Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

Introducción al Razonamiento Sobre Ontologías

Prof. Iván J. Flores Vitelli

ND 2011-01

Centro de Ingeniería de Software y Sistemas (ISYS)
Caracas, Abril 2011.

Resumen

Durante la última década las Ontologías son utilizadas en aplicaciones para las áreas de procesamiento del lenguaje natural, *e-comerce*, integración de información inteligente, consulta de información, integración de bases de datos, bioinformática, educación y en la web semántica, entre otras. Dichas Ontologías proporcionan un vocabulario y organización de conceptos que representan un marco de trabajo conceptual para el análisis, discusión o consulta de información de un domino. Sin embargo, existen la necesidad de realizar tareas de razonamiento en ellas, por lo cual se debe integrar módulos o herramientas de razonamiento de acuerdo al desarrollo Ontológico realizado. En este trabajo se pretende dar una introducción a las distintas maneras de hacer inferencia sobre Ontologías y a las herramientas existen para ello.

Palabaras Claves: Ontologías, Razonamiento, Lógica de Primer Orden, Lógica Descriptiva, Reglas.

Contenido

Introducción

Actualmente estamos viviendo una época donde el conocimiento y su procesamiento juegan un papel importante, llegando hasta definir el poderío de una nación. La ciencia, medicina, ingeniería y negocios impulsan una alta calidad de vida, pero para lograrlo es necesario disponer de personal altamente calificado y especializado. Por esta razón, en el mundo cada vez más se están construyendo sistemas inteligentes que capturan el conocimiento experto y razonan de manera similar a los humanos, ayudando a la resolución de estas tareas especializadas.

En el proceso de construir dichos sistemas es necesario representar el conocimiento acerca del domino, verificar la validez o consistencia de esta representación y a partir de ella, inferir nuevo conocimiento consistente y válido.

Hoy en día, existen diversos formalismos para representar conocimiento, como lo son: marcos conceptuales (Minsky, 1975), redes semánticas (Quillian, 1967), reglas de producción (Newell y Simon, 1973), lógica de primer orden y lógica descriptiva. Cada uno de estos formalismos proveen diversos niveles de expresividad referente al cocimiento y complejidad de cálculo para inferir nuevo conocimiento aplicando distintas estrategias de inferencia.

Durante la última década en las áreas de la Ingeniería del Conocimiento, Inteligencia Artificial y Ciencias de la Computación se ha utilizado las Ontologías en aplicaciones relacionadas con la representación del conocimiento, procesamiento del lenguaje natural, *e-comerce*, integración de información inteligente, consulta de información, integración de bases de datos, bioinformática, educación y en la web semántica.

Diversas metodologías, lenguajes y herramientas se utilizan para poder construir e integrar estas Ontologías con sistemas especializados en áreas específicas. En particular este trabajo se refiere a cómo se pueden utilizar las Ontologías e implementar algún mecanismo de inferencia sobre éstas, bien sea aplicando herramientas ya diseñadas para este fin o implementado algún mecanismo en particular.

Este Trabajo se divide en dos secciones. La primera, explica qué es una Ontología, sus características, diferentes formas de modelarlas y lenguajes utilizados para su representación. La segunda, se refiere a los distintas formas de realizar inferencia según el modelo utilizado y algunas herramientas diseñadas para tal fin.

1 Ontologías

El término ontología tiene su origen en la filosofía, disciplina que trata de dar una explicación sistemática de la existencia; proviene de la conjunción de los términos griegos "ontos" y "logos" que significan existencia y estudio, respectivamente. En la última década, este término ha ganado relevancia entre los Ingenieros de Conocimiento tomando una interpretación particular; y es por ello que, en 1995, Guariano y Giaretta proponen utilizar la palabra "Ontología" (con O mayúscula) para referirse a ella en el contexto de la Ingeniería del Conocimiento.

Es común que cada comunidad que desarrolla Ontologías adopte una definición propia dependiendo de sus necesidades. Entre las tantas definiciones que se pueden encontrar, la más aceptada es la propuesta por Gruber (1993): "una Ontología es una especificación formal y explícita de una conceptualización compartida". Los términos utilizados en esta definición se basan principalmente en lo siguiente:

- *Conceptualización*: Modelo abstracto de un fenómeno, que puede ser visto como un conjunto de reglas informales que restringen su estructura. Por lo general se expresa como un conjunto de conceptos (entidades, atributos, procesos), sus definiciones e interrelaciones (Uschold y Gruninger, 1996).
- *Formal*: Organización teórica de términos y relaciones usados como herramienta para el análisis de los conceptos de un dominio.
- *Compartida*:Se refiere a la captura del conocimiento consensual que es aceptado por una comunidad.
- *Explícita*: Concierne a la especificación de los conceptos y a las restricciones sobre éstos.

En el 2001, Hendler propone la siguiente definición: "una Ontología es un conjunto de términos de conocimiento, que incluye un vocabulario, relaciones y un conjunto de reglas lógicas y de inferencia sobre un domino en particular". La importancia de la definición de Handller son las relaciones y el conjunto de reglas, expresando que las Ontologías describen el significado de las relaciones entre conceptos y permiten de alguna manera formas de razonamiento.

1.1 Característica de las Ontologías

Las Ontologías exhiben características especiales para la representación del conocimiento y el procesamiento de éste. Según Chandrasekaran et al. (1999), Gruber (1993b), Guarino (1995), McGuinness (2002), y Schreiber y colaboradores (1994) destacan las siguientes:

- Las Ontologías proveen un vocabulario común y sin ambigüedades para referirse a los términos en el área aplicada, pudiéndose compartir o reutilizar éstos entre diferentes aplicaciones que hagan uso de la Ontología.

- Además de un vocabulario común, especifican una taxonomía o herencia de conceptos que establecen una categorización o clasificación de las entidades del domino. Una buena taxonomía es simple y fácil de recordar, separa sus entidades de forma mutuamente excluyente, y define grupos y subgrupos sin ambigüedad.

- El vocabulario y la taxonomía representan un Marco de Trabajo Conceptual para el análisis, discusión o consulta de información de un domino.

- Una Ontología incluye una completa generalización/especificación de sus clases y subclases, las cuales están formalmente especificadas (incluyendo sus relaciones e instancias) asegurando la consistencia en los procesos deductivos.

- Las Ontologías son implementadas en un lenguaje específico de representación Ontológica (*ontology representation languages*) de manera que la especificación de sus clases, relaciones entre éstas y sus restricciones dependerán de las características de dicho lenguaje.

1.2 Modelado de Ontologías

Las Ontologías pueden ser modeladas con diferentes técnicas de modelado de conocimiento y a su vez implementadas con diferentes lenguajes (Gómez, Fernandez y Corcho, 2004). Ellas pueden ser altamente informales, expresadas en lenguaje natural, semi-informales (expresadas en un lenguaje natural restringido y estructurado), semi-formales (expresadas en un lenguaje artificial y formalmente definido: RDF, Ontolingua, OWL, etc...) y rigurosamente-formales si poseen términos definidos meticulosamente con semántica formal, teoremas y propiedades que les dan un sentido completo.

Según Gómez y colaboradores (Gómez, Fernandez y Corcho, 2004), a principios de los 90, las Ontologías fueron construidas utilizando principalmente técnicas de representación de conocimiento en Inteligencia Artificial (IA), como son los Marcos y la Lógica de Primer Orden. En los últimos años, se han usado otras técnicas de representación de conocimiento basadas en lógica descriptiva para modelarlas y se implementan utilizando distintos lenguajes para tal fin. Otras maneras que también se han aprovechado para modelar Ontologías, son el Lenguaje de Modelado Unificado (UML) en Ingeniería de Software y Diagramas E/R, utilizados en Base de Datos para modelar conceptos y las relaciones entre éstos. A continuación se expone un breve resumen de cada una de las maneras mencionadas

- Modelado de Ontologías Utilizando Marcos y Lógica de Primer Orden

En 1993, Gruber propuso modelar Ontologías utilizando marcos y lógica de primer orden, identificando cinco clases de componentes de una Ontología:

- Clases: son la base de la descripción del conocimiento en las Ontologías, ya que expresan los conceptos (ideas básicas que se intentan formalizar) del dominio. Las clases usualmente se organizan en taxonomías pudiéndose aplicar mecanismos de herencia para su manejo.

- Relaciones: Representan las interacciones entre los conceptos del dominio. Las Ontologías por lo general contienen relaciones binarias; el primer argumento de la relación se conoce como el dominio y el segundo como el rango.

