UNIVERSITAT JAUME I
Departament de Llenguatges i Sistemes Informàtics

Scalable methods to analyze Semantic Web data

Ph. D. dissertation
Victoria Nebot Romero

Supervisor
Dr. Rafael Berlanga Llavori

Castellón, December, 2012

To my family.
To Rafa.

Abstract

Semantic Web data is currently being heavily used as a data representation format in scientific communities, social networks, business companies, news portals and other domains. The irruption and availability of Semantic Web data is demanding new methods and tools to efficiently analyze such data and take advantage of the underlying semantics. Although there exist some applications that make use of Semantic Web data, advanced analytical tools are still lacking, preventing the user from exploiting the attached semantics.

The main objective of this dissertation is to provide a formal framework that enables the multidimensional analysis of Semantic Web data in an scalable and efficient manner. The success of multidimensional analysis techniques applied to large volumes of structured data in the context of business intelligence, especially for data warehousing and OLAP applications, has prompted us to investigate the application of such techniques to Semantic Web data, whose nature is semi-structured and contain implicit knowledge.

Multidimensionality is based on the fact/dimension dichotomy. Data are modeled in terms of facts, which are the analytical metrics, and dimensions, which are the different analysis perspectives, and are usually hierarchically organized. We believe that the construction of a multidimensional view of Semantic Web data driven by the semantics encoded in the data themselves and the user's requirements, empowers and enriches the analysis process in a unique manner, as it brings about new analytical capabilities not possible before. Aggregations and display operations typical of multidimensional analysis tools, such as changing the granularity level of the displayed data, or adding a new analysis perspective to the data, will be performed based on the semantic relations encoded in the data. This is possible thanks to the mapping of the data to a conceptual multidimensional space.

We base our research on the hypothesis that Semantic Web data is an emerging knowledge resource worth exploiting, and that the knowledge encoded in Semantic Web data can be leveraged to perform an efficient, scalable and full-fledged multidimensional analysis. Scalability is achieved by two means. On one hand, we provide

an ontology indexing model over ontologies that allows to manage implicit knowledge in a compact format. Therefore, operations that require reasoning can be efficiently solved using the indexes. On the other hand, we have developed several scalable modularization techniques that build upon the previous indexes and allow to extract and work only with the ontological subsets of interest.

The previous indexing and modularization methods assist in the analytical tasks by enabling the extraction of multidimensional data from semantic sources based on the user query. That is, the methods are used to make the extraction of facts and dimensions from Semantic Web data efficient and scalable. However, identifying facts, dimensions, measures and well-shaped dimension hierarchies from the graph structure that underlies SW data is a big challenge due to the mismatch between the graph model that underlies Semantic Web data and the multidimensional model. Therefore, the notions of fact and dimension are revisited in the Semantic Web context and both facts and dimensions are defined from a logical viewpoint.

Facts are formally defined as multidimensional points quantified by measure values, all of which are logically reachable from the subject of analysis defined by the user. To this end, we use the notion of aggregation path, which is less restrictive than the traditional multidimensional constraints usually imposed between facts and dimensions, and define different interesting subgroups for analysis. We also detect the summarizability of the extracted facts and produce correctly aggregated results.

Dimensions are defined as direct acyclic graphs composed by nodes that are semantically related and adhere to the conceptual specification of the user. That is, nodes are sub-concepts of the dimension type and the edges are subsumption relations. Two alternative methods allow to re-shape the extracted dimension hierarchies to favor aggregation while preserving the semantics of the dimension values as much as possible.

The flexibility introduced in the discovery of facts and dimensions allows us to analyze data that have complex relations, which escape to the traditional multidimensional model and cannot be otherwise analyzed. Moreover, instead of building a huge, one time data warehouse, our method for fact and dimension extraction is based on several indexes and precomputed data, which allows us to efficiently materialize only the facts and dimensions required by an analytical query. This provides up-to-date, dynamic and customized results to the user.

The experimental evaluation performed on each of the components of the framework demonstrates that the proposed analysis frame-

work scales to large ontological resources. Likewise, the developed use cases show the potential of the multidimensional analysis of Semantic Web data.

Resumen

Introducción

La Web se ha convertido en una de las mayores fuentes de conocimiento, tanto a nivel científico y empresarial, como a nivel cotidiano. Inicialmente, la Web se planteó como un recurso de consulta de información con contenidos más o menos estáticos, el cual se componía de una serie de documentos enlazados, típicamente ricos en texto. La primera gran evolución de la Web desembocó en la Web 2.0, cuyo objetivo es crear una Web más dinámica que facilite a los usuarios interactuar y compartir información. A través de las tecnologías Web 2.0 se permite a los usuarios crear y gestionar contenidos en la Web. Parte del éxito de la Web 2.0 se debe a la utilización de lenguajes semi-estructurados como XML para el intercambio de información, los cuales permiten dar cierta estructura al contenido. Sin embargo, se ha demostrado que las tecnologas XML son insuficientes para gestionar de forma eficiente la avalancha de información publicada en la Web, puesto que XML es capaz de estructurar sintácticamente un documento pero no su contenido. Ello ha propiciado la evolución de la Web hacia una Web más inteligente, la Web Semántica, o Web 3.0. El objetivo que persigue la Web Semántica es describir los contenidos y la información presente en la Web utilizando formalismos lógicos para que ésta pueda ser "entendida" y procesada de forma automática por las máquinas. Aunque este objetivo es muy ambicioso, ya se dispone de tecnologías adecuadas para la descripción semántica de los contenidos, tales como RDF y OWL. Además, existen iniciativas como Linked Open Data (LOD) que promueven la publicación y enlace de datos en formato semántico en la Web. Tal ha sido el éxito de este tipo de iniciativas que distintas organizaciones de ámbitos muy diversos han decidido publicar sus datos en la Web en formato semántico. Entre las organizaciones que se han adherido a esta iniciativa podemos mencionar: distintos gobiernos como el de Reino Unido o los Estados Unidos, los cuales han publicado gran cantidad de datos acerca de temas tan diversos e interesantes como los niveles de ozono de las distintas ciudades o la situación financiera de las empresas; medios de comunicación como la BBC, que pone a

disposición de los usuarios datos sobre sus programas de televisión y radio enlazados con otros recursos de conocimiento; varias comunidades científicas del ámbito de la biomedicina como Bio2RDF, DailyMed, Diseasome o DrungBank entre muchas otras, han publicado gran cantidad de datos sobre el genoma humano, medicamentos junto con su composición química y su uso, enfermedades y sus relaciones genéticas, etc. El dominio biomédico es especialmente complejo, y es por ello que la conceptualización de estos datos puede acarrear avances importantes.

La existencia de este tipo de datos anotados semánticamente abre nuevas áreas de investigación para el desarrollo de aplicaciones que permitan explotar el conocimiento implícito de los datos. Sin embargo, la explotación de datos anotados semánticamente es bastante compleja debido a su estructura subyacente en forma de grafo y a la semántica implícita, la cual requiere de técnicas de razonamiento que suelen ser costosas computacionalmente.

Objetivos

En esta tesis se proponen una serie de métodos para al análisis de datos anotados semánticamente de forma escalable y eficiente. El éxito que han tenido las técnicas de analisis multidimensional aplicadas a grandes cantidades de datos estructurados, mayormente las técnicas de almacenes de datos y OLAP, nos ha hecho plantearnos la aplicación de dichas técnicas sobre datos anotados semánticamente. De esta forma, en un hospital en donde la información de los pacientes esté anotada semánticamente y enlazada con datos biomédicos, el médico podrá realizar un análisis multidimensional a nivel conceptual para poder analizar el impacto de administrar ciertos fármacos a pacientes con un determinado tipo de enfermedad. Es decir, el médico podrá escoger sus variables de estudio o dimensiones, tales como el tipo de enfermedad, el sexo del paciente, o el tipo de medicamento administrado, y estudiar el impacto que tienen en distintos indicadores o medidas de análisis registradas en los pacientes, tales como los valores de los análisis clínicos o los niveles de recuperación de la enfermedad.

Gracias a las anotaciones semánticas, los datos de los pacientes se enriquecen con información del dominio y se trasladan a un espacio conceptual multidimensional, en donde es posible realizar un análisis guiado por la semántica de los datos. De este modo, el médico puede hacer resúmenes de los datos a distintos niveles de granularidad, por ejemplo, puede visualizar los niveles de colesterol de los pacientes que tienen enfermedades cardiovasculares y, a continuación, aumentar el nivel de detalle para visualizar solo los niveles de los pacientes que tienen una angina de pecho. Este tipo de análisis es posible gracias a las relaciones semánticas existentes entre los tipos de enfermedades, que indican que una angina de pecho es un tipo de enfermedad cardiovascular. Este tipo de conocimiento es el que se encuentra implícito en las ontologías de dominio y es necesario explotar. Sin embargo, la estructura en forma de grafo que subyace

a los datos anotados semánticamente junto con la información semántica que llevan asociada requiere de nuevas técnicas de procesamiento que permitan acceder a los datos objeto de análisis de forma eficiente y transformarlos en una estructura multidimensional preservando la semántica de los mismos.

Por tanto, el objetivo de esta tesis es proporcionar un marco formal para el análisis multidimensional de datos anotados semánticamente de forma eficiente y escalable.

Metodología

Para poder llevar a cabo un análisis multidimensional como el especificado anteriormente, en esta tesis se investigan distintos métodos de manipulación de datos anotados semánticamente. En particular estos métodos se orientan a la gestión eficiente de los datos, pues es bien sabido, que los datos en formato lógico requieren de técnicas de razonamiento computacionalmente costosas. La tesis propone la aplicación de un sistema de indexación basado en intervalos sobre los axiomas inferidos de las ontologías que permita responder de forma eficiente a consultas sobre relaciones de ascendencia/descendencia entre conceptos. Asimismo, se han desarrollado varias técnicas de modularización que permiten aislar las partes de las ontologías que son de interés para el análisis, siempre preservando la estructura y la lógica subyacente. Estos métodos de manipulación de datos anotados semánticamente son utilizados para hacer la extracción de hechos y dimensiones escalable. Los hechos se definen de manera formal como puntos multidimensionales cuantificados por medidas, los cuales son lógicamente alcanzables desde el sujeto de análisis. Las dimensiones se definen como grafos formados por conceptos semánticamente relacionados y con una estructura que favorece la agregación de los datos. Una vez se han obtenido los hechos y las dimensiones de análisis, la generación de cubos puede ser llevada a cabo por herramientas OLAP convencionales.

Aportaciones

La tesis realiza una revisión de la evolución de las tecnologías de análisis multidimensional, desde el análisis de datos estáticos y estructurados en tablas relacionales hasta la apertura a datos semi-estructurados en formato XML. Asimismo, se analizan las aproximaciones que utilizan tecnologías de la Web Semántica para mejorar los procesos de análisis. Sin embargo, no se han encontrado trabajos hasta la fecha cuyo objetivo sea realizar un análisis multidimensional sobre datos anotados semánticamente abordando el problema en toda su complejidad. Las aproximaciones de análisis existentes se orientan a datos que, o bien ya poseen una estructura multidimensional porque han sido directamente derivados de bases de datos existentes, o su estructura es muy sencilla.

La aportación principal de la tesis es la definición de un marco formal para el análisis multidimensional de datos anotados semánticamente de forma eficiente y escalable. Los principales componentes de este marco son: un módulo de indexación de ontologías, un módulo de extracción de fragmentos de ontologías y el módulo de análisis.

El módulo de indexación de ontologías está basado en un sistema de indexación que, aplicado sobre los axiomas de subsunción inferidos de las ontologías, permite tener acceso a las relaciones de *ascendencia/descendencia* entre conceptos de una forma rápida y compacta. Además, estos índices nos permiten realizar consultas sencillas propias de la Lógica de Descripciones sobre los índices de forma rápida. Asimismo, las instancias son indexadas con los índices anteriores para poder realizar consultas conjuntivas de forma eficiente.

El módulo de extracción de fragmentos está compuesto por una serie de técnicas de modularización que permiten seleccionar solo la parte de conocimiento que resulta de interés para el análisis. Estas técnicas se basan en el sistema de indexación previo para construir los módulos y están orientadas a la extracción de módulos de forma eficiente que preservan tanto la estructura como ciertas propiedades lógicas. La tesis realiza una revisión previa del estado del arte en modularización y las nuevas técnicas desarrolladas surgen como una alternativa que combina tanto aspectos lógicos como estructurales en los módulos resultantes. Las técnicas de extracción de módulos desarrolladas pueden utilizarse tanto dentro del marco de análisis definido en esta tesis, como de manera autónoma.

El módulo de análisis es el que se encarga de extraer hechos y dimensiones de análisis de acuerdo a su definición lógica y con la ayuda de las técnicas de indexación y modularización anteriores. El análisis multidimensional se basa en la dicotomía hecho/dimensión. La información se modela en términos de hechos, que son eventos que contienen medidas de análisis interesantes (por ejemplo, visita de un paciente), y las dimensiones, que son las distintas perspectivas de análisis de los hechos (por ejemplo, la enfermedad del paciente). Los elementos de las dimensiones se suelen organizar jerárquicamente en niveles, de forma que los análisis pueden realizarse a distintos niveles de granularidad. Típicamente, los almacenes de datos ya estructuran los datos de forma multidimensional y las herramientas de análisis OLAP proporcionan una serie de operaciones que permiten la construcción de cubos a través de la selección de dimensiones y medidas, y la manipulación de los mismos. Sin embargo, esta tesis aborda el problema de la extracción de datos en forma multidimensional a partir de datos anotados semánticamente. En particular, nuestro estudio se centra en la definición y extracción de hechos que sean válidos a nivel lógico y en la construcción de dimensiones a partir de las ontologías de la forma más automática posible. Para facilitar el dinamismo, los hechos y dimensiones se extraen a partir de los requerimentos del usuario, que se expresan en forma de consulta a nivel conceptual. Esta extracción ad hoc de los hechos y dimensiones contrasta con el análisis tradicional, en donde los hechos y dimensiones

los define a priori el ingeniero del almacén de datos y los análisis de los usuarios se restringen a dichos hechos y dimensiones. El método diseñado para extraer hechos se basa en la alcanzabilidad a nivel lógico de las dimensiones y medidas definidas por el usuario. Es decir, un hecho está compuesto por un punto multidimensional (conjunto de valores de dimensión) y sus respectivas medidas, los cuales son lógicamente alcanzables desde el sujeto de análisis. Para ello, se utiliza la noción de *camino de agregación*, que es mucho menos restrictiva que las dependencias funcionales tradicionalmente requeridas entre hechos y dimensiones, y se distinguen distintos tipos de caminos de agregación interesantes para el análisis. Por otra parte, se han diseñado dos métodos para la extracción de jerarquías de dimensión que se basan en las relaciones semánticas codificadas en las ontologías y permiten la extracción de jerarquías con una estructura lo más adecuada posible para el análisis multidimensional, siempre preservando la semántica de los elementos que la componen.

La flexibilidad que se proporciona en la identificación de hechos y dimensiones permite realizar un análisis multidimensional sobre datos que son complejos por naturaleza y cuyas relaciones escapan al modelo multidimensional de análisis tradicional, el cual está basado en la sumarizabilidad de los datos. En nuestro caso, aunque los hechos extraídos no sean sumarizables, somos capaces de proporcionar una respuesta correcta a la consulta del usuario. Para asegurar la escalabilidad y la eficiencia del análisis multidimensional se hace uso extensivo de los índices y las técnicas de modularización sobre los datos anotados semánticamente.

Conclusiones y trabajo futuro

El marco formal de análisis de datos anotados semánticamente propuesto en esta tesis constituye una potente herramienta de análisis susceptible de ser utilizada en diversos dominios en los que, hasta la fecha, no existen aplicaciones que sean capaces de gestionar de forma eficiente datos anotados semánticamente y proporcionar herramientas de análisis sofisticadas similares a las utilizadas en la inteligencia de negocio. El caso de uso desarrollado en la tesis muestra el potencial del análisis sobre este nuevo tipo de datos y los diversos experimentos validan la vialilidad de la propuesta, y demuestran la eficiencia de los métodos desarrollados.

No obstante, los métodos desarrollados poseen ciertas limitaciones que es conveniente estudiar y que pueden dar lugar a investigaciones futuras. Actualmente, el sistema de indexación diseñado no soporta de manera eficiente las actualizaciones. Cuando se añade una nueva entidad o axioma a una ontología, ésta se vuelve a indexar. Por otra parte, las técnicas de modularización desarrolladas están orientadas básicamente a la extracción de relaciones de subsunción entre conceptos, quedando las propiedades relegadas a un segundo plano. Esta decisión se tomó por cuestiones de aplicabilidad de los fragmen-

tos, ya que, aunque existen otras técnicas de modularización en donde las propiedades son consideradas ciudadanos de primer orden, los fragmentos que se extraen son mucho más grandes, lo cual implica un sobrecoste de procesamiento a la vez que disminuye su reusabilidad.

La extracción de hechos se basa en la alcanzabilidad de las dimensiones y medidas desde el sujeto de análisis. Esta alcanzabilidad se define de manera formal utilizando la noción de caminos de agregación. Aunque la tesis clasifica los caminos de agregación en varios tipos para limitar el espacio de búsqueda, sería conveniente estudiar la posibilidad definir nuevos tipos de caminos de agregación que se ajusten a los requerimientos del usuario. Con respecto a los dos algoritmos desarrollados para la extracción de jerarquías de dimensión, queda pendiente la estratificación de las jerarquías en niveles de forma automática.

Por último, aunque durante la tesis se sugiere que los métodos desarrollados podrían implementarse sobre un marco de procesamiento paralelo (por ejemplo, MapReduce), sería interesante realizar un estudio más a fondo sobre su viabilidad, ya que estas técnicas están siendo aplicadas recientemente a problemas de escalabilidad tipo Web.

La tesis demuestra la hipótesis de que las anotaciones semánticas de los datos proporcionan una información valiosa que es posible explotar de manera escalable y eficiente para realizar tareas de análisis multidimensional aprovechando la semántica de los datos. Este primer resultado nos alenta a ampliar las perspectivas de análisis a otras técnicas de soporte a la toma de decisiones, como es la minería de datos, debido a las semejanzas que hemos observado entre el proceso de extracción de hechos y el proceso de obtención de transacciones de datos anotados semánticamente.

Otra linea de investigación futura que ha surgido del desarrollo de esta tesis consiste en la utilización de la anotación semántica para la extracción de conocimiento que se encuentra implícito en texto. De esta forma colaboramos en la creación de datos anotados semánticamente, los cuales serán susceptibles de ser analizados mediante métodos como el que desarrolla esta tesis.

Palabras clave: Web Semántica, ontologías, lógica de descripciones, OWL, modularización de ontologas, análisis multidimensional, escalabilidad.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Rafael Berlanga. I specially thank him for admitting me in his research group (TKBG) back in 2006, where my research career started. Since then, he has been extremely patient with me, supportive, and more importantly, he has made immense scientific contributions in most of the work I have done throughout the thesis, which would not have been at all possible without his excellent and friendly guidance. Likewise, I really appreciate that he always had time for me.

I would also like to thank my colleagues in the TKBG group for their support. These years of hard work have been much nicer thanks to the motivating and inspiring working environment. It has been a pleasure to work with all of you.

Thanks to Prof. Gerhard Weikum for the opportunity to work in his research group at MPI on an interesting project which, in a mutual benefit, allowed me to combine the research work for the PhD with other exciting IE research lines. I also want to thank the people in the Databases and Information Systems group there, who made me feel like at home.

I thank both my thesis committee members and reviewers: Roxana Dánger, Torben Bach Pedersen and Juan Carlos Trujillo. It is a pleasure for me that they accepted to be part of this.

This thesis is also dedicated to my friends, with whom I have shared really good moments and have always been there for listening.

And last but not least I would like to thank my family, which had an important role, supporting and encouraging me. I would like to acknowledge specially my parents and my partner Rafa for being there, this work would have not be completed without their support and patience.

This work was mainly supported by the FPU grant of the Ministerio de Economa y Competitividad (AP2007-02569). Partial funding was received from the agreement of the research group TKBG with Maat G Know Knowledge S.L. through the project "Health-e-Child" (IST 2004-027749) and from research projects from the

Ministerio de Economa y Competitividad (TIN2008-01825/TIN, TIN2011-24147).

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Research context

The World Wide Web (WWW) appeared as the result of the need to integrate many disparate information systems. The former idea was to provide an abstract space to exchange information among different systems. Its rapid evolution and success since 1989, when it was first conceived, has drastically changed the availability of electronically accessible information. The Web can be considered the last most revolutionary invention in the human communication domain. The reasons for its rapid success and world-wide acceptance lay in its simplicity but powerful way of representing networked information. At the same time, this simplicity and lack of constraints on the Web has led to a situation that requires new solutions.

The ever growing amount of data that is being placed on the Web has made it increasingly difficult to find, access, present and analyze the information required by the users. This is because content is primarily presented in a human-readable form. The enormous proportions that the Web has acquired conform an immense source of knowledge worth exploiting. However, more elaborated mechanisms should be layered on top of the Web in order to efficiently exploit it and extract its full potential.

Already in 2001, Tim Berners-Lee, Director of the WWW Consortium, referred to the future of the WWW as the *Semantic Web* (SW) - an extended web of machine-readable information and automated services that extends far beyond current capabilities [19]. The explicit representation of the semantics underlying data, programs, pages, and other web resources, will enable a knowledge-based web that provides a qualitatively new level of service. Automated services will improve in their capacity to assist users in achieving their goals by "understanding" more of the content on the web and thus providing more accurate filtering, categorization and searching of information sources. This process will ultimately lead to an extremely knowledgeable system that

features various specialized reasoning services. These services will support us in nearly all aspects of our daily life - making access to information as pervasive and necessary as access to electricity is today.

Although it has taken almost a decade for the SW to really take off, we are at a point where SW technologies are mature enough to be deployed in real applications such as web portals, information retrieval, information integration and business intelligence (BI) among others. Ontologies are the backbone technology of the SW. They formalize the knowledge of an application domain by first defining the relevant concepts of the domain (i.e., the terminological knowledge or TBox), and then using these concepts to specify properties of objects occurring in the domain (i.e., the assertional knowledge or ABox). All this is accomplished by using formal knowledge representation languages. The concept of ontology is not new, as the first ontologies were developed in Artificial Intelligence (AI) during the nineties to facilitate knowledge sharing and reuse. However, it is during the last few years that ontologies have found interesting application scenarios in the context of the SW.

Many efforts have been devoted to the construction of terminological ontologies, which conceptualize the general knowledge of specific domains. One especially successful field where large ontologies have been developed is that of Biomedicine. However, the formalization of the concepts of a domain (i.e., the terminological knowledge) is not enough for the realization of the SW. We also need tools to populate the SW with assertional data, that is, to provide semantic annotations of content using the terminological ontologies. These annotations make data meaning explicit by situating it in a conceptual framework. Throughout this dissertation we refer to both the terminological and assertional data with the term *SW data*.

SW technologies enable to attach semantics to resources, ranging from very simple to very complex annotations depending on the expressivity required. They provide standard formats for knowledge representation (e.g., RDF[1], RDFS[2] and OWL[3]) and automatic reasoning. Currently, the de facto metadata representation language on the SW is RDF. Moreover, community efforts such as Linked Open Data[4] are promoting the publication and linkage of data on the Web using RDF. As a result, hundreds of millions of documents embedding RDF metadata in different formats (e.g., RDF/XML, RDFa, Microformats, N-Triples and Turtle) are being continuously published and the tendency is to keep growing. The variety of institutions that have shown interest in publishing RDF content on the Web range from public institutions such as governments to e-commerce industries and research communities. Figure 1.1 shows the linked data cloud in 2010, where a relevant part is data from the Life Sciences domain.

[1]RDF: `http://www.w3.org/TR/rdf-concepts/`
[2]RDFS: `http://www.w3.org/TR/rdf-schema/`
[3]OWL: `http://www.w3.org/TR/owl-features/`
[4]http://linkeddata.org/

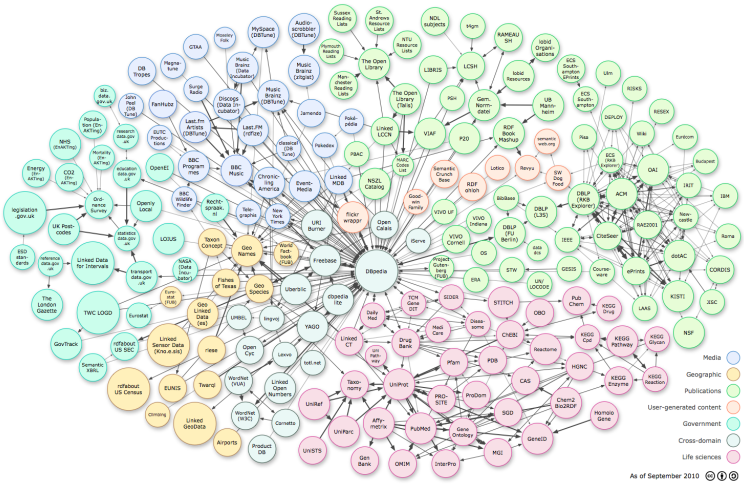

Figure 1.1: Linked Data Cloud in 2010.

1.2 Motivation and Objectives

SW data is already out there. The next natural step is to develop methods and tools to efficiently analyze and take profit from the implicit knowledge encoded in SW data.

Although there exist tools that ease the use of SW data such as ontology editors (e.g., Protégé[5], NeOn Toolkit[6]), reasoning systems (e.g., FaCT++[7], Pellet[8], Hermit[9]) and storage and query systems (e.g., Sesame[10], BigOWLIM[11], Jena[12]), most of these tools treat ontologies as monolithic entities and provide little support for managing and accessing ontologies in a modular and efficient manner. Moreover, these tools are lacking advanced analytical capabilities further than pattern-based querying.

The main objective of this dissertation is to provide a formal framework that enables the analysis of SW data in an scalable and efficient manner. The success of multidimensional (MD) analysis techniques applied to large volumes of structured data in the context of BI, especially for data warehousing (DW) and OLAP applications, has prompted us to investigate the application of such

[5]Protégé: `http://protege.stanford.edu/`
[6]NeOn toolkit: `http://www.neon-toolkit.org`
[7]FaCT++ reasoner: `http://owl.man.ac.uk/factplusplus/`
[8]Pellet reasoner: `http://clarkparsia.com/pellet/`
[9]HermiT reasoner: `http://hermit-reasoner.com/`
[10]Sesame: `http://www.openrdf.org/`
[11]BigOWLIM: `http://www.ontotext.com/owlim`
[12]Apache Jena: `http://jena.apache.org/`

techniques to SW data, whose nature is semi-structured and contain implicit knowledge. Multidimensionality is based on the fact/dimension dichotomy. Data are modeled in terms of facts, which are analytical metrics, and dimensions, which are the different analysis perspectives, and are usually hierarchically organized. We believe that the construction of an MD view of SW data driven by the semantics encoded in the data themselves empowers and enriches the analysis process in a unique manner, as it brings about new analytical capabilities not possible before. Aggregations and display operations typical of MD analysis tools, such as changing the granularity level of the displayed data, or adding a new analysis perspective to the data, will be performed based on the semantic relations encoded in the ontologies. This is possible thanks to the mapping of the data to a conceptual MD space.

