad de Razonamiento	Razonamiento SHROIQ (D)	------	Razonamiento SHIQ y SHF	Razonamiento SHIQ
Lenguaje de consulta	SPARQL	nRQL	-----	SPARQL
Extraer datos de las KB Relacionales	Si	Si	Si	Si
Lenguaje	Java	Comercial	C++	Comercial
Soporte DIG	Si	Si	Si	Si
Tipo de Licencia	Libre	No Libre	Libre	Libre
Soporte de Reglas	SWRL-DL	SWL	No	SWRL-DL
Disponibilidad del API	Si	Si	No	Si
Multiplataforma	Si	No	No	Si
Tipo de Consultas	A-Boxes y T-Boxes	A-Boxes y T-Boxes	T-Boxes	A-Boxes extensas

Tabla 3: Características de los razonadores más desarrollados.

El siguiente es un resumen de las características relacionadas con los razonadores menos desarrollados y que fueron anteriormente caracterizados:

CARACTERÍSTICAS	RAZONADORES		
	IRIS	MSPASS	CEL
Algoritmos de Razonamiento	Datalog	-----	Polinomio-tiempo
Expresividad de Razonamiento	-----	-----	-----
Lenguaje de consulta	-----	-----	-----
Extraer datos de las KB Relacionales	Si	-----	-----
Lenguaje	Java	C++	-----
Soporte DIG	No	No	Si
Tipo de Licencia	Libre	Libre	-----
Soporte de Reglas	WSML	-----	-----
Disponibilidad del API	Si	-----	No
Multiplataforma	-----	-----	Si

Tabla 4: Características de los razonadores menos desarrollados.

CONCLUSIONES DEL CAPÍTULO

Luego del desarrollo del capítulo pueden ser enunciadas las siguientes conclusiones:

- El estudio detallado de las características de los razonadores permite contar con elementos para llevar a cabo una elección del más efectivo para ser aplicado en el proceso de *stream reasoning.*
- Entre los razonadores que han sido estudiados, *Pellet* se distingue por contar con características esencialmente importantes y que deben ser tenidas en cuenta a la hora de efectuar una selección.

Capítulo 3

Propuesta para el stream reasoning

INTRODUCCIÓN

Los niveles de expresividad alcanzados por los lenguajes para la RC han impuesto nuevos retos para los desarrolladores de los razonadores. Para la aplicación de un razonador de las *DLs* en el proceso de *stream reasoning,* es necesario seleccionar un motor de inferencia que trabaje de manera eficiente sobre la información que sea procesada. En este capítulo se analiza de manera detallada las características de los razonadores, para efectuar una selección eficaz y lograr describir de manera general cómo debe desarrollarse el proceso de *stream reasoning*.

3.1 Selección del Razonador

- **Análisis de los razonadores**

El propósito de la presente investigación es la selección de un razonador para implementarlo en el proceso *stream reasoning*. Para este fin se han tenido en cuenta una serie de criterios que resultan claves para la propuesta de soluciones eficientes. Los elementos tomados a consideración son enunciados a continuación:

- ✓ El algoritmo de *Tableaux* se ha convertido en un estándar para los procesos de inferencia.
- ✓ El tipo de licencia con que cuenta cada razonador es importante y es un elemento que será tenido en cuenta en la investigación. La misma determinará la elección del razonador para poderle implementar cambios y aplicarlo a la vida real en los procesos de inferencia de los datos que circulan en tiempo real.

- ✓ El lenguaje de consulta que posee cada razonador es un aspecto esencial para poder extraer los datos de la ontología.
- ✓ Se tendrá en cuenta que el razonador sea multiplataforma, ya que esta propiedad permite contar con herramientas capaces de funcionar en varias plataformas. Este elemento constituye una ventaja incuestionable porque nos brinda la libertad de escoger el mejor dominio donde se trabaje para poder realizar un razonamiento con calidad.
- ✓ La existencia del *API* en un razonador permite el trabajo directo con el mismo y el desarrollo de mejoras.

Las características anteriormente expuestas son algunas de las que serán evaluadas para la selección del razonador. Al mismo se le aplicarán mejoras con el objetivo de optimizar los indicadores de eficiencia y escalabilidad para el *stream reasoning*.

- **Selección del razonador**

Para la selección del razonador sólo serán tenidos en cuenta los que más se han desarrollado actualmente (*Pellet*, *Kaon2*, *FaCT++, RacerPro*). Esto se debe a que la selección deberá estar centrada en un razonador que sea óptimo para la inferencia, que cuente con apoyo de la comunidad de la Web Semántica y que tenga propiedades como la licencia libre permitiendo así su modificación.

En base al criterio de accesibilidad a la información, vemos que *FaCT++,* a pesar de ser *open source,* no cuenta con mucha información de ayuda. Otras desventajas que presenta este razonador es que no posee una gran comunidad de soporte, ni ofrece suficiente información al usuario en caso de errores. Otros elementos desfavorables son que no es multiplataforma y no posee disponibilidad de *API*. Aunque es un poco más eficiente que *Kaon2* en su interfaz *DIG*.

Por otro lado se encuentra el razonador RacerPro, el mismo no puede ser seleccionado ya que su tipo de licencia es privativa. Originando que sea imposible realizarle cambios.

El razonador seleccionado en esta investigación será *Pellet*. El mismo cuenta con una licencia libre, interfaz *DIG*, permite extraer datos de las *KB* relacionales y es multiplataforma. Pero la razón fundamental que se tuvo en cuenta a la hora de su selección es que cuenta con un gran apoyo por parte de la

comunidad de desarrolladores; por lo cual tendrá la posibilidad de acelerar la madurez del razonador. Pero su ventaja principal radica en que permite realizar un razonamiento incremental sobre los datos, evitando de esta manera tener que comenzar siempre desde el inicio cada tarea de razonamiento.