- Funciones: Son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de una Ontología.

- Instancias: Representan objetos determinados de un concepto.

- Axiomas: Se usan para modelar sentencias que son siempre ciertas. Los axiomas permiten, junto con la herencia de conceptos, inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos. En especial los axiomas son usados para verificar la consistencia de la Ontología. También son utilizados para inferir nuevo conocimiento.

- Modelado de Ontologías utilizando Lógica Descriptiva

La Lógica Descriptiva es un formalismo lógico. La teoría de la lógica descriptiva está dividida en dos partes: los TBox y los ABox. Los TBox contienen la terminología referente al conocimiento y son construidos con declaraciones que describen propiedades generales de los conceptos. Los ABox contienen conocimiento extendido, y especifican las individuos del domino del discurso. En otras palabras, los TBox contienen las definiciones de los conceptos y roles, mientras que los ABox contienen las definiciones de las instancias (Gómez, Fernandez y Corcho, 2004).

- Modelado de Ontologías utilizando UML

UML puede ser utilizado como técnica para modelar Ontologías por su fácil comprensión y uso extendido por la comunidad de personas relacionadas con el desarrollo de software y la gran cantidad de herramientas existentes para modelado con UML. UML, combinado con un Lenguaje de Relaciones entre Objetos (OCL), puede expresar la especificación de la Ontología. Generalmente los Diagramas de clases pueden ser usados para representar los conceptos y sus atributos, así como las relaciones entre los conceptos y los axiomas a través de un OCL. Los diagramas de Objetos pueden ser usados para representar las instancias (Gómez, Fernandez y Corcho, 2004).

- Modelado de Ontologías utilizando Diagramas E/R

Los diagramas de Entidad Relación Extendidos son comúnmente utilizados para modelar Ontologías donde las clases son representadas a través de las Entidades-ER y éstas a su vez pueden ser organizadas en taxonomías utilizando la relación de generalización entre las Entidades-ER. Los atributos de las clases son representados por los Atributos-ER y, a través de las Relaciones-ER entre las Entidades-ER, se pueden definir las relaciones entre las clases. Los Axiomas pueden ser representados a través de restricciones de integridad o usando notación complementaria, como lo es lógica de primer orden, reglas de producción, etc (Gómez, Fernandez y Corcho, 2004).

1.3 Lenguajes para Representar Ontologías

Ramos y Nuñez (2007) mencionan que los lenguajes para codificar Ontologías deben contemplar ciertos aspectos tales como: sintaxis bien definida, semántica específica, suficiente expresividad, fácilmente traducible entre los lenguajes ontológicos y permitir eficiencia para realizar razonamientos.

Entre los lenguajes para representar Ontologías existen XML, RDF, OIL, DAML + OIL y OWL, de los cuales la World Wide Web Consortium (W3C) recomienda RDF y OWL; éstas se describen brevemente a continuación.

Lenguaje RDF

RDF es una recomendación de la W3C para representar metadatos en la Web. Proporciona un medio para agregar semántica a un documento sin referirse a su estructura. RDF es una infraestructura para la codificación, intercambio y reutilización de metadatos estructurados (Fensel y otros, 2000).

El modelo de datos de RDF está formado por recursos (objetos) y pares de atributos/valores. Un recurso representa cualquier entidad que pueda ser referenciada por un URI (Identificador Único de Recursos). Los atributos representan las propiedades de los recursos, y sus valores pueden ser entidades atómicas (por ejemplo: strings, enteros) u otros recursos.

Un modelo RDF puede ser representado como un grafo dirigido, donde los recursos y los valores constituyen los nodos, y los atributos constituyen los arcos, formando una red semántica. En la Figura 1 se muestra un ejemplo sobre el concepto de dirección

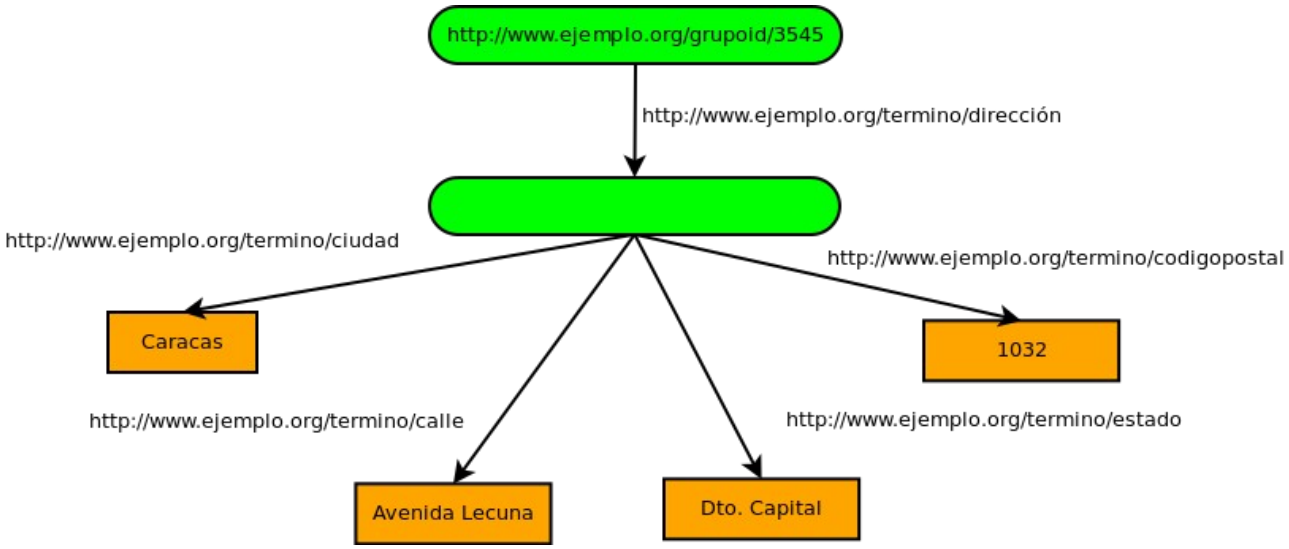

Figura 1. Grafo del concepto dirección (http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/)

El conjunto de elementos formado por un recurso, un valor y un atributo se le denomina tripleta, sentencia o *RDF Graph;* en el ejemplo anterior se tienen 5 tripletas. La primera tripleta entre los nodos http://www.ejemplo.org/grupoid/3545 y el nodo sin uri, indica que que existe el recurso denominado "grupoid/3545" y un término dirección compuesto por cuatro atributos (el nodo en blanco es utilizado como enlace para especificar los atributos de la dirección). El resto de los nodos representa los valores de los atributos de los término de la dirección. Como conclusión se puede observar que existen cinco términos en la representación: dirección, ciudad, calle, estado y código postal.

Lenguaje OWL

Es un lenguaje de marcado semántico desarrollado por la W3C para publicar y compartir Ontologías sobre la World Wide Web (http://www.w3.org/TR/owl-features/). Es una extensión del vocabulario de RDF y se deriva de DAML+OIL. OWL está diseñado para ser utilizado por aplicaciones que necesitan procesar el contenido de la información en lugar de sólo

presentarla a las personas (McGuinness y Van Harmelen, 2004). OWL proporciona tres sub-lenguajes diseñados para ser utilizados por comunidades específicas de desarrolladores y usuarios. La característica que define a cada lenguaje es su expresividad, como se explica a continuación:

• OWL Lite: Es el sub-lenguaje con sintaxis más simple, su intención es ser utilizado en situaciones donde se requiera una jerarquía de clases y restricciones simples.

• OWL DL: Es mucho más expresivo que OWL Lite y está basado en lógica descriptiva. Proporciona la máxima expresividad posible sin perder la completitud computacional (todas las conclusiones pueden ser deducidas) y la posibilidad (todos los cálculos se realizan en un tiempo finito).

• OWL Full: Es el sub-lenguaje más expresivo, su intención es ser utilizado en situaciones donde una alta expresividad es más importante que la capacidad de garantizar la completitud computacional y la posibilidad.

En la Figura 2 de la siguiente página se muestran las primitivas del Lenguaje OWL-Lite y OWL-DL

2 Razonamiento sobre Ontologías

Uno de los aspectos de la inteligencia es la capacidad de razonar, es decir, la capacidad de obtener nueva información de la ya disponible. En la IA existen diversas técnicas para hacer razonamiento aplicando diversas estrategias de inferencia sobre el conocimiento.