As the purpose of MD analysis is to give an accurate, intelligent snapshot of the data, we require a minimum quality in the data to ensure that the results are precise. Adding semantics to the data helps in this matter, as the more semantics you add, the more unambiguous data become and reasoning techniques can be executed to draw inferences and check the consistency of the data. Therefore, we mainly aim at analyzing SW data that is expressed in OWL. However, the benefits of adding semantics to the data comes at the expense of complexity issues regarding aspects such as usability, scalability, efficient reasoning, analysis and so on. In this thesis we develop methods that aid the MD analysis of expressive SW data at large scale.

1.3 Hypothesis

This thesis addresses the problem of MD analysis over SW data. Hence, the overall goal is to provide a formal framework that enables such MD analysis from a logic-based viewpoint in an scalable and efficient manner. Towards this end, we have analyzed why this involves a big research challenge and how to solve it. This analysis has led to the main hypothesis, which captures the main research question of this thesis. The hypothesis is used to derive concrete methods that enable the problem of scalable MD analysis over SW data.

Hypothesis: *The knowledge encoded in SW data can be leveraged to perform a full-fledged MD analysis of such data in a scalable an efficient manner.*

As MD analysis is based on the arrangement of data into facts and dimensions, we need to investigate methods that are able to access and extract specific subsets of SW data, always preserving the underlying semantics. However, the full exploitation of SW data using scalable methods is far from trivial due to their special features and the requirements on the data imposed by the developed applications. The biggest challenge is to deal with the implicit semantics in a scalable manner. SW data are generally based on formal descrip-

tions, which enable to derive new logical consequences through the process of reasoning. However, reasoning techniques over large datasets are computationally expensive. Therefore, new indexing structures are needed to allow efficient querying and management of the data, thus minimizing the use of the reasoner. Towards that end, we have developed an ontology indexing model that is applied to the inferred ontologies and allows fast responses about relationships between concepts, as well as answers to a restricted subset of description logic (DL) queries. Instance data is also indexed to efficiently perform conjunctive query answering.

Another challenge specific of SW data is the extraction of only the required subsets of the data in a scalable manner. For MD analysis purposes, the ideal modules should be extracted from the user's requirements and should show a good compromise between not only the preservation of semantics, but also the preservation of the structure and the size of the resulting module. Towards that end, we have developed several ontology modularization techniques that build on top of the previous ontology indexes. These modularization techniques allow the user to efficiently extract and work with ontology subsets, thus favoring reuse and scalability. The ontology modularization techniques show a good trade-off between logical and structural properties, thus they are a good alternative to the current modularization approaches.

The previous ontology manipulation methods assist in the extraction of facts and dimensions from SW data, based on the MD conceptual query of the user. The main challenge specific to the MD analysis of SW data is their semi-structured nature. SW data in their simplest form are composed by triple statements of the form subject, predicate, object, (s, p, o), where the predicate p expresses a relation between the resource s and the object o, which can be another resource or a literal. A collection of interconnected triples constitutes a labeled directed graph, where nodes are the subjects and objects of assertions, and the edges are properties. This graph topology clearly contrasts with the MD model on which traditional analysis tools are based. The MD model views the data in terms of facts and dimensions. Facts are the metrics that business users use for making business decisions. Dimensions are the attributes that qualify facts. They are the different analysis perspectives and give structure to the facts by arranging themselves into hierarchies so that the user can navigate the facts at different granularities. Identifying facts, dimensions, measures and well-shaped dimension hierarchies from the graph structure that underlies SW data is a big challenge. Also, as the web is continuously changing and growing, the developed methods should take into account the user requirements while being as automatic as possible so that the results can be reproduced to reflect changes in the data. To meet the previous challenges, the notions of fact and dimension are revisited in the SW context and both facts and dimensions are defined from a logical viewpoint. Facts are formally defined as analytical metrics supported by a set of dimensions that are all logically reachable from the subject of analysis defined by the user by means of aggregation paths. The

notion of aggregation path is less restrictive than the functional dependency between facts and dimensions usually required by traditional MD analysis. This flexibility allows us to capture data that have complex relations and cannot be otherwise analyzed. However, the extracted facts can contain duplicate information, which is an issue that we need to deal with. Dimensions are defined as directed acyclic graphs where nodes are sub-concepts of the dimension type defined by the user and edges are the semantic relations between the concepts. The concepts (i.e., dimension values) are arranged in a hierarchical structure that favors aggregation.

The developed methods to achieve scalable MD analysis of SW data assume the existence of a reasoner that is able to compute inferences in a reasonable amount of time. These inferences are then indexed by our methods so that the analysis becomes efficient and scalable. However, we are aware that reasoning is hard and it depends on the language expressivity. To cope with this scalability issue, several approaches have emerged, which aim at performing scalable reasoning either by approximation [51], by reducing non-determinism [87], or by reducing the expressivity [11]. Therefore, the availability of a scalable reasoner is a fair assumption.

As we are interested in performing MD analysis over SW data which is spread among decentralized information sources, we assume that the sources provide way to identify and access the data through standard data exchange protocols. Moreover, although data are described using the same data model (RDF), each data source might provide its own schema (conceptualization), ranging from loosely to strictly defined. The problem of finding mappings and alignments between different terminological resources has been extensively studied in the literature. Thus, we assume that these data sources are interlinked and the relationships (e.g., mappings or alignments) are explicitly defined by the data sources themselves.

1.4 Contributions

This thesis includes a review on the evolution of MD analysis techniques over different types of data, from the analysis of static, structured data residing in relational tables, to the analysis of external and semi-structured data, mainly coming from the Web. For completeness, we also review the main approaches that make use of SW technologies to enhance the traditional analytical processes. We have not found so far any approaches that tackle the problem of MD analysis of SW data in all its complexity. The few approaches that tackle MD analysis of SW data deal with data that either already possess an MD structure because it has been directly derived from data bases, or their structure is very simple, almost flat.

The main contribution of this work is a formal framework that enables MD analysis of SW data in a scalable and efficient manner. Both scalability

and efficiency are achieved by tailored ontology indexing and modularization methods that support the analysis of SW data. These methods allow to efficiently treat ontological data and minimize the use of a reasoner, which has usually high computational cost. The experiments demonstrate that the proposed framework scales to large ontological resources. The main components of the framework are: the Ontology Indexing Model (OIM), the Ontology Module Extraction Techniques (OMETs) and the MD Analysis Component (MDAC).

The OIM has been devised to allow efficient querying and management of SW data, thus minimizing expensive calls to a reasoner. The indexing schema proposed is applied to the inferred ontologies and is based on intervals that compactly encode hierarchical relationships among concepts. We also provide an intervals' algebra to operate with the ontology indexes. Thanks to this algebra, basic operations involving ancestor/descendant relationships, as well as a restricted subset of DL queries and conjunctive queries can be efficiently performed using the index.

The OMETs is composed by several modularization techniques that ease the use and reusability of ontologies by allowing to extract only specific fragments. We provide a review of the literature on the main existing modularization approaches emphasizing the different criteria that they focus on (i.e., logical vs. structural properties). The developed techniques are based on the previous OIM and are developed to cover the need for modules that enable the MD analysis. Therefore, these modules are extracted from the user's requirements and show a good trade-off between size, scalability, preservation of structure and preservation of original semantics. The OMETs can be use both inside the analysis framework defined in this thesis or as a standalone tool.

The MDAC is in charge of extracting facts and dimensions according to their logical definition with the aid of the previous indexing and modularization methods. Typically, DWs subject to analysis have already an MD structure, and the analytical OLAP tools provide a series of operations that allow the construction of cubes through the selection and manipulation of the facts and dimensions. However, this thesis tackles the problem of extracting and shaping SW data into a suitable MD structure. More precisely, our study is focused on the challenges in defining and extracting logically valid facts and dimensions directly from SW data as automatically as possible, remaining scalable and taking into consideration the user's requirements expressed conceptually. To enable dynamism, facts and dimensions are extracted based on the user requirements, which are expressed as a conceptual query. This ad hoc extraction of facts and dimensions clearly contrasts with the traditional MD analysis, where facts and dimensions are defined a priori by the DW engineer.

The developed method to extract facts is based on the logical reachability of the dimensions and measures defined by the user. That is, a fact is a numeric measure together with its supporting dimensions. All of them have to be reachable from the subject of analysis. To this end, we use the notion of aggregation path, which is less restrictive than the traditional MD constraints

imposed between facts and dimensions, and classify the aggregation paths in different subtypes interesting for analysis. By allowing facts to be composed by dimensions and measures reachable by aggregation paths, we are enabling the MD analysis of data that have complex relations and cannot be analyzed by traditional MD analysis tools. However, this flexibility can lead to facts that are not summarizable, as they contain duplicated information. We identify when the extracted facts are not summarizable and are still able to provide correct results to the user MD query by handling duplicated data.

On the other hand, we model dimensions as directed acyclic graphs, in an attempt to capture as much semantics as possible from the ontologies. Nodes are sub-concepts of the dimension type and edges are subsumption relations between the concepts. We have developed two alternative methods to extract dimension hierarchies based on the semantic relations encoded in the ontologies. These methods allow the extraction of rich dimension hierarchies, which are later re-shaped to provide good aggregation power, at the same time that preserve the semantics of the dimension values.

To ensure scalability and efficiency of the MD analysis of SW data, we make extensive use of the indexing and modularization approaches proposed.

1.5 Outline of the thesis

This thesis has been organized in six chapters (including this one). Chapters two to five contain the contributions of the thesis. Chapter six outlines the conclusion and future research lines. A brief overview of each chapter is shown below.

Second Chapter: Background

This chapter introduces the general concepts and related work investigated in the thesis. The chapter is divided in three parts. The first part introduces the foundations and main aspects related to the SW (e.g., the SW vision, ontologies, ontology languages and description logics). The second part defines the concept of ontology modularization and reviews the main modularization approaches, focusing on ontology module extraction and its two different trends, logical vs. structural approaches. The third part of the chapter introduces basic analysis concepts and walks the reviewer through the evolution of traditional MD analysis to analytics over the SW.

Third Chapter: Framework

This chapter groups the proposed methods for efficiently managing and analyzing SW data into a common framework. The main components of the framework are: the OIM module, the OMETs module and the MDAC. The

chapter also presents an application scenario and use case that will be followed along the rest of the thesis.

Fourth Chapter: Indexing and modularization approaches for ontologies

The first part of this chapter proposes the OIM based on intervals that can be applied to the TBox and ABox of large ontologies. By using this index, we obtain fast response times to queries about relationships between concepts, as well as to conjunctive queries. Moreover, the index is able to answer a restricted subset of DL queries, which are otherwise expensive if we use a reasoner. In the second part of the chapter, we define our notion and analytical requirements for a module and, according to these, present four different modularization approaches. These modularization approaches differ from the existing ones in that they reach a good trade-off between the analytical requirements identified.

Fifth Chapter: Multidimensional analysis of Semantic Web data

This chapter describes the process of identification and extraction of facts and dimensions from SW data from a logical viewpoint. The chapter starts with the definition of an MD query specified by the analyst according to the conceptual descriptions of the sources (i.e., concepts and roles). Then, we introduce the notion of aggregation path and a classification of interesting aggregation paths, as this is the main foundation to extract facts. Facts are extracted by accessing the dimension and measure values reachable from the subject of analysis by means of aggregation paths. The notion of summarizability is also investigated in the resulting facts, as well as how to handle duplicated information. Dimension hierarchies are automatically extracted from the relations of concepts specified in the ontologies and are shaped to meet as much as possible the requirements imposed by MD analysis.

Sixth Chapter: Conclusions

This chapter outlines the main conclusions and identifies some areas of future work based on some open issues of the work presented in this thesis, as well as others that have emerged during the development of the thesis.

Chapter 2

Background

This chapter has been split in three main parts that cover relevant aspects investigated in this dissertation. The first part, Section 2.1, covers the main foundations of the SW and the enabling technologies. The second part of the chapter, Section 2.2, is devoted to the concept of modularization (Section 2.2.2) and the study of the different approaches, namely Ontology Partitioning (Section 2.2.3) and Ontology Module Extraction (Section 2.2.4). The last part of the Chapter, Section 2.3, provides the background on analysis concepts (Section 2.3.1) and discusses the evolution of the main analytical approaches devised for a semi-structured scenario such as the Web (Section 2.3.2), putting special emphasis in those analytical processes where SW technologies are used to enhance the analysis process (Section 2.3.3). Finally, we discuss the methods where the main source of analysis is SW data (Section 2.3.4) and outline new trends.

2.1 Semantic Web Foundations

The SW is grounded on theoretical principles borrowing to several fields of computer science such as programming languages, data bases, structured documentation, logic and artificial intelligence. In information systems, ontologies are conceptual yet computational models of a domain of interest that build on knowledge representation techniques. They play a key role in the SW, where they support the meaningful annotation of web content and resources. This section gives a brief overview of the foundations of the SW. Section 2.1.1 describes the vision of the SW. Section 2.1.2 introduces the concept on ontology and its applications in the SW scenario. In Section 2.1.3 we introduce the main ontology languages for the SW. Finally, Section 2.1.4 is devoted to DL, the main logic-based language in the SW.

2.1.1 The Semantic Web Vision

We live in an information society and access to information lies at the heart of most human activity. However, several factors impose new challenges in the efficient management of information. Both the amount and the complexity of information has increased enormously. The clear example is the Web, which is without question the most popular information service over the Internet. The usual way for users to find the information they are looking for in the Web is by typing a set of keywords in a search engine. Although the popularity of search engines is indisputable, the precision of the returned results has decreased, making access to the relevant information more difficult. The main problem stems from the human-readable orientation of the Web. HTML is mainly focused on presentation and visual aspects and does not provide any semantics to the annotated elements. Therefore, search engines rely mainly on syntactic means for content matching with user queries. Matching is based on direct comparison of query keywords and the words that appear in web documents. Moreover, the ever growing rate of the Web, which has doubled its size only during the last couple of years, complicates matters with the addition of new contents that range from structured to semi-structured and unstructured. Also, the underlying data may be of low quality (e.g., incomplete or inconsistent). Heterogeneous information is being increasingly distributed and it is consumed not only by humans, but also by machines. In this new scenario, the traditional vision of the Web is not sufficient anymore to fulfill today's information management requirements.

An extension of the traditional Web was conceived to provide web resources with knowledge that is machine-processable. Tim Berners-Lee, the inventor of the Web, defines the term Semantic Web as follows [19]:

> The Semantic Web is an extension of the current web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation.

To achieve the vision of the SW, resources are annotated by structured and machine-processable metadata, which are assigned a well-defined meaning and are interpreted by means of ontologies.

2.1.2 Ontologies

Since ancient times, philosophers have been concerned with fundamental questions about the existence and the general categories for all things that exist. *Ontology* is the discipline that has traditionally studied such matters. In AI, ontologies are computational artifacts that encode knowledge about a particular domain in a machine-processable form using knowledge representation techniques. In the SW community the most accepted definition is based on [47].

Definition 2.1. *An ontology is a formal explicit specification of a shared conceptualization of a domain of interest.*

This short definition encapsulates important characteristics of an ontology that deserve an explanation.

- *formal*: refers to the knowledge representation language used to specify the ontology, which should provide formal semantics to ensure machine-readability.
- *explicit*: knowledge should be made explicit so that computers can interpret it.
- *shared*: an ontology is a conceptualization reached by an agreement among a community.
- *conceptualization*: the knowledge in an ontology is specified in terms of symbols that represent concepts and their relations, which correspond to the elements in a human mental model.
- *domain of interest*: the knowledge encoded in an ontology refers to the specification of a particular domain.

Ontologies formalize the knowledge in a domain by means of a set of components: *concepts*, *relations*, *instances* and *axioms*. Concepts represent the general abstractions or categorizations used to describe objects. Relations semantically connect concepts and instances. Instances represent the particular objects of the world. The axioms are a set of statements expressed in terms of the previous elements.

In the following, we show some applications in which ontologies provide an alternative way for efficiently managing information:

- *Information retrieval.* The semantic information contained in both documents and queries can be leveraged by mapping these to ontological concepts and relations, thus increasing the precision of the results.
- *Information integration.* Ontologies can be used to mediate and integrate information sources with different schemas.
- *Content management.* Ontologies can be used as the common domain-specific vocabulary to annotate data, allowing automatic processing and machine readability.
- *Knowledge management.* Ontologies can serve as the conceptual backbone that connects individual knowledge management systems.
- *Question answering and expert systems.* Domain ontologies can be used to formalize expert knowledge about a certain domain so that domain-specific questions can be answered by reasoning over such knowledge.

2.1.3 Ontology Languages for the Semantic Web

Ontologies play a key role in the context of the SW. The idea of the SW is to add a layer of meaning to the Web so that the knowledge becomes machine understandable [19]. This can be achieved by annotating web content with machine-interpretable meta data such that computers are able to process this content on a semantic level. Thus, ontologies provide the domain vocabulary (i.e., concepts, relations and instances) in terms of which semantic annotation is formulated. But having an ontology is not enough. We also need to adopt a standard ontology language that provides both a shared syntax and a shared semantics to interpret this syntax. This section introduces the ontology languages that are used for representing and querying knowledge within the SW. These are RDF (Resource Description Framework), its extension RDFS (Resource Description Framework Schema) and OWL (Web Ontology Language).

2.1.3.1 RDF/RDFS

RDF [67] allows for the description of resources and how they relate to each other. RDF specifies a data model for publishing metadata as well as data on the Web and utilizes XML as serialization syntax for data transmission. It is a model and syntax for annotating web resources designed for the exchange of information over the Web and it is the base layer for building the SW. The underlying structure of RDF is a collection of triples, each consisting of a subject, a predicate and an object, which form an RDF graph. An RDF triple states that there is some relationship, indicated by the predicate, holding between the subject and the object of the triple.

The RDFS [25] defines a simple modeling language on top of RDF. It provides primitives that allow to express set membership of objects in property and class extensions. That is, RDFS uses classes, subsumption relationships on both classes and properties, and global domain and range restrictions for properties as modeling primitives. However, RDFS is too weak to describe resources in sufficient detail and it only serves to create lightweight ontologies.

2.1.3.2 OWL

OWL [114] is the widely accepted ontology language of the SW. Its syntax is compatible with existing web standards such as XML, RDF, and RDFS and its semantics are formally specified to support reasoning. OWL has a richer set of operators than RDFS and richer semantics. Therefore, complex concepts can be built up in definitions out of simpler concepts by using such logical operators. Furthermore, its formal semantics allows the use of a reasoner which can check whether the statements and definitions in the ontology are mutually consistent.

The formal underpinning of OWL is based on formal logic and comes in three flavors, which have a direct correspondence with DL:

- OWL Lite is the smallest subset. Reasoning with OWL Lite is efficient but it has low expressive power. Hence, it is mainly devised to support simple classification hierarchies and constraints.
- OWL DL imposes some limitations on the full use of OWL to allow decidability in reasoning tasks.
- OWL Full has more expressive power than OWL DL and reasoning in such language is undecidable.

OWL 2 [86] is a revision of the former OWL. It introduces some improvements such as new DL constructs, a better specification of the language and more flexibility in the use of annotations. The underlying logics of OWL2 is $\mathcal{SROIQ}$ [52]. OWL 2 also defines three new profiles, which may better meet certain performance requirements or may be easier to implement. In the following, we describe the three profiles of OWL 2 [85]. The choice of which profile to use in practice will depend on the structure of the ontologies and the reasoning tasks[1] at hand.

- OWL 2 EL is primarily designed for classification tasks (subsumption/ instance checking) with large ontologies. Basic reasoning problems can be performed in time that is polynomial with respect to the size of the ontology. Dedicated reasoning algorithms for this profile are available and have been demonstrated to be implementable in a highly scalable way.
- OWL 2 QL is targeted to applications that use very large volumes of instance data, and where query answering is the most important reasoning task. Using a suitable reasoning technique, sound and complete conjunctive query answering can be performed in LogSpace with respect to the size of the data (assertions). As in OWL 2 EL, polynomial time algorithms can be used to implement the ontology consistency and class expression subsumption reasoning problems.
- OWL 2 RL is aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate OWL 2 applications that can trade the full expressivity of the language for efficiency, as well as RDF(S) applications that need some added expressivity. OWL 2 RL reasoning systems can be implemented using rule-based reasoning engines. All reasoning tasks can be solved in time that is polynomial with respect to the size of the ontology.

[1] We refer to the reader to Section 2.1.4.1 for an overview of the main reasoning tasks in DL

2.1.4 Description Logics

The logical foundation of OWL is formed by a subset of first-order logic called Description Logics (DL), which is a family of knowledge representation formalisms. Due to its nature of decidability, DLs have proved useful in a wide range of applications in computer science regarding knowledge representation.

Baader [10] summarizes the main underlying characteristics of DL in the following way:

> Description Logics is the most recent name for a family of knowledge representation formalisms that represent the knowledge of an application domain (the world) by first defining the relevant concepts of the domain (its terminology), and then using these concepts to specify properties of objects and individuals occurring in the domain (the world description). As the name indicates, one of the characteristics of these languages is that, unlike some of their predecessors, they are equipped with a formal, logic-based semantics. Another distinguishing feature is the emphasis on reasoning as a central service: reasoning allows one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base. DLs support inference patterns that occur in many applications of intelligent information processing systems, and which are also used by humans to structure and understand the world: classification of concepts and individuals.

The terminological knowledge in DLs is represented by *concepts*, which are unary predicates such as HUMAN, and *roles*, which are binary predicates such as hasChild. Concepts denote sets of individuals and roles denote binary relations between individuals. Based of atomic concepts (denoted by A) and atomic roles (denoted by R), complex concept descriptions (denoted by C) are built inductively using concept constructors. The first and second columns of Table 2.1 show the name and syntax of the main DL constructors.

Different DLs can be constructed by taking a different subset of constructors. The last column in Table 2.1 shows different DLs. The language $\mathcal{AL}$ was introduced as the minimal language that is of practical interest. For example, the following $\mathcal{AL}$-concept description represents all women that have at least one human child, i.e., who are mothers: WOMAN $\sqcap \exists$ hasChild.HUMAN

The expressive power of a DL depends on the provided constructors from which concepts and relations can be composed, and the kinds of axioms supported. The other languages of this family are extensions of $\mathcal{AL}$. For example $\mathcal{ALC}$ extends $\mathcal{AL}$ by general concept negation ($\neg C$) and full existential quantification ($\exists R.C$), and is thus the most basic DL closed under Boolean operators. The DL $\mathcal{SHIQ}$ is $\mathcal{ALC}$ plus extended cardinality restrictions, and transitive and inverse roles. For historical reasons, the sublanguage of $\mathcal{AL}$ obtained by disallowing atomic negation is called $\mathcal{FL}^-$ and the sublanguage of $\mathcal{FL}^-$ obtained by disallowing limited existential quantification is called $\mathcal{FL}_0$.

Constructor	Syntax	Semantics	Expressivity
Domains			
Universe	$\top$	$\Delta^{\mathcal{I}}$	$\mathcal{FL}^-, \mathcal{AL}, \mathcal{S}$
Empty	$\bot$	$\emptyset$	$\mathcal{FL}^-, \mathcal{AL}, \mathcal{S}$
Concrete Domain	Φ	$\Phi^{\mathcal{D}} \subseteq \Delta^{\mathcal{D}}$	$(\mathcal{D})$
Nominals	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$	$\mathcal{O}$
Roles and Features			
Atomic Role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	$\mathcal{FL}_0, \mathcal{FL}^-, \mathcal{AL}, \mathcal{S}$
Inverse Roles	R^-	$\{(a,b) \mid (b,a) \in R^{\mathcal{I}}\}$	$\mathcal{I}$
Role Composition	$R \circ S$	$\{(a,c) \mid (a,b) \in R^{\mathcal{I}}, (b,c) \in R^{\mathcal{I}}\}$	$\mathcal{R}$
Role Conjunction	$R \sqcap S$	$R^{\mathcal{I}} \cap S^{\mathcal{I}}$	$\mathcal{R}$
Role Disjunction	$R \sqcup S$	$R^{\mathcal{I}} \cup S^{\mathcal{I}}$	$\mathcal{R}$
Role Complement	$\neg R$	$\{(a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \notin R^{\mathcal{I}}\}$	$\mathcal{R}$
Feature	u	$u^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{D}}$	$(\mathcal{D})$
Concept constructors			
Atomic Concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	$\mathcal{FL}_0, \mathcal{FL}^-, \mathcal{AL}, \mathcal{S}$
Negation of atomic concepts	$\neg A$	$\Delta^{\mathcal{I}} - A^{\mathcal{I}}$	$\mathcal{AL}, \mathcal{S}$
Negation of concepts	$\neg C$	$\Delta^{\mathcal{I}} - C^{\mathcal{I}}$	$\mathcal{C}, \mathcal{S}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}, \mathcal{S}$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$\mathcal{FL}_0$, $\mathcal{FL}^-, \mathcal{AL}, \mathcal{S}$
Concepts from roles			
Limited existential quantification	$\exists R$	$\{a \mid \exists R(a,b) \in R^{\mathcal{I}}\}$	$\mathcal{FL}^-, \mathcal{AL}, \mathcal{S}$
Universal quantification	$\forall R.C$	$\{a \mid \forall R(a,b) \in R^{\mathcal{I}}, b \in C_{\mathcal{I}}\}$	$\mathcal{FL}_0, \mathcal{FL}^-, \mathcal{AL}, \mathcal{S}$
Full existential quantification	$\exists R.C$	$\{a \mid \exists R(a,b) \in R^{\mathcal{I}}, b \in C_{\mathcal{I}}\}$	$\mathcal{E}, \mathcal{S}$
Card. Restrictions	$(\leq nR)(\geq nR)$	$\{a \mid \sharp\{b \mid (a,b) \in R^{\mathcal{I}}\} \leq n\}$	$\mathcal{N}$
Q. Number Restrictions	$(\leq nR.C)\ (\geq nR.C)$	$\{a \mid \sharp\{b \mid (a,b) \in R^{\mathcal{I}}, b \in C^{\mathcal{I}}\} \leq n\}$	$\mathcal{Q}$
Existential Restrictions for concrete domains	$\exists u.\Phi$	$\{a \mid \exists (a,b) \in u^{\mathcal{I}}\}$	$(\mathcal{D})$
Universal Restrictions for concrete domains	$(\forall u.\Phi)$	$\{a \mid \forall (a,b) \in u^{\mathcal{I}}\}$	$(\mathcal{D})$
Card. Restrictions for concrete domains	$(\leq n\, u.\Phi)\ (\geq n\, u.\Phi)$	$\{a \mid \sharp\{x \mid (a,x) \in u^{\mathcal{I}}, x \in \Phi^{\mathcal{D}}\} \leq n\}$	$(\mathcal{D})$
Axioms			
Concept Hierarchy	$C \sqsubseteq D$	$\mathcal{I} \models (C^{\mathcal{I}} \subseteq D^{\mathcal{I}})$	$\mathcal{H}$
Concept Equivalence	$C \equiv D$	$\mathcal{I} \models (C^{\mathcal{I}} = D^{\mathcal{I}})$	$\mathcal{AL}, \mathcal{S}$
Disjoint Concepts	$disjoint(C, D)$	$\mathcal{I} \models (C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset)$	$\mathcal{R}$
Role Hierarchy	$R \sqsubseteq S$	$\mathcal{I} \models (R^{\mathcal{I}} \subseteq S^{\mathcal{I}})$	$\mathcal{H}$

Table 2.1: DL constructors semantics

The formal semantics of the syntactic elements of a DL is considered in terms of model-theoretic semantics. Thus, an interpretation $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$, with $\Delta^{\mathcal{I}}$ a non-empty set, called the domain of the interpretation, and $.^{\mathcal{I}}$ a function that assigns to each individual a an object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ and that interprets (possibly) complex concepts and roles as indicated in the third column of Table 2.1.