3.2 Análisis del razonador *Pellet*

Pellet es un razonador multiplataforma con doble licencia; *LGPL* para las aplicaciones de código abierto y licencia propietaria para aplicaciones privadas. En esta investigación nos basaremos en la licencia libre ya que nos brinda la posibilidad de crear soluciones y modificar el razonador de acuerdo a los objetivos de la investigación.

Actualmente éste es el único razonador que comprueba satisfactoriamente la coherencia de la información. El núcleo del razonador está basado en el algoritmo de *Tableuax,* anteriormente caracterizado. Pellet lleva a cabo un conjunto de tareas para poder desarrollar sus servicios de razonamiento.

- **Servicios de Inferencia Lógica del razonador *Pellet***

Al igual que el resto de los razonadores, *Pellet* implementa un conjunto de servicios de inferencia para llevar a cabo el razonamiento. Estos servicios son enunciados a continuación:

- ✓ **Chequeo de Consistencia:** Garantiza que una ontología no contenga hechos contradictorios.
- ✓ **Satisfactibilidad de Concepto:** Comprueba si es posible que una clase tenga instancia. Si la clase es insatisfactible; luego definir una instancia provocaría que una ontología sea incompatible.
- ✓ **Clasificación Incremental:** Calcula las relaciones entre cada subclase de la clase nombrada, para crear jerarquía de clases completas. Esta jerarquía de clases se puede utilizar para responder a consultas tales como: conseguir todas o sólo las subclases directas de una clase.
- ✓ **Realización:** Encuentra la clase más específica a la que pertenece un individuo. En otras palabras, calcula los tipos directos para cada uno de los individuos. Este método sólo es posible realizarlo después de la Clasificación Incremental ya que los tipos directos se definen con respecto a una

jerarquía de clases. Empleando la clasificación jerárquica también es posible obtener todas las clases a las que pertenece el individuo.

- **Análisis y selección de Servicios de Razonamiento**

Entre los cuatro servicios de inferencia expuestos anteriormente son seleccionados dos de ellos para su valoración y posterior modificación: chequeo de consistencia y clasificación incremental. Esta elección se basa en que ambos son los que podrían apoyar el manejo de la información en tiempo real. Serán evaluados y se determinará cuál de ellos puede ser modificado para mejorar los indicadores de eficiencia y escalabilidad en el proceso de s*tream reasoning,* aunque es válido aclarar que ambos pueden ser seleccionados. Este proceso quedará determinado a partir de la valoración de cuál o cuáles servicios influyen en el trabajo directo con las ontologías. A continuación serán caracterizados los servicios seleccionados:

- ✓ **Chequeo de Consistencia**

La consistencia lógica es una propiedad de un conjunto de axiomas, puede definirse como verdades que no necesitan demostración. Un conjunto de axiomas es consistente si a partir de él no puede deducirse simultáneamente una proposición (p) y su contraria ($\neg p$). Un razonador semántico chequea la consistencia lógica buscando una solución tal que junto con su negación pueda considerarse teorema.

La consistencia de un argumento es la necesidad de que todas las premisas tengan que ser necesariamente y a la vez, verdaderas. Aunque el chequeo de consistencia también se centra en el argumento, si es consistente, pueda ser válido o no válido. La consistencia tiene que ver con que las implicaciones lógicas del sistema no sean autocontradictorias. Un razonador semántico, en definitiva, busca que exista coherencia en una *KB*.

Un comprobador de consistencia tiene un documento de entrada; y devuelve un resultado luego de su comprobación. Este resultado se traduce en las palabras: consistente, inconsistente y desconocido.

El chequeo de consistencia es una tarea importante y es factible que se aplique cuando hay cambios muy frecuentes en las instancias, pero no permite trabajar con las ontologías a profundidad. Se centra únicamente en verificar que el conjunto de axiomas de entrada no contenga contradicciones. Esta desventaja incide directamente en que resulte imposible seleccionar este servicio de inferencia como

candidato para realizar cambios con el fin de mejorar el *stream reasoning.*

- ✓ **Clasificación Incremental**

Este servicio es fundamentalmente aplicado cuando ocurren cambios (adiciones y eliminaciones) en la jerarquía de clases de una ontología. La eficiencia de esta tarea se ve determinada por la estructura de la ontología y por las actualizaciones que se realicen.

Una ventaja fundamental de los medios de razonamiento incremental es que permite realizar cambios en una ontología sin tener que comenzar desde el principio. El razonamiento incremental engloba dos tareas fundamentales: el chequeo incremental y la clasificación incremental.

Como anteriormente se explicó la clasificación incremental es empleada para actualizar incrementalmente la clasificación de los axiomas cuando se efectúan cambios en la ontología.

Este servicio es el que será seleccionado en la investigación, por su notable importancia en el manejo a profundidad de las ontologías. Su empleo a la hora de efectuar cambios en la jerarquía de clases es otra de las razones por las cuales fue seleccionado, para realizar las modificaciones y obtener mejores indicadores de escalabilidad y eficiencia.

3.3 Clasificación Incremental en las *DLs*

El razonador *Pellet* implementa dos algoritmos que resultan fundamentales para poder llevar a cabo el razonamiento sobre corrientes de datos. Los mismos trabajan de manera incremental sobre los cambios que sean efectuados sobre la ontología. El algoritmo de Clasificación Incremental será el analizado por las razones que anteriormente fueron expuestas.