Para cada formalismo de representación del conocimiento existen varias técnicas que permiten la obtención de nuevo conocimiento. Éstas responden en gran parte a la naturaleza del mismo, pero también dependerá del tipo de manipulación que se vaya a realizar, según los objetivos que se persigan. Para razonar con Ontologías principalmente se utilizan tres técnicas: Razonamiento con Lógica de Primer Orden (LPO), Razonamiento con Lógica Descriptiva (DL) y Razonamiento con Reglas.

2.1 Razonamiento con Lógica de Primer Orden

En (Russell y Norvig, 2004) se indica que la Lógica de Primer Orden puede utilizarse para representar los modelos que pertenecen a un dominio particular a través de sentencias y realizar razonamiento sobre éste a través de dichas sentencias para producir nuevos modelos del domino, como se ve en la siguiente Figura 3.

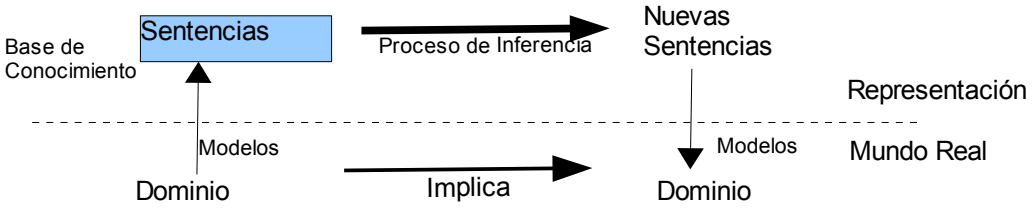

Figura 3. Razonamiento con LPO

OWL DL

Class expressions allowed in: rdfs:domain, rdfs:range, rdfs:subClassOf
owl:intersectionOf, owl:equivalentClass, owl:allValuesFrom, owl:someValuesFrom

Values are not restricted (0..N) in: owl:minCardinality, owl:maxCardinality, owl:cardinality

owl:DataRange, rdf:List, rdf:first, rdf:rest, rdf:nil

owl:hasValue (*daml:hasValue*)
owl:oneOf (*daml:oneOf*)
owl:unionOf (*daml:unionOf*), owl:complementOf (*daml:complementOf*)
owl:disjointWith (*daml:disjointWith*)

OWL Lite

owl:Ontology (*daml:Ontology*),
owl:versionInfo (*daml:versionInfo*),
owl:imports (*daml:imports*),
owl:backwardCompatibleWith,
owl:incompatibleWith, owl:priorVersion,
owl:DeprecatedClass,
owl:DeprecatedProperty

owl:Class (*daml:Class*),
owl:Restriction (*daml:Restriction*),
owl:onProperty (*daml:onProperty*),
owl:allValuesFrom (*daml:toClass*) (only with class identifiers and named datatypes),
owl:someValuesFrom (*daml:hasClass*) (only with class identifiers and named datatypes),
owl:minCardinality (*daml:minCardinality*; restricted to {0,1}),
owl:maxCardinality (*daml:maxCardinality*; restricted to {0,1}),
owl:cardinality (*daml:cardinality*; restricted to {0,1})

owl:intersectionOf (only with class identifiers and property restrictions)

owl:ObjectProperty (*daml:ObjectProperty*),
owl:DatatypeProperty (*daml:DatatypeProperty*),
owl:TransitiveProperty (*daml:TransitiveProperty*),
owl:SymmetricProperty,
owl:FunctionalProperty (*daml:UniqueProperty*),
owl:InverseFunctionalProperty (*daml:UnambiguousProperty*),
owl:AnnotationProperty

owl:Thing (*daml:Thing*)
owl:Nothing (*daml:Nothing*)

owl:inverseOf (*daml:inverseOf*),
owl:equivalentClass (*daml:sameClassAs*) (only with class identifiers and property restrictions),
owl:equivalentProperty (*daml:samePropertyAs*),
owl:sameAs (*daml:equivalentTo*),
owl:sameIndividualAs,
owl:differentFrom (*daml:differentIndividualFrom*),
owl:AllDifferent, owl:distinctMembers

RDF(S)

rdf:Property
rdfs:subPropertyOf
rdfs:domain
rdfs:range (only with class identifiers and named datatypes)
rdfs:comment, rdfs:label, rdfs:seeAlso, rdfs:isDefinedBy
rdfs:subClassOf (only with class identifiers and property restrictions)

Fgura 2. Primitivas del Lenguaje OWL-DL y OWL-LITE (Gómez y colaboradores, 2004)

La LPO se vale de la representación de los objetos, relaciones entre dichos objetos y la teoría del cálculo de predicados para realizar inferencias sobre el domino. Los elementos básicos de la LPO son símbolos que representan los objetos y relaciones, nombrados a continuación:

- Un conjunto finito de *objetos*: *{a,b,c,d,...}*
- Un conjunto finito de *variables*: *{x,y,z,x1,x2,y9,....}*
- Un conjunto finito de *constructores (Functions): {f,g,o,l,b,...}*
- Un conjunto finito *de predicados o fórmulas*: *{P,A,S,N,...}*
- El conjunto de conectivas lógicas heredadas de la lógica proposicional:*{¬, ∧, ∨, ⇒, ⇔}*
- El conjunto de cuantificadores: *{∀,∃}*
- Los paréntesis izquierdo y derecho: {(,)}
- La igualdad {=}, que a veces se considera parte de la lógica de primer orden. En ese caso, el símbolo de igualdad se incluye entre los símbolos y se comporta sintácticamente como un predicado binario. Este caso se llama a veces lógica de primer orden con igualdad.

Para representar el domino se define el vocabulario de los predicados, constructores y objetos y se construyen sentencias compuestas por términos y constructores. Un término es una variable que toma el valor de algún objeto del dominio o un constructor "*functions*" de la forma f(t1,...,tn) cuyo valor es otro término dentro del domino. Las fórmulas son predicados de la forma P(t1,...,tn) cuyo valor es evaluado como verdadero o falso. Las sentencias también pueden ser compuestas utilizando las conectivas lógicas y los cuantificadores.

Razonar con LPO consiste en determinar nuevas sentencias válidas como consecuencia lógica de las ya existentes. Para razonar con LPO se utiliza el conjunto de reglas para cálculo de predicados y la inferencia, que indica que dada una sentencia o un conjunto de ellas representadas como hipótesis se puede obtener otra u otras como conclusión al aplicar dicha regla a la hipótesis.

Para entender un poco cómo se realiza la inferencia, es necesario recordar los conceptos de equivalencia, validez y satisfacibilidad. La equivalencia significa que dos sentencias α y β son equivalentes lógicamente si tienen los mismos valores de verdad para el mismo conjunto de modelos; la validez indica que una sentencia es válida si es verdadera para todos los modelos del domino (esto se conoce también con el nombre de tautología); y por último, satisfacibilidad indica que una sentencia es satisfactoria si es verdadera para algún modelo del domino.

En general el método para realizar inferencia en en LPO es convertir las sentencias expresadas en lógica de primer orden, en sentencias que pueden ser inferidas con la lógica proposicional. Esto se traduce principalmente en inferir sentencias no cuantificadas, es decir, eliminar los cuantificadores *∀ y ∃ .* Luego de este proceso, se pueden aplicar algunos métodos de inferencia como lo es el algoritmo de resolución.

- Algoritmo de Resolución

Una idea intuitiva de la resolución es: Si se sabe que se cumple "P ó Q" y se sabe que se cumple "¬P ó R" entonces se puede deducir que se cumplirá "Q ó R".

Antes de aplicar este algoritmo es necesario aplicar ciertos mecanismos que nos permiten transformar todas las sentencias que representan nuestro domino en sentencias que pueden

ser procesadas por el algoritmo de resolución. Estos mecanismos se centran en la forma normal conjuntiva (FNC), la sustitución y la unificación.

La FNC es convertir una sentencia a otra equivalente donde sus términos estén en forma disyuntiva. Entre los pasos a seguir para pasar una sentencia a FNC se encuentra: Eliminación de las implicaciones, anidar las negaciones, estandarizar las variables, *Skolemizar*, eliminar los cuantificadores universales y distribuir la conjunción respecto a la disyunción.