DL Knowledge Base

A DL knowledge base (KB) consists of two components, the TBox ($\mathcal{T}$), containing intensional knowledge, and the ABox ($\mathcal{A}$), containing extensional knowledge [10]. The TBox defines the terminology (vocabulary) of an application domain. The ABox contains assertions about named individuals in terms of the vocabulary.

The terminological axioms of the TBox can be of two types: *definitions* and *inclusion* axioms. The inclusion axioms ($\sqsubseteq$) state inclusion relations between DL concepts. For example, one can state that a woman is a human being: WOMAN $\sqsubseteq$ HUMAN. Definitions allow to give a name to a concept description. For example, we can define a mother as a woman that has at least one child that is a human being: MOTHER $\equiv$ WOMAN $\sqcap \exists$ hasChild.HUMAN. Here, WOMAN and HUMAN are *primitive* concepts and MOTHER is a *defined* concept. If a TBox contains an axiom of the form $C \sqsubseteq D$ where C is a complex concept, then this axiom is referred to as a *general concept inclusion axiom* (GCI) and the TBox is referred to as a *general TBox*.

The assertional axioms of the ABox can also be of two types: concept assertions and role assertions. By a concept assertion, one states that a certain individual a belongs to a concept C, written $C(a)$. By a role assertion, one states that an individual c is a filler of the role R for an individual b, written $R(b, c)$. For example, we can state that *Mary* is a mother, and *John* is a man who is the son of *Mary*: MOTHER$(Mary)$, MAN$(John)$, hasChild$(Mary, John)$.

2.1.4.1 Reasoning with DL

For DLs various reasoning tasks are usually considered. These tasks allow to draw new conclusions about the knowledge base or check its consistency. In this section standard reasoning problems are introduced. The main reasoning services supported by DLs can be reduced to satisfiability [10]. The first two are related to the TBox and the rest to the ABox.

- *Concept Satisfiability.* A concept is satisfiable if it admits instances. That is, C is satisfiable iff there is some model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$.
- *Concept Subsumption.* A concept C is subsumed by another D, $\mathcal{T} \models C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. Concept equivalence and concept disjointness can be reduced to concept subsumption as follows: $\mathcal{T} \models C \equiv D \longleftrightarrow \mathcal{T} \models C \sqsubseteq D, D \sqsubseteq C$ and $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset \longleftrightarrow \mathcal{T} \models (C \sqcap D) \sqsubseteq \bot$

- *Consistency.* An ABox $\mathcal{A}$ is consistent w.r.t. a TBox $\mathcal{T}$ iff there exists some model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$.
- *Instance checking.* An individual a is an instance of C iff for every model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$, $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- *Realization.* For all individual i of an ABox $\mathcal{A}$ compute their most specific concept names w.r.t. a TBox $\mathcal{T}$ such that $\mathcal{T} \models C(a)$ and C is least w.r.t the subsumption ordering.

Most of the previous reasoning tasks are of great importance in the context of this dissertation, especially for analysis purposes. For instance, checking satisfiability of concepts in a KB describing business data will show if the business concepts make sense or whether there are contradictions. Also, knowing if a concept is more general than another one (concept subsumption) can help devising possible aggregations of data. Reasoning tasks over the ABox are also of paramount importance as an inconsistent ABox could draw new and erroneous inferences that would result in an inaccurate analytical view of the data. Reasoning tasks can help in this matter to detect quality issues in SW data subject to analysis.

2.1.4.2 DL Reasoning Complexities

This section briefly introduces various complexity results on reasoning with DLs. Although the topic of the thesis is not on (the scalability of) reasoning methods, reasoning may be used to check inconsistencies and to draw new inferences. However, as completeness of inferences and scalability are in conflict for expressive ontologies, the user should be aware of the reasoning complexities of the different DL languages and consider the trade-off between completeness and scalability when performing the analysis of data.

For DLs without full negation, e.g., $\mathcal{AL}$, all inferences can be reduced to subsumption. If a DL offers both intersection and full complement, satisfiability becomes the key inference of terminologies, since all other inferences can be reduced to satisfiability. Now the question is how difficult it is to deal with the reasoning problems introduced in the previous section for more expressive languages. While studies about the complexity of reasoning problems initially were focused on polynomial-time versus intractable, the focus nowadays has shifted to very expressive logics such as $\mathcal{SHIQ}$ whose reasoning problems are ExpTime-hard or worse. Exponential-time behavior is mainly due to disjunctions, where we can alternatively assign an element of a model to several classes and we have to check every alternative. However, if an AND source such as the existential restriction is present, we may have to expand the model with new elements. The complexity of the $\mathcal{S}$ family of languages that are relevant for the SW is listed in Table 2.2.

DL	Complexity
$\mathcal{S}$	PSPACE-complete (without TBox)
$\mathcal{SI}$	PSPACE-complete
$\mathcal{SH}$	EXPTIME-complete
$\mathcal{SHIF}$	EXPTIME-complete
$\mathcal{SHIQ}$	EXPTIME-complete
$\mathcal{SHOIQ}$	NEXPTIME-hard

Table 2.2: Complexity of Satisfiability for $\mathcal{S}$ languages.

The presence of a cyclic TBox can also increase the complexity of reasoning problems. Even for $\mathcal{AL}$, the presence of a general TBox leads to EXPTIME-hardness, which also effects description logics like $\mathcal{SH}$, which allow the internalization of general inclusion axioms. An extension with datatypes also effects the complexity. Given that we distinguish datatype properties, satisfiability is decidable if the inference problems for the concrete domain are decidable. Finally, adding role composition to some DLs derived from the $\mathcal{AL}$ family already leads to undecidability.

2.2 Modularization in SW data

This section is devoted to all the different aspects concerned with ontology modularization. In Section 2.2.1 we motivate the research in ontology modularization and identify several criteria used in different approaches. Then, in Section 2.2.2, we formally introduce the concept of ontology modularization. Section 2.2.3 reviews the main literature on ontology partitioning and Section 2.2.4 reviews ontology module extraction approaches. In particular, we analyze and discuss the differences between the traversal (Section 2.2.4.1) and the logical (Section 2.2.4.2) approaches, which are summarized in the discussion (Section 2.2.5).

2.2.1 Motivation

Ontologies are increasingly being incorporated in different disciplines such as knowledge management, e-Commerce and e-Science. The tasks for which they are used (e.g., question answering, reasoning, knowledge selection, integration, etc.) impose different constraints on the ontologies. The distributed nature of the SW, in which ontologies play a key role, imposes further constraints in the way ontologies are dealt with. Moreover, the design, maintenance, reuse, and integration of ontologies are complex tasks [36]. Therefore, there is a need for new tools and methodologies that ease the use of ontologies in an efficient manner.

Ontology modularization has been extensively studied as a means to identify a relevant fragment of an existing ontology. Modularization approaches differ significantly in terms of the concrete goal of the modularization and consequently, in terms of the criteria used to extract modules. However, we identify general situations where modularization has proved useful:

- *Distributed Systems.* In distributed environments like the SW, the question of modularization arises naturally. Ontologies are distributed in different places but need to interact. Having references to remote ontologies can lead to semantic inconsistencies. The introduction of modules with local semantics alleviates this problem.

- *Large ontologies.* Modularization can also help to manage very large ontologies, which are quite common, specially in Biomedicine. Modules give the user a better understanding and allow her to focus only on relevant parts of the ontology. Also, many visualization tools can benefit from modules by displaying only the necessary elements. Another argument for modularization is reuse. When having a large ontology two choices are available: either reuse the whole ontology, with the unnecessary overhead that this implies, or create the definitions that we need from scratch, which seems inappropriate if there already exists a standard ontology covering the domain. The alternative is to reuse only the relevant part of the ontology, a module.

- *Reasoning.* DL reasoners do not scale well with the size of ontologies. Thus, there is a motivation to reduce the size of the ontology, which can be done by extracting a module. Reasoning in a distributed environment can also cause problems. The introduction of modules with local semantics will help to localize the reasoning process.

Research in ontology modularization is quite extensive and the developed techniques are based on different formal and informal modularization criteria, usually driven by the application scenario of the modules. As stated in [40], there is no universal way to modularize an ontology and the choice of a particular technique or approach should be guided by the requirements of the application or scenario relying on modularization. While there is no agreement on the criteria to follow to evaluate modules, [40] classifies the most relevant criteria into types:

- *Logical criteria.* Some part of the ontology modularization community is concerned with the preservation of the logical properties of the modules, since modules are viewed as logical theories. In particular, *correctness* and *completeness* are studied. Correctness states that every axiom entailed by the module should also be entailed by the original ontology. Completeness refers to the ability of a module to infer the same entailments as the original ontology with respect to some signature.

- *Structural criteria.* Criteria based on the structure of the resulting module are also of great importance in some applications. As simple as it may appear, the *size* of a module is an important indicator, as it directly affects the maintainability and robustness of the module. Other structural criteria such as the *intra-module distance* have been explored, as the closeness of the elements in a module can be relevant for certain applications.

- *Application criteria.* Some criteria are selected based on the constraints imposed by the particular application that will use the modules. Some approaches rely on *assumptions on the ontology.* For example, there are approaches that are restricted to a certain ontology language or require a well-designed ontology, while others are agnostic to the language. The *level of user interaction* also varies among different approaches. Some are totally automatic while others need the user to tune some parameters or consider the modularization an interactive process. The *use of the module* made by the application also influences its construction. Modules used for reasoning will have different requirements from modules used for visualization purposes. In this respect, Palmisano et al. [111] perform a task-oriented evaluation of several module extraction techniques. They identify as common tasks for modules *instance retrieval*, *subclass retrieval* and *superclass retrieval.* Finally, the *performance* of the modularization process should be taken into account, as some applications will require dynamic construction of modules while others will do it as a batch process.

In the context of this dissertation, large amounts of ontological data (e.g., SW data) constitute the input data subject to analysis. Therefore, the need for modularization approaches that efficiently handle these data arises naturally. Modularization is needed to select only the relevant part of knowledge that is of interest for the analysis without losing semantic power. Moreover, as the analytical tools impose a series of restrictions over the data, the extracted data should be shaped to meet these requirements. As a result, we need efficient modularization techniques that reach a good compromise between logical and structural properties.

2.2.2 Ontology modularization

An ontology $\mathcal{O}$ can be defined as a pair $\mathcal{O} = (Ax(\mathcal{O}), Sig(\mathcal{O}))$ where $Ax(\mathcal{O})$ is a set of axioms and $Sig(\mathcal{O})$ is the signature of $\mathcal{O}$. The signature of an ontology $\mathcal{O}$ is the set of entity names occurring in the axioms of $\mathcal{O}$, i.e., its vocabulary. Different ontology modularization approaches have different assumptions about the definition of an ontology module. The assumption adopted for the following discussion is that a module is considered to be a significant and self-contained sub-part of an ontology. Therefore, a module $M_i(\mathcal{O})$ of an ontology

$\mathcal{O}$ is also a set of axioms (i.e., an ontology), with the minimal constraint that $Sig(M_i(\mathcal{O})) \subseteq Sig(\mathcal{O})$[2].

There are two different approaches that have been considered for the modularization of existing ontologies: ontology partitioning[3], which divides an ontology into a set of partitions or modules, and module extraction, which extracts a subset of an ontology focusing on a given set of elements.

2.2.3 Ontology Partitioning

Ontology partitioning is the process of splitting up an ontology $\mathcal{O}$ into a set of modules $\{M_1, M_2, ..., M_n\}$ such that each M_i is an ontology and the union of all modules is semantically equivalent to the original ontology $\mathcal{O}$. Note that some approaches being labeled as partitioning methods do not actually create disjoint partitions. In the following we present several approaches for ontology partitioning that have been developed for different purposes.

The work in [143] presents a method that produces sparsely connected modules based on a series of dependency measures between concepts. These measures are based on the structure of the class hierarchy among others. The idea is that strongly interconnected concepts should be part of the same partition. This approach can be used to support maintenance and reuse of very large ontologies by providing the possibility to individually inspect smaller parts of the ontology. The weak points are the dependency measures, which are agnostic with respect to the semantics of the ontology, and the termination point of the partitioning algorithm, which is rather arbitrary. These parameters can be tuned according to the requirements of a given application.

Contrary to [143], which do not consider the semantics of the ontology, other ontology partitioning methods such as [37] are focused on the problem of modular, distributed reasoning under DL, that is, they emphasize on the correctness and completeness of the local reasoning. This has led to the development of different logic frameworks for reasoning about distributed ontologies, which can be used in ontology partitioning algorithms. In [37] the problem of partitioning an OWL ontology using $\mathcal{E}$-connections is studied. The approach aims at preserving the completeness of local reasoning within all created partitions. $\mathcal{E}$-connections [71] is a formalism that allows to connect disjoint domains (partitions) by means of *link* properties. Reasoning can be performed on each partition or over a combination of linked partitions. The Distributed Description Logics (DDL) formalism [23, 134] provides mechanisms for referring to ontology concepts and for defining *bridge rules* that encode subsumption between concepts of different ontologies. Context OWL (C-OWL) [24] is an extension of DDLs that suggests several improvements, such as a richer family of bridge rules, allowing bridging between roles, etc. In contrast, C-OWL does

[2]Note that in some modularization approaches it is not always the case that $M_i(\mathcal{O})) \subseteq O$

[3]Although ontology partitioning is not the focus of this study, it is included for completeness

not allow to reuse foreign concepts in restrictions as in $\mathcal{E}$-connections. Another approach called Package-based Description Logics (P-DL) [12] tries to overcome the limitations introduced by $\mathcal{E}$-connections and C-OWL by allowing both subsumption between different ontologies, and foreign concepts in restrictions. The work in [144] defines modular ontologies in terms of a subset of DDL and provides rationales for the restrictions applied. They compute subsumption relations between external concepts off-line and store them as explicit axioms in the local ontologies. However, this modular approach can be computationally very expensive because it has exponential cost in the worst case. Although the previous logic frameworks could be used in ontology partitioning methods, they often extend OWL with non-standard syntax and therefore, semantics and scalability are not ensured, which contradicts the interoperability spirit of the SW.

For the purposes of this dissertation, ontology partitioning is not useful because the partitions are usually created without considering the user requirements. In an analytical process as the one that occupies this thesis, the extracted modules should take into account the user's requirements, as modules are used as an approach to extract only the knowledge of interest for the analysis.

2.2.4 Ontology Module Extraction

Ontology module extraction consists in extracting a subset from an ontology $\mathcal{O}$, the module M, that covers a specified signature Sig such that $Sig \subseteq Sig(M) \subseteq Sig(\mathcal{O})$. The module M is supposed to be the relevant part of $\mathcal{O}$ with respect to Sig.

The techniques for module extraction can be divided in two groups: traversal approaches and logical approaches. Traversal approaches consider the ontology as a graph and extract a module by traversing the graph. Logical approaches address the modularization from a logical perspective and are concerned with preserving logical properties in the modules such as coverage and minimality. Thus, these approaches explicitly consider the semantics of the ontology.

2.2.4.1 Traversal Approaches

A common, simple approach to modularize an ontology is to traverse the ontology structure (i.e., set of axioms), and apply heuristics to identify a sub-graph. Even though useful, such approaches do not take into consideration the underlying semantics of the ontology, and hence, do not generate modules that are complete (see Section 2.2.1). Nonetheless, these algorithms are tractable, intuitive to the user, and are useful for some specific applications. We discuss some of the most representative graph traversal-based modularization approaches in the remainder of this section.

Noy and Musen [109] propose a traversal view extraction from an ontology. The user selects a concept as starting point for the traversal and specifies which relations to explore and to what extent. Other traversal parameters can also be manually configured. The algorithm was integrated in PROMPT [108] and is basically targeted to assist users in manually extracting and inspecting parts of an ontology.

d'Aquin et al. [39] presented a modularization algorithm that is integrated into the larger process of knowledge selection. Knowledge selection aims to dynamically retrieve the relevant components from online ontologies to automatically annotate a web page that is currently being viewed in a web browser. The algorithm is based on exploiting the hierarchical relationships in an ontology. This algorithm requires no user interaction and relies on inferences during the modularization process. The objective is to extract the smallest part of the ontology covering an input set of terms via a fixed-point algorithm. To minimize the size of the modules, the class hierarchy is created only by including the most specific common superclasses of the classes contained in the module.

Seidenberg and Rector [132] developed a technique for extracting modules from the ontology GALEN [124] based on an input signature given by the user. Starting from a target concept, the algorithm extracts all its super and subclasses. Then, links across the hierarchy from any of the previously traversed classes are followed, and the targets of these links are also upwardly traversed. This process continues until there are no more links left to follow. The focus on GALEN makes it unclear how to generalize and apply the algorithm to other ontologies.

The approach proposed by Doran et al. [43] extracts a module from a single user-supplied concept that is self-contained, concept-centered and consistent. The ontology is transformed into an abstract graph model, thus the approach is agnostic with respect to the ontology language. The traversal is done recursively down the *is-a* hierarchy with some conditions changing to suit the language that the ontology is expressed in.

OntoPath [61] represents our first effort in extracting ontology fragments. The system was focused on the storage and management of ontologies within a graph-based database. According to the user requirements, which are expressed by a XPath-like query, OntoPath retrieves a graph-based fragment.

2.2.4.2 Logical-based Approaches

These techniques for extracting modules are based on rigorous logical foundations. In particular they emphasize on the following properties:

- *Correctness.* A module $M_i(\mathcal{O})$ of an ontology $\mathcal{O}$ should contain only the knowledge that is present in $\mathcal{O}$. That is, every axiom entailed by $M_i(\mathcal{O})$ should be also entailed by $\mathcal{O}$.

- *Completeness.* A module $M_i(\mathcal{O})$ of an ontology $\mathcal{O}$, extracted for a partic-

ular signature Sig, should contain all the information relevant to the elements of Sig. That is, every entailment of $\mathcal{O}$ that concerns only elements of Sig is preserved in $M_i(\mathcal{O})$. This ensures that there is no difference in the logical consequence from importing $M_i(\mathcal{O})$ versus $\mathcal{O}$ based on Sig.

Correctness of a module is trivial to satisfy since a module computed by extracting a subset of axioms from an ontology will only contain information about that ontology. Defining completeness is not trivial and has been defined based on the notion of conservative extension introduced by Lutz et al. [79]:

Definition 2.2 (Conservative Extension)**.** *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be $\mathcal{L}-TBoxes$, and let $\mathcal{S} \subseteq Sig(\mathcal{T}_1)$ be a signature. Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a $\mathcal{S}-$conservative extension of $\mathcal{T}_1$ if for all $C_1, C_2 \in \mathcal{L}(\mathcal{S})$, we have $\mathcal{T}_1 \models C_1 \sqsubseteq C_2$ iff $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$.*

The main intuition is that all the entailments regarding the signature of the ontology module are the same as if you take the union of the ontology module and the original ontology. That is, conservative extensions ensure local completeness of the modules such that knowledge contained in each module is not altered even after their integration. Conservative extensions depend on the description logic $\mathcal{L}$. In [79] a purely model-theoretic version of conservative extension was also studied, which does not depend on the language. However, they are problematic from an algorithmic viewpoint because they are highly undecidable even in the basic description logic $\mathcal{ALC}$.

In order to gain tractability one must sacrifice expressivity. The $\mathcal{EL}$ family of DLs has recently gained research interest due to its polynomial behavior[4]. Following this line, Konev et al. [69] developed a polynomial algorithm, MEX, for extracting conservative extensions from acyclic terminologies formulated in $\mathcal{EL}$ or $\mathcal{ELI}$. Although these DLs restrict the expressivity, they are widely being used in the biomedical domain to develop ontologies such as the Gene Ontology[5] and SNOMED[6].

Other approaches are based on approximations to address the problem of conservative extensions in more expressive DLs. This is the case of [36], where locality-based modules are defined to guarantee the properties of *coverage* and *safety* at the expense of minimality, which is guaranteed in conservative extensions. Coverage and safety are defined in [34]. The intuition of coverage is that the module should contain all the information about the terms of the signature such that it makes no difference between reusing the module or reusing the original ontology. The safety property guarantees that the meaning of the extracted terms is not changed when the module is imported by other ontologies.

In [35] two variants of locality are defined. Syntactic locality is based on the syntactic structure of the axioms and it can be computed in polynomial time.

[4]The EL profile has been defined in OWL 2 to address scalability issues in basic reasoning problems.

[5]`http://www.geneontology.org`

[6]`http://snomed.org`

Semantic locality is based on the interpretation of the axioms and is PSPACE-complete. Jiménez-Ruiz et al. [60] propose two different locality conditions for extracting ontology modules, depending on the interpretation of the entities outside the signature: $\top$-locality and $\bot$-locality. An axiom is considered $\top$-local if it does not define new sub-concepts for a given concept C and $\bot$-local if it does not define new super-concepts. Therefore, for a given signature, modules extracted using $\top$-locality will contain subconcepts (*lower modules* (LM)) and modules extracted using $\bot$-locality will contain superconcepts (*upper modules* (UM)). A third type of module named *lower of upper module* (LUM) is also identified, which is the result of extracting an upper module M for a signature S and then a lower module for S in M. This third type of module is the most restrictive and it is introduced in order to make the module smaller.

2.2.5 Discussion

The different approaches for ontology modularization presented in this section implement their own intuition about what a module should contain and what should be its qualities. As a result, each modularization technique has been driven by different criteria, usually imposed by the requirements of the application or scenario relying on modularization.

Focusing on ontology module extraction, the different starting assumptions made by both the traversal and logical approaches makes it difficult to draw a fair comparison. While logical approaches are more concerned with guaranteeing certain properties useful for reasoning, traversal approaches make emphasis in other criteria such as the size of the module or a compact distribution of the concepts, which suits better other applications. Therefore, the selection of a particular technique should be driven by the application requirements.

From the previous investigation it appears that there is a big gap between logical and traversal techniques, as they regard different criteria. As the objective of this thesis is to perform MD analysis over SW data driven by the underlying semantics, we require modularization approaches that are able to efficiently extract suitably-shaped modules for analysis that also preserve the semantics as much as possible. This fact has motivated us to investigate alternative modularization techniques that bridge this gap between logical and structural properties. In Chapter 4 we present a framework for ontology modularization that reaches a good compromise between logical and structural modularization criteria.

2.3 Multidimensional analysis over Semantic Web data

This section is devoted to all the different aspects concerned with the MD analysis of data and its development from structured and controlled scenarios

to semi-structured and knowledge-enriched SW data. In Section 2.3.1 we introduce several basic concepts related to MD analysis and discuss traditional approaches. Then, in Section 2.3.2, we discuss MD analysis techniques over data expressed in XML format, distinguishing approaches that target at homogeneous XML from those that deal with the harder problem of heterogeneous XML. Section 2.3.3 reviews the main literature that enhances traditional MD analysis with SW technologies and, in Section 2.3.4, we review techniques that aim at analyzing native SW data, which are the focus of this dissertation. In particular, we analyze and discuss approaches that follow a traditional architecture for this new type of data and point out to new trends for scaling the analysis. Finally, Section 2.3.5 summarizes the presented work and discusses the main points and limitations.

2.3.1 Basic concepts

The term business intelligence (BI) refers to all the decision support technologies aimed at making better informed and faster decisions in the enterprise environment. During the past two decades, businesses have been increasingly leveraging their data with sophisticated analysis techniques to get comprehensive knowledge and gain insight of their data. DW and OLAP are now mature technologies and have been traditionally applied in the field of BI.

The classic definition of a DW introduced by Inmon states that a DW is a subject-oriented, integrated, nonvolatile, and time variant collection of data in support of management's decisions [55]. Kimball introduced another widely accepted definition of a DW as a copy of transaction data specifically structured for query and analysis [66]. These definitions involve the construction of a huge repository where an integrated view of data is given at a particular time period, which is optimized for analysis purposes.

Traditional DW systems have three main components: the DW, the ETL tools and the analysis tools (see Figure 2.1). By means of the ETL tools, data coming from different sources are gathered, integrated (e.g., homogenized, cleaned, etc.) and loaded into the DW. The DW is the core of a BI system. It is a huge repository providing an integrated view of the data. Later, analysis tools (i.e., OLAP tools, data mining tools, etc.) can access the DW and exploit the information.

In this dissertation, we focus on the exploitation of information by OLAP tools, which are intended to ease analysis and navigation of the information in the DW. The term OLAP was first coined by Codd. In [33], Codd presented 12 rules to evaluate OLAP systems and emphasized the main characteristic of OLAP: the MD analysis.

Multidimensionality is based on the *fact/dimension* dichotomy. Figure 2.2 shows an example of an MD view of data. OLAP tools conceptually model the information in terms of facts, the central entities for the desired analysis (e.g., a sale), and dimensions, which provide contextual information for the facts

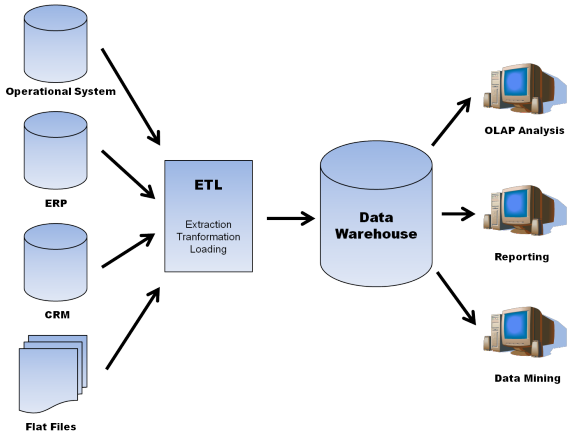

Figure 2.1: Traditional DW architecture

(e.g., the products sold). The dimensions are also known to be the different points of view (i.e., MD point) from where a fact can be analyzed. A *cube* consists of fact instances (or simply facts), where each fact is identified by a MD point (a point for each dimension) and quantified by measure values. Usually, the dimensions are hierarchically organized into levels. For instance, products can be grouped into product categories. Typically, the facts have associated numerical measures (e.g., the quantity sold or the total price), and queries aggregate fact measure values up to a certain level (e.g., total profit by product category and month). This provides the user an easy-to-understand and dynamic visualization of data by applying specific MD operators. For instance, the "roll-up" increases the aggregation level (e.g., from month to year), the "drill-down" decreases the aggregation level (e.g., from month to day), the "slice" performs a selection of a dimension (e.g., select product='car'), etc.