De acuerdo a la lógica que sea empleada y al nivel de expresividad que se quiera lograr a la hora de procesar la información; han sido definidos una serie de algoritmos que implementan la Clasificación Incremental. En este caso el razonador seleccionado emplea la lógica $\mathcal{SROIQ}$ y se propone, para ser aplicado a *Pellet*, el algoritmo de Clasificación Incremental usando Módulos para los axiomas.

Su funcionamiento se basa en los módulos que crea donde solamente almacena un conjunto de axiomas. Simplificando el manejo de la ontología y las inferencias que se realicen sobre esta, y por consiguiente

disminuyendo el tiempo del proceso. El algoritmo cuenta entre sus ventajas con la flexibilidad y la facilidad de implementación.

Actualmente *Google* desarrolla investigaciones en este sentido por la necesidad de disminuir los tiempos de respuestas. Obteniendo resultados que constituyen verdaderos aportes al trabajo con la información de la web. Surgiendo de esta manera el *framework Haddop* que trabaja con el motor de inferencia *MapReduce*; que se ocupa de la información distribuida pero no del procesamiento continuo.

Para la exposición de este algoritmo serán adoptadas las siguientes convenciones a la hora de nombrar las ontologías: ($\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_n$) para referirse a los subconjuntos de una ontología dada y ($\mathcal{O}^1, \mathcal{O}^2, \ldots, \mathcal{O}^n$) para referirse a las diferentes ontologías. [18]

- ✓ **Algoritmo de extracción de Módulos para $\mathcal{SROIQ}$**

Procedure extract_module ***($\mathcal{O}$; $\mathcal{S}$)***

Input:

O: ontology;

$\mathcal{S}$***:*** signature;

Output:

O1: a module for $\mathcal{S}$ in $\mathcal{O}$

1: $\mathcal{O}_1$:=Ø ; $\mathcal{O}_2$:= O

2: **while not empty** ($\mathcal{O}_2$) **do**

3: α := select_axiom ($\mathcal{O}_2$)

4: **if** local(α ; S ∪ Sig($\mathcal{O}_1$)) **then**

5: $\mathcal{O}_2$:= $\mathcal{O}_2$ \ { α }

6: **else**

7: $\mathcal{O}_1$:= $\mathcal{O}_1$ ∪ { α }

8: $\mathcal{O}_2$:= $\mathcal{O}$ \ $\mathcal{O}_2$

9: **end if**

10: **end while**

11: **return** $\mathcal{O}_1$

- ✓ **Algoritmo de Clasificación Incremental usando Módulos**

Procedure inc_classify_mod ($\mathcal{O}_1$, $\Delta^+\mathcal{O}$, $\underline{\sqsubset}_1$)

Input:

$\mathcal{O}_1$: an ontology

$\Delta\mathcal{O} = (\Delta^-\mathcal{O}, \Delta^+\mathcal{O})$: removed / added axioms

$\underline{\sqsubset}_1$: subsumption relations in $\mathcal{O}^1$

A→ $\mathcal{O}^1_A$: a module for every A ∈ $CN^T(\mathcal{O}^1)$

Output:

$\mathcal{O}^2$: The result of applying the change $\Delta\mathcal{O}$ to $\mathcal{O}_1$

$\underline{\sqsubset}_2$: subsumption relations in $\mathcal{O}^2$

A→ $\mathcal{O}^2_A$: a module for every A ∈ $CN^T(\mathcal{O}^2)$

---

1. $\mathcal{O}^2 := (\mathcal{O}^1 \setminus \Delta^-\mathcal{O}) \cup \Delta^+\mathcal{O}$
2. **for each** D ∈ CN($\mathcal{O}^2$) \ CN($\mathcal{O}^1$) **do**
3. $\mathcal{O}^1_D = \mathcal{O}^1_T$
4. **for each** T $\underline{\sqsubset}_1$ *B* **do** *D* $\underline{\sqsubset}_1$ *B*:= **true**
5. **for each** A $\underline{\sqsubset}_1$ ⊥ **do** A $\underline{\sqsubset}_1$ *D*:=**true**
6. **end for**
7. M^- := Ø M^+:= Ø
8. **for each** A ∈ $CN^T(\mathcal{O}^2)$ **do**

9. **for each** $\alpha \in \Delta^{-}\mathcal{O}$ **do**
10. **if not** local (α, Sig($\mathcal{O}^1_A$)) **then**
11. $\mathbb{M}^{-} := \mathbb{M}^{-} \cup \{A\}$
12. **end if**
13. **end for**
14. **for each** $\alpha \in \Delta^{+}\mathcal{O}$ **do**
15. **if not** local (α, Sig($\mathcal{O}^1_A$)) **then**
16. $\mathbb{M}^{+} := \mathbb{M}^{+} \cup \{A\}$
17. **end if**
18. **end for**
19. **end for**
20. **for each** $A \in CN^{T}(\mathcal{O}^2)$ **do**
21. **if** $A \in \mathbb{M}^{-} \cup \mathbb{M}^{+}$ **then**
22. $\mathcal{O}^2_A$:= extract_module ({A}, $\mathcal{O}^2$)
23. **else**
24. $\mathcal{O}^2_A := \mathcal{O}^1_A$;
25. **end if**
26. **for each** $B \in CN(\mathcal{O}^2) \cup \{\underline{\bot}\}$ **do**
27. **if** ($A \in \mathbb{M}^{-}$ **and** $A \sqsubseteq_1 B$) **or**
28. ($A \in \mathbb{M}^{+}$ **and** $A \not\sqsubseteq_1 B$) **then**
29. $A \sqsubseteq_2 B$:= subsumes ($\mathcal{O}^2_A$,< A, B>)
30. **else**
31. $A \sqsubseteq_2 B := A \sqsubseteq_1 B$
32. **end if**