Una sustitución Φ es un conjunto finito de la forma $\{x_1/o_2, x_2/o_2, ..., x_2/o_n\}$ donde cada x_i es una variable y cada o_i es un objeto o término del domino con las variables x_i distintas entre si. Dicho esto, se puede aplicar una sustitución Φ a una sentencia dada. Por ejemplo: Si se tiene la sentencia α = *Rey(x) y Codicioso(x) --> Malvado(x)* y la sustitución Φ= *{x/juan}* se obtiene la sentencia *Rey(juan) y Codicioso(juan) --> Malvado(Juan)* la cual se puede escribir como *Sust(Φ, α)*.

En cuanto a la unificación, ésta consiste en encontrar una sustitución *Φ* que haga que sentencias diferentes se hagan idénticas; ahora bien, si existen distintos unificadores se debe encontrar el unificador más general.

Explicado los mecanismos de FNC, sustitución y unificación; se pude describir el algoritmo de resolución para LPO, el cual se basa en la siguiente regla:

$$\frac{l_1 \vee \cdots \vee l_k, m_1 \vee \cdots \vee m_n}{Sust(\theta, l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

donde $\theta = Unificar(l_i, \neg m_j)$; *l* es un literal y l_i y *m* son literales complementarios.

La idea del algoritmo de resolución es mostrar la validez de una proposición estableciendo que la negación de la proposición es insatisfacible.

El algoritmo es el siguiente:

Función Resolución(BC,p) devuelve lógico

// entradas BC, base de conocimiento, p sentencia a probar

clausulas= el conjunto de sentencias de la BC en FNC y ¬p;

satisfacible = verdad;

Mientras Existan dos sentencias S_i y S_j tales que contienen dos términos t_i y t_j unificables, y tales que, después de haber aplicado la sustitución correspondiente permite aplicar la regla de resolución entonces

θ = unificar(t_i , ¬t_j);

S_{nueva} = Sust(θ, Regla(S_i , S_j)); ***// Regla() aplica la regla de resolución***

Si S_{nueva} == vacía entonces satisfacible = falso;

clausulas = clausulas U S_{nueva};

Fmientras;

Ffunción Resolución;

Como ejemplo supóngase que se tiene el siguiente enunciado: "La ley dice que es un crimen para un americano vender armas a países hostiles. El país de Nono, un enemigo de América, tiene algunos misiles, y todos sus misiles fueron vendidos por el Coronel West, que es americano." y queremos probar que West es un criminal (Russell y Norvig, 2004).

Vocabulario

Americano(x): x es americano

Arma(y) :y es un arma

Vende(x,y,z): x vende arma y a país z

Criminal (x): x es un Criminal

Hostil(z): z es país hostil

Tiene(y,x) : país y tiene arma x

Misil (x): x es un misil

Enemigo (x,América): x es enemigo de América

West: Coronel West

Nono: País de Nono

Se define el vocabulario del domino

Base de Conocimiento

S_1: $Americano(x) \wedge Arma(y) \wedge Vende(x, y, z) \wedge Hostil(z) \Rightarrow Criminal(x)$

S_2: $\exists x\, Tiene(Nono, x) \wedge Misil(x)$

S_3: $Misil(x) \wedge Tiene(Nono, x) \Rightarrow Venede(West, x, Nono)$

S_4: $Misil(x) \Rightarrow Arma(x)$

S_5: $Enemigo(x, América) \Rightarrow Hostil(x)$

S_6: $Americano(West)$

S_7: $Enemigo(Nono, América)$

Se construye la base de conocimiento definiendo las sentencias.

Sentencia P

p: $Criminal(West)$

Se define la sentencia a probrar.

Aplicación del algoritmo de resolución

1. FNC de BC

S_1: $\neg Americano(x) \vee \neg Arma(y) \vee \neg Vende(x, y, z) \vee \neg Hostil(z) \vee Criminal(x)$

S_2: $\neg Misil(x) \vee \neg Tiene(Nono, x) \vee Vende(West, x, Nono)$

S_3: $\neg Enemigo(x, América) \vee Hostil(x)$

S_4: $\neg Misil(x) \vee Arma(x)$

S_5: $Teine(Nono, M_1)$

S_7: $Misil(M_1)$

S_8: $Americano(West)$

S_9: $Enemigo(Nono, América)$

Se transforma la Base de Conocimiento a Forma Normal Conjuntiva.

2. Resolución(BC,p)

Clausulas

S_1: $\neg Americano(x) \vee \neg Arma(y) \vee \neg Vende(x, y, z) \vee \neg Hostil(z) \vee Criminal(x)$

S_2: $\neg Misil(x) \vee \neg Tiene(Nono, x) \vee Vende(West, x, Nono)$

S_3: $\neg Enemigo(x, América) \vee Hostil(x)$

S_4: $\neg Misil(x) \vee Arma(x)$

S_5: $Teine(Nono, M_1)$

S_7: $Misil(M_1)$

S_8: $Americano(West)$

S_9: $Enemigo(Nono, América)$

S_{10}: $\neg Criminal(West)$

Se aplica el algoritmo de resolución a la Base de Conocimiento en FNC junto con la sentencia p negada a probar.

Ciclo 1

Seleccionar S_1,S_{10}

unificar($Criminal(x)$, $\neg\neg Criminal(West)$) = {x/West}

θ = {x/West}

Sust(θ, Regla(S_1 ,S_{10}))= $\neg Americano(west) \vee \neg Arma(y) \vee \neg Vende(West, y, z) \vee \neg Hostil(z)$

S_{nueva} = $\neg Americano(west) \vee \neg Arma(y) \vee \neg Vende(West, y, z) \vee \neg Hostil(z)$

Clausulas

S_1: $\neg Americano(x) \vee \neg Arma(y) \vee \neg Vende(x, y, z) \vee \neg Hostil(z) \vee Criminal(x)$

S_2: $\neg Misil(x) \vee \neg Tiene(Nono, x) \vee Vende(West, x, Nono)$

S_3: $\neg Enemigo(x, América) \vee Hostil(x)$

S_4: $\neg Misil(x) \vee Arma(x)$

S_5: $Teine(Nono, M_1)$

S_7: $Misil(M_1)$

S_8: $Americano(West)$

S_9: $Enemigo(Nono, América)$

S_{10}: $\neg Criminal(West)$

S_{11}: $\neg Americano(west) \vee \neg Arma(y) \vee \neg Vende(West, y, z) \vee \neg Hostil(z)$

Ciclo 2

Seleccionar S_8,S_{11}

unificar($Americano(West)$, $\neg\neg Americano(West)$) = {ø}

θ = {ø}

Sust(θ, Regla(S_8 ,S_{11}))= $\neg Arma(y)\vee\neg Vende(West, y, z)\vee\neg Hostil(z)$

S_{nueva} = $\neg Arma(y)\vee\neg Vende(West, y, z)\vee\neg Hostil(z)$

Clausulas = Clausulas U S_{nueva} = (S_{12})

Ciclo 3

Seleccionar S_4,S_{12}

unificar($Arma(x)$, $\neg\neg Arma(y)$) = {x/y}

θ = {x/y}

Sust(θ, Regla(S_4 ,S_{12}))= $\neg Misil(y)\vee\neg Vende(West, y, z)\vee\neg Hostil(z)$

S_{nueva} = $\neg Misil(y)\vee\neg Vende(West, y, z)\vee\neg Hostil(z)$

Clausulas = Clausulas U S_{nueva} = (S_{13})

Ciclo 4

Seleccionar S_7,S_{13}

unificar($Misil(M_1)$, $\neg\neg Misil(y)$) = {y/M_1}

θ = {y/M_1}

Sust(θ, Regla(S_7 ,S_{13}))= $\neg Vende(West, M_1, z)\vee\neg Hostil(z)$

S_{nueva} = $\neg Vende(West, M_1, z)\vee\neg Hostil(z)$

Clausulas = Clausulas U S_{nueva} = (S_{14})

Ciclo 5

Seleccionar S_2,S_{14}

unificar($Vende(West, x, Nono)$, $\neg\neg Vende(West, M_1, z)$) = {x/M_1, z/Nono}

θ = {x/M_1, z/Nono}

Sust(θ, Regla(S_2 ,S_{14}))= $\neg Misil(M_1)\vee\neg Tiene(Nono, M_1)\vee\neg Hostil(Nono)$

S_{nueva} = $\neg Misil(M_1)\vee\neg Tiene(Nono, M_1)\vee\neg Hostil(Nono)$

Clausulas = Clausulas U S_{nueva} = (S_{15})