MD modeling has been for quite some time an active area of research. It was firstly introduced by Ralph Kimball at the logical level [66] and later by Matteo Golfarelli at the conceptual level [45]. Since then, many approaches have introduced or improved MD models either at the logical or the conceptual level [118, 146, 3]. Early MD methods view the modeling of facts and dimensions as a complex and manual process performed by the DW designer from the initial user requirements and the data sources. Other approaches try to assist the designer and sometimes even automate this complex process either by automatically analyzing the data sources (i.e., supply-driven approaches) or by formalizing the user requirements (i.e., requirement-driven approaches). Hybrid approaches combine both strategies. In the end, the DW is designed and loaded with the factual and dimensional data. Business end-users can

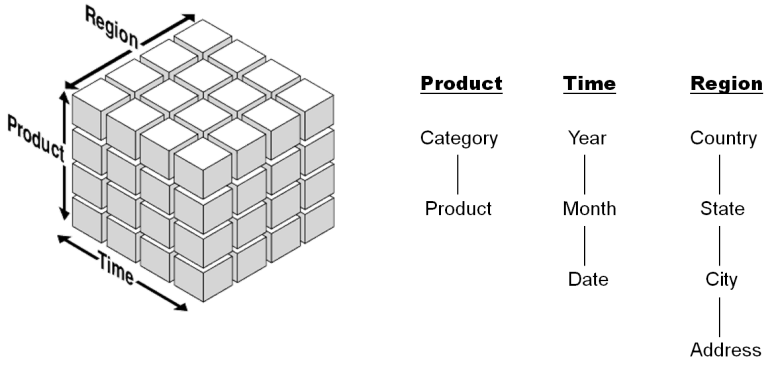

Figure 2.2: Multidimensional view of data

then pose their MD queries based on the facts and dimensions of the DW. A thorough review of MD modeling methods is presented in [127].

Traditionally, a fact has been conceived as an event of interest characterized by different perspectives (i.e., dimensions). Therefore, each fact must be related to each analysis dimension by a many-to-one relationship. That is, every instance of the fact is related to, at least and at most, one instance of an analysis dimension, and every dimension instance may be related to many instances of the fact. Therefore, many automated and semi-automated approaches for MD modeling have focused on the discovery of functional and inclusion dependencies from relational OLTP systems. They take as starting point a relational schema (i.e., logical schema) and look for many-to-one relationships in the data [121, 57, 82]. However, using the logical schema to derive an MD schema of a DW has disadvantages. The logical schema is driven by design issues, which will have an impact on the quality of the discovered MD schemas, as the usual way to represent functional and inclusion dependencies is by means of foreign and candidate key constraints. The previous limitation can be avoided by discovering the MD model from a conceptual formalization of the domain. Adding a conceptual layer on top of a relational system has been discussed in the literature and the benefits are obvious, as it provides more and better knowledge about the domain. Some approaches for MD modeling use the UML or ER diagrams of the data sources to derive the MD schema [27, 45, 84]. Unfortunately, these approaches are hard to automate because these conceptual formalizations are thought to graphically represent the domain and not for querying and reasoning. A recent approach [126] uses ontologies as the conceptual layer to automate the design of the DW from relational sources. We will analyze this work in Section 2.3.3.

Regarding the implementation of the MD model, there are two main trends: using the relational technology (ROLAP) and/or using a (usually proprietary)

MD implementation (MOLAP). In ROLAP, the data is stored in relational tables. In order to map the MD data cubes into tables, different logical schemas have been proposed. The *star* and the *snowflake schemas* are the most commonly used. The star schema consists of a fact table plus one dimension table for each dimension. Each tuple in the fact table has a foreign key column to each of the dimension tables and numeric columns that represent the measures. The snowflake schema extends the star schema by normalizing and explicitly representing the dimension hierarchies. In the MOLAP alternative, special data structures (e.g., MD arrays) are used for the storage instead. The combination of ROLAP and MOLAP is known as Hybrid OLAP (HOLAP). In the HOLAP approach, detailed data is usually stored in relational tables, whereas the MOLAP strategy is applied to manage aggregated data. ROLAP implementations as shown in Kimball's book [66] have been the reference architecture for years, as relational databases are a mature and well-established technology. However, these implementations also have some drawbacks (mainly their performance for query answering), which has made these architectures to lose popularity. In any case, the simplicity of the star schema has remained and dominates the MD modeling landscape both at the conceptual and logical level.

2.3.2 DW and OLAP on the Web

DW and OLAP have been successfully applied within the database community for analysis purposes, but always under a well-controlled and structured scenario. MD modeling has been traditionally restricted to structured, relational data within the company. However, the eruption of XML and other richer semi-structured formats like RDF has shifted the attention of the DW community to a much more heterogeneous and open scenario than that of traditional BI applications. The work in [56] outlines the opportunity and importance of using unstructured and semi-structured data (either textual or not) in the decision making process. These data could still come from the sources in the company, but also from the Web. It is clear the benefit of enriching our MD model with information coming from the Web, since it can provide new points of view, new aggregation levels, or even new measures to analyze. Currently no one questions the need of adding all this external information to the traditional corporate analysis processes. In the following, we distinguish between approaches aimed at analyzing homogeneous and well-structured XML and approaches that address heterogeneous XML sources.

Analytics for homogeneous XML sources.

Earlier work about XML DW [44, 22, 135] focused on translating XML Schema into relational structures in order to apply existing algorithms for creating a DW conceptual schema from relational sources. However, relational databases are not the best choice for complex data that is deeply hierarchical, irregular and recursive. Although it is possible to build a relational implementation

of this type of data the translation may not be trivial, leading to some loss of information. In order to avoid this, a lot of research has considered the design of the DW directly from XML sources. Among the first attempts we find [58], [122] and [46, 147]. These approaches address the physical integration of XML sources into an MD schema starting from the document type definitions (DTDs) or XML Schemas that describe the structure of the XML documents. Once the XML data is loaded into the database, the traditional OLAP techniques can be used for querying. Although these approaches provide good query performance, they are not suitable for dynamic environments such as the Web, as physically integrating data is typically a long, time-consuming process. To overcome this issue, Pedersen et al. [117] aim at the logical integration of OLAP and XML data sources. Their approach allows the execution of OLAP operations that involve data contained in external XML data, which is accessed by using XPath[50]. In this way, XML Web data can be used as dimensions and/or measures of the OLAP cubes. This implies extending the existing OLAP techniques to allow the execution of queries that involve online XML data. In [113] it is assumed that each XML document describes a single dimension or a single fact record and XQuery[50] is used for query processing. In general, most of the previous approaches assume XML documents are given along with either a DTD or XML Schema and they usually work well with very homogeneous and controlled XML data (e.g., XML documents generated from relational data sources). Unfortunately, data in real world scenarios are usually incomplete and much more heterogeneous. A review and deep discussion of these and other DW approaches for XML and Web data can be found in [120].

Analytics for heterogeneous XML sources.

As the XML document landscape covers from highly structured and homogeneous data to heterogeneous documents, some literature has focused on more complex XML scenarios where documents have a high structural heterogeneity. In [74, 150] it is shown that XQuery is not the most suitable query language for data extraction from heterogeneous XML data sources, since the user must be aware of the structure in the underlying documents. The lowest common ancestor (LCA) semantics can be applied to extract meaningful related data in a more flexible way. The work in [74, 150] applies some restrictions over the LCA semantics. In particular they propose smallest lowest common ancestor (SLCA) [150] and meaningful lowest common ancestor MLCA [74] whose general intuition is that the LCA must be minimal. However, in [88, 106] they showed that these approaches still produced undesired combinations between data items in some cases (e.g., when a data item needs to be combined with a data item at a lower level of the document hierarchy). In order to alleviate the previous limitations they propose the smallest possible context (SPC) data strategy where they redefine the notion of closeness of data item occurrences

in an XML document.

Although XML has been adopted as the de-facto standard for publishing Web data, it only formalizes the structure of a document and not its content. The tags do not have formally defined semantics and thus their meaning is not well-defined. This fact has prompted the appearance of other semantic tagging languages (i.e., RDF/S, OWL) to add machine understandable and semantic annotations to Web documents in order to access knowledge. This is a great opportunity for the DW community, as there is a strong agreement about bringing more semantics to the analytical processes. As DW mainly involves the integration of disparate information sources, semantic issues are highly required for effectively discovering and merging data.

The remainder of this chapter discusses the first approaches that combine DW and OLAP technologies with SW data. In particular, we distinguish between approaches that incorporate semantics in some parts of the traditional analysis architecture (Section 2.3.3) and approaches aimed at directly analyzing SW data (Section 2.3.4).

2.3.3 Extending DW and OLAP with Semantic Web technologies

Although the prime objective of the literature discussed in this section is not to analyze SW data, we consider it relevant in the context of this dissertation because they are pioneers in using SW technologies to enhance the traditional DW and OLAP analysis processes. First, we discuss approaches that use ontologies as middleware to represent and query business data. Then, we review literature that focuses on enhancing specific parts of the DW process by taking advantage of SW technologies. In particular, we review approaches that address the conceptual design of the ETL process, and approaches that focus on the MD design of the DW. The work in [18] emphasizes the benefits of combining SW technologies with BI and surveys the main approaches.

Using ontologies to represent and query business data

State-of-the-art research in the BI area proposes the use of ontologies as a semantic middleware for integrating data from heterogeneous information systems. In [141] a layered architecture is proposed where each data source schema is independently mapped to a technical ontology (TO) and the various TOs are connected to a business ontology (BO) to relate technical concepts to business-level concepts. At the application layer, the user can specify queries based on a graph representation of the BO, which contains business relevant vocabulary that is familiar to business users and therefore, intuitive and easy to understand. In [133] the authors differentiate between the domain ontology, which provides the terminology of the business domain, and the BI ontology, which models the concepts used to describe how the data is organized in data sources

(i.e., OLAP concepts) and to map such data to the concepts described in the domain ontology. Neumayr et al. [104] present a BI front-end extended with SW technologies that assists and guides the business analyst. The system is enabled with reasoning and several kinds of knowledge are explicitly represented by ontologies, including both internal and external organization knowledge, the semantics of measures and scores, knowledge about insights gained from previous analysis and so on.

In most of the previous approaches, SW technologies are thought to provide a conceptual view that integrates the data and makes querying easier. However, specific wrappers that map each of the data sources to the ontologies must be manually built and maintained.

Extending the ETL process with ontologies

During the initial steps of a DW project, the main goal is to construct a conceptual ETL design that describes the corresponding data transformations needed to map the data sources to the target DW concepts. For achieving that, it is imperative to identify and understand the semantics of both the data sources and the target data stores. Several approaches have been proposed for using SW technology to the design and construction of the ETL part [138, 139]. Naturally, most of them deal with the conceptual part of the ETL design, since the SW paradigm seems as a promising means to overcome the lack of handy ways for capturing the semantics of an ETL process. The prevailing so far idea in using SW technology for ETL suggests using a global ontology for mapping all the involved data stores to it. However, the use of an OWL ontology, instead of a global schema provides a formal model on which automated reasoning mechanisms may be applied. Furthermore, [138, 139] point out that in ETL it is not sufficient to consider the integration problem as a query rewriting problem, since the transformations taking place in real-case ETL scenarios usually include operations, such as the application of functions, that cannot be captured by a query rewriting process. In [140, 136] the authors propose ontologies to formally and explicitly specify the semantics of the data source and the DW schema and thus, through reasoning, automate in a large extent the ETL generation. However, the ontology does not exist a priori, and requires designers to explicitly indicate mappings between the data sources to be integrated and the DW schema, which must be known. In this case, ontologies act as a meta-model that guides the transformation process.

DW MD design driven by ontologies

The design of a DW is a complex and prone-to-fail task that requires some level of expertise and domain knowledge. Although several methods that use ontologies to assist the design of DWs have been proposed, we focus on the most prominent and close approaches to our research.

Given that ontologies provide a semantically rich formalism, approaches such as [126] try to incorporate semantics in the design of a DW MD schema by taking as starting point a domain ontology that describes the data sources. Instead of looking for functional dependencies (which constitute typical fact-dimension relations) in the sources, they are derived from the ontology in a semi-automatic way by using an ad hoc algorithm. The application of this work is valid in scenarios where a single ontology of reduced size, with multiplicity restrictions, is used for annotating the source data. Note that multiplicity information is rarely found in the source ontologies. A refined approach to fully exploit DL reasoning services and compute functional dependencies is discussed in [129]. This work restricts the language of the input domain ontology to $DL-Lite_A$ [8] to compute functional dependencies based on role compositions by exploiting its query answering services for conjunctive queries. This work addresses only the design phase, overlooking the process of data extraction and integration to populate the MD schema. The framework in [128] proposes to extract an MD schema from each domain ontology and later conciliate those results in a single, detailed MD schema. Thus, the semantic integration of the resulting schemas must be performed a posteriori. Notice that this approach is mainly supply-driven. Recently, the authors have incorporated the user requirements into this framework [130].

The approach in [65] introduces a methodology for designing DWs from ontology-based operational databases. The authors propose a pre-defined global ontology into which data source ontologies are loosely integrated. User requirements are expressed as queries over the global ontology, which are executed to build up the DW conceptual model (local ontology). In this approach, reasoning is applied to classify and validate the classes of the local ontology and they use modularity to build the local ontology. As a limitation, the methodology supposes the existence of the global ontology and mappings between the global and local ontologies must be performed. Moreover, the generation of the MD schema is built based on a set of simple heuristics applied to the user requirements, which are expressed using an ontological query language.

2.3.4 DW and OLAP over Semantic Web Data

The previous section has discussed approaches that make use of semantics to enhance some phase of the analysis of traditional data sources. The focus here is different, as we discuss how existing approaches address the analysis of data sources that are natively semantic. These semantic data sources have become available on the Web thanks to initiatives such as the Web of Linked Data, which promotes a mechanism for making data available on the Web using languages such as RDF and the HTTP protocol. We discuss the state-of-the-art literature ranging from approaches that follow the traditional DW architecture to analyze these new sources to lighter approaches that break with the traditional DW architecture and perform more ad hoc analysis to meet the

scalability and freshness requirements imposed by both the needs of analysts and dynamic environments typical for SW data.

Traditional SW data analysis

Niemi et al. [105] are among the first to completely implement the traditional DW workflow (i.e., the ETL process, the population of the DW and the cube construction) using SW technologies. Although this research could be categorized under Section 2.3.3 because it uses SW technologies to enhance the DW and OLAP process, we discuss it in this section because they address the complete analysis process, from the data sources to the analyst, and they transform the target data sources into RDF, which later populates the DW. In order to integrate data from different sources, the authors consider RDF as the common data format. They suggest starting with mapping the sources to an RDF/OWL ontology. Then, the user needs to design the structure of the OLAP cube. Next, the OLAP cube is constructed based on RDF queries issued on the data sources. At the instance level, the combined result of such queries represents an instance of the OLAP cube. As a constraint, the method requires the mappings to translate the data to RDF format. An extension to this work discusses with more detail the method for automating the construction of OLAP schemas [107]. Again, the source and target schemas are considered as known. They are aligned by converting the data in RDF using ontology mappings. Then, the relevant source data are extracted using RDF queries generated with the ontology describing the OLAP schema. At the end, the extracted data are stored in a database and analyzed using typical OLAP techniques. Both approaches aim at an end-to-end design approach, but they have two main limitations. First, they both require prior knowledge of the source and target schemas and second, they consider only simple data transformations.

The work in [101] proposes an MD framework for analyzing semantic annotations from a logical viewpoint using ontologies. In this approach semantic annotations are based on application and domain ontologies. They are kept in a semantic DW (SDW) expressed in RDF/OWL format. The user can build an MD integrated ontology (MIO) containing the required analysis measures and dimensions by selecting concepts and properties from the available ontologies in the SDW. Then, the necessary logical modules are extracted to set up a global ontology from which facts and dimensions will be validated and generated from the annotations in the SDW. Although this approach follows the traditional DW architecture, it has several distinguishing features. Firstly, if data sources are already expressed in RDF/OWL, then their inclusion in the SDW is straightforward. In such scenario where multiple ontologies co-exist together, ontology mappings and merging strategies can be semi-automatically calculated as in current semantic integration models. Second, as the MD design is performed according to the user requirements, which are expressed in terms

of the current ontologies in the SDW, the resulting models fit their specific needs at any time. However, the manual identification of the MD concepts (facts, dimensions, measures, roll-up relationships) by the user is not straightforward, as it requires knowledge about the ontology. Also, the alternatives for the instantiation of the MIO to build OLAP cubes are sketched but not implemented.

The work in [63] proposes a preliminary approach to analyze through OLAP queries a specific type of SW data, statistical Linked Data (sLD). These data are expressed in RDF and annotated using the RDF Data Cube vocabulary (QB)[7], which is a vocabulary for annotating and publishing data in an MD format. The paper proposes a traditional DW architecture to store and analyze sLD, where the sources are selected by the user using SPARQL[8] and the system creates and instantiates an MD model based on the QB annotations. In this approach there is no MD design because sLD is already expressed as MD data, therefore, the user requirements are not taken into account. Moreover, reasoning is not addressed. The same authors present in [64] an approach that directly defines OLAP operations over the source data (i.e., sLD). They define an MD model based on QB and map OLAP operations in this MD model to SPARQL queries over the sources. As data is queried on demand and no materialization is done, freshness of results is always guaranteed. However, the approach suffers from efficiency issues as the computational cost falls upon the SPARQL processor, which shows inefficient for complex MD queries.

An alternative analysis approach is presented in [38], which combines information extraction (IE) techniques with logical reasoning. The work proposes an MD model specially devised to select, group and aggregate the instances of an ontology. The result of these operations is a set of tuples, whose members are instances of the ontology concepts. They also present the adaptation of a feature selection algorithm to discover interesting potential analysis dimensions. This algorithm builds the dimension hierarchies by selecting the relationships in the ontology that maximize the information gain. The main limitation of this work is that the MD operations (i.e., aggregations) are performed inside the ontological formalism, which can lead to undecidability.

Scaling Semantic Web data analysis

Traditional DW and OLAP has focused on creating MD models and DWs that imply high modeling efforts and are devoted to process structurally similar queries about day-to-day business data. These approaches fail to incorporate external data that is not considered in the pre-defined data model. Some of the approaches presented previously try to overcome this issue by taking advantage of the conceptual layer that SW data provides and letting the analyst create the MD schema according to her requirements.

[7] http://www.w3.org/TR/vocab-data-cube/
[8] http://www.w3.org/TR/rdf-sparql-query/

Recently, some efforts such as *Live BI* [31], have dealt with the changing business environment by continuously processing data streams, whose results can be piped directly to the analysis layer without loading it first into a DW. Although these approaches are able to provide if not fresh, current answers, the dynamic extensibility of the system may not be trivial, as it usually involves the development of a wrapper for each new added stream. Similar to this idea is the new BI scenario proposed in [77], *Situational BI*, where the analytical tools should be able to integrate information from several data sources to satisfy the users' analytical queries, which are specific and momentary. However, these approaches have not yet tackled the problem from a semantic viewpoint. We believe that these approaches would enormously benefit from adding semantics into the analytical processes.

The MD analysis of SW data, which is the issue that this thesis deals with, shares some of the previous requirements (e.g., freshness) and new ones that require new solutions. One of them is given by the semi-structured nature of the data model, which consists of RDF labeled graphs (i.e., graphs where edges express binary relations between nodes). Much effort has been devoted to the design and implementation of scalable RDF stores. Earlier work on RDF storage engines includes main memory stores [30] and also stores laying on top of relational databases [26, 149]. Recently, native stores have been developed with sophisticated storage and indexing schemas for achieving Web scale performance. Among the most prominent, we cite vertically partitioned column stores [2] and stores using multi-indexing techniques [148, 103, 49]. At the same time, other research efforts have focused on the development of systems oriented to massive OWL storage, most of them using relational technology as back-end [53, 78, 112, 125, 76]. However, as massive-scale data sources are becoming common, the previous techniques may pose limitations for ad hoc processing and analysis on the Web.

Analysis of SW data implies powerful processing mechanisms able to handle large joins derived from constructs such as pattern matching, grouping and aggregation. The join operations are needed to transform data into n-ary relations suitable for OLAP operations. Thus, massive parallel processing systems seem a good candidate. The MapReduce programming model [41] is gaining momentum for processing large analytical workloads. However, this model is not appropriate to compute costly joins, as it was designed to work with a single dataset. Several optimizations have been proposed to make the problem of join computing more efficient [4, 5, 42]. To achieve scalable ad-hoc analytics of RDF data, the authors in [142, 123] have developed a system, RAPID, based on the MapReduce framework, whose operators support efficient expression and parallelization of complex analytical queries. They have extended Yahoo's Pig Latin language [110] with query primitives for dealing with the graph structured nature of RDF, that is, the MD-join operator. This approach allows Web scale processing and freshness of results, as the MD operations are performed ad hoc over the MapReduce framework based on the user query. However, the perfor-

mance and suitability of such techniques for complex processing compared to databases is still being questioned [115].

Another inherent issue in the analysis of SW data is reasoning. Reasoning allows making explicit all the implicit knowledge derived from the semantic annotations. Scalable reasoning is a crucial task that should be integrated in the analysis of SW data. Reasoning can be performed either at query time (backward reasoning) or beforehand (forward reasoning or materialization). Materialization strategies seem appropriate only for small or medium-sized data sets, where the storage costs are admissible. The advantage of materialization is mainly efficiency at query time, which is desirable in analytical tools. On the other hand, backward reasoning does not incur on storage costs but offers poor performance at query time. The complexity of reasoning also depends on the expressivity of the logic used (see Section 2.1.4.2). The more expressive the logic, the more computationally expensive reasoning becomes.

Up until now, reasoning has been incorporated to analytical processes by means of traditional reasoners, which have been designed with a centralized architecture where the execution is carried out by a single machine. However, with the advent of large-scale SW data, distributed reasoning is being approached. Distributed reasoning is significantly more challenging because it requires new algorithms to efficiently share both data and computation. Some preliminary approaches have been devised for distributed reasoning that seek a good trade-off between logic complexity and performance. This is the case of [145], where the MapReduce framework is used to perform materialization of a quite restricted logic. However, materializing inferences of large scale SW data can lead to an inadmissible increase of the storage requirements. A promising research direction that has not yet been fully explored is the use of indexing mechanisms to manage inferred data without the need of materialization. In this dissertation, we shed some light in this direction. Similarly, the study and application of RDF compression techniques could also be beneficial in this matter [80, 72].

2.3.5 Discussion

This section has presented the main state-of-the-art approaches that combine the fields of MD analysis (i.e., DW and OLAP) with the SW from different viewpoints. Either to extend and enhance the traditional analytical processes or as a knowledge-enriched data source, the truth is that SW technologies can only improve and bring more insight into the analytical tools.

Focusing on the MD analysis of SW data, only a few approaches have addressed this issue, mainly in a conservative manner. However, the analysis scenario has changed, and so the analytical requirements. The analysis is no longer restricted to well-defined, structured and static data sets. The availability of large-scale SW data is demanding new analytical tools that are able to dynamically access these data and provide fresh results according to the ana-

lyst requirements at any time. Therefore, the traditional analytical processes where a huge DW is created by integrating a set of pre-defined data sources by means of complex and expensive ETL processes is not suitable anymore. Some state-of-the-art literature have addressed Web-scale analytics by using parallel processing over a MapReduce framework. Although the first results are promising, it is not clear if these approaches are the most suitable to the kind of operations required by OLAP tools. Moreover, this framework introduces new challenges such as distributed reasoning that need to be dealt with.

In this dissertation the problem of MD analysis of SW data is approached by reaching a compromise between scalability, freshness of results and user requirements. Scalability is achieved by two means. On one hand, we provide indexing mechanisms over ontologies that allow to manage implicit data in a compact way, thus, operations requiring reasoning can be efficiently solved using the indexes. On the other hand, we have developed several modularization techniques that build upon the previous indexes and allow to extract and work only with the ontological subsets of interest. Moreover, instead of building a huge, one time DW, we only materialize the facts and dimensions required by the user during an analytical session. This provides up-to-date and customized results to the user.

Chapter 3

Framework

This chapter presents the developed methods for scalable analysis of SW data under an integrated framework. We explain each of the components of the framework in Section 3.1. Then, we introduce the application scenario and use case for the developed framework in Section 3.2.

3.1 Component-based framework

The methods that this thesis investigates for efficient and scalable management and analysis of SW data are provided as modules in a component-based framework. Fig. 3.1 shows such framework.

The OIM is composed by the indexing and querying modules. This component is formalized and explained in Chapter 4. The indexing module is shown at the bottom of the figure and is composed by a series of methods applied to the ontology axioms. This phase is usually performed off-line only once per ontology. A DL reasoner is used to compute the inferred ontology hierarchy. Then, the interval labeling scheme allows to index this inferred hierarchy in a compact and compressed format. The ABox assertions are also indexed according to the TBox indexes. As a result, we obtain the TBox indexes and the indexed instance store.

The querying module is divided into the TBox and ABox querying. Part of the TBox querying module is an interval's algebra that allows to perform efficient queries about ancestors and descendants of concepts. The DL querying module has been designed to answer a restricted subset of DL queries by using only the indexes. The ABox querying module allows to perform conjunctive query answering over named concepts and properties of the ontology.

The OMETs are a series of modularization techniques that build upon the OIM. In particular, we have developed four different modularization approaches (i.e., S, SCA, ASA and ASA-ST) to account for different structural and logical requirements. Moreover, the DAG component allows to extract modules with

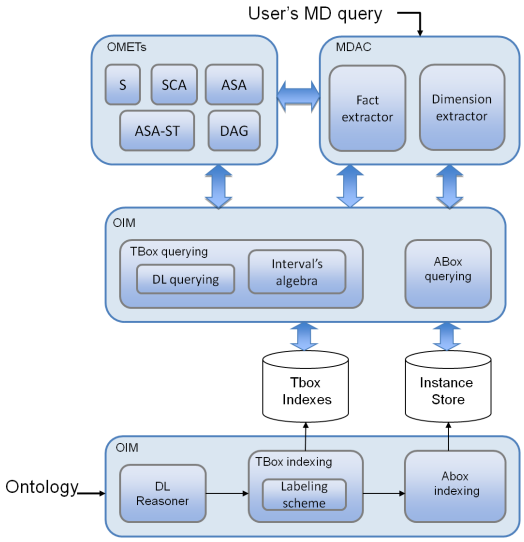

Figure 3.1: Component-based framework for the management and analysis of SW data.

a graph-based structure instead of a tree. This component is also described in Chapter 4.

Finally, the MDAC allows to perform MD analysis of instance data driven by the semantics encoded in the ontology axioms. The module is composed by the fact extractor and the dimension extractor. The fact extractor is in charge of mapping the elements of the MD query specified by the user to the elements in the ontologies and finding semantic paths (i.e., aggregation paths) that logically connect them. As a result, the fact extractor composes facts with the instance data that adheres to these semantic paths, which in turn accounts for the MD query of the user. This task is performed by using the TBox indexes to extract the semantic paths that connect the MD elements and the indexed instance store to retrieve the facts. The dimension extractor module is in charge of extracting truly semantic hierarchical dimensions based on the subsumption relations of the ontology. The hierarchies are re-shaped to favor aggregation while preserving as much as possible the semantics of the concepts. This task is performed by using the modularization techniques, which in turn make use of the TBox indexes.