33. **end for**
34. **end for**
35. **return** $\mathcal{O}^2$, $\underline{\sqsubset}_2$, A→ $\mathcal{O}^2_A$

- **Ventajas del empleo del Algoritmo de Clasificación Incremental por Módulos**

La incorporación del algoritmo de Clasificación Incremental usando Módulos constituye una avanzada en el proceso de *stream reasoning*. Entre las ventajas que reviste su incorporación al razonador Pellet se encuentra:

- ✓ Facilita el trabajo con la ontología porque se enfoca en los módulos donde se encuentran los axiomas de interés y evita de esta manera el análisis general de los elementos.
- ✓ Reduce el tiempo de inferencias y clasificación de la ontología.
- ✓ Convierte el *stream reasoning* en un proceso escalable y eficiente.

De manera general la primera vez que la ontología es clasificada, *Pellet* computará los módulos para cada clase. Cuando la ontología es modificada y se ve afectada la jerarquía de clases, se determina el módulo que se ha afectado. Posteriormente sólo será modificado ese módulo, que lógicamente siempre será menor que la ontología original. Esta tarea es esencialmente importante para trabajar con ontologías que contengan una jerarquía de clases compleja y/o grande.

3.4 Proceso de *stream reasoning* empleando las *DLs*

Esta investigación propone la ejecución de un *stream reasoning*, con el empleo de uno de los razonadores de las *DLs*: *Pellet*. Actualmente este razonador no es capaz de razonar sobre corrientes de datos; razón por la cual la perspectiva radica en procesar corrientes de datos con el apoyo de *Pellet* y de esta manera lograr eficiencia y escalabilidad en el proceso de *stream reasoning*.

En el capítulo anterior fueron abordados temas relacionados con el *stream reasoning*. Para enfocarnos en el mismo debemos tener en cuenta dos elementos básicos para el procesamiento de corrientes de datos, que ya han sido expuestos: Las *windows* y el *continuous processing* de la información. En este punto de la

investigación se definirá cómo serán procesados los datos en tiempo real para poder aplicar *Pellet* en el razonamiento.

El primer elemento a tener en cuenta para la tarea que se va a llevar a cabo es que se trabajará con *data stream*. Las características de estas corrientes de datos exigen que el proceso se enfoque en una ventana de tiempo que será definida. Este elemento permitirá centrar el proceso en sólo un segmento de la corriente de información y trabajar únicamente con la información que contenga esta ventana. A continuación se muestra de manera gráfica el procedimiento:

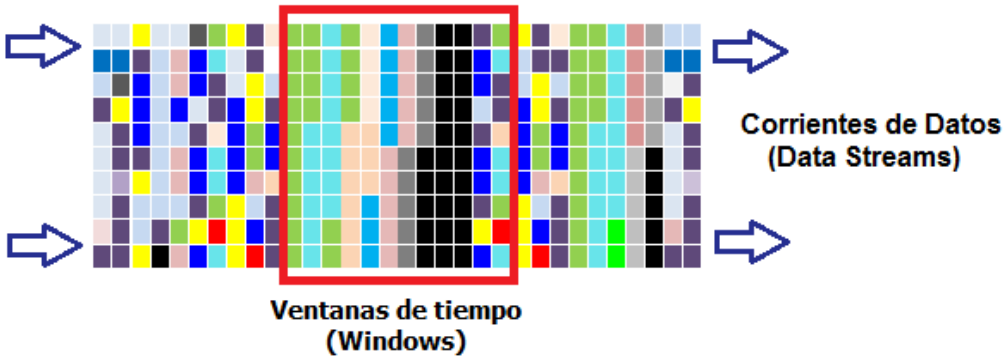


Fig. 11: Windows de una data stream.

Otro elemento cardinal a tener en cuenta es el rango en el que van a desarrollarse las tarea de inferencia dentro de las corrientes de datos. Es necesario definir, siempre que se desee realizar un *stream reasoning*, cuáles serán los límites del *Continuous Processing*.

A continuación se propone el pseudocódigo que deberá ser tomado en cuenta para la implementación de una *SDL* (*Stream Reasoning_ Description Logic)*, con el empleo del razonador *Pellet* y apoyándose en uno de los algoritmos que este implementa y que permite manejar con eficiencia los cambios que se realicen en las ontologías: Clasificación Incremental.

- ✓ **Algoritmo para la creación de un *stream reasoning* para ser aplicado a Pellet**

Entradas:
DS: Corriente de Datos;
T_i: Ventana de Tiempo;
T_{cp}: Procesamiento Continuo.

Salidas:
$\mathcal{O}$: Ontología;
$\mathcal{I}$: Inferencias;

1. **Desde** corriente_datos
2. **Si** esta_Rango$\mathcal{T}cp$ (T_i)
3. **Registrar** data_stream ***cada*** T_i
4. **Llamar al procedimiento** Pellet.inc_classify_mod
5. **Actualizar** ($\mathcal{O}$, $\mathcal{I}$)
6. **Guardar** ($\mathcal{O}$, $\mathcal{I}$)
7. **Retornar** $\mathcal{O}$, $\mathcal{I}$

Este algoritmo ha sido graficado con el objetivo de representar el proceso de *SDL* y está compuesto por los elementos siguientes:

[1]. El elemento principal serán las *data streams*, las mismas viajarán de un lado a otro en períodos cortos de tiempo. Serán recibidas de disímiles puntos a partir de sensores.