Ciclo 6

Seleccionar S_7,S_{15}

unificar($Misil(M_1)$, $\neg\neg Misil(M_1)$) = {ø}

θ ={ø}

Sust(θ, Regla(S_7 ,S_{15}))= $\neg Tiene(Nono, M_1) \vee \neg Hostil(Nono)$

S_{nueva} = $\neg Tiene(Nono, M_1) \vee \neg Hostil(Nono)$

Clausulas = Clausulas U S_{nueva} = (S_{16})

Ciclo 7

Seleccionar S_5,S_{16}

unificar($Tiene(Nono, M_1)$, $\neg\neg Tiene(Nono, M_1)$) = {ø}

θ ={ø}

Sust(θ, Regla(S_5 ,S_{16}))= $\neg Hostil(Nono)$

S_{nueva} = $\neg Hostil(Nono)$

Clausulas = Clausulas U S_{nueva} = (S_{17})

Ciclo 8

Seleccionar S_3,S_{17}

unificar($Hostil(x)$, $\neg\neg Hostil(Nono)$) = {x/Nono}

θ ={x/Nono}

Sust(θ, Regla(S_3 ,S_{17}))= $\neg Enemigo(Nono, América)$

S_{nueva} = $\neg Enemigo(Nono, América)$

Clausulas = Clausulas U S_{nueva} = (S_{18})

Ciclo 9

Seleccionar S_9,S_{18}

unificar(, $Enemigo(Nono, América)$, $\neg Enemigo(Nono, América)$) = {ø}

θ = {ø}

Sust(θ, Regla(S_9 ,S_{18}))= ø

S_{nueva} = ø

Clausulas = Clausulas U S_{nueva} = (S_{19})

Finalmente como S_{19} = ø entonces Resolución(BC,p) = Falso, por lo tanto es insatisfacible y en consecuencia *Criminal(West)* es verdad; quedando demostrado que el Coronel West es un Criminal.

2.2 Razonamiento con Lógica Descriptiva (DL)

La Lógica Descriptiva (DL) es un formalismo de representación del conocimiento caracterizado por describir los conceptos relevantes de un domino particular y utilizar éstos para especificar las propiedades de los objetos e individuos del dominio (Baader y colaboradores, 2003).

Básicamente un Sistema Basado en Lógica Descriptiva (SBDL) se caracteriza por tener una base de conocimiento (KB) compuesta por dos componentes. El primero, el Tbox, el cual indica la terminología del dominio. El segundo, el Abox, que contiene las aserciones acerca de los individuos particulares del domino con base en el vocabulario expresado en la terminología. En la Figura 4, se puede observar la arquitectura de un SBDL.

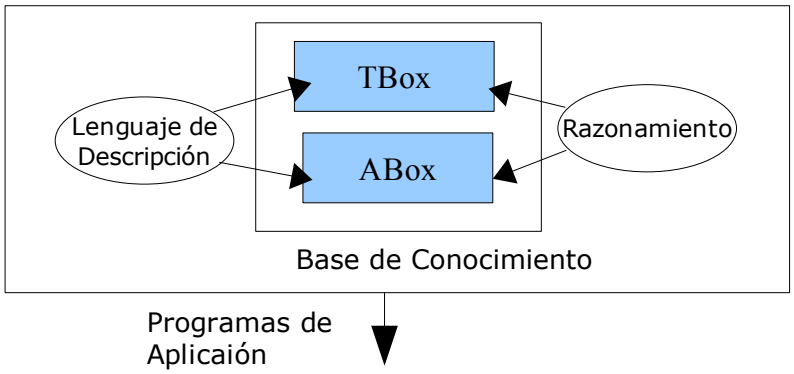

Figura 4. Arquitectura de un SBDL.

Un vocabulario consiste en una serie de conceptos y roles. Los conceptos especifican a los individuos y los roles a las relaciones entre éstos. Para construir un SBDL se utiliza un lenguaje descriptivo particular el cual caracteriza un SBDL de otro.

Además, los SBDL brindan tareas de razonamiento acerca de la terminología. Generalmente una tarea de razonamiento indica si una aserción se satisface (no es contradictoria) o una es más general que otra, es decir, si una subsume a la otra.

- Lenguajes de Descripción

Los descriptores elementales de un Lenguaje de Descripción (LD) son los conceptos y los roles atómicos. Los descriptores complejos pueden ser construidos inductivamente por conceptos constructores. Los LD se diferencian según los constructores que provean. En la Tabla 1 se muestran los distintos lenguajes con sus constructores, sintaxis y nombres correspondientes.

Para construir un SBDL se utiliza la combinación de cualquiera de estos lenguajes. Por ejemplo, SHIQ es un lenguaje que combina 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.

Constructor	**Sintaxis**	**Lenguaje**			
Concepto	A	FL_0	FL^-	AL	S
Rol	R				
Intersección	C $\cap$ D				
Valor de Restricción	$\forall$ R.C				
Cuantificador Existencial	$\exists$ R				
Universo o Tope	T				
Fondo o Vacío	$\perp$				
Negación Atómica	$\neg$ A				
Negación	$\neg$ C			C	
Unión	C $\cup$ D			U	
Restricción Existencial	$\exists$ R.C			E	
Restricción Numérica	($\geq$ n R) ($\leq$ n R)			N	
Nominales	{a1,....,an}			O	
Herencia de Roles	R $\subseteq$ S			H	
Rol Inverso	R^-			I	
Restricción Numérica Calificada	($\geq$ n R.C) ($\leq$ n R.C)			Q	

Tabla 1. Lenguajes de Descripción (Gómez, 2004)

- *Tareas de razonamiento*

Existen diferentes tipos de razonamiento para los SBDL y éstos están basados principalmente en la inferencia lógica. Cuando se modela un dominio se construye una terminología T. Si se desea construir nuevos conceptos en base a T seguramente es necesario verificar si son consistentes o contradictorios respecto a T, propiedad conocida como Satisfacibilidad. También, puede ser necesario saber si un concepto es más general que otro (subsunción) o si es equivalente o disjunto. Formalmente se definen estas propiedades como sigue:

Si T es un Tbox, e I es una función de interpretación en un conjunto no vacío del domino del discurso, se tiene:

- *Satisfacibilidad:* Un Concepto C es Satisfacible respecto a T si existe un modelo I de T tal que C^I es no vacío. En este caso se dice también que I es un modelo de C.

- Subsunción: Un concepto C se subsume a otro concepto D con respecto a T si $C^I \subseteq D^I$ para todo modelo I de T. En este caso se escribe T $\models$ C $\subseteq$ D.

- Equivalencia: 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:* Dos conceptos C y D son disjuntos respecto a T si $C^I \cap D^I = \varnothing$ para todo modelo I de T.

Después de haber diseñado la terminología del Tbox y verificado la satisfacibilidad y subsunción de los conceptos, se construyen el Abox con dos tipos de aserciones sobre los individuos; los conceptos de aserción y los roles de aserción con la forma C(a) y R(a,b) respectivamente, donde a y b son individuos, C es el concepto y R es el rol.

Definidos los conceptos de Subsunción, Satisfacibilidad, Eqivalencia y Disyunción se puede describir el mecanismo básico de razonamiento de todos los SBDL. Este mecanismo consiste en verificar la subsunción de conceptos, lo cual es suficiente para verificar otras relaciones entre conceptos según la siguiente proposición:

- Reducción por Subsunción

- C es insatisfacible sii C se subsume a $\perp$
- C y D son equivalentes sii C se subsume a D y D se subsume a C
- C y D son disjuntos sii C $\cap$ D se subsume a $\perp$

Otra forma de comprobar la subsunión es a través de la *insatisfacibilidad* mediante la siguiente proposición:

- Reducción por insatisfacibilidad

- C se subsume a D sii C $\cap \neg$ D es insatisfacible
- C y D son equivalentes sii (C $\cap \neg$ D) y ($\neg$ C $\cap$ D) son insatisfacibles
- C y D son disjuntos sii C $\cap$ D es insatifacible

Generalmente los algoritmos utilizados para realizar este tipo de razonamiento son de alta complejidad, entre los cuales se encuentran los algoritmos estructurales por subsunción y los basados en tableaux. Para poder aplicar estos algoritmos y reducir la complejidad asociada a ellos, es necesario aplicar ciertas técnicas de optimización tales como normalización, absorción, optimización de clasificaciones, entre otras (Baader y colaboradores, 2003).

2.3 Razonamiento con Reglas

Los sistemas basados en reglas fueron desarrollados por Newell y Simmon (1973). Este tipo de representación de conocimiento se relaciona más con las heurísticas y formas de proceder de los expertos en contraposición de las representaciones declarativas de los esquemas anteriores (Alonso y colaboradores, 2004).