The next section describes an application scenario where this analysis framework has been successfully deployed.

3.2 Application scenario

The application scenario selected to develop our analysis framework is Biomedicine, in which vast and complex domain ontologies are being developed. In particular, we focus on the Health-e-Child integrated project (HeC) [137]. HeC was an European funded project that aimed at improving personalized healthcare in selected areas of paediatrics, particularly focusing on integrating and providing decision support tools for medical data across disciplines, modalities, and vertical levels such as molecular, organ, individual and population. Such decision support tools are mainly focused on traditional diagnosis/prognosis tasks and patient follow-up. The project focused on some carefully selected diseases in three different categories; pediatric heart diseases, inflammatory diseases and brain tumours. This scenario regards three main types of data sources: well standardized records coming from hospital information systems (e.g., HL7-conformant records), highly heterogeneous semi-structured clinical reports, and a variety of unstructured data such as DICOM files, ECG data, X-ray and ultrasonography images. All data are delivered to the HeC infrastructure in XML format, usually lacking of any schema for the semi- and unstructured data sources. Unfortunately, most of the interesting analysis dimensions are located in the schema-less data sources.

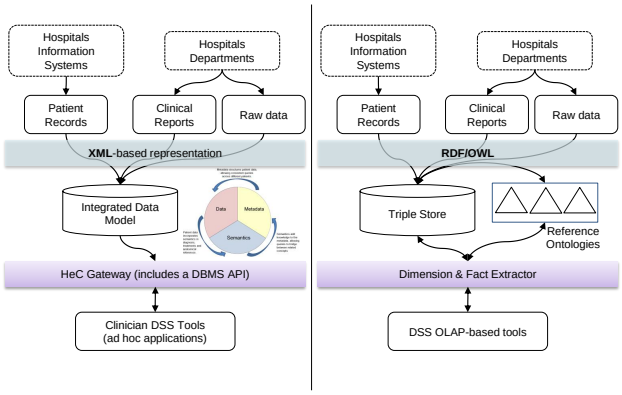

Figure 3.2: HeC data integration architecture (left hand) versus the SW integration architecture (right hand)

The integration solution proposed in the HeC project formerly consisted in defining a flexible integrated data model [15], which is built on top of a grid-aware relational database management system (see left hand side of Figure 3.2). As clinical reports present very irregular structures (see the example presented in Figure 3.3), they are decomposed and classified into a few organizational units (i.e., patient, visit, medical event and clinical variables), thus disregarding much information that could be useful for analysis. Additionally, this data

model also includes tables that map clinical variables to concept identifiers from the UMLS Metathesaurus [20]. These tables are intended to link patient data to bibliographic resources such as MEDLINE [1]. It is worth mentioning that a similar scenario and solution were proposed in the OpenEHR initiative[1].

```
Ultrasonography
     WristExamination
        date '10/10/2006'
        hasUndergoneWrist True
        rightWrist False
        leftWrist True
     WristScore
        wristExamined 'Left'
        PannusAndJointEffusion
           distalRadioUlnarJoint
              result 'No Synovial thikening and no joint effusion'
           radioCarpalJoint
              result 'Only synovial pannus without joint efussion
           midCarpalCMCJ
              result 'None'
        Synovitis
           distalRadioUlnarJoint 'Mild'
           radioCarpalJoint 'None'
           midCarpalCMCJ 'Severe'
        BoneErosion
           distalUlna
              erosionDegree 0
           carpalBones
              erosionDegree 1
...
```

Figure 3.3: Fragment of a clinical report of the Rheumatology domain.

In this dissertation we adopt a different integration architecture (see right hand side of Figure 3.2), which follows the current trends in biomedical [21] and bioinformatics data integration [62], and relies entirely on SW technology. In this architecture, the integrated data model is defined as an *application ontology* that models the health care scenario (e.g., patients, visits, reports, etc.) This ontology can import concepts defined in external knowledge resources (*reference ontologies* in Figure 3.2). For example, we can import a disease taxonomy from UMLS, or a classification of the body parts from GALEN [124]. Finally, the biomedical data is semantically annotated according to the application ontology and stored as RDF triples (i.e., triple store).

For practical purposes, the TBox and the ABox are treated separately. Notice that while the ABox is usually very dynamic for it is constantly updated, the TBox hardly changes over time. We assume that the instance store is always consistent w.r.t. the associated ontology. Figure 3.4 shows a fragment of the application ontology designed for patients with rheumatic diseases, whereas Table 3.1 shows a fragment of an instance store associated to this ontology. In this case, the instance store is expressed as triples $(subject, predicate, object)$, where a triple of the form $(a, type, C)$ corresponds to a DL assertion $C(a)$, and otherwise the triple (a, R, b) represents the relational assertion $R(a, b)$.

The UMLS Metathesaurus is used as domain ontology as it is probably the most comprehensive biomedical knowledge resource. For the sake of clarity,

[1] `http://www.openehr.org/home.html`

$$
\begin{aligned}
Patient &\sqsubseteq\ =1\ hasAge.string && (3.1)\\
Patient &\sqsubseteq\ =1\ sex.Gender && (3.2)\\
Patient &\sqsubseteq \forall\ hasGeneReport.GeneProfile && (3.3)\\
GeneProfile &\sqsubseteq \forall\ over.Gene\ \sqcap\ \forall\ under.Gene && (3.4)\\
Patient &\sqsubseteq \forall\ hasHistory.PatientHistory && (3.5)\\
PatientHistory &\sqsubseteq \exists familyMember.Family\_Group \sqcap \\
&\quad \exists\ hasDiagnosis.Disease\_or\_Syndrome && (3.6)\\
Patient &\sqsubseteq \exists\ hasVisit.Visit && (3.7)\\
Visit &\sqsubseteq\ =1\ date.string && (3.8)\\
Visit &\sqsubseteq \forall\ hasReport.(Rheumatology\ \sqcup\ Diagnosis\ \sqcup \\
&\quad Treatment\ \sqcup\ Laboratory) && (3.9)\\
Rheumatology &\sqsubseteq \exists\ results.(Articular\ \sqcup\ ExtraArticular\ \sqcup \\
&\quad Ultrasonography) && (3.10)\\
Rheumatology &\sqsubseteq\ =1\ damageIndex.string && (3.11)\\
Ultrasonography &\sqsubseteq \forall\ hasAbnormality.Disease\_or\_Syndrome && (3.12)\\
Ultrasonography &\sqsubseteq \forall\ location.Body\_Space\_or\_Junction && (3.13)\\
ArticularFinding &\sqsubseteq \exists\ affectedJoint.Body\_Space\_or\_Junction && (3.14)\\
ArticularFinding &\sqsubseteq \forall\ examObservation.string && (3.15)\\
Diagnosis &\sqsubseteq \exists\ hasDiagnosis.Disease\_or\_Syndrome && (3.16)\\
Treatment &\sqsubseteq\ =1\ duration.string && (3.17)\\
Treatment &\sqsubseteq \exists\ hasTherapy.DrugTherapy && (3.18)\\
DrugTherapy &\sqsubseteq\ =1\ administration.AD && (3.19)\\
DrugTherapy &\sqsubseteq\ =1\ hasDrug.Pharmacologic\_Substance && (3.20)\\
AD &\sqsubseteq \exists\ dosage.string\ \sqcap\ \exists\ route.string\ \sqcap\ \exists\ timing.string && (3.21)\\
Laboratory &\sqsubseteq \exists\ bloodIndicants.(\ \exists\ cell.Cell\ \sqcap \\
&\quad \exists\ result.string\ \sqcap\ \exists\ test.Lab\_Procedure) && (3.22)\\
Rheumatoid\_Arthritis &\sqsubseteq Autoimmune\_Disease && (3.23)\\
Autoimmune\_Disease &\sqsubseteq Disease\_or\_Syndrome && (3.24)\\
&\ldots && (3.25)
\end{aligned}
$$

Figure 3.4: Ontology axioms (Tbox).

Table 3.2 shows only a small excerpt of axioms of UMLS with the "is-a" relationships among diseases and drugs related to the application scenario. The abbreviations used in the ontology are shown in Table 3.3.

3.2.1 Use Case

In the previous application scenario, we propose a use case that will serve as running example to illustrate all the developed methods.

The use case is focused on the MD analysis of the efficacy of different drugs in patients diagnosed with inflammatory diseases. The analyst of this use case expresses her analysis requirements at the conceptual level in terms of dimensions and measures, similarly as in traditional MD analysis. Figure 3.5 depicts the conceptual model of the analyst requirements where the central subject of analysis is *Patient*. The analyst is interested in exploring the patient's follow-up visits according to different perspectives (i.e., dimensions) such as gender, the drugs prescribed and the diseases diagnosed. She is particularly interested in obtaining figures about the registered articular damage and the number of

subject	predicate	object
PTNXZ1	hasAge	"10"
PTNXZ1	sex	Male
VISIT1	date	"06182008"
VISIT1	hasReport	RHEX1
RHEX1	damageIndex	"10"
RHEX1	results	ULTRA1
ULTRA1	hasAbnormality	"Malformation"
ULTRA1	hasAbnormality	Knee
VISIT1	hasReport	DIAG1
DIAG1	hasDiagnosis	Arthritis
VISIT1	hasReport	TREAT1
TREAT1	hasDrugTherapy	DT1
DT1	hasDrug	Methotrexate
PTNXZ1	hasVisit	VISIT2
VISIT2	date	"08202008"
VISIT2	hasReport	RHEX2
RHEX2	damageIndex	"15"
RHEX2	results	ULTRA2
ULTRA2	hasAbnormality	"Malformation"
ULTRA2	hasAbnormality	Knee
RHEX2	results	ULTRA3
ULTRA3	hasAbnormality	"Rotation 15degrees"
ULTRA3	hasAbnormality	Right_Wrist
VISIT2	hasReport	DIAG2
DIAG2	hasDiagnosis	Systemic_Arthritis
VISIT2	hasReport	TREAT2
TREAT2	hasDrugTherapy	DT2
DT2	hasDrug	Methotrexate
TREAT2	hasDrugTherapy	DT3
DT3	hasDrug	Corticosteroids
...	...	...

Table 3.1: Semantic annotations (Abox).

#	axioms	#	axioms
1	Uv $\sqsubseteq$ D&D	17	etanercept $\sqsubseteq$ Bio-M
2	JI $\sqsubseteq$ D&D	18	anakirna $\sqsubseteq$ Bio-M
3	Rh $\sqsubseteq$ D&D	19	infliximab $\sqsubseteq$ Bio-M
4	AD $\sqsubseteq$ D&D	20	DMARDs $\sqsubseteq$ methotrexate
5	P-JCA $\sqsubseteq$ JI	21	NSAIDs $\sqsubseteq$ aspirin
6	CRA $\sqsubseteq$ JI	22	NSAIDs $\sqsubseteq$ ibuprofen
7	CRA $\sqsubseteq$ Rh	23	corticosteroids $\sqsubseteq$ Drugs
8	CRA $\sqsubseteq$ AD	24	DMARDs $\sqsubseteq$ Drugs
9	JRA $\sqsubseteq$ CRA	25	NSAIDs $\sqsubseteq$ Drugs
10	A-SD $\sqsubseteq$ CRA	26	Bio-M $\sqsubseteq$ Drugs
11	P-JRA $\sqsubseteq$ P-JCA	27	CRA $\sqsubseteq \exists$ may_treat.anakirna
12	P-JRA $\sqsubseteq$ JRA	28	CRA $\sqsubseteq \exists$ may_treat.etanercept
13	O-JRA $\sqsubseteq$ JRA	29	CRA $\sqsubseteq \exists$ may_treat.infliximab
14	JI $\sqsubseteq \exists$ may_treat.NSAIDs	30	CRA $\sqsubseteq \exists$ may_treat.methotrexate
15	JI $\sqsubseteq \exists$ may_treat.corticosteroids	31	Uv $\sqsubseteq \exists$ contraindicated_drug.fluorometholone
16	fluorometholone $\sqsubseteq$ corticosteroids		

Table 3.2: Ontology axioms for the use case.

Abbrev.	Name
D&D	Diseases and Disorders
Uv	Uveitis
JI	Joint Inflammation
Rh	Rheumatism
AD	Autoimmune Disease
P-JCA	Polyarticular Juvenile Chronic Arthritis
CRA	Chronic Rheumatic Arthritis
JRA	Juvenile Rheumatoid Arthritis
A-SD	Adult-onset Still's Disease
P-JRA	Polyarticular Juvenile Rheumatoid Arthritis
O-JRA	Olygoarticular Juvenile Rheumatoid Arthritis
Bio-M	Biological Medication
DMARDs	Disease Modifying Anti-Rheumatic Drugs
NSAIDs	Non-Steroidal Anti-Inflammatory Drugs

Table 3.3: Abbreviations of concepts from ontology in Table 3.2.

patients (i.e., measures).

The method described in Chapter 5 provides the means to effectively extract facts and dimensions according to the user's requirements so that SW data can be explored and analyzed using OLAP-style capabilities. Facts and dimensions are extracted from the data with the help of the indexing and modularization approaches developed in Chapter 4. Once the facts and the dimensions are extracted, the user will be able to feed this information to an OLAP server and create cubes and queries over the cubes to analyze patient data and discover useful patterns and trends. However, the extracted facts and dimensions may not be summarizable, meaning that the use of pre-aggregation techniques can lead to wrong results. We warn the user about the summarizability property and, in the case where facts are not summarizable, we still provide a correct answer to the initial MD query.

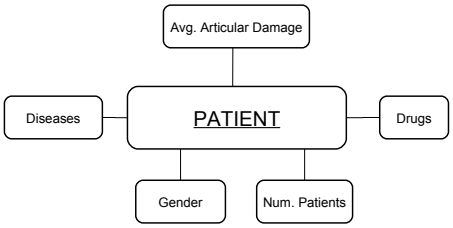

Figure 3.5: Use case analysis requirements.

Supposing that the extracted facts and dimensions are summarizable and are fed to an OLAP server that supports MDX, we show an example of the creation of a MD cube using an MDX-like syntax[2].

```
CREATE CUBE [cubeArticularDamage1]
FROM [patient_dw]
(
   MEASURE [patient_dw].[avgArtDamIndex],
   MEASURE [patient_dw].[numPatients],
   DIMENSION [patient_dw].[Drug],
   DIMENSION [patient_dw].[Disease]
   DIMENSION [patient_dw].[Gender]
)
```

By exploring this cube the analyst can find out interesting patterns such as which drugs mitigate the articular damage depending on the disease type.

[2]The specification in MDX of the dimension hierarchies and the measures are out of the scope of this example.

For example, we show a potential query that the analyst may pose over the previous cube.

```
SELECT
   [Drug].[Hierarchy1].[''NSAIDS''].Members On Axis(0),
   [Disease].[Hierarchy2].[''Auto-immune''].Members On Axis(1),
FROM [patient_dw]
```

Through this query, the analyst is rolling-up the drugs to the NSAIDS level and also rolling-up the diseases to auto-immune ones. Therefore, she wants to obtain measures (i.e., the average articular damage index and the number of patients) for patients that have been prescribed with NSAIDS drugs and have auto-immune diseases.

Next, we give an overview of the process that has to be followed in order to obtain facts and dimensions that are able to solve queries such as the previous one. We differentiate between the fact and the dimension extraction. Both processes are detailed in Chapter 5. Previously to obtain facts, we must extract the raw data that will be aggregated to compose facts. These data are arranged into tuples (i.e., data tuples) and are derived according to the user MD query. The user query (i.e., the subject of analysis, dimensions and measures) are translated into a series of conceptual descriptions over the TBox. Then, semantic connections must be looked for between the MD elements. In particular, we look for aggregation paths between the subject of analysis and each dimension and measure. This ensures that the extracted data are logically connected through semantic paths. As a result, we obtain a reachability graph, which is a graph rooted in the subject of analysis and which contains aggregation paths to each of the dimensions and measures. This process is efficiently performed by using the TBox indexes described in Chapter 4. The reachability graph is used as a guide to extract from the ABox instance tuples and the posterior data tuples. This process is performed by using an efficient query answering mechanism also proposed in Chapter 4. Data tuples are the raw data used to compose the aggregated information (i.e., facts). A fact is defined as a MD point quantified by a set of aggregated measures. Therefore, all data tuples that share the same dimension values are collapsed into a fact, and the respective measure values are aggregated using the aggregation function defined by the user for each measure. As previously mentioned, we warn the user about the summarizability of the resulting facts. In case these are not summarizable, we still provide correctly aggregated facts to the user, with the only condition that these aggregations cannot be further used to aggregate data.

Dimension extraction concerns the extraction of dimension hierarchies for the different dimensions defined by the user. A dimension is defined as an directed acyclic graph of nodes, where nodes are sub-concepts of the dimension type specified by the user, and the edges correspond to the semantic relations (e.g., "is-a" relationships) between the nodes. Dimension hierarchies are

extracted by using the modularization techniques proposed in Chapter 4. Traditional dimension hierarchies usually satisfy a series of restrictions to ensure summarizability. However, as we believe that the summarizability constraints are too restrictive and hinder the aggregation possibilities provided by a truly semantic dimension, we do not transform the dimension hierarchies into summarizable ones but only re-shape the dimensions to favor both dense regions and good aggregation nodes, while preserving as much as possible the semantics.

As a result, we are able to extract a facts and several dimension hierarchies from semantically enriched data that can be fed to a traditional OLAP tool to respond to interesting MD queries (taking into account the summarizability property).

Chapter 4

Indexing and modularization approaches for ontologies

This chapter describes the indexing and modularization approaches developed for large ontologies. In Section 4.1, we motivate the need for modules and identify the main analytical requirements that have driven the development of the modularization approaches. Section 4.2 describes the OIM proposed to index and query ontologies, and Section 4.3 contains the four OMETs devised to extract modules that meet the analytical requirements identified. In Section 4.4 we summarize the preservation of semantics of the developed methods. Finally, Section 4.5 presents the experimental evaluation and Section 4.6 the discussion.

4.1 Introduction

Ontologies play a key role in the SW by providing a common domain vocabulary and standard semantics that facilitate interoperability and automatic machine processing. The increasing use of ontologies in different domains is demanding new tools that ease the access, management, use and reuse of ontologies so that real benefits can be achieved. Ontology modularization has attracted a lot attention among the research community as a means to efficiently manage and reuse ontologies. There is a wide range of ontology modularization approaches, which are usually driven by the application requirements of the modularization process. Specifically, two differentiated trends are observed: logical and traversal approaches. While logical approaches are concerned with the preservation of logical properties, traversal approaches consider more important structural criteria such as the size or the compactness of the modules.

The focus of this chapter is on developing indexing and modularization techniques that overcome both the logical and traversal approaches and extract modules useful for analysis. In the following, we identify a series of criteria that have driven the development of such techniques:

- *User requirements*: The proposed modularization techniques should be driven by the user requirements while being as automatic as possible. Any analytical task comes up from the user's needs, which are usually expressed in terms of analysis requirements. Therefore, the modularization techniques should take into account such requirements when extracting the module. The usual way to express the user's requirements is by creating an input signature, composed by entities from the ontology. This signature is the starting point from where the module should be extracted.

- *Performance*: The proposed modularization techniques should be scalable. It is well known that logical formalisms that underly ontologies, such as DL, suffer from scalability issues. However, the SW is the main application scenario of ontologies, where scalability becomes crucial. Therefore, modularization approaches are of no use if they cannot deal with at least medium-size ontologies. We perceive scalability as crucial requirement even if some logical aspects have to be sacrificed.

- *Semantics preservation*: The extracted modules should preserve semantics as long as it does not affect the other regarded criteria. However, we do not always require complete semantics preservation as there are already existing approaches aimed at that, which usually lead to high computational cost.

- *Size*: The size of the resulting module should be small enough with respect to the original ontology in order to justify its construction. The size of the module is crucial in many applications such as visualization and summarization tasks.

- *Preservation of structure*: The extracted modules should provide a context for the signature entities. In analytical applications, the structure of the data is fundamental to perform aggregations.

The developed indexing and modularization framework has been designed by taking into account the previous criteria. In particular, we present the OIM that allows to efficiently index large ontologies. The index is based on an interval labeling schema that is applied to the ontology inferred hierarchy. This index encodes information about the descendants and ancestors of each concept in a compact format. An interval algebra has been designed to efficiently perform operations involving ancestor/descendant relations, to perform conjunctive queries, and to perform a restricted subset of DL queries. Based on

the OIM, we provide four different OMETs to cover the different requirements of the user. These make use of the OIM to efficiently extract compact ontology modules. The techniques have proved to be scalable and the modules adhere to previously stated criteria. The main foundations of this chapter have been published in [89, 91].

4.2 Ontology Indexing Model

One of the main requirements of the developed modularization techniques and, in general, of the methods developed in this thesis, is that they offer a good performance in on-line scenarios. Allowing inferences of SW data during the extraction process would incur in a prohibitively cost and would hinder scalability. Therefore, we resort to indexing mechanisms over ontologies (i.e., both the TBox and the ABox) where all the inferences are made *a priori* to extraction. That is, we index the inferred TBox and also, the ABox assertions (not the materialized inferred ABox) based on the TBox indexes. The remaining of this section presents the TBox and ABox indexing mechanisms.

4.2.1 TBox indexing

In this section we present our approach for indexing large TBoxes. First, we discuss how the inferred model of a TBox can be obtained and its underlying graph structure. Then, we introduce the interval labeling schema designed to index and query the inferred graph. Over this labeling schema, we define an interval algebra, which consists of a set of interval operations that allow to efficiently perform queries about ancestors/descendants and some basic DL queries over the index. This indexing mechanism is the basis for the OMETs that will be described later.

4.2.1.1 Ontology inferred model

We assume that the ontology has been normalized and all anonymous concepts and roles have been named. As disjoint axioms of type $disjoint(A, B)$ imply $A \sqcap B \sqsubseteq \bot$, we transform them into $X \equiv A \sqcap B$ and $X \sqsubseteq \bot$, being $\bot$ a named concept. After that, the TBox contains two kinds of axioms: class definitions $A \equiv C$, where A is a concept name and C is a concept description or concept name, and subsumption axioms $A \sqsubseteq B$, where both A and B are concept names. The same applies to roles. The ABox contains property instantiations of the form $R(x, y)$, and facts of the form $A(x)$, with R and A being a property and concept name. Notice that these structural changes do not affect the semantics and the normalized ontology is semantically equivalent. From now on, we refer with $N_{c'}$ to the set of all named concepts in the ontology, which is composed by the initial atomic concepts, N_a, the complex named concepts, N_c, and the

set of new anonymous named concepts, N_{anon}. Similarly, we refer with $N_{r'}$ to the set of all named roles.

As we assume an ontology encoded in a language with DL semantics, it is possible to perform automated reasoning over the ontology using a DL reasoner. A DL reasoner performs various inferencing services, such as computing the inferred ancestors of a concept, determining whether or not a concept is consistent, deciding whether or not one concept is subsumed by another, etc. We are interested in obtaining the inferred concept hierarchy. This reasoning process can be considered as a pre-processing step performed only once per ontology. Also, the reasoner performs realization of the ABox, that is, it computes the most specific types for all individuals. The inferred concept hierarchy can be computed using alternative methods to standard DL reasoners, which are usually based on tableaux algorithms. In particular, [9] use a set of completion rules that applies to a normalized TBox to compute all the ancestors of each concept under the restricted description language $\mathcal{EL}^{++}$.

In any case, the inferred concept hierarchy (containing anonymous named concepts correctly classified) is modeled as a directed acyclic graph (DAG). We consider a directed graph $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. Without loss of generality, we assume that G has no self loops (v, v). A directed acyclic graph is a directed graph containing no cycles. Each node of the DAG represents a set of concept names (because multiple concept names may be logically equivalent), and edges correspond with subsumption relationships. Each node also keeps track of the definitions of its concept names in a structure called *annotations*.

Figure 4.1 shows the underlying graph structure with the annotations of the inferred concept hierarchy only for the diseases part of the ontology shown in Table 3.2[1]. We take this graph as reference to develop the subsequent examples.

4.2.1.2 Interval labeling schema

From a logical viewpoint the subsumption relationship is both transitive and reflexive. That is, $\mathcal{O} \models A \sqsubseteq B, B \sqsubseteq C \implies \mathcal{O} \models A \sqsubseteq C$ and $\mathcal{O} \models A \sqsubseteq A$, respectively. In the graph representation, a classified concept A subsumes a classified concept B if either: i) both A and B are in the label of some node $v \in V$ or ii) A is in the label of some node $v \in V$, there is an edge $(v, w) \in E$ in the graph, and the concepts(s) in the label of node w subsume B.

Considering the previous statements, we apply a node labeling schema over the graph representation of the inferred concept hierarchy to capture the subsumption relationships between concepts in the ontology. This labeling schema is based on Agrawal's interval schema [6], which allows the efficient management of transitive and reflexive relationships by materializing the transitive closure in a compact and compressed way. This approach can be applied to directed trees and DAGs, which is the underlying structure of the inferred class

[1]For the sake of simplicity, we obviate the inferred concept hierarchy for the drugs

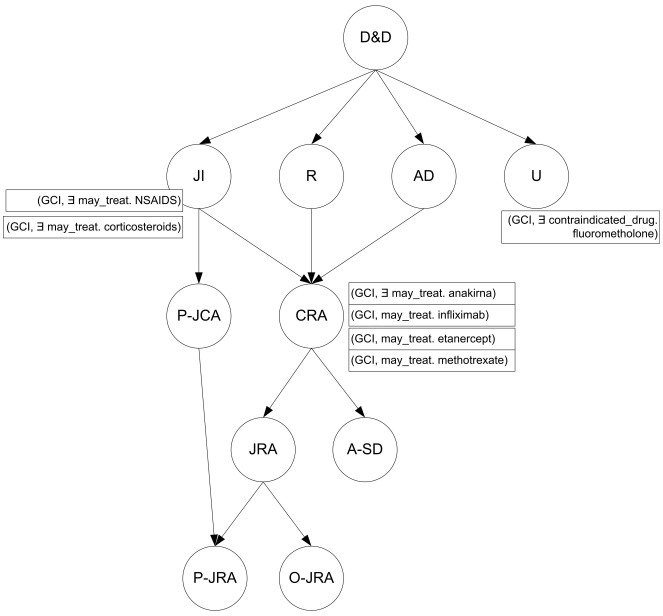

Figure 4.1: Excerpt of the inferred concept hierarchy generated from the ontology in Table 3.2.

hierarchy of ontologies [32]. The *transitive closure* of $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges is a graph $G^+ = (V, E^+)$ such that for all $v, w \in V$ there is an edge $(v, w) \in E^+$ if and only if there is a path from v to w in G.

In our labeling schema, the interval that is associated with a node v is $[pre(v), maxpre(v)]$, where $pre(v)$ is the preorder number of v and $maxpre(v)$ is the highest preorder number of v's descendants. The preorder number is taken as the node's unique identifier. This labeling variation has been taken from Schubert [131] and it is identical to the schema proposed by Agrawal. Figure 4.2 shows the compressed transitive closure mapped into intervals of the subsumption relationships.