[2]. El Procesamiento Continuo de las tareas de inferencia se desarrollarán únicamente en el rango que sea definido; especificando de esta manera su comienzo y su final.

[3]. La ventana de tiempo determinará cuál será la información que será procesada. Será registrada únicamente la información que esté dentro de la ventana de tiempo.

[4]. Aplicación del razonador *Pellet* con el algoritmo de clasificación incremental usando módulos y se almacena la ontología y las inferencias.

[5]. Como resultado quedará la ontología y las Inferencias que se le realicen a la misma. Este último elemento se puede traducir en las relaciones de inferencia que existan entre las clases de la ontología.

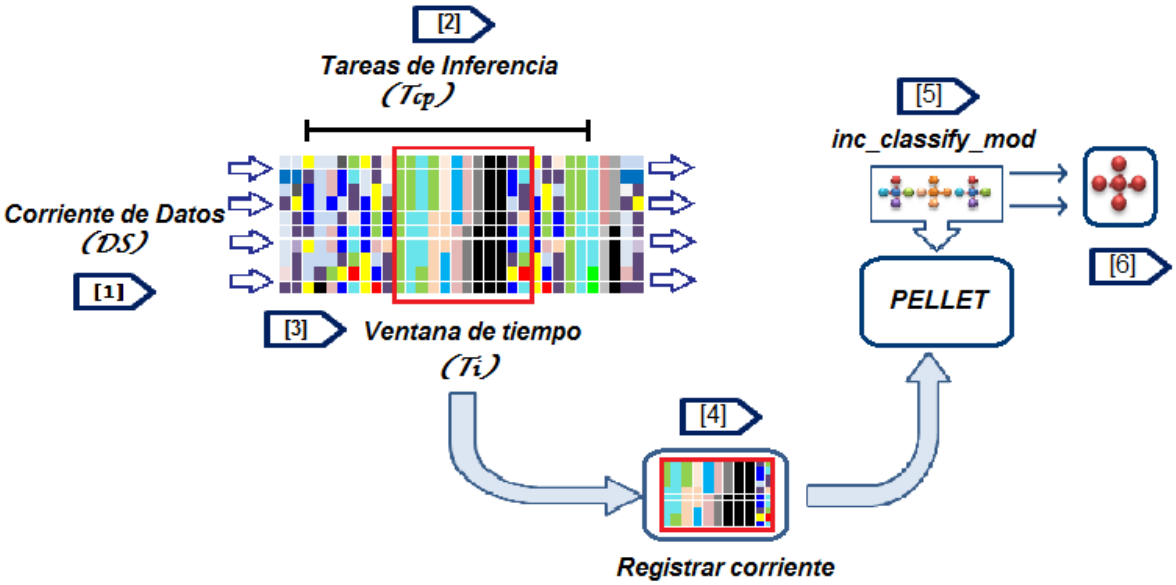


Fig. 12: Modelo del proceso de Stream Reasoning_ Description Logic.

3.5 Editor de ontologías seleccionado

Protegé es el editor de ontologías que se propone para posteriores pruebas de los procedimientos que en esta investigación se formulan. Considerándose acertado si se tiene como premisas su gratuidad y su condición de código abierto; empleando una *KB* en su funcionamiento. La plataforma de *Protegé* soporta dos formas principales de modelado: a través de marcos y a través de editores *OWL*. Las ontologías en este editor pueden ser exportadas en una variedad de formatos; incluyendo *RDF(S), OWL* y *XML Schema*. Está basado en *Java* y proporciona un entorno de *plug and play* que lo hace más flexible para la creación rápida de prototipos y desarrollo de aplicaciones. Cuenta con un gran apoyo de la comunidad de desarrolladores y académicos; ha sido aplicado en áreas tan diversas como la biomedicina, la geografía, la astronomía y la defensa.

Se sugiere que la versión que se emplee para comprobar la *SDL*, que sea implementada, sea superior a la versión 3.4.1. Esta sugerencia se basa en que las mismas traen integrado el *Pellet*, siendo esta una característica esencial ya que evitará la instalación de este razonador y por consiguiente facilitará el proceso de validación.

3.6 Selección y análisis de ontología

Las ontologías se basan en la conceptualización de un dominio determinado. En su nivel más básico, la conceptualización corresponde a una enumeración de todos los individuos relevantes. A estos individuos se les podrá asociar propiedades que determinarán características específicas de cada uno de ellos. Las clases estarán compuestas por un conjunto de individuos que compartan características en común.

Estos últimos elementos son los componentes principales a la hora de definir una ontología, aunque los sistemas para el tratamiento de ontologías proporcionan también la posibilidad de definir subclases especializadas.

Entre los principales impulsores de las investigaciones relacionadas con los razonadores semánticos, se encuentra Ian Horrocks. En sus investigaciones expone la importancia de seleccionar ontologías, en el momento de trabajar con los motores de inferencia, que sean reflejo típico de la vida real y que resulten expresivas. Para la selección de la ontología la investigación se basó fundamentalmente en el hecho de que el trabajo manual con la misma no resultará factible como lo sería con ejemplos pequeños del mundo real.