La arquitectura de este tipo de representación de conocimiento generalmente consta de una base de hechos, una base de reglas y un motor o máquina de inferencia como indica la Figura 5.

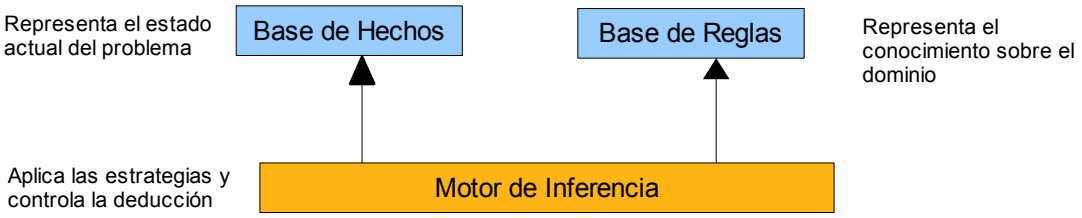

Figura 5. Razonamiento con Reglas

La base de hechos se modifica a medida que se apliquen los procesos de inferencia y pasa por tres estados:

1. Estado Inicial, que representa la situación original del problema
2. Estados finales, que representan las situaciones objetivos que se pretenden alcanzar.
3. Estados Intermedios o memoria de trabajo que contiene la descripción de la solución en curso de resolución. La ejecución de las reglas modifica este estado y añade o elimina hechos.

La base de reglas contiene la parte fundamental de la base de conocimiento y las reglas describen los elementos de deducción básico que utilizará el sistema. La forma más sencilla de representación es del tipo "*SI condiciones Entonces acciones*"

El motor de inferencia actúa sobre los dos componentes anteriores aplicando una estrategia de deducción determinada y controlando el proceso de deducción. El funcionamiento básico del proceso de deducción consiste en seleccionar alguna regla que pueda aplicarse a una situación actual, hasta que los hechos de la base de hechos satisfagan una condición de terminación o se ejecute una regla de parada.

La estrategia controla que sea aplicada una única regla, seleccionando la más adecuada y luego se activa, incorporando o eliminando hechos de la base de hechos. La fase de selección es la más compleja y comprende las siguientes etapas:

1. *Restricción:* Su finalidad es acotar el conjunto de reglas que deben examinarse para mejorar la eficiencia del sistema.
2. *Equiparación o filtrado:* Consiste en examinar las reglas que han pasado sobre la etapa de restricción, evaluando la condición en la base de hechos para examinar las reglas que pueden ser aplicadas. Para ello, se realiza una equiparación de variables con constantes, y se construye el conjunto conflicto de reglas que se pueden aplicar.
3. *Resolución del conjunto conflicto:* De las reglas pertenecientes al conjunto conflicto se debe seleccionar la regla a ser aplicada, para ello debe utilizarse alguna o varias estrategias de decisión.

Las estrategias de decisión pueden ser la siguientes:

- Orden Lineal Explícito. Cada regla lleva asociada una prioridad y se selecciona según la prioridad.
- Selección de la regla más específica. Se selecciona la regla con más elementos en la condición.
- Selección de la regla más General. Se selecciona la regla con menos elementos en la condición.
- Selección de la regla en donde los hechos en la base de hechos sean los más nuevos o

los más antiguos.

- Aplicar el principio de refracción. Una regla no puede ser aplicada de nuevo a menos que exista un conjunto conflicto que no contenga la regla entre el ciclo actual y el ciclo que aplicó la regla. Con esto se logra que no se creen bucles de deducción que no llevarían a ninguna solución.

La estrategia de deducción tiene dos alternativas: encadenamiento hacia adelante o encadenamiento hacia atrás. En el encadenamiento hacia adelante, el motor de inferencia crea el conjunto conflicto con aquellas reglas cuyo antecedente satisfacen la base de hechos actual. En el proceso de resolución, se selecciona una regla y se ejecutan las acciones especificadas en el consecuente. En cambio, en el encadenamiento hacia atrás se trata de probar una premisa partiendo de una situación final o un hecho que se debe llegar a demostrar. Lo que se analiza es el consecuente desde el estado objetivo. Si coincide con la base de hechos se termina la búsqueda; si no, se siguen buscando reglas recursivamente intentando resolver el antecedente nuevo de las reglas dispuesto en la base de hechos. La meta final es reemplazar secuencialmente por la conjunción de submetas equivalentes.

Para ilustrar el razonamiento con reglas, supongamos que se tienen dos variables *x* e *y;* las constantes A,B,C,D; una base de hechos con el conjunto de constantes (A,C,B) y una base de reglas definida así (Alonso y colaboradores, 2004):

R1: Si (A,$x,B) $\wedge$ (B,$y,C) entonces Añadir (A,$x,$y) y Borrar(A,$x,B)
R2: Si (A,C,$x) entonces Añadir (B,$x,D)
R3: Si (B,B,D) $\wedge$ (A,$x,$y) Entonces Añadir(A,$x,$y) y Añadir(A,B,B)
R4: Si (A,$x,B) entonces Añadir($x,C,C)

donde Borrar() y Añadir() significa borrar y añadir el conjunto dado a la base de hechos respectivamente. Además, el motor de inferencia aplica el principio de refracción junto con orden lineal explicito respecto al ordinal de la regla al proceso de inferencia. Si se quiere determinar el hecho (A,C,C) el proceso de inferencia sería el siguiente:

Base Hecho: {(A,C,B)}

Ciclo 1
Conjunto Conflicto {R2 con $x=B , R4 con $x=C}
Selección de la Regla (R2 con $x=B)
Base Hecho: {(A,C,B) , (B,B,D)}
Ciclo2
Conjunto Conflicto {R2 con $x=B, R3 con $x=C y $y=B, R4 con $x=C}
Selección de la Regla (R3 con $x=C y $y=B)
Base de Hecho: { (A,C,B),(B,B,D),(A,B,B) }
Ciclo3
Conjunto Conflicto {R2 con $x=B, R3 con $x=C y $y=B, R3 con $x=B y $y=B, R4 con $x=C , R4 con $x=B}
Selección de la regla (R3 con $x=B y $y=B)
Base de Hecho: { (A,C,B),(B,B,D),(A,B,B)}
Ciclo4
Conjunto Conflicto { R2 con $x=B, R3 con $x=C y $y=B, R3 con $x=B y $y=B, R4 con $x=C , R4 con $x=B }
Selección de la regla (R4 con $x=C)
Base de Hecho: { (A,C,B),(B,B,D),(A,B,B),(C,C,C)}

Ciclo5
Conjunto Conflicto {R2 con $x=B, R3 con $x=C y $y=B, R3 con $x=B y $y=B, R4 con $x=C , R4 con $x=B}
Selección de la regla (R4 con $x=B)
Base de Hecho: { (A,C,B),(B,B,D),(A,B,B),(C,C,C),(B,C,C)}
Ciclo6
Conjunto Conflicto {R1 con $x=C y $y=C, R1 con $x=B y $y=C, R2 con $x=B, R3 con $x=C y $y=B, R3 con $x=B y $y=B, R4 con $x=C , R4 con $x=B}
Selección de la regla (R1 con $x=C y $y=C)
Base de Hecho: { (B,B,D),(A,B,B),(C,C,C),(B,C,C),**(A,C,C)**}

2.4 Herramientas de razonamiento utilizadas con Ontologías

En las siguientes secciones se expone brevemente tres herramientas distintas de razonamiento utilizadas para el desarrollo de aplicaciones que hacen uso de representaciones Ontológicas.

2.4.1 JENA

JENA es un Framework de Java para desarrollar aplicaciones de Web Semántica. Ella provee un ambiente para manipulación de Ontologías representadas en RDF, RDFS y OWL (http://jena.sourceforge.net/index.html).

Su arquitectura puede verse en la Figura 6.