For indexing DAGs, disjoint components can be hooked together by creating a virtual root node. The compression schema first finds a spanning tree T for the given graph (solid edges). Then it assigns an interval to each node based on the preorder traversal of T. Next, all nodes of the graph are examined in the reverse topological order so that for every edge from node v to w, all the intervals associated with node w are added to the intervals associated with node v, taking into account that if one interval is subsumed by another, the subsumed interval is not added. In the figure, interval $[6, 7]$ is associated to node JRA when labeling the spanning tree. Then, during the reverse topological traversal,

node JRA inherits interval $[4, 4]$ from node $P\text{-}JRA$.

For trees, this schema requires $O(n)$ storage, only twice the storage for the tree itself since one interval is enough to encode the descendants of a node. When dealing with DAGs, a node v will have associated a set of intervals, and in the worst case, the storage required will be $O(n^2)$. However, this situation is unlikely because Agrawal's approach for DAGs finds the optimum spanning tree, that is, the spanning tree that leads to minimum amount of intervals per node and thus, minimum storage requirements.

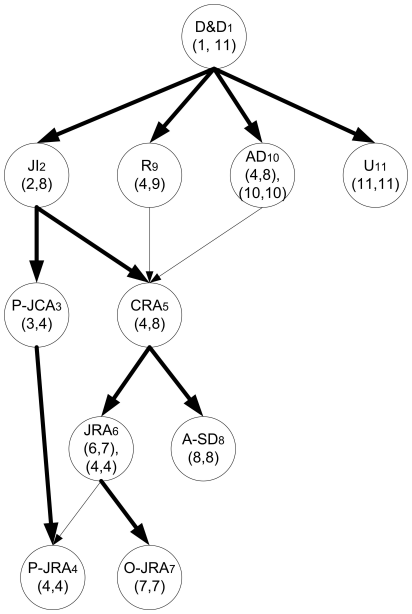

Figure 4.2: Interval encoding of subsumption relationships of the ontology. The subscript of the node's name denotes its identifier (preorder number).

The previous labeling schema encodes the subsumption relationships of the concepts of an ontology as intervals. By applying simple interval operations over the nodes, information about descendants and common descendants of ontology concepts can be immediately obtained. Information about the descendants of a given concept is very useful, specially for query answering, where the instances that belong to a given concept C are the instances that belong to C or any of its descendant concepts.

Equally interesting would be to dispose of such information about the ancestors of ontology concepts. By retrieving ancestors of a given set of concepts we are able to put the concepts into context and locate them in the ontology. This is specially interesting for modularity, where an upper module from a set of concepts is defined as the fragment that holds ancestor information about

the concepts. To that end, we have applied the same interval labeling schema over the reversed graph representation of the subsumption relationships. That is, the edges now point from a child to a parent node. Notice that a reversed tree becomes a DAG, and a DAG is still a DAG. In addition, a virtual root node is created to hook together what are leaf nodes in the original structure. Then, the same labeling schema described previously is applied to this reversed structure. As now edges denote ancestor relationships, the labeling schema will encode ancestor concepts. Figure 4.3 shows the corresponding reversed graph of Figure 4.2 with the ancestors information encoded as intervals. Notice that the node identifier is its preorder number and both the original and the reversed structure have their own preorder indexing system. The intervals in the reversed structure lose their compactness, thus resulting in extra storage requirements compared to the original structure. This occurs because the amount of intervals needed to encode all the relationships depends on the rate of multiple parents, and this rate tends to be larger when the structure is reversed.

We proceed exactly the same way with the DAG representing the entailed sub-property relation between all named roles. Information about sub-roles is also needed to respond to conjunctive queries.

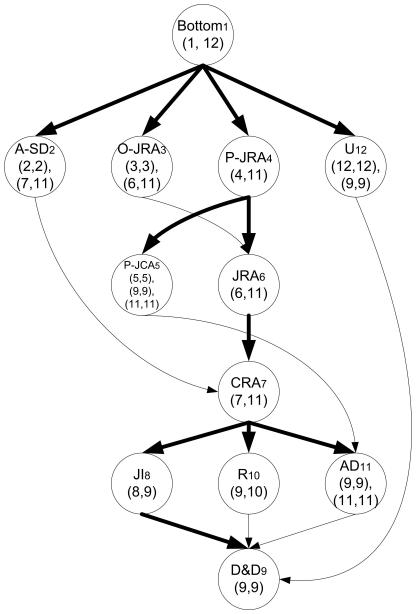

Figure 4.3: Interval encoding of ancestor nodes for the source ontology.

4.2.1.3 Node descriptors

We encapsulate all the information regarding ancestors and descendants of each concept in a descriptor function per node $v \in V$:

$$descriptor(v) = \langle descpre(v), descintervals(v), \\ ancpre(v), ancintervals(v), topo(v) \rangle$$

where $descpre(v)$ is the preorder number of node v in the original structure, $descintervals(v)$ is the set of intervals encoding v's descendants, $ancpre(v)$ is the preorder number of v in the reversed structure, $ancintervals(v)$ is the set of intervals encoding v's ancestors and $topo(v)$ is the topological order of v, which has been calculated using a standard algorithm.

The descriptors of the role nodes only have information about descendants:

$$descriptor(v) = \langle descpre(v), descintervals(v) \rangle$$

4.2.1.4 Interval algebra

The previous interval labeling schema associates each named concept $C \in N_{c'}$ in the ontology with a node in both subsumption graphs and a descriptor function $descriptor(C)$. This function maps each concept $C \in N_{c'}$ into the descendants labeling schema space, named $\mathcal{LS}^-$, and the ancestors labeling schema space, named $\mathcal{LS}^+$. Therefore, we have two separate indexes. We use the functions $\cdot^{\mathcal{LS}^-}$ ($\cdot^{\mathcal{LS}^+}$) to represent a concept in each space, $C^{\mathcal{LS}^-}$ and $C^{\mathcal{LS}^+}$. In both spaces we have the functions *id* and *int* that map a concept to an identifier (i.e., its preorder number), $id(C^{\mathcal{LS}^{-(+)}})$, and to a set of intervals, $int(C^{\mathcal{LS}^{-(+)}})$, respectively. As both graphs represent all the subsumption relations between any pair of concepts $C, D \in N_{c'}$ and the subsumption relation is transitive (i.e., captured by the interval labeling schema), any subsumption relation between a pair of concepts $C, D \in N_{c'}$ can be checked by using either the $\mathcal{LS}^-$ or the $\mathcal{LS}^+$ index. The same applies to the named roles, except that they only have the $\mathcal{LS}^-$ index. This is formalized as follows:

Definition 4.1. *Given a normalized TBox $\mathcal{T}$ with $N_{c'}$ being the named concepts, $\mathcal{LS}^-$ and $\mathcal{LS}^+$ are two indexes with a pair of associated functions for each index, $id : N_{c'} \to \mathbb{N}$ and $int : N_{c'} \to 2^{\mathbb{N}\times\mathbb{N}}$, such that, for each pair of concepts C, D in $N_{c'}$, we have that $\mathcal{T} \models C \sqsubseteq D$ iff $id(C^{\mathcal{LS}^-})$ is in $int(D^{\mathcal{LS}^-})$ and $id(D^{\mathcal{LS}^+})$ is in $int(C^{\mathcal{LS}^+})$.*

Definition 4.2. *Given a normalized TBox $\mathcal{T}$ with $N_{r'}$ being the named roles, $\mathcal{LS}^-$ is an index with a pair of associated functions, $id : N_{r'} \to \mathbb{N}$ and $int : N_{r'} \to 2^{\mathbb{N}\times\mathbb{N}}$, such that, for each pair of roles R, S in $N_{r'}$, we have that $\mathcal{T} \models R \sqsubseteq S$ iff $id(R^{\mathcal{LS}^-})$ is in $int(S^{\mathcal{LS}^-})$.*

As a result, we can check entailments between named concepts and roles by using directly the indexes.

The next step consists in designing an interval algebra to efficiently operate with the $\mathcal{LS}^-$ and $\mathcal{LS}^+$ representations of the concepts[2]

First, we introduce some definitions and functions related to the normalization of ontology axioms:

- An *atomic expression* is an expression of the form $\{(\neg)A, (\neg)\{\exists, \forall, \geq n, \leq n, = n\}R.((\neg)B)\}$, where A and R are atomic concepts and roles, and B is an atomic expression, respectively.
- The function $norm : \mathcal{T} \rightarrow \mathcal{T}$ takes as input a TBox an replaces the complex concepts with their definitions until concept descriptions contain only atomic expressions.
- The *disjunctive normal form* (DNF) of a concept description is a disjunction of conjunctions of simple concept descriptions.
- The *conjunctive normal form* (CNF) of a concept description is a conjunction of disjunctions of simple concept descriptions.

Now, we present some auxiliary functions to operate with the $\mathcal{LS}^-$ and $\mathcal{LS}^+$ representations of the concepts.

- The function $flatten : 2^{\mathbb{N}\times\mathbb{N}} \rightarrow 2^{\mathbb{N}}$ takes a set of natural intervals and flattens the structure into a set of natural numbers. For example, given the interval set $\{(2,3),(9,12)\}$ it returns the set $\{2,3,9,10,11,12\}$.
- The function $[\ \ ]^{\mathcal{LS}^{-(+)}} : 2^{\mathbb{N}} \rightarrow 2^{N_{c'}}$ takes as input a set of natural numbers (i.e., identifiers) and returns the set of concepts represented by each identifier either in $\mathcal{LS}^-$ or $\mathcal{LS}^+$ space. For example, from the $\mathcal{LS}^-$ graph in Figure 4.2, we observe that $[\{4,6,7\}]^{\mathcal{LS}^-}$ returns the concepts $\{P-JRA, JRA, O-JRA\}$, whereas $[\{4,6,7\}]^{\mathcal{LS}^+}$ from the $\mathcal{LS}^+$ graph in Figure 4.3 returns $\{P-JRA, JRA, CRA\}$.
- The function $expand : N_{c'} \rightarrow 2^{N_{c'}}$ takes as input a concept represented in either $\mathcal{LS}^-$ or $\mathcal{LS}^+$ and returns the set of concepts represented by its associated intervals, which in $\mathcal{LS}^-$ are descendants and in $\mathcal{LS}^+$ are ancestors. This function is equivalent to perform the operation $[flatten(int(C^{\mathcal{LS}^{-(+)}}))]^{\mathcal{LS}^{-(+)}}$. For example, from Figure 4.2 we see that $expand(CRA^{\mathcal{LS}^-})$ returns $\{CRA, JRA, A-SD, P-JRA, O-JRA\}$, whereas in Figure 4.3, $expand(CRA^{\mathcal{LS}^+})$ returns $\{CRA, JI, R, AD, D\&D\}$.

[2]From now on, we refer to only the $\mathcal{LS}^-$ and $\mathcal{LS}^+$ representations of the concepts for simplicity. However, the following functions can be also used over the $\mathcal{LS}^-$ index of the roles when possible.

- The function $neg : N_{c'} \rightarrow \neg N_{c'}$ takes as input a named concept and simply returns the logical negated concept. For example, $neg(A)$ returns $\neg A$.

We have designed an interval algebra over the elements of $\mathcal{LS}^-$ and $\mathcal{LS}^+$ that allows to perform interesting computations over the named concepts $N_{c'}$ of the ontology.

Definition 4.3. *Let $C^{\mathcal{LS}^{-(+)}}$ and $D^{\mathcal{LS}^{-(+)}}$ be the $\mathcal{LS}^{-(+)}$ representation of two named concepts $C, D \in N_{c'}$.*

1. *The operator $C^{\mathcal{LS}^{-(+)}} \otimes D^{\mathcal{LS}^{-(+)}}$ denotes the classic intersection of set theory and is equivalent to perform $[flatten(int(C^{\mathcal{LS}^{-(+)}}) \cap int(D^{\mathcal{LS}^{-(+)}}))]^{\mathcal{LS}^{-(+)}}$, that is, it admits as a result only concepts $E^{\mathcal{LS}^{-(+)}}$ whose $id(E^{\mathcal{LS}^{-(+)}})$ values are in the intersection of $int(C^{\mathcal{LS}^{-(+)}})$ and $int(D^{\mathcal{LS}^{-(+)}})$.*
2. *The operator $C^{\mathcal{LS}^{-(+)}} \oplus D^{\mathcal{LS}^{-(+)}}$ denotes the classic union of set theory and is equivalent to perform $[flatten(int(C^{\mathcal{LS}^{-(+)}}) \cup int(D^{\mathcal{LS}^{-(+)}}))]^{\mathcal{LS}^{-(+)}}$, that is, it admits as a result only concepts $E^{\mathcal{LS}^{-(+)}}$ whose $id(E^{\mathcal{LS}^{-(+)}})$ values are in the union of $int(C^{\mathcal{LS}^{-(+)}})$ and $int(D^{\mathcal{LS}^{-(+)}})$.*
3. *The operator $\ominus$ behaves differently in $\mathcal{LS}^-$ and $\mathcal{LS}^+$, therefore, we distinguish the two cases:*
   - 3.1. *$\ominus C^{\mathcal{LS}^-}$ returns the set of concepts $\{expand(neg(D)^{\mathcal{LS}^-})/D \in expand(C^{\mathcal{LS}^+})\}$.*
   - 3.2. *$\ominus C^{\mathcal{LS}^+}$ returns the set of concepts $\{expand(neg(D)^{\mathcal{LS}^+})/D \in expand(C^{\mathcal{LS}^-})\}$.*

The previous operators are very useful to efficiently perform set operations over named concepts. For example, the operator $\otimes$ allows to extract the common ancestors (in the $\mathcal{LS}^+$ space) and common descendants (in the $\mathcal{LS}^-$ space) of a pair of concepts $C, D \in N_{c'}$.

4.2.1.5 Querying the TBox

In this section, we approach the more difficult problem of answering DL queries using the previous indexes and algebra. We restrict the problem to queries about atomic concepts that use the intersection ($\sqcap$), union ($\sqcup$) and negation ($\neg$) DL operators. We refer to these queries with the name $Q_{\sqcup,\sqcap,\neg}$. We formalize the problem as follows:

Definition 4.4. *Given a TBox $\mathcal{T}$ that has been indexed and mapped to the $\mathcal{LS}^-$ and $\mathcal{LS}^+$ spaces, and a DL query $Q_{\sqcup,\sqcap,\neg}$ over atomic concepts of $\mathcal{T}$, we want to find named concepts C such that $\mathcal{T} \models C \sqsubseteq Q_{DL}$ and $\mathcal{T} \models Q_{DL} \sqsubseteq C$ using the indexes $\mathcal{LS}^-$ and $\mathcal{LS}^+$, respectively.*

The first intuition is to use the previously defined operators $\otimes$, $\oplus$ and $\ominus$ to solve the intersections, unions and negations of the DL query, respectively. Let us analyze the suitability of such operations. First, we refer to queries about descendants, which will use the $\mathcal{LS}^-$ space. According to the model theoretic semantics of DL, the descendants of $C \sqcap D$ are concepts X such that $X \sqsubseteq C \sqcap D$. From this formula, we infer $X \sqsubseteq C$ and $X \sqsubseteq D$, which means that concepts X are descendants of both C and D. As these two subsumptions are between named concepts, by Definition 4.1, they correspond to edges in the graphs. That is, all the descendants of $C \sqcap D$ are captured by the graph structure. Therefore, to obtain the concepts X, we just need to intersect the descendants of C and D in $\mathcal{LS}^-$, that is, $C^{\mathcal{LS}^-} \otimes D^{\mathcal{LS}^-}$.

The descendants of $C \sqcup D$ are concepts X such that $X \sqsubseteq C \sqcup D$. From this formula we cannot infer further subsumption relations among X, C and D, therefore, the $\mathcal{LS}^-$ graph does not contain edges expressing such information. However, as we know that concepts X are descendants of $C \sqcup D$, we obtain concepts X with the union of the descendant of C and D, that is, $C^{\mathcal{LS}^-} \oplus D^{\mathcal{LS}^-}$. The results of such operation are sound but not complete. We illustrate this issue in Figure 4.4. The left part shows an example of an ontology and all the inferred axioms. The $\mathcal{LS}^-$ graph is built by considering each named concept a node and each inferred axiom an edge. The middle part of the figure shows the indexed $\mathcal{LS}^-$ graph. The right part shows an example query where the $\mathcal{LS}^-$ index fails to give a complete answer. We observe that the union of the elements of the query gives a partially correct answer. However, we are missing named concepts that are entailed by the concepts in this partial answer. In the example, the partial answer is $\{B, A, D, C\}$ but, as $A \sqcup C$ entails E due to the original axiom $E \equiv A \sqcup C$, the concept E and its descendants should also be part of the answer. These kinds of entailments are not encoded in the graph but should be taken into account in order to produce complete answers.

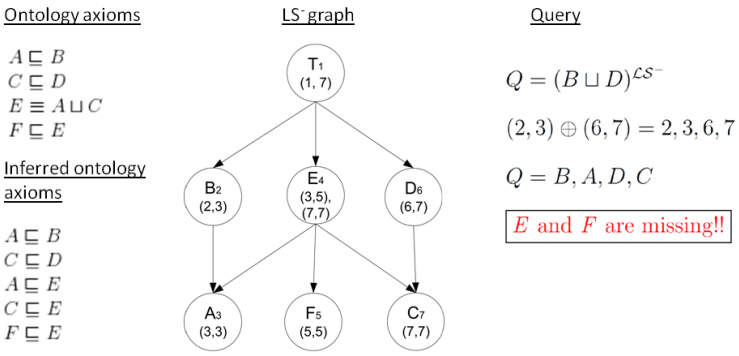

Figure 4.4: Example of query about descendants where the $\mathcal{LS}^-$ index gives an incomplete answer.

Now, we analyze queries about ancestors, for which we use the $\mathcal{LS}^+$ space. The ancestors of $C \sqcup D$ are concepts X such that $C \sqcup D \sqsubseteq X$. From this formula, we infer $C \sqsubseteq X$ and $D \sqsubseteq X$, which means that concepts X are ancestors of both C and D. As these two subsumptions are between named concepts, by Definition 4.1, they correspond to edges in the labeled graphs. That is, all the ancestors of $C \sqcup D$ are captured by the graph structure. Therefore, to obtain the concepts X, we just need to intersect the ancestors of C and D, that is, $C^{\mathcal{LS}^+} \otimes D^{\mathcal{LS}^+}$.

The ancestors of $C \sqcap D$ are concepts X such that $C \sqcap D \sqsubseteq X$. From this formula we cannot infer further subsumption relations between X, C and D, therefore, the $\mathcal{LS}^+$ graph does not contain edges expressing such information. However, as we know that concepts X are ancestors of $C \sqcap D$, we obtain concepts X with the union of the ancestors of C and D, that is, $C^{\mathcal{LS}^+} \oplus D^{\mathcal{LS}^+}$. Similarly as with the operation $C \sqcup D$ in $\mathcal{LS}^-$, by using only the index $\mathcal{LS}^+$ to retrieve ancestor of queries of type $C \sqcap D$, we obtain sound but not complete answers. This is illustrated in Figure 4.5.

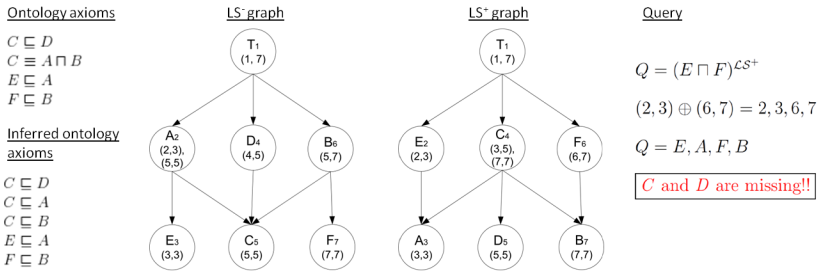

Figure 4.5: Example of query about ancestors where the $\mathcal{LS}^+$ index gives an incomplete answer.

As in the previous figure, the ontology axioms and the inferred axioms are on the left part and both the $\mathcal{LS}^-$ and $\mathcal{LS}^+$ graphs are shown in the middle. The right part contains an example query that produces incomplete results. The reason is similar as in the previous example. Here, the $\mathcal{LS}^+$ index gives $\{E, A, F, B\}$ as the result for the ancestors of $E \sqcap F$. However, as $A \sqcap B$ entails C, the concept C and its ancestors should be in the query answer.

Finally, we analyze the implications of dealing with negations. We only deal with negated atomic concepts. When querying for descendants, $\mathcal{LS}^-$, the only implications that $\neg A$ can generate are given by the existence of an axiom $A \sqsubseteq B$. As this axiom is equivalent to $\neg B \sqsubseteq \neg A$, it implies that $\neg B$ is a descendant of $\neg A$ that must be considered. Fortunately, the operator $\ominus$ is able to find such descendants of $\neg A$. The querying about ancestors is analogous. In $\mathcal{LS}^+$, the only implications that $\neg A$ can generate are given by the existence of an axiom $B \sqsubseteq A$. As this axiom is equivalent to $\neg A \sqsubseteq \neg B$, it implies that

$\neg B$ is an ancestor of $\neg A$ that must be considered. Fortunately, the operator $\ominus$ is able to find such descendants and ancestors of $\neg A$ in both $\mathcal{LS}^-$ and $\mathcal{LS}^+$ spaces.

To summarize, using the indexes to solve queries with unions in the $\mathcal{LS}^-$ and intersections in the $\mathcal{LS}^+$ can give incomplete results due to the limitations of the graph representation. Also, negated atoms can imply new descendants and ancestors that need to be considered. In order to solve the previous problem, we have designed an algorithm that gives complete answers about descendants and ancestors of queries of type $Q_{\sqcup,\sqcap,\neg}$. For both the retrieval of descendants and ancestors, the problem is analogous and the algorithm is split in three parts. First, a partial query answer, $Q^{\mathcal{LS}^{-(+)}}$, is given by using the operators of the interval algebra. Then, the partial answer is extended with concepts entailed by itself using structural notions about the ontology axioms. Finally, either the descendants or the ancestors of the result are also extracted, as they are also valid answers. First, we define a partial query answer to a query about descendants:

Definition 4.5. *Given an indexed TBox $\mathcal{T}$ with the $\mathcal{LS}^-$ index and a query $Q_{\sqcup,\sqcap,\neg}$ in disjunctive normal form, $Q = \alpha_1 \sqcup ... \sqcup \alpha_n$, where $\alpha_i = c_{i1} \sqcap ... \sqcap c_{im}$ such that $c_{ik} \in N_a$, $1 \leq i \leq n$, $1 \leq k \leq m$, a partial query answer to $\mathcal{T} \models C \sqsubseteq Q$ is given by the following expression:*

$$Q^{\mathcal{LS}^-} = \alpha_1^{\mathcal{LS}^-} \oplus ... \oplus \alpha_n^{\mathcal{LS}^-} \text{ where } \alpha_i^{\mathcal{LS}^-} = {c_{i1}}^{\mathcal{LS}^-} \otimes ... \otimes {c_{im}}^{\mathcal{LS}^-}$$

.

As the problematic operation in $\mathcal{LS}^-$ is the union, we transform the user query into disjunctive normal form so that the unions are pushed outside and are dealt with at the end. The elements α_i of the query are intersections of atomic concepts, which can appear negated. The negated atomic concepts are treated with the $\ominus$ operator and the intersections with the $\otimes$ operator. Then, the unions are solved with the $\oplus$ operator but, as we have shown, these results are incomplete. The negated atoms can give rise to other negated atoms or negated named concepts. If they do not have a representation in the $\mathcal{LS}^-$ graph, they are assigned an artificial representation where $id(C) = 0$ and $int(C) = \{(0, 0)\}$. The result of the partial query answer is a set of concepts that must be normalized to atomic expressions with the function *norm*.

Now, we define a function that, given a set of atomic expressions C_{set}, extends the input set with concepts from the ontology that are entailed by any combination of atomic expressions in C_{set}

Definition 4.6. *Given a normalized TBox in conjunctive normal form where concept definitions have the shape $C \equiv \beta_1 \sqcap ... \sqcap \beta_n$, where $\beta_j = c_{j1} \sqcup ... \sqcup c_{jm}$, such that c_{jm} is an atomic expression, we define the function RNC^-, which takes as input a set of atomic expressions C_{set} and extends it with named concepts C that have some β_j such that for all $c_{jk} \in \beta_j$, $c_{jk} \in C_{set}$.*

The function RNC^- is able to extract ontology concepts that are entailed by a combination of atomic expressions given as input. In this context, we apply the function RNC^- to the normalized partial query answer $Q^{\mathcal{LS}^-}$ in order to retrieve entailed concepts that cannot be extracted from the graph structure.

Finally, we define the answer to a query as follows:

Definition 4.7. *Given an indexed TBox $\mathcal{T}$ with the $\mathcal{LS}^-$ index and a query $Q_{\sqcup,\sqcap,\neg}$ in disjunctive normal form, the answer to the query is defined as:*

$$Ans(Q) = \{expand(C^{\mathcal{LS}^-})/C^{\mathcal{LS}^-} \in RNC^-(norm(Q^{\mathcal{LS}^-}))\}$$

The answer to a query $Q_{\sqcup,\sqcap,\neg}$ is defined as the result of applying the RNC^- function to the normalized partial query answer and extracting their descendants with the *expand* function, as they are also valid answers to the query. The *expand* function operates over concepts that have a representation in the $\mathcal{LS}^-$ ($\mathcal{LS}^+$), that is, over $N_{c'}$. If a concept returned by RNC^- does not belong to $N_{c'}$, the result of expanding that concept is empty. During the processing of the query we may come across a concept C together with its negated form $\neg C$. If that happens, it means the query is unsatisfiable and the result is empty.

Now, let us explain the implementation of the RNC^- function with more detail. Instead of using reasoning techniques to check if some combination of concepts in $Q^{\mathcal{LS}^-}$ entails another named concept, we base our algorithm on structural equivalence and implement it through an inverted index. The construction of the inverted index is as follows: first, the ontology is normalized so that concept descriptions are expanded to their atomic expressions with the function *norm*. Then, they are put into conjunctive normal form. Therefore, all concept descriptions have the shape $X_j \equiv \beta_1^j \sqcap ... \sqcap \beta_n^j$, where $\beta_i^j = c_{i1}^j \sqcup ... \sqcup c_{im}^j$, and c_{ik}^j is an atomic expression. Then, we build an inverted index where each entry has the shape $c \rightarrow [X_i.\beta_j^i.k]$, where c is an atomic expression, X_i is a complex named concept, β_j^i is a disjunction of atomic expressions of X_i and k is the number of atomic expressions in β_i. That is, each atomic expression c points to a list of concept definitions (and the corresponding disjunctions of atomic expressions) that contain it. Having this inverted index, we reduce the checking of new entailments of the partial query answer to a check over this index. Therefore, the RNC^- function retrieves concept definitions X such that any of their disjunctions of atomic expressions β_i is covered by the atomic expressions of the partial query answer. The construction of the inverted index does not add any extra computational complexity, as the process is linear with the size of the ontology.

The algorithm for answering queries about ancestors follows the same line of thought.