La ontología que se propone está relacionada con la educación y posee una gran cantidad de clases y subclases; englobando de esta manera un gran número de datos. La misma emplea el lenguaje *OWL*, el mismo es un estándar dentro de los lenguajes para ontologías y es uno de los que admite la herramienta Protegé. La dimensión de la ontología va a requerir un mayor esfuerzo del editor de ontologías y por lo tanto permitirá probar el tiempo de ejecución que requiere el razonador para realizar cada una de las tareas que se planteen, y por consiguiente establecer una comparación antes y después de la puesta en ejecución del *SDL*.

A continuación se propone la ontología a emplear. Las clases de la misma definen algunos conceptos básicos de la Investigación Académica: [19]

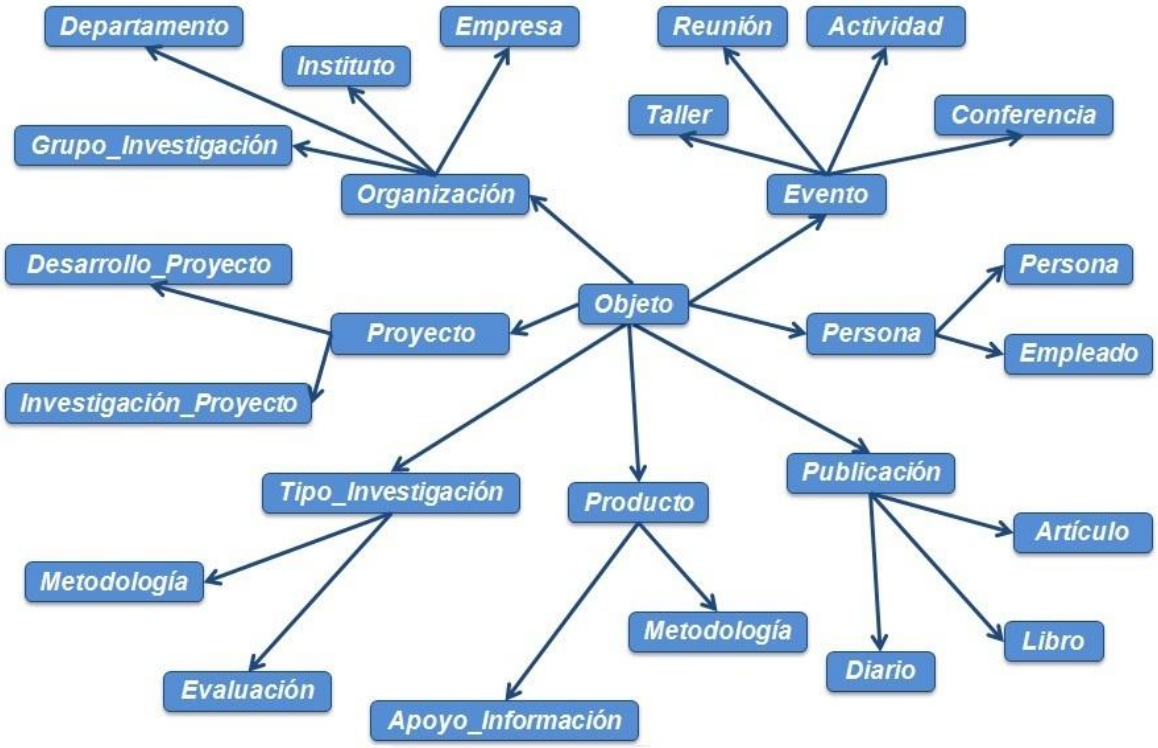


Fig 13: Ontología Propuesta.

Para poder comprobar el funcionamiento del algoritmo que se implemente, es necesario realizar cambios en la estructura de la ontología anteriormente propuesta. Posibilitando el trabajo con el algoritmo de clasificación incremental usando módulos y de forma paralela ir procesando las *data streams*.

Para realizar dichos cambios se deben tener en cuenta los 5 componentes que conforman las ontologías y que fueron abordados en el primer capítulo.

CONCLUSIONES DEL CAPÍTULO

El desarrollo de este capítulo le da paso a las siguientes conclusiones:

- Fue seleccionado *Pellet* como el razonador da las *DLs* más efectivo para ser aplicado en el proceso de *stream reasoning*.
- La aplicación del algoritmo de Clasificación Incremental usando Módulos que se propone, mejora el

proceso de inferencia cuando se producen cambios en la jerarquía de clases de una ontología.

- El pseudocódigo propuesto guía el proceso a seguir para llevar a cabo el *stream reasoning*.
- La ontología que se expone es factible para llevar a cabo validaciones del algoritmo que sea implementado, tomando como referencia el pseudocódigo propuesto.

Conclusiones

Luego de la culminación de la presente investigación se arribó a las conclusiones que a continuación se exponen:

- Luego del estudio de las herramientas existentes que utilizan las *DLs* para el razonamiento se seleccionó al razonador *Pellet* para ser aplicado en el proceso de *stream reasoning*.
- Luego del análisis de las técnicas que emplea *Pellet* para inferir se seleccionó la Clasificación Incremental para realizar mejoras en el razonamiento.
- Se analizaron las características del procesamiento de corrientes de datos y se propuso un algoritmo que guía el proceso de stream reasoning y mejora los indicadores de escalabilidad y eficiencia.

Recomendaciones

El desarrollo de esta investigación abre paso a futuras investigaciones y luego de ser concluida emergen nuevos elementos a tener en cuenta para trabajos posteriores, con la finalidad de que sea implementado un mecanismo para el *stream reasoning.* Derivándose las siguientes recomendaciones:

- Implementar el procedimiento para el *stream reasoning* conjuntamente al razonador *Pellet*, como parte de una de los 15 algoritmos que debe desarrollar la tutora de la presente investigación en su tesis doctoral.
- Emplear la ontología propuesta, o una similar, para realizar las validaciones.
- Ejecutar cambios en la estructura de la ontología para comprobar el funcionamiento de la clasificación incremental usando módulos en el proceso de *stream reasoning*.