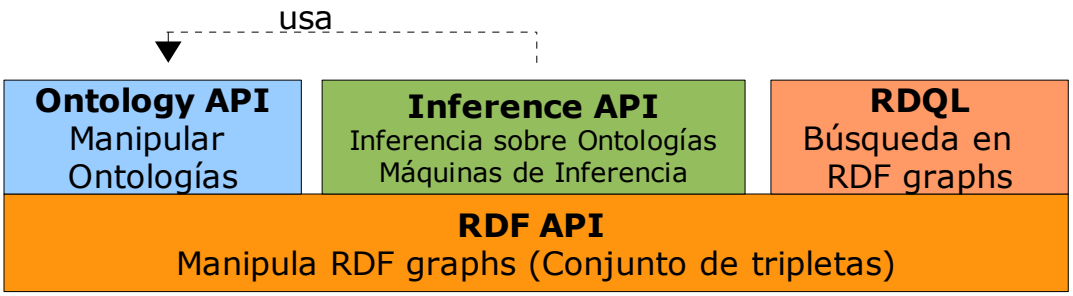

Figura 6. Arquitectura de JENA

JENA consta de cuatro Módulos: *Inference API* o *API de Inferencia*, utilizado para inferir nuevo conocimiento sobre representaciones RDF; *RDF API*, para manipular los RDF graph (conjunto de tripletas RDF); *Ontology API*, para manejar Ontologías representadas en RDF y OWL; *RDQ*, para realizar búsquedas sobre los RDF graphs.

- *La API de Inferencia*

La API de inferencia se caracteriza por ofrecer distintos tipos de razonadores para Ontologías e intenta crear un modelo de Inferencia, el cual inferirá los hechos sobre un modelo dado. En la Figura 7 se puede visualizar cómo es este proceso.

La Inferencia, se hace en dos etapas. Primero, se debe configurar la máquina de inferencia indicando las reglas y la forma en que se realizará la inferencia para instanciar un razonador. Segundo, una vez instanciado el razonador, se suministran los hechos y se obtiene un modelo de inferencia que inferirá los nuevos hechos.
Para razonar con Ontologías es necesario que ésta sea cargada en el Framework, haciendo uso

de la API de Ontología, para que JENA intente crear el modelo RDF que utilizará.

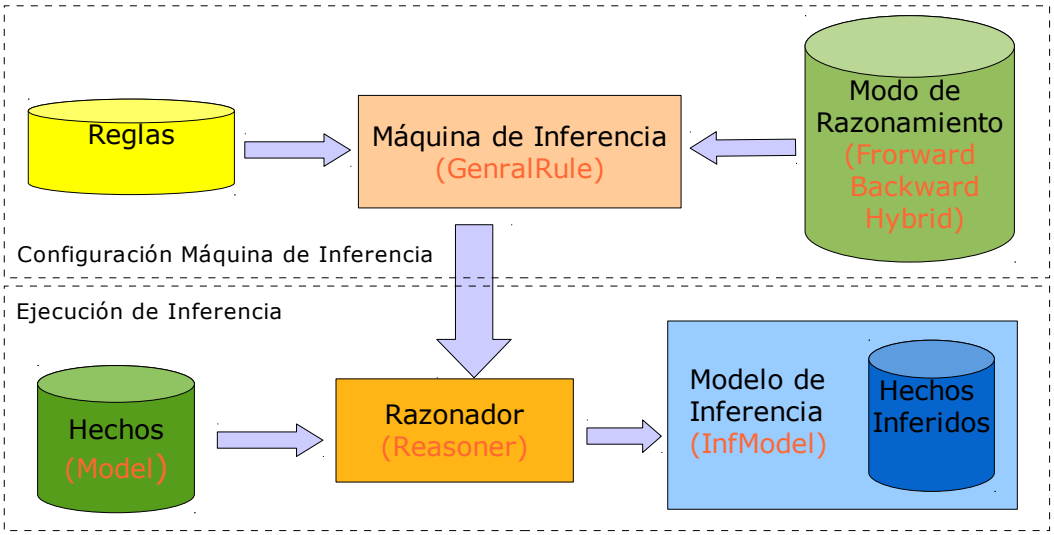

Figura 7. Proceso para realizar inferencia en JENA

Razonadores de JENA

a) *Razonador general de Reglas*
Es un razonador general basado en reglas y en éste se basan los demás razonadores que ofrece JENA. Este razonador puede inferir hechos utilizando encadenamiento hacia adelante, hacia atrás e híbrido. La estructura de las reglas están definidas por una lista de antecedentes (body), una lista de consecuentes (head), un nombre para la regla (opcional) y una dirección que representa la forma en que se resuelve la regla.

En el la Figura 8 se puede ver la sintaxis general de una regla.

b) *Razonador RDF(S)*
Razonador basado en reglas que soporta todas las implicaciones sobre RDFS.

Puede funcionar de tres modos diferentes:

- Full: implementa RDFS axiomas y Reglas. Este es uno de los modos más costosos debido a que los datos en Graph deben ser chequeados junto con sus propiedades. También genera todas las aserciones de todos los recursos existentes en los datos.

- Default: Este omite los chequeos realizados por el modo Full.

- Simple: sólo maneja la transitividad entre Subclases y Subpropiedades y omite los axiomas.

```
Regla :=   bare-rule | [ bare-rule ] | [ nombre : bare-rule ]

bare-rule :=   term, ... term -> hterm, ... hterm    // forward rule
       | bhterm <- term, ... term    // backward rule

hterm    :=   term | [ bare-rule ]

term     :=   (node, node, node)     // triple pattern
       | (node, node, functor)       // extended triple pattern
       |  builtin(node, ... node)    // invoke procedural primitive

bhterm     :=   (node, node, node)       // triple pattern

functor  :=   functorName(node, ... node)  // structured literal

node     :=   uri-ref                    // e.g. http://foo.com/eg
       | prefix:localname                // e.g. rdf:type
       |  <uri-ref>                      // e.g. <myscheme:myuri>
       |  ?varname                       // variable
       |  'a literal'                    // a plain string literal
       |  'lex'^^typeURI                 // a typed literal, xsd:* type names
supported
       |  number                         // e.g. 42 o 25.5,
```

Figura 8. Sintaxis general de una regla en JENA

c) *Razonador OWL*
Razonador para OWL que sólo puede ser utilizado para razonar sobre OWL/Lite.

d) *Razonador Transitivo*
Éste es utilizado para soportar datos almacenados con clases y propiedades. La implementación sólo soporta propiedades de rdfs:subPropertyOf o rdfs:subClassOf.

- Operaciones con el modelo de inferencia (InfModel)

- *Validación*: Verificar si existen inconsistencias sobre la ontología a través del modelo de inferencia.

- *Lista de sentencias extendidas:* La API de JENA generalmente accede a la información de las tripletas. Pero si se desea hacer referencia a expresiones que no están descritas en el modelo de datos, las búsquedas de JENA no lo permiten. Esto se soluciona a través de la API de Inferencia para definir un conjunto de aserciones que serán usadas temporalmente.

- *Relaciones directas e indirectas:* Son operaciones para obviar las relaciones indirectas y sólo operar con relaciones directas.

- *Deducciones:*Se refiere a crear un conjunto de nuevas reglas a partir del conjunto de reglas dadas.

- *Acceso a los datos originales y deducciones:* En ocasiones es necesario obtener el modelo del conjunto de datos originales o el modelo del conjunto de datos con sus deducciones.

- *Control de Procesamiento:* En algunas ocasiones es necesario cambiar el modo como la máquina de inferencia trabaja para obtener más rendimiento o menos uso de la memoria o volver a la forma original de trabajo.

2.4.2 PELLET

Es una herramienta de razonamiento para Ontologías especificadas en OWL-DL basadas en los LD SHOINQ y SROIQ. PELLET es implementado en Java y de libre distribución a través del sitio web: http://pellet.owldl.com. Ofrece una variedad de tareas de razonamiento y compatibilidad con otras herramientas de razonamiento como por ejemplo JENA.

Los componentes de PELLET pueden observarse en la Figura 9. Pallet en sí es un *core* para razonar sobre lógica descriptiva, basado en distintas implementaciones del algoritmo tableux, para chequear la consistencia de la base de conocimiento y otros servicios.

Además PELLET implementa distintas técnicas de optimización para DL como: Normalización, Simplificación, Absorción, Ramificación Semántica, *Backjumping*, Estatus de Cache Satisfacible, Búsqueda *Top-Bottom* para clasificación y Mezcla de Modelos (Baader y colaboradores, 2003).