Definition 4.8. *Given an indexed TBox $\mathcal{T}$ with the $\mathcal{LS}^+$ index and a query $Q_{\sqcup,\sqcap,\neg}$ in conjunctive normal form, $Q = \alpha_1 \sqcap ... \sqcap \alpha_n$, where $\alpha_i = c_{i1} \sqcup ... \sqcup c_{im}$*

such that $c_{ik} \in N_a$, $1 \leq i \leq n$, $1 \leq k \leq m$, *a partial query answer to* $\mathcal{T} \models Q \sqsubseteq C$ *is given by the following expression:*

$$Q^{\mathcal{LS}^+} = \alpha_1^{\mathcal{LS}^+} \oplus ... \oplus \alpha_n^{\mathcal{LS}^+} \text{ where } \alpha_i^{\mathcal{LS}^+} = {c_{i1}}^{\mathcal{LS}^+} \otimes ... \otimes {c_{im}}^{\mathcal{LS}^+}$$

.

In $\mathcal{LS}^+$, the intersection is the operation that does not give complete results. Therefore, the user query is transformed into a conjunctive query to push outside the intersections, as opposed to the $\mathcal{LS}^-$. The negated atoms are solved with the $\ominus$ operator. The elements α_i are unions and are solved with the $\otimes$ operator. Then, the intersections are solved with the $\oplus$ operator. The normalization of the result gives rise to the partial query answer $Q^{\mathcal{LS}^+}$.

Then, the function RCN^+ is similarly defined.

Definition 4.9. *Given a normalized TBox in disjunctive normal form where concept definitions have the shape* $C \equiv \beta_1 \sqcup ... \sqcup \beta_n$, *where* $\beta_j = c_{j1} \sqcap ... \sqcap c_{jm}$ *and* c_{jm} *is an atomic expression, we define the function* RCN^+, *which takes as input a set of atomic expressions* C_{set} *and extends it with named concepts* C *that have some* β_j *such that for all* $c_{jk} \in \beta_j$, $c_{jk} \in C_{set}$.

The behavior of the RNC^+ function is the same as that of RNC^-. The only difference stems from the construction of the inverted index. In this case, the construction of the inverted index is done based on the disjunctive normal form of the ontology concepts, $X_j \equiv \beta_1^j \sqcup ... \sqcup \beta_n^j$, where $\beta_i^j = c_{i1}^j \sqcap ... \sqcap c_{im}^j$, and c_{ik}^j is an atomic expression. The shape of the index is identical, $c \rightarrow [X_i.\beta_j^i.k]$, where c is an atomic expression, X_i is a complex named concept but β_j^i is a conjunction of atomic expressions of X_i. Having this inverted index, the RNC^+ function retrieves concept definitions X such that any of their conjunctions of atomic expressions β_j is covered by $Q^{\mathcal{LS}^+}$

Finally, we define the answer to a query as follows:

Definition 4.10. *Given an indexed TBox* $\mathcal{T}$ *with the* $\mathcal{LS}^+$ *index and a query* $Q_{\sqcup,\sqcap,\neg}$ *in conjunctive normal form, the answer to the query is defined as:*

$$Ans(Q) = \{expand(C^{\mathcal{LS}^+})/C^{\mathcal{LS}^+} \in RNC^+(norm(Q^{\mathcal{LS}^+}))\}$$

The answer to a query $Q_{\sqcup,\sqcap,\neg}$ is defined as the result of applying the RNC^+ function to the normalized partial query answer and extracting their ancestors, as they are also valid answers to the query.

4.2.2 ABox querying

In this section we present our approach for efficiently querying large ABoxes based on the previous TBox indexes. The method presented is an alternative

to materializing all the inferences, which can result in a prohibitive storage cost. First, we discuss how we store ABox assertions and then, we explain how conjunctive queries are solved.

We assume the ABox has been reasoned over and contains concept assertions $A(x)$ with A being the most specific named concept. However, all the ABox inferences are not materialized as materialization may increase the storage requirements to an unmanageable size. We keep two separate relational tables, one for the concept assertions and one for the property assertions to account for the inferences.

The table T_C has two attributes (x, id) and contains tuples $(x, id(C^{\mathcal{LS}^-}))$ for each ABox concept assertion $C(x)$. That is, we keep for every individual the identifiers in $\mathcal{LS}^-$ of the concepts that it belongs to. The column x is indexed by the relational back-end.

The table T_P has three attributes (x, y, id) and contains tuples $(x, y, id(r^{\mathcal{LS}^-}))$ for each ABox property assertion $r(x, y)$. That is, we keep for every property assertion the two individuals (or the individual and the literal) involved and the identifiers in $\mathcal{LS}^-$ of the roles that relate them. The columns (x, y) are indexed by the relational back-end.

The previous tables allow us to answer conjunctive queries over named concepts and properties efficiently. In the following algorithms we use relational algebra terminology to solve the conjunctive queries. Algorithm 1 shows how to retrieve individuals x such that $\mathcal{O} \models C(x)$. First, all the intervals identifying the descendants of C are accessed with the $\mathcal{LS}^-$ index. Then, with simple range queries with the intervals over T_C, all individuals that belong to C or to any of its subconcepts are retrieved.

Algorithm 1 Instance retrieval

Procedure RETRIEVECONCEPT(C)
Input: C, named concept
Output: Res, set of individuals

```
1: Res = {}
2: S = int(C^{LS^-})
3: for (l, h) ∈ S do
4:     Res+ = Π_(x)(σ_{l≤id≤h}(T_C))
5: return Res
```

Algorithm 2 shows how to retrieve property assertions $r(x, y)$ such that $\mathcal{O} \models r(x, y)$. The right hand side of the algorithm shows the function $retrieveAux$, which given a role r, retrieves property assertions about r or any of its subroles in a similar way as Algorithm 1. That is, the intervals that identify the subproperties of r are accessed with the $\mathcal{LS}^-$ index and then, range queries are performed over the table T_P to retrieve the assertions. However, this function does not return complete results, as we have to also take into account ABox inferences produced by inverse roles and role chains, which are not materialized to keep the size of the ABox manageable. These inferences are efficiently

Algorithm 2 Property retrieval

Procedure RETRIEVEROLE(r)
Input: r, named role
Output: Res, tuples (x r y)
1: $Res = \{\}$
2: $Res+ =$ RETRIEVEAUX(r)
3: $s = int(r^{\mathcal{LS}^-})$
4: $subroles = [flatten(s)]^{\mathcal{LS}^-}$
5: **for** $s \in roles$ **do**
6: **if** $isInverse(s)$ **then**
7: $s_1 = removeInverse(s)$
8: $Res+ = \Pi_{(y,x)}($ RETRIEVEAUX$(s_1))$
9: **if** $isChain(s)$ **then**
10: $(s_1, s_2) = unchain(s)$
11: $tmp_1 =$ RETRIEVEAUX(s_1)
12: $tmp_2 =$ RETRIEVEAUX(s_2)
13: $Res+ = \Pi_{(x_1,y_2)}(tmp_1 \bowtie_{y_1=x_2} tmp_2)$
14: **return** Res

Procedure RETRIEVEAUX(r)
Input: r, named role
Output: Res, tuples (x r y)
1: $Res = \{\}$
2: $s = int(r^{\mathcal{LS}^-})$
3: **for** $(l, h) \in s$ **do**
4: $Res+ = \Pi_{(x,y)}(\sigma_{l \leq id \leq h}(T_P)$
5: **return** Res

responded at query time by making use of the indexes. The left part of the algorithm shows the algorithm that gives complete results for queries about property assertions. First, there is a call to *retrieveAux* in order to retrieve the assertions about r or any subrole. Then, each of the subroles s of r is inspected and the two problematic situations are handled. If s is an inverse role, the property assertions of s without the inverse are retrieved with the function *retrieveAux*. Notice that the resulting tuples are projected with the order of the x and y attributes switched. If s is a role chain, the roles that imply s are obtained and for each of them, a call to *retrieveAux* obtains the temporary property assertions, which are then joined and projected over the first and last attribute[3]. As a result, we obtain all the property assertions implied by the role r.

4.2.3 Implementation

The algorithms for the OIM are implemented in Python 2.6. The OWL ontologies are parsed with a custom OWL parser. In order to compute inferences

[3]For brevity, we only show the procedure with role chains of size 2, but the algorithm could be easily extended to role chains of size n

over the ontology we use Pellet reasoner. The returned subsumption hierarchy is represented as a graph using NetworkX 1.7 python package. The interval indexes are calculated and stored in MySQL[4] using a simple key-value schema, where the key is the concept and the value is the data returned by the *descriptor* function defined in 4.2.1.3. This way, data can be loaded in main memory into a hash table, which allows fast access to the information about a concept. If the data does not fit in main memory, the algorithms and the interval indexes could be distributed and efficiently accessed using key-value stores[75]. The Abox assertions are also stored in MySQL using a simple schema.

4.3 Ontology Module Extraction Techniques

The OMETs presented in this section offer a good trade-off between logical and structural aspects. They have been designed following the criteria shown in Section 4.1. For the purposes of this dissertation, a module is a subset of explicit or implicit axioms extracted from an ontology that provide a context for the input signature specified by the user. The four different modularization techniques are implemented as Python programs that rely on the OIM presented in the previous section and are the result of applying different operations over an extended signature. They all have a common pre-processing step where the user's input signature is extended. Then, one of the four ontology module extraction techniques is applied depending on the user requirements. The subsumption relationships of the concepts in the extracted module are restricted to a tree structure as it may be a requirement of a specific application. Optionally, the remaining subsumption relationships can be added to obtain a DAG.

We use the application scenario and use case described in Sections 3.2 and 3.2.1, respectively, to better illustrate each of the module extraction techniques. The use case proposed is focused on the MD analysis of patients with inflammatory diseases. As one of the dimensions is the disease, we build the example modules using a toy signature composed by inflammatory diseases. The modularization techniques can be used inside the analysis framework presented in this thesis to extract dimension hierarchies or as standalone tool. In fact, another interesting use case apart from extracting useful analysis hierarchies that fit the application scenario would be the discovery of relations as well as possible treatments and contraindications among the inflammatory diseases (i.e., *Polyarticular Juvenile Rheumatoid Arthritis* ($P-JRA$), *Adult-onset Still's Disease* ($A-SD$) and *Uveitis* (Uv)) by extracting a module. A module extracted from a domain ontology starting from these concepts and relations provides information about how these concepts are related to each other. In addition, the small size of the module, where only relevant concepts and relationships are added, enables the clinician to better visualize and understand the diseases' in-

[4] `MySQL: http://www.mysql.com`

teractions. The following sections explain in detail the steps followed to extract modules.

4.3.1 Extension of the input signature

The pre-processing step common to the four different module extraction techniques, is the extension of the user's input signature. The following definitions formalize this process.

Let N_C and N_R be countably infinite and disjoint sets of *concept names* and *role names*, respectively [68]. In general, C and D denote concepts, and r denotes a role name. Concepts in a description logic $\mathcal{L}$ are built up starting from the concept names in N_C, and applying the concept constructors (see concept constructors of DL in Table 2.1) A *signature* is a finite subset of $N_C \cup N_R$.

Definition 4.11 (Signature of an ontology)**.** *The signature $Sig(\mathcal{O})$ of an ontology $\mathcal{O}$ is the set of concept and role names which occur in $\mathcal{O}$.*

Definition 4.12 (Signature of a concept)**.** *The signature $Sig(C)$ of a concept C is the set of concept names which occur in the description of C. For example, if $C \equiv A \sqcap B$, $sig(C) = \{A, B\}$.*

Definition 4.13 (Input signature)**.** *The input signature Sig^{INPUT} for an ontology $\mathcal{O}$ is a tuple (SC_0, SP_0) where $SC_0 \subseteq sig(\mathcal{O})$ is the set of concepts specified by the user and $SP_0 \subseteq sig(\mathcal{O})$ is the set of roles the user is interested in w.r.t. SC_0. SC_0 is called the basic signature.*

Although the input signature contains both concepts and roles, the interpretation given to them is not the same as in other modularity approaches, such as [34] and [68]. The intuitive idea of input signature in this work is the following: the signature concepts define the core concepts of the module from which we want to obtain their definition and taxonomic relationships. On the other hand, the signature roles indicate the relevant roles the user is interested in w.r.t. the signature concepts. In other words, if a signature concept is related to some of the signature roles (either contains the role restriction in its definition or inherits it) the module should contain the concept along with the role restriction.

Following the use case, the objective is to extract an analysis hierarchy for the diseases *P-JRA*, *A-SD* and *Uv* enriched with the roles *may_treat* and *contraindicated_drug* when possible. Clinicians have to specify the intended content of the module in the form of an input signature as follows:

$$Sig^{INPUT} = (\{P\text{-}JRA, A\text{-}SD, Uv\},\ \{may\_treat,\ contraindicated\_drug\})$$
$$C_0\ (Basic\ signature) = \{P\text{-}JRA, A\text{-}SD, Uv\}$$

Modules are intended to preserve semantic information (i.e., subsumption and role restrictions) over the concepts of the input signature. To ensure this,

we extend the input signature with additional concepts and roles before applying the module extraction techniques.

Definition 4.14 (Extended signature)**.** *The extended signature* $Sig^{EXT} = (SC^*, SP^*)$ *for an ontology* $\mathcal{O}$ *is an extension of* Sig^{INPUT} *defined as follows:*

$$SC^* = SC_0 \cup \{C \mid \exists\, C' \in SC_0,\ r \in SP^* : \mathcal{O} \models C' \sqsubseteq C \ \wedge\ C \sqsubseteq \exists r.D \textit{ is axiom of } \mathcal{O}\}$$
$$SP^* = \{r \mid \exists\, r' \in SP_0 : \mathcal{O} \models r \sqsubseteq r'\}$$

In Sig^{EXT} we extend the concepts of the basic signature by including concepts connected to them through the subsumption relationship (ancestors) which have an asserted axiom in their definition containing some role in SP^*. Finally, SP^* contains the set of initial roles specified by the user in the input signature plus their subroles. Algorithm 3 shows how to calculate the extended signature by using the interval algebra operations of Section 4.2.1.4. Sig^{EXT} is the input to the OMETs described in Section 4.3.2.

We obtain $Sig^{EXT} = (SC^*, SP^*)$ for the use case as follows: SP^* is SP_0 because there is no property taxonomy. SC^* is the result of the union of SC_0 (basic signature) and ancestors of signature concepts which have an asserted restriction in their definition involving some property of SP^*, which are $\{JI, CRA\}$. Therefore, the extended signature is as follows: $Sig^{EXT} = (\{P\text{-}JRA,\ A\text{-}SD, Uv, \underline{CRA}, \underline{JI}\},\ \{may\_treat,\ contraindicated\_drug\})$

Algorithm 3 Compute extended signature

Procedure ExtendedSignature(Sig^{INPUT},Sig^{EXT})
Input: $Sig^{INPUT} \leftarrow (SC_0, SP_0)$, the input signature
Output: $Sig^{EXT} \leftarrow (SC^*, SP^*)$, the extended signature

```
1: SC* ← SC0
2: SP* ← SP0
3: for r ∈ SP0 do
4:     SP* ← SP* ∪ expand(r^{LS^-})
5: for C ∈ SC0 do
6:     C_ANC ← expand(C^{LS^+})
7:     for D ∈ C_ANC do
8:         for r ∈ SP* do
9:             if r ∈ D.annotations then
10:                SC* ← SC* ∪ {D}
11: return (SC*, SP*)
```

4.3.2 Ontology Module Extraction Techniques

In this section we describe four different OMETs that rely on the indexing mechanism and the interval algebra operations defined in previous sections to extract ontology modules. As the evaluation section shows, the use of one technique or another will depend on the specific application requirements. The four techniques differ in the number of concepts that are included in the module to provide a context for the extended signature Sig^{EXT}, which varies from none (in the *Signature* technique) to all the ancestors (in the *All Signature Ancestors* technique). Afterwards, the subsumption relations among the module concepts are established by using the interval algebra and the definitions of concepts in the module that include some property of the extended signature are also extracted. The subsumption relationships can be restricted to form a tree or a DAG structure depending on the user's needs. For each technique, we show a figure with the corresponding use case module with both tree and DAG structure (left and right side of the figure, respectively).

4.3.2.1 Signature (S).

The first approach is the most basic one and extracts the most compact modules. Indeed, its output consists of the concepts from Sig^{EXT} along with their subsumption relationships made explicit. In other words, this technique retrieves the inferred transitive closure of the subsumption relationship of concepts from Sig^{EXT}. This is calculated with Algorithm 4, which computes a spanning tree based on the subsumption relationships among concepts making use of Definition 4.1. The computational complexity of this algorithm is quadratic in the worst case.

Algorithm 4 Compute spanning tree based on subsumption relationships

Procedure SPANNINGTREE(L)
Input: L, list containing output nodes sorted by their preorder number
Output: G, containing a spanning tree of the output nodes

1: $G \leftarrow \emptyset$
2: Stack $parents \leftarrow \emptyset$
3: $C \leftarrow nextNode(L)$
4: $D \leftarrow nextNode(L)$
5: **while** L **do**
6: **if** $id(D^{\mathcal{LS}^-}) \in int(C^{\mathcal{LS}^-})$ **then** ▷ $\mathcal{T} \models D \sqsubseteq C$ according to Definition 4.1
7: $addEdge(G, edge(C, D))$
8: $push(parents, D)$
9: $C \leftarrow D$
10: $D \leftarrow nextNode(L)$
11: **else**
12: $pop(parents)$
13: $C \leftarrow top(parents)$
14: **return** G

Figure 4.6 shows the skeleton of the module extracted for the running use

case with this technique. In all the subsequent figures we use the following convention: darker nodes represent the input signature concepts, lighter nodes are the nodes added to form the extended signature (Sig^{EXT}) and blank nodes represent ancestors which, depending on the technique, are added to the module. In this technique, the only blank node added is a *root* node to join together the module nodes. None of the ancestors of Sig^{EXT} are considered. Moreover, solid edges represent the spanning tree computed with Algorithm 4 and dashed edges correspond to the remaining subsumption relationships to form a DAG.

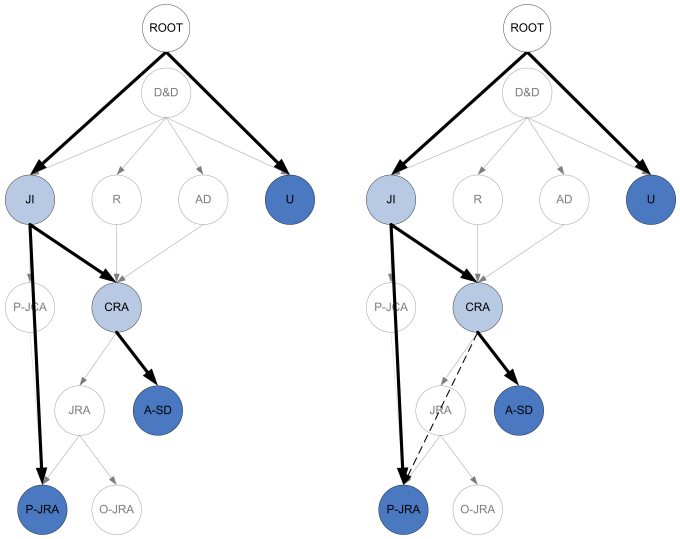

Figure 4.6: Output module for the S technique (tree and DAG structure, respectively).

4.3.2.2 Signature common ancestors (SCA).

This technique extracts all the concepts which are common ancestors of any pair of concepts from Sig^{EXT}. That is, for each pair of concepts C, D, we perform the operation $C^{\mathcal{LS}^+} \otimes D^{\mathcal{LS}^+}$. The number of common ancestors computations is quadratic in the worst case w.r.t. $|Sig^{EXT}|$. After the identification of the common ancestors, Algorithm 4 computes a spanning tree of the subsumption relationship with all the concepts from Sig^{EXT} plus common ancestors. Figure 4.7 shows the resulting module for the use case.

4.3.2.3 All signature ancestors (ASA).

In order to alleviate the computational cost of the previous technique, we propose to extract all ancestors from concepts of Sig^{EXT}, that is, for each concept

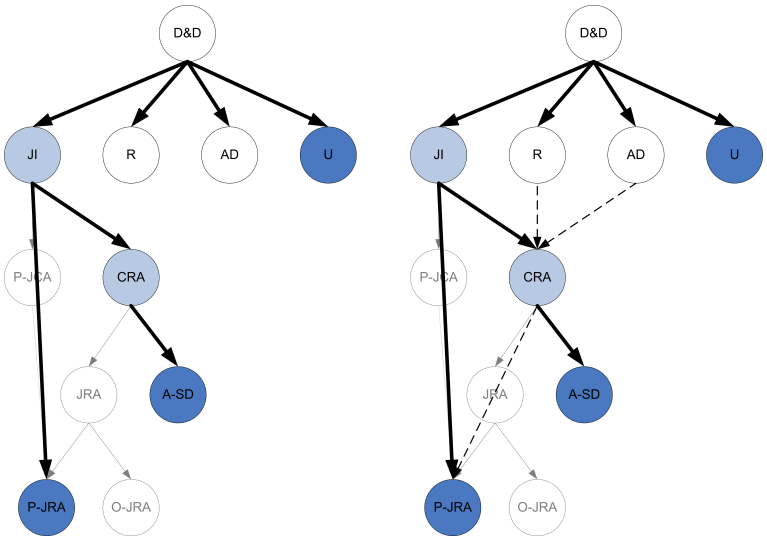

Figure 4.7: Output module for SCA technique (tree and DAG structure, respectively).

C, we get $expand(C^{\mathcal{LS}^+})$. Then, the extracted concepts are similarly organized according to their subsumption relationships as in the previous techniques using Algorithm 4. The output module contains all concepts from Sig^{EXT} plus all their ancestors organized by the subsumption relationships (see Figure 4.8). Notice that the resulting module can be much larger than the one of the previous technique, as we do not restrict the extracted ancestors to have at least two descendants in Sig^{EXT} but just one. Figure 4.8 shows the result of applying this technique to the use case.

4.3.2.4 All signature ancestors spanning tree (ASA-ST).

In the previous approach, extracting all ancestors from concepts of Sig^{EXT} can result in an excessive amount of irrelevant nodes in the output module. Thus, this last technique tries to improve this situation by just selecting ancestor nodes that relate concepts of Sig^{EXT} through the subsumption spanning tree calculated. This approach can be considered an extension of ASA; indeed Algorithm 5 is applied to the ASA output module as a cleaning process. Figure 4.9 shows the ASA-ST module in which nodes R, AD and JRA have been removed from the output of ASA. As removed nodes do not participate in the transitive closure of the signature, the transitive closure is not altered.

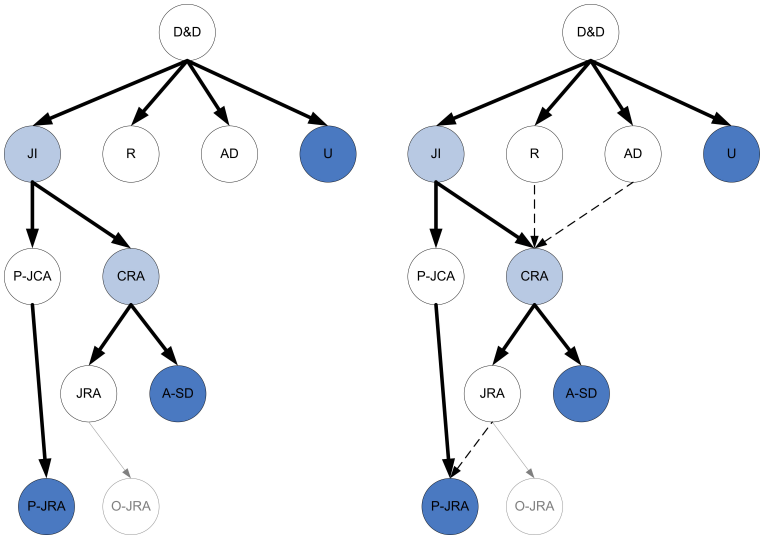

Figure 4.8: Output module for ASA technique (tree and DAG structure, respectively).

Algorithm 5 Remove nodes not related to the Sig^{INPUT} through the subsumption spanning tree

Procedure RemoveNon-SPTNodes(S, G)
Input: S, the input signature, G the spanning tree calculated with ASA technique
Output: G, with just nodes related to the signature through the subsumption spanning tree

```
1: Stack fringe ← ∅
2: for root r in G do
3:     if subtree(r) does not contain nodes from S then
4:         deleteNodes(G, subtree(r))
5: for root r in G do
6:     fringe ← subtree(r)
7:     while fringe do
8:         n ← pop(fringe)
9:         if subtree(n) does not contain nodes from S then
10:            deleteNodes(G, subtree(n))
11:        else
12:            push(fringe, subtree(n))
13: return G
```

4.3.3 Obtaining a DAG.

The output module from the four previous techniques consists of a tree structure for simplicity (see left graph of each of the previous figures). Notice that some applications (e.g., traditional DW dimensions) require a tree hierarchy

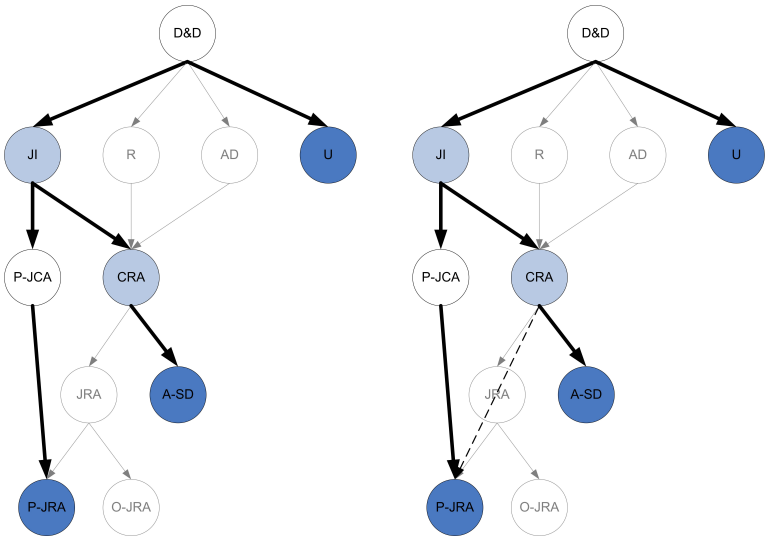

Figure 4.9: Output module for ASA-ST technique (tree and DAG structure, respectively).

rather than a DAG. However, in many cases it is necessary to get the complete DAG structure. For this purpose, Algorithm 6 has been designed.

Algorithm 6 Search additional edges (subsumpt. rels.) to get a DAG

Procedure GetDAG(G)
Input: G, the tree obtained by some of the previous techniques
Output: G, with DAG-structure

```
1: sortedNodes ← topologicalOrder(G)
2: for node C in sortedNodes do
3:     ancestors ← expand(C^{LS+})
4:     while ancestors do
5:         nearestAncestor ← getNearestAncestor(C, ancestors)
6:         if C or nearestAncestor in Signature then
7:             addEdge(G, edge(nearestAncestor, C)
8:         nodesToRoot ← getNodesToRoot(nearestAncestor)
9:         deleteFrom(ancestors, nodesToRoot)
10: return G
```

In this algorithm, for each node in the graph G sorted by increasing topological order, we identify its nearest ancestor, which is the ancestor whose topological order is lower and closer to the topological order of the node. Then, if some of the two nodes belong to the signature, we add this additional relationship to G as a new edge. In order not to include redundant edges to the node being examined, we delete from its ancestors list all the ancestors of the

nearest ancestor node identified, since these relationships are already implicit in the subsumption hierarchy. This algorithm does not add an extra temporal complexity because all operations have a linear cost w.r.t. the number of nodes if they make use of the *descriptor* functions of each node. Modules with DAG structure are shown in the right part of each of the Figures 4.6-4.9.