Referencias Bibliográficas

[1]. *On the relationship between description logic and predicate logic*. **Alexander Borgida**. ACM, 1994.

[2]. *Las Bases de Datos y la Web Semántica: Ficción o Realidad*. **María del Mar Roldán García, José F. Aldana- Montes**. Málaga, 2010.

[3]. *Description Logics: Basics, Applications and More.* **Ian Horrocks**. University of Manchester. UK, 2008.

[4]. *Lógicas Descriptivas y Ontologías*. **Edna Ruckhaus**. Universidad Simón Bolívar, Departamento de Computación. Venezuela, 2005.

[5]. *Towards Expressive Stream Reasoning*. **Stefano Ceri, Emanuele Della Valle, Frank Van Harmelen**. Amsterdam, 2009.

[6]. *Toward Distributed and Stream Reasoning on Web of Data*. **Liudmila Reyes Álvarez and José F. Aldana Montes**. Universidad de las Ciencias Informáticas, 2010.

[7]. *Towards Expressive Stream Reasoning*. **H. Stuckenschmidt, S. Ceri**. Zentrum fuer Informatik. Germany, 2010.

[8]. *DBOWL: Persistencia y Escalabilidad de Consultas y Razonamiento en la Web Semántica.* **G. M. Roldán y M. J. Aldana**. Universidad de Málaga. España, 2009.

[9]. *Pellet: A Practical OWL-DL Reasoner*. **Sirin, Parsia, Cuenca, Kalyanpur and Katz**. Manchester, 2009.

[10]. *Análisis de técnicas de aprendizaje adaptativo con ontolgías*. **Lama and Sánchez**.2008.

[11]. *SPARQL Query Language for RDF*. **Prud'hommeaux**. http://www.w3.org/TR/rdf-sparql-query, 2008.

[12]. *The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications*. **Knublauch, Fergerson, Noyand Musen**. Proceedings of the 3rd International Semantic Web Conference, Hiroshima. Japón, 2009.

[13]. *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. **Gómez, Fernández and Vicente**. Facultad de Informática, Universidad Politécnica de Madrid. España, 2004.

[14]. *Reducing SHIQ- Description Logic to Disjunctive Datalog Programs.* **Motik and Sattler**. Proceedings of the 9th International Conference on Knowledge Representation and Reasoning. 2005.

[15]. *Reducing SHIQ- Description Logic to Disjunctive Datalog Programs*. **Hustad, Motik and Sattler**. Proceedings of the 9th International Conference on Knowledge Representation and Reasoning. 2004.

[16]. *Análisis de técnicas de aprendizaje adaptativo con ontología*. **Lama y Sánchez**. Proyecto Suma learning multimodal y adaptativo, 2009.

[17]. *A Polynomial-time Reasoner for Life Science Ontologies*. **Baadeand Suntisrivaraporn**. 2002.

[18]. *Incremental Classification of Description Logics Ontologies*. **Bernardo Cuenca Grau, Christian Halaschek- Wiener, Yevgeny Kazakov, Boontawee Suntisrivaraporn**. University of Oxford. UK, 2010.

[19]. *Ontologías.* **Ian Horrocks**. http://web.comlab.ox.ac.uk/people/Ian.Horrocks/, 2009.

Bibliografía Consultada

[1]. *Transparencias de Ian Horrocks*. **IanHorrocks**. [Online]. http://www.cs.man.ac.uk/~horrocks/Slides/, 2009.

[2]. *Introducción a las Ontologías en la Web Semántica: Fundamentación lógica, Lenguajes y Herramientas.* **Antonio Paredes Moreno**. Dpto. Economía Financiera y Dirección de Operaciones, Universidad de España. España, 2010.

[3]. *Deductive and Inductive Stream Reasoning for Semantic Social Media Analytics.* **Davide Barbieri, Daniele Braga, Stefano Ceri and Emanuele Della Valle.** Polytechnic of Milan. España, 2010.

[4]. *"Incremental Reasoning on Streams and Rich Background Knowledge".* **D. Barbierietal**. Proc. Extended Semantic Web Conf, 2010.

[5]. *"Towards Type System for Semantic Streams".* **M. Mendler and S. Scheele**. CEUR, 2009.

[6]. *Proc. 1st Int'l Workshop Stream Reasoning.* **CEUR.** http://zunsite.informatik.rwthaachen.de/publications/ CEUR-WS/Vol-466, 2009.

[7]. *"Commonsense Spatial Reasoning about Heterogeneous Events in Urban Computing", Proc. 1st Int'l Workshop Stream Reasoning*. **M. Palmonari and D. Bogni**. CEUR, 2009.

[8]. *"Stream Reasoning in DyKnow: A Knowledge Processing Middleware System," Proc. 1st Int'l Workshop Stream Reasoning*. **F. Heintz, J. Kvarnstrom, and P. Doherty**. CEUR, 2009.

[9]. *"Answering Reachability Queries on Streaming Graphs," Proc. 1st Int'l Workshop Stream Reasoning*. **G. Unel, F. Fischer, and B. Bishop**., CEUR, 2009.

[10]. *"A First Step towards Stream Reasoning," Proc. Future Internet Symp*. **E. Della Valle**. Springer, 2008.