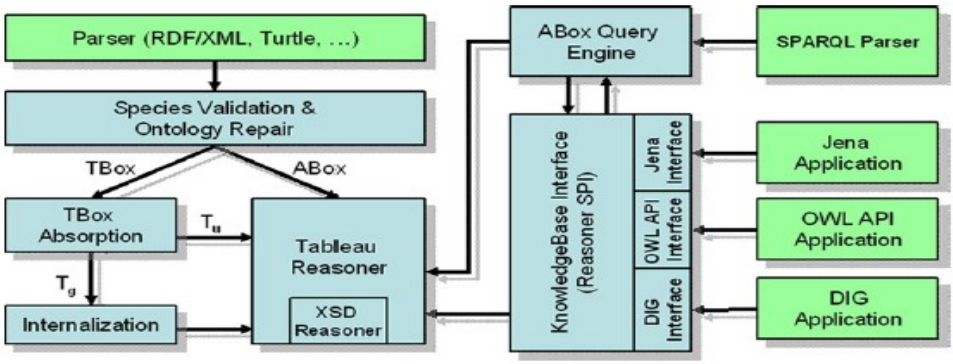

Figura 9. Arquitectura de PELLET (Sirin y colaboradores, 2007)

Sirin y colaboradores (2007) expone las siguientes características y capacidades de PELLET:

- *ABOX query:* PELLET incluye una sistema de búsqueda que puede responder querys expresados en SAPRQL o RDQL. Así como también optimizarlos o reordenarlos.

- *Razonamiento con tipos de datos*: Pallet soporta razonamiento sobre tipos de datos construidos por el usuario siempre y cuando éstos estén basados en los tipos de datos numéricos y/o fecha-tiempo, además de los tipos de datos definidos por XML Schema.

- *Refinamiento de axiomas*: El refinamiento de axiomas no es un estándar de inferencia

sobre DL, se refiere a la justificación sobre cualquier aserción dada por el razonador sobre la Base de Conocimiento. Es decir, dada una Ontología y cualquiera de sus axiomas lógicos, el refinamiento de axiomas indica las premisas suficientes para alcanzar este axioma. Esto puede ser usado con fines de mejorar o corregir la Ontología.

- *Integración con el formalismo de reglas*: Acoplamiento del Log de datos del razonador con la aplicación de Reglas-DL. Esto se refiere a permitir que las clases DL puedan ser usadas en el cuerpo de una regla.

- *Razonamiento con múltiples Ontologías (E-connect):* Pallet implementa un mecanismo denominado E-connect para realizar conexiones a múltiples Ontologías y poder razonar sobre múltiples Ontologías.

2.4.3 FaCT++

Fact++ es una herramienta experimental basada en desarrollo de algoritmos *tableux* recientes e implementaciones de nuevas técnicas de optimización para razonamiento en Ontologías especificadas en OWL-DL . Esta herramienta está implementada usando librerías en C++ bajo licencia de libre distribución. Para tener acceso a esta herramienta se puede acceder en http://code.google.com/p/factplusplus/.

Tsarkov y Horrocks (2006) explican que FaCT++ usa el estándar de interfaz DIG para OWL (http://dig.sourceforge.net/) y funciona de la siguiente manera: para razonar con Fact++ el sistema primero preprocesa la Ontología dada donde es normalizada y transformada en una representación interna aplicando una serie de optimizaciones. Luego, se crea la clasificación de conceptos, es decir, se establece la subsunción. Aquí, también se utilizan técnicas de optimización para reducir la subsunción de conceptos. Finalmente, el clasificador utiliza el chequeo de satisfacibilidad para resolver los problemas de subsunción para dos pares de conceptos dados. Este clasificador es el core del sistema el cual está altamente optimizado para realizar razonamiento.

Conclusiones

El uso de las Ontologías proporcionan un vocabulario y una organización de conceptos particular que representan un marco de trabajo conceptual para el análisis, discusión o consulta de información de un domino. Sin embargo, existen necesidades que implican el uso de tareas de razonamiento en dichas Ontologías; bien sea para la verificación del modelo Ontológico, inferir nuevo conocimiento o resolver problemas análogamente como lo haría un experto en el área.

Para realizar estas tareas de razonamiento en Ontologías, es necesario conocer el tipo de formalismo de representación y el lenguaje utilizado para su desarrollo. El formalismo de representación indica la estrategia de inferencia a utilizar y el lenguaje la expresividad y complejidad del cálculo a realizar.

En la actualidad, existen diversas herramientas desarrolladas para realizar tareas de inferencia con Ontologías, pero éstas no son de caracter generalista, cada una de ellas presentan diferentes características que ayudaran o no a alcanzar los objetivos. Entre las características que se deben analizar se tiene:

- El lenguaje de representación Ontológica que puede interpretar (RDF, RDFS, OWL-LITE, OWL-DL, Etc..).

- Mecanismos de optimización que utiliza para interpretar la Ontología, ya que éstos tendrán consecuencia en la complejidad de cálculo, es decir, el calculó se podrá realizar en tiempos aceptables.

- Las estrategias de inferencia en que se basan su funcionamiento, para saber si el modelo ontológico es adecuado para la herramienta.

- El lenguaje en que fue desarrollado a efectos de compatibilidad con la aplicación.

Referencias

Alonso Amparo, Güijarro Bertha, Lozano Tello Adolfo, Palma José y Tabeada Maria. (2004). “Ingeniería del Conocimiento Aspectos Metodológicos”. Pearson Prentice Hall. IBSN84- 205-4192-3. España, pp. 30-36.

Chandrasekaran, B., Josephson, J.R., & Benjamins, V.R. (1999), What are ontologies, and why do we need them? IEEE Intelligent Systems , Vol. 14, no. 1, pp. 20–26.

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, (2003), The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, Cambridge, UK, ISBN 0-521-78176-0

Fensel D, Horrocks I, Van Harmelen F, Decker S, Erdmann M y Klein M. (2000). OIL in a Nutshell. In Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW-2000).

Gómez-Pérez A, Fernández-López M y Corcho M. (2004). Ontological Engineering. Springer Verlag London.

Gruber T. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Available as Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University. Disponible en fecha febrero de 2007 en http://citeseer.ist.psu.edu/gruber93toward.html

Gruber, T. (1993b), A translation approach to portable ontology specifications, Knowledge Acquisition , Vol. 5, no. 2, pp. 199–220.

Guarino N, Giaretta P (1995), Ontologies and Knowledge Bases: Towards a Terminological Clarification . In: Mars N (ed) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing (KBKS’95). University of Twente, Enschede, The Netherlands. IOS Press, Amsterdam, The Netherlands, pp 25–32

Guarino, N. (1995), Formal ontology, conceptual analysis and knowledge representation, International Journal of Human-Computer Studies , Vol. 43, no. 5/6, pp. 625–640.

Hendler, J. (2001), Agents and the semantic web, IEEE Intelligent Systems , Vol. 16, no. 2, pp. 30–37.

McGuinness D. y Van Harmelen F. (2004). OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004. Disponible en fecha enero 2007 en: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3

McGuinness, D.L. (2002), Ontologies come of age, in Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential , eds. D. Fensel, J. Hendler, H. Lieberman, & W. Wahlster, MIT Press, Boston, MA, pp. 1–18.

Minsky M. (1975), A framwork for representing knowledge. The Psycology of Computer Vision, pp 211-217

Newell A., Simon F. (1973) Production System of Control Structure, Academic Press.

Quilluan M. R. (1967), RDS concept: A theory and simulation of some basic semantic capabilities. Behavioral Science, 12:410-430.

Ramos E. y Nuñez H. (2007), ONTOLOGÍAS: componentes, metodologías, lenguajes, herramientas y aplicaciones, Lecturas en Ciencias de la Computación, Universidad Central de Venezuela, ISSN 1316-6239.

Russell S. y Norvig P.,(2004), Inteligencia Artificial: un enfoque moderno, Pretice Hall, México, Segunda Edición

Schreiber, G., Wielinga, B., de Hoog, R., Akkermans, H., y Van de Velde, W. (1994), CommonKADS: A comprehensive methodology for KBS development, IEEE Expert , Vol. 9, no. 6, pp. 28–37.

Sirin E., Parsia B., Cuenca Grau B., Kalyanpur A., Katz Y. (2007), Pellet: A practical OWL-DL reasoner, Journals of Web Semantic, pp 51-53.

Tsarkov D. Y Horrocks I. (2006), FaCT++ Description Logic Reasoner: System Description, School of Computer Science The University of Manchesterr, UK, disponible en:http://www.comlab.ox.ac.uk/people/ian.horrocks/Publications/download/2006/TsHo06a.pdf

Uschold M & Gruninger M. (1996). Ontologies: Principles, Methods and Applications. AIAI- TR-191. Knowledge Engineering Review. Vol 11 Number 2. Disponible en fecha febrero de 2007 en http://citeseer.ist.psu.edu/uschold96ontologie.html