4.3.4 Implementation

The algorithms for the OMETs are implemented in Python 2.6. They are based on the OIM presented in the previous section, that is, ontologies are first indexed. Then, modules are extracted from user's signatures and presented as OWL files.

4.4 Semantics preservation

The OIM developed in Section 4.2 preserves all the entailments between named concepts and named roles of a given ontology by Definition 4.1. Therefore, both the indexes $\mathcal{LS}^-$ and $\mathcal{LS}^+$ reach soundness and completeness over named ontology concepts and roles.

The query module developed in Section 4.2.1.5 reaches soundness and completeness over queries of atomic concepts with the constructors $\sqcup$, $\sqcap$ and $\neg$, when applied to ontologies indexed with the OIM.

The modularization techniques presented in Section 4.3 are focused on providing a structure and context to the input signature that is suitable for analysis, thus, they are focused on the retrieval of ancestors. However, they also preserve the subsumption entailments between the signature concepts. Therefore, they reach soundness and completeness over the signature concepts.

A different issue that comes to mind is the utilization of the query module of Section 4.2.1.5 to perform queries over the modules extracted in Section 4.3. In order for the query module to return complete results to queries, it is necessary that the module contains all ancestors directly entailed by the reasoner and also the ancestors entailed by a combination of signature concepts. However, this requirement contrasts with the philosophy of keeping a structured and reduced module.

4.5 Evaluation

In this section we describe a series of experiments performed over three selected target ontologies. The experiments prove that both the OIM and the OMETs are scalable and able to extract modules according to the criteria in Section 4.1.

The experiments were performed on a Linux server with 8 1.86GHz Intel(R) Xenon(R) processors, 33GB of RAM, Ubuntu 10.04.4, Kernel Linux 2.6.32-45.

The selected ontologies are UMLS (version 2007AC), GALEN and NCI Thesaurus (version v07.10d). These ontologies have been selected due to their different sizes and levels of expressivity. Table 4.1 shows some statistics about the number of classes and properties for each one. The fourth column refers to the total number of property restrictions over classes found in the ontology and the last column is an average on the range cardinality of a property (i.e., average number of concepts that are related to each concept through each property). Well-structured and expressive ontologies have a range cardinality around 1. As it can be observed, UMLS has the largest size although it is the one with loosely defined semantics. On the other hand, GALEN is the smallest but the most expressive one.

Ontology	# classes	# properties	# restrictions	avg. range card.
UMLS	1.036.963	238*	306.622	9.8
GALEN	2748	413	25.707	1.09
NCI	63564	190	57.641	1.34

Table 4.1: Statistics about target ontologies. *Only inheritable properties are included.

4.5.1 Statistics about the OIM

The OIM assigns a descriptor to each node of the inferred subsumption relationships of an ontology, which have a DAG structure. The descriptor holds information about the descendants, ancestors and the topological order of the node, encoded in a compact format using intervals. The amount of space (i.e., intervals) needed to encode this information depends on the structure of the ontology (more specifically on the structure of the subsumption relationships). Table 4.2 shows some statistics about the size of the generated indexes (descriptors) for the selected ontologies. We observe the number of node descriptors is a little bit lower than the number of concepts in the ontology due to equivalent concepts, which are placed in the same node[5]. The maximum number of intervals in a descriptor can be quite large, although the average for both the descendants and ancestors space remains low. Therefore, the space requirements for the designed indexes are linear when dealing with large ontologies.

Onto.	Num. of desc.	Avg. desc.	Max. desc.	Avg. anc.	Max. anc.	Depth
UMLS	293,042	2.28	2,326	11.46	114	27
GALEN	2,945	1.02	12	3.74	6	12
NCI	60,565	1.30	320	4.97	20	16

Table 4.2: Statistics about the number of intervals per descriptor.

[5]This does not happen to GALEN because new descriptor nodes are created to hold anonymous classes.

The total performance time to index an ontology depends on two factors: the reasoning system used to infer the subsumption relationships and the indexing process itself. The reasoning time varies depending on the size and expressivity of the ontology, whereas the indexing process takes a few minutes for the three ontologies. In any case, the indexing of the ontologies can be performed as a batch process, as it is done only once. A different issue is how to update the index when the ontologies change, instead of re-building the indexes from scratch. However, this type of incremental indexing algorithms is still an open issue even for the most efficient state-of-the-art indexing approaches [151].

Specific experiments that make use of the $\mathcal{LS}^-$, $\mathcal{LS}^+$ are shown in the following chapter, as the indexes are used to speed up the fact and dimension extraction process.

4.5.2 UMLS modularizaton experiments

UMLS can be considered one of the largest reference ontologies in Biomedicine. It comprises domain knowledge from many different vocabularies, which makes it an ideal candidate to test the scalability of the OIM and the OMETs. To the best of our knowledge, there are no ontology modularization techniques that have performed their experiments using UMLS, therefore, no comparisons are shown.

Experimental settings The use of UMLS in the experimental evaluation is also motivated by the requirements of the application scenario (see Section 3.2), where visualization and analysis tools require manageable modules that provide a context for the different perspectives of the biomedical research, called vertical levels, namely: population, individual disease, clinical attributes, tissue, cellular and molecular levels. The input signatures for each vertical level should contain the most relevant concepts for each particular disease. For this experiment we take two diseases: *Juvenile Idiopathic Arthritis* (JIA) and *Tetralogy of Fallot* (TOF). Due to the large scale of the experiment, the input signatures to extract the modules are not manually specified by the user. Instead, we resort to automatic and semi-automatic approaches to collect the appropriate concepts for each target disease and vertical level. As a result, 100 different signatures have been defined.

Size performance The size of the modules is an important criteria because it directly affects maintainability and re-usability. Besides, it enables a better visualization and understandability in user-oriented applications. Some ontology modularization techniques, specially the ones concerned with preserving logical properties, such as the $\bot$-locality based approach in [60], extract all the axioms syntactically related to the signature, often resulting in too large modules to be useful for specific applications. The presented OMETs are designed to offer a good trade-off between logical and structural aspects.

The experiment showed in Figure 4.10 analyzes the size of the extracted modules of each of the modularization techniques proposed w.r.t. the size of their input signature. As expected, the basic technique S does not add any extra concept, which results in very compact modules. On the other hand, ASA generates the largest modules, as it includes all the ancestors and subsumption relationships involving concepts of the signature. By rejecting the subsumption relationships that do not relate concepts of the signature ($ASA-ST$), the module's size is reduced by half. This decrease is due to the fact that UMLS mixes different classifications over the same concepts, thus, concepts have multiple parents. However, for keeping signature concepts related, just one of these classifications will usually cover the signature and a large amount of concepts can be discarded. Finally, SCA also obtains quite reduced modules because it only includes in the module the subsumption relationships of common ancestors of concepts of the signature. The overall tendency of the modules' size is linear with respect to the input signature for the four modularization techniques.

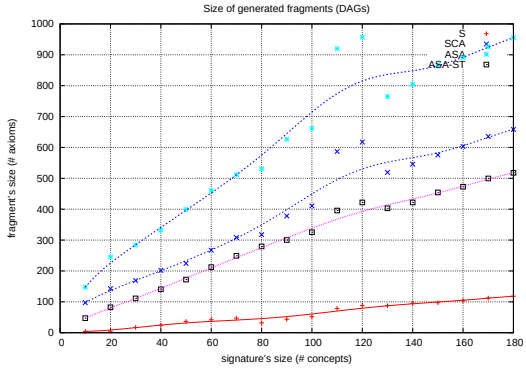

Figure 4.10: Signatures' size vs. modules' size for each modularization technique.

Time performance The required time to extract a module from an ontology is an important aspect specially in user-oriented applications, where the user needs to dynamically interact and extract subsets of interest from an ontology. Again, in these applications, logical properties of the resulting modules may need to be sacrificed in exchange for good time performance. Figure 4.11 shows the time required by each technique to generate modules for different signature sizes. SCA has the largest temporal complexity, which is quadratic in the worst case w.r.t. the number of signature concepts. This is due to the computation of common ancestors for each pair of concepts. On the other hand, S, ASA and $ASA-ST$ show a very efficient performance, being S the most efficient. Notice that for signatures smaller than a hundred concepts, SCA is even faster

than ASA and $ASA - ST$. This property is very remarkable and should be taken into account when generating modules from small input signatures.

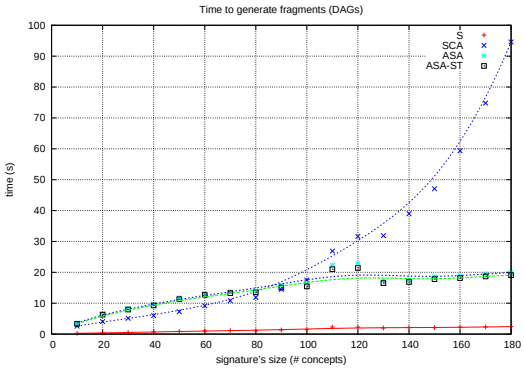

Figure 4.11: Signatures' size vs. time for each technique.

Structural criteria performance Some applications require modules to provide a simplified view of the original ontology structure relating the signature concepts. This is particularly relevant in applications where reducing the distance between the input concepts facilitates their joint visualization and helps in understanding their relationships in the original ontology. Therefore, we have designed a quality measure for modules based on the structural preservation. This measure calculates the distance of the signature concepts in the subsumption hierarchy of the module. It rewards the classification of the signature concepts and punishes unrelated concepts. We define the conceptual density (CD) of a signature as follows:

$$CD(Signature) = \sum_{x_i, x_j \in Signature, x_i \neq x_j}^{N} \frac{1}{d(x_i, x_j)} \frac{1}{N^2} \tag{4.1}$$

where $d(x_i, x_j)$ is the taxonomic distance between concepts x_i and x_j. The taxonomic distance has been calculated as follows:

$$d(x_i, x_j) = \begin{cases} (t_{x_i} - t_{nca(x_i,x_j)}) + (t_{x_j} - t_{nca(x_i,x_j)}) & \text{if } nca(x_i, x_j) \neq \{root\} \\ \infty & \text{otherwise} \end{cases} \tag{4.2}$$

where t_x is the topological order of node x and $nca(x_i, x_j)$ is one of the nearest common ancestors between concepts x_i and x_j. Thus the taxonomic distance is the length of the path between two concepts traversing through their nearest common ancestor. If their nca happens to be the $root$ node, it means both

concepts are unconnected and therefore, their taxonomic distance is infinite. A high CD means the signature concepts are strongly connected in the module while a lower CD means signature concepts are loosely connected (through long paths) or even unconnected in the module.

For this experiment, we have taken 25 signatures from the previous dataset and have compared their CDs in the original ontology (i.e., UMLS) and in the extracted modules. Figure 4.12 shows the results. The first column (Sig) is taken as reference, as it measures the structural density of the signatures in the original ontology. The CD of modules generated with ASA is expected to be the same as in the original ontology, since this technique replicates the subsumption relationships of signature concepts as in the ontology. $ASA - ST$ differs slightly from Sig, but on average, the CD of the signature in the modules is similar to that of Sig. The last column shows the distribution of the CDs with S. As expected, this technique is the one with most variance. Modules extracted with this technique contain just signature concepts, thus, the structure is only preserved if signature concepts have subsumption relationships among themselves in the ontology (highest CD). Otherwise, concepts will be unconnected in the module (lowest CD). Finally, SCA is the one that always improves the CD in the module w.r.t. to the CD in the original ontology. That is, signature concepts preserve their subsumption relationships as in the original ontology but are closer to each other (more compact modules).

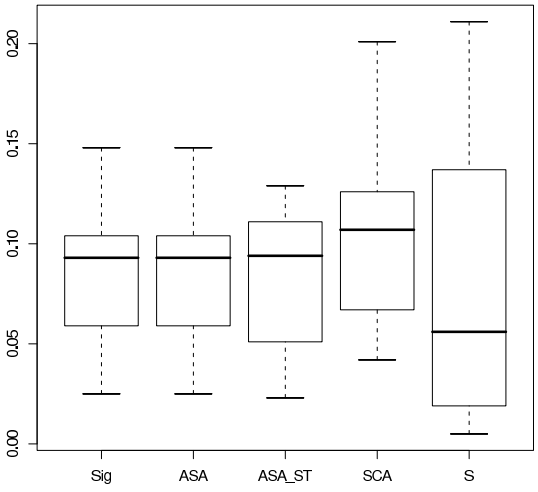

Figure 4.12: Comparison of CD of signatures for each technique.

4.5.3 GALEN and NCI modularization experiments

Both GALEN and NCI are biomedical ontologies smaller than UMLS but more expressive. These ontologies have been subjected to several modularization experiments. In particular, we undertake some of the experiments proposed in [60]. We show a comparison of the results of two of our modularization techniques, S and SCA, with the logical approach in [60] (Locality Lower of Upper Modules (LUM)), and the modules extracted with the traversal approach of Seidenberg-Rector Segmenter[6] [132].

The experiments are also carried out in the context of the Health-e-Child project's user scenario and they focus on the JIA[7] diseases. Here, signatures have been built from medical protocols only, by selecting 40 classes from GALEN and 48 from NCI. These sets have been also split into several subsets considering in each subset a different vertical level (e.g., genes, cells, drugs, etc.). Table 4.3 shows the number of classes of each signature selected for NCI and GALEN.

	Number of Classes of Signatures	
Id. Sig.	**GALEN**	**NCI**
#1	All classes: 40	All classes: 48
#2	Cells and Proteins: 7	Cells: 5
#3	Joints: 11	Genes and Proteins: 20
#4	Diseases and Signs: 13	Diseases and Signs: 7
#5	Procedures and Tests: 9	Drugs: 13

Table 4.3: Selected signatures for NCI and GALEN.

The comparison of the modularity techniques is carried out both w.r.t. the size and the CD of the output modules, in order to ensure that the extracted modules are small and provide a compact representation. Tables 4.4 and 4.5 compare the size of the modules obtained for each of the four techniques using the signatures selected for GALEN and NCI, respectively. In particular, we measure the number of class axioms of each module, which includes sub-class axioms, equivalent classes axioms and GCI axioms. Tables 4.6 and 4.7 show the CD of the signatures over the original ontology and over the modules for GALEN and NCI, respectively.

In general, the smallest modules extracted from GALEN are the ones by S, while SCA and the *Segmenter* lead to larger CDs. Comparing the three mentioned approaches, SCA is the one that presents the best trade-off between size and CD. For the case of fragments from NCI, LUM modules are the smallest ones, although they also have the smallest CDs, which is not a desirable property. This is due to the fact that LUM modules are so restrictive about the size of modules that, in many cases, the signature concepts are left unconnected.

[6] GALEN Segmenter: `http://www.co-ode.org/galen/`
[7] In the reference article JIA is named as JRA

Although modules preserve the signature entailments (e.g. signature concepts do not participate in any common entailment), these modules may not be useful for many applications. If module's size is a strong requirement, S is the next approach that extracts small modules while keeping the CDs above LUM modules. On the other hand, as shown by GALEN modules, SCA also extracts small and manageable modules, and it is the one that leads to higher CDs from the four compared techniques.

Signature	**|sig|**	**SCA**	**S**	**LUM**	**Segmenter [132]**
#1	40	208	**177**	372	416
#2	7	37	**24**	28	63
#3	11	67	**58**	279	299
#4	13	86	**68**	71	111
#5	9	52	42	**41**	103

Table 4.4: Comparison of the number of class axioms retrieved from GALEN.

Signature	**|sig|**	**SCA**	**S**	**LUM**	**Segmenter [132]**
#1	48	245	**174**	252	539
#2	5	32	22	**4**	50
#3	20	123	107	**0**	305
#4	7	34	17	**3**	67
#5	13	59	31	**19**	192

Table 4.5: Comparison of the number of class axioms retrieved from NCI.

Signature	**CD sig**	**SCA**	**S**	**LUM**	**Segmenter [132]**
#1	0.116	**0.118**	0.048	0.059	0.116
#2	0.182	**0.182**	0.136	0.136	**0.182**
#3	0.302	0.304	0.275	0.310	**0.322**
#4	0.144	**0.146**	0.109	0.091	0.144
#5	0.243	**0.243**	0.198	0.198	**0.243**

Table 4.6: Comparison of the CD of modules from GALEN.

Overall, SCA seems to work well on relatively large signatures, obtaining small modules that highly preserve the structure among the signature concepts. If the modules's size is a major requirement, S can generate very small modules, although the structural preservation in this technique depends greatly on the selected signature.

Signature	CD sig	SCA	S	LUM	Segmenter [132]
#1	0.037	**0.046**	0.018	0.015	0.037
#2	0.300	**0.427**	**0.427**	0.100	0.300
#3	0.095	**0.104**	0.050	0.048	0.095
#4	0.089	**0.128**	0.000	0.000	0.089
#5	0.115	**0.161**	0.030	0.022	0.115

Table 4.7: Comparison of the CD of modules from NCI.

4.6 Discussion

This chapter has presented a framework for efficiently indexing and modularizing large ontologies that meets a series of requirements imposed by analytical applications.

The OIM proposed is based on a compact interval labeling schema that is applied to the ontology inferred hierarchy. Therefore, queries about entailments between named concepts can be efficiently answered directly using the indexes. An interval algebra is also provided to perform interesting operations that involve computations about ancestors and descendants of concepts over the indexes. An interesting extension to the OIM is a query module to perform a constrained subset of DL queries directly over the indexes. This query module returns sound and complete answers for queries built from atomic concepts with the intersection, union and negation constructors. The ABox is indexed with the TBox indexes so that conjunctive queries can be efficiently solved.

The different OMETs have been developed to cover the need for modules that reach a good trade-off between the requirements imposed by analytical applications. These requirements can be summarized in the following sentence: the *user analysis requirements*, expressed as a signature, should drive the extraction of modules of *reduced size* that *preserve the semantics* of the signature elements to some extent while providing a suitable context and *structure* for such elements, all this remaining *scalable*. The lack of modularity approaches aimed at covering the previous requirements has prompted the development of the OMETS and the OIM that sustain them.

The usefulness of the developed framework is demonstrated by the different applications that make use of it:

- The framework presented in [90] is a direct application of the indexing and modularization approaches presented in this chapter to build compact and customized logic-based ontologies from large knowledge resources from the user requirements expressed as a free-text query.

- In [16], the modularization approaches are used to build tailored application ontologies that bring semantics to a set of data acquisition forms.

- The work in [13, 14] aims at building semantic conceptual spaces as a means to explore and link unstructured biomedical resources. In this work, the indexing and modularization approaches presented in this chapter are used to build the skeleton of the conceptual space, which is composed by a set of dimensions (i.e., different viewpoints) that have a hierarchical structure to allow summarization operations.
- In [70], the modularization approaches are used to enhance the process of semantic annotation of natural language text by previously extracting a small fragment from the knowledge resource that embraces all the candidate concepts for annotation related to the text beforehand.
- The work in [59] presents a scalable ontology matching system where the the interval labeling schema is applied to the ontologies and provides an interface to efficiently answer queries about taxonomic relationships.

Chapter 5

Multidimensional analysis of Semantic Web data

This chapter describes the developed method to analyze SW data from an MD perspective. In Section 5.1, we motivate the need for having tools that provide an analytic view of SW data and identify the challenges addressed. Section 5.2 is devoted to the specification of the MD query of the user. This query reflects the user's requirements in an MD form (i.e., based on dimensions and facts) and it is specified at the conceptual level by selecting concepts and properties from the available ontologies. In Section 5.3 we lay the foundations of the MD arrangement of SW data based on the semantics of the ontology axioms. The aggregation paths are the semantic paths expressed in the ontology schema (i.e., TBox) that go from the subject of analysis to each dimension and measure and allow instances to hook together and form data tuples from which facts will be derived. However, computing all the possible aggregation paths between the subject of analysis and each dimension and measure is prohibitive and usually unnecessary from a practical viewpoint. Therefore, we make a classification of interesting aggregation paths for the user and, in Section 5.4, we propose an efficient implementation for the computation of aggregation paths based on this classification. Section 5.5 deals with the extraction of facts at the instance level (i.e., from the ABox) based on the previously calculated aggregation paths. Section 5.6 describes the process of deriving dimension hierarchies from the user conceptual specification of the dimensions. The dimension hierarchies are composed by sub-concepts of the dimension type with truly semantic relations among themselves, that is, subsumption relationships. Here, we elaborate on the method to build well-shaped dimension hierarchies that favor aggregation. Finally, in Section 5.7, we present the experiments that validate and demonstrate the applicability of the developed method and Section 5.8 shows the discussion.

5.1 Introduction

The irruption and availability of SW data is demanding new methods and tools to efficiently analyze and provide richer insights into the current business processes. Although there exist some applications that make use of SW data, advanced analytic tools are still lacking, preventing the user from exploiting the attached semantics. The success of the well-known discipline of MD analysis over traditional and structured data sources has prompted us to investigate the application of such techniques to more open and semi-structured scenarios such as the SW. We address the problem of MD analysis over SW data and, more precisely, our study is focused on the challenges in designing and extracting facts and dimensions from SW data that are valid from a logical viewpoint. The presented method meets the requirements imposed by modern analytic applications: 1) it takes into consideration the user requirements, which are expressed by means of a conceptual MD query, 2) it automates the process of the fact and dimension extraction, relieving the user from this task and, 3) it deals with the possible ambiguity of the user MD query by exploring possible and logically valid MD configurations (by means of aggregation paths), ordered by interestingness.

However, the full exploitation of SW data by tools based on MD analysis (i.e., OLAP tools) is far from trivial due to the special features of SW data and the requirements imposed by these tools. The MD model views dimensions as the different analysis perspectives, usually composed by leveled and tree-shaped hierarchies, and facts as metrics functionally dependent on the dimensions. This is to ensure the summarizability property [73, 81], which refers to the possibility of accurately computing aggregate values with a coarser level of detail from values with a finer level of detail. The lack of summarizability can lead to incorrect results, and therefore erroneous analysis decisions. To ensure correctness, we have to pre-compute the total results for all the aggregations that we need fast answers to, while other aggregates must be computed from the base data. Traditionally, the MD model has relied upon relational sources with a direct mapping of the facts and dimensions to relational tables. The search of facts and dimensions has been restricted to the search of functional and inclusion dependencies between the mapped tables to ensure summarizability.

The semi-structured and graph topology of SW data clearly contrasts with the relational model upon which MD models rely. The basic structure in the SW is a triple statement of the form $(subject, predicate, object)$, where the *predicate* expresses a relation between the *subject*, which is a resource, and the *object*, which can be another resource or a literal. As opposed to the relational model where data are tuple-based and organized in tables, here data entities are scattered and have connections among themselves by means of relations, which give place to graph structures. The flexibility offered by this semi-structured model is specially suitable to model domains with complex relations. We believe that adhering to the traditional MD model to extract facts and dimensions in

scenarios where data have complex relations is too restrictive, as traditional MD models impose several integrity constraints that may not apply in such scenarios (e.g., the functional dependency between the facts and the dimensions, or the tree and level-shaped structure of the dimensions).

In attempt to capture and analyze SW data that is complex by nature, we propose new methods to extract facts and dimensions, which are not based on the traditional summarizability constraints. Roughly, a fact is identified by a MD point (a dimension value for each dimension) and quantified by measure values. A valid fact satisfies the following conditions: 1) the dimension values belong to the conceptual types of the dimensions specified by the user, 2) both the dimension values and the base measures are logically reachable from the subject of analysis by means of an aggregation path and 3) the dimension values of each fact share the same contexts, which are the closest hooking instances in the paths from the subject instances to each dimension value. Notice that the notion of aggregation path is broader than the functional dependency constraint imposed by traditional MD analysis, which can break the summarizability property of the extracted facts, as they can contain duplicated information that may give incorrect aggregation results. However, even though the extracted facts are not summarizable, MD analysis over these facts is still useful and can help drawing conclusions about data that cannot be otherwise analyzed. In this situation we re-calculate the measures by taking into account the fact duplicities so that the results given to the user are correct. We also warn the user about the non-summarizability of the extracted facts.

Dimensions have also been subjected to a series of restrictions to ensure summarizability. The dimension hierarchies must be *covering*, *onto* and all the paths *strict*. Moreover, they are static and usually pre-defined. Again, these constraints hinder the use of knowledge-rich structures to aggregate data. In a dynamic environment as the SW, we allow dimension hierarchies to be dynamically extracted from the knowledge sources based on the user MD query. Later, these hierarchies are re-shaped to perform meaningful aggregations while preserving as much as possible the semantics.

Apart from the previous structural mismatch between the traditional MD model and the topology of SW data, we need to address other challenges that arise when dealing with SW data. The most obvious is the management of implicit data. Reasoning mechanisms must be applied so that implicit data can be made explicit and enhance the analysis process. However, scalability is known to be an issue when reasoning with large amounts of SW data. We try to overcome this issue by applying the indexes developed in Chapter 4 to the ontologies, in order to handle basic entailments and thus, minimizing the use of the reasoner.

Other challenges that we address are related to usability aspects. In particular, when dealing with SW data, the user typically needs to specify a structured query in a formal language like SPARQL. However, the end user does not often know the query language and the underlying data graph structure.

Even if she did, languages such as SPARQL do not account for the complexity and structural heterogeneity often common in the data. We overcome this issue by providing the user with a simple mechanism to specify her analysis requirements at the conceptual level. We ask the user to compose a conceptual MD query by selecting concepts and properties of interest from the available ontologies. However, this flexibility offered to the user in the specification of the requirements can lead to an ambiguous specification (i.e., the concepts and properties selected might be used in several contexts in the ontologies). On the other hand, the user might have limited knowledge about the domain and her specification might be too general or abstract. Our method overcomes both types of imprecision by taking into account all possible interpretations of the concepts and letting the user select the intended meaning.

In summary, the developed method provides the means for efficiently analyzing and exploring large amounts of SW data by combining the inference power from the annotation semantics with the analysis capabilities provided by MD models (i.e., aggregations, navigation, and reporting). We formally present how SW data should be arranged in a well-defined conceptual MD schema, so that sophisticated queries can be expressed and evaluated. The work presented in this chapter is an extension of [99].

When talking about semantic annotations, we mean annotations that refer to an explicit conceptualization of entities in the respective domain. These relate the syntactic tokens to background knowledge represented in a model with formal semantics (i.e., an ontology) [17]. When we use the term *semantic*, we thus have in mind a formal logical model to represent knowledge. The work is not tied to a specific ontology language. For notation and terminology we use the expressive description logic $\mathcal{SROIQ}$ [52], which corresponds to the OWL 2 DL, the most expressive member of the OWL family where inferencing is still decidable. In fact, most of the popular DLs are sublanguages of $\mathcal{SROIQ}$.

The following sections explain in detail the foundations that underlie