[11]. *Description Logic Programs: Combining Logic Programs with Description Logic*. **Horrocks, Volz and Decker**. UK, 2009.

[12]. *Especificación OWL de una ontología para teleeducación en la web semántica*. L. R. Romero. Universidad Politécnica de Valencia, Departamento de Comunicaciones. España, 2007.

[13]. *Reducing SHIQ- Description Logic to Disjunctive Datalog Programs.* **Hustadt, Motik and Sattler**. Proceedings of the 9th International Conference on Knowledge Representation and Reasoning. UK, 2004.

[14]. *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. **Gomez, Fernandez, & Vicente**. Facultad de Informática, Universidad Politécnica de Madrid. España, 2009.

[15]. *Release Notes for RacerPro 1.9.2 beta*. **Racer Systems GmbH &Co**. RacerProNotes, 2007.

[16]. *Página oficial de FaCT*. **FaCT++**. http://owl.man.ac.uk/factplusplus/, 2007.

[17]. *Protege-OWL API Razonamiento*. **APIProtegé**. http://protegewiki.stanford.edu/wiki/ProtegeReasonerAPI, 2009.

[18]. *OWL Web Ontology Language Overview.* **W3C Recommendation**. http://www.w3.org/TR/owl-features/, 2008.

[19]. *Distributed Reasoning Techniques for the Semantic Web*. **Zhenning Shangguan, Joshua Shinavier, Giovanni Thenstead and Jesse Weaver**. Advanced Semantic Technology, 2009.

[20]. *Description Logic Reasoning*. **Ian Horrocks**.University of Manchester.UK, 2006.

[21]. *Introducción al Razonamiento sobre Ontologías.* **Iván J. Flores Vitelli**. Universidad Central de Venezuela. Caracas, 2011.

[22]. *Description Logics. In Handbook of Knowledge Representation*. **F. Baader, I. Horrocks, and U. Sattler**. Elsevier, 2007.

[23]. *Estudio comparativo y evaluación de razonadores para Lenguajes Ontológicos en la Web Semántica*. **Rosario Elizabeth Guamán Tandazo**. Universidad Técnica Particular de Loja. Ecuador, 2010.

[24]. *Razonadores en la Web Semántica*. **Gonzálo A. Aranda Corral**. Seminario IntegraWeb, 2007.

[25]. *Especificación OWL de una ontología para teleducación en la Web Semántica*. **Roberto Romero Llop**. Tesis Doctoral. Valencia, 2007.

[26]. *Semantics reasoners: a survey review*. **Claudia Milena Rodríguez, William Cano Montaño y José Mauricio Martínez**. INGENIUM: Revista de la Facultad de Ingeniería. Colombia, 2010

Glosario de Términos

- **Conocimiento:** El conocimiento puede ser definido como el conjunto de hechos y principios acumulados por la humanidad, o el acto, hecho o estado de conocer. Es la familiaridad con el lenguaje, conceptos, procedimientos, reglas, ideas, abstracciones, lugares, costumbres y asociaciones, unida a la habilidad de utilizar estas nociones en forma efectiva para modelar diferentes aspectos del universo que nos rodea.
- **Eficiencia:** Capacidad que tiene un lenguaje de alcanzar los objetivos y metas programadas con el mínimo de recursos disponibles y tiempo, logrando de esta forma su optimización.
- **Escalabilidad**: Propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para extender el margen de operaciones sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos. Se puede definir de igual manera como la acción de deducir una cosa a partir de otra.
- ***Framework*:** Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.
- **Inferencia:** Evaluación que realiza la mente entre expresiones bien formadas de un lenguaje (EBF), que al ser relacionadas intelectualmente como abstracción, permiten trazar una línea lógica de condición o implicación lógica entre las diferentes EBF. Determinando la verdad o falsedad de algunas de estas expresiones.
- ***Java*:** Leguaje Orientado a Objetos, permite crear programas que funcionan en cualquier equipo tipo ordenador y sistema operativo. Se usa *Java* para crear programas especiales llamados *Applets*, que se incorporan en páginas web para hacerlas interactivas.

- **Motor de inferencia:** Es un programa de control cuya función es seleccionar las reglas posibles para satisfacer el problema, para ello se vale de ciertas estrategias de control sistemático o de estrategias heurísticas.
- ***OWL*:** Lenguaje de Ontologías Web, tiene mayor capacidad para expresar significado y semántica que *XML*, *RDF*, y *RDF-S*; va más allá de estos lenguajes en su capacidad para representar contenido interpretable por un ordenador en la Web.
- **Razonamiento:** Conjunto de actividades mentales que consiste en la conexión de ideas de acuerdo a ciertas reglas y que darán apoyo o justificarán una idea.
- ***RDF*:** Originalmente diseñada como un modelo de metadatos, pero se ha vuelto un método general para el modelado de información. Permite describir metadatos en sitios web, ofreciendo interoperabilidad entre las aplicaciones que intercambian información en lenguaje máquinas por la web.
- ***RDF SCHEMA*:** (Esquema *RDF*), lenguaje primitivo de ontologías que proporciona los elementos básicos para la descripción de vocabularios.
- ***W3C*:** Siglas de *Word Wide Web Consorcio*, un consorcio fundado en 1994 para dirigir la web hacia su pleno potencial mediante el desarrollo de protocolos comunes que promueven su evolución y aseguren su interoperabilidad.
- ***XML*:** Lenguaje de programación desarrollada por *W3C*, es una versión de SGML diseñado específicamente para los documentos de la web; permite que los diseñadores creen sus propias etiquetas, dando la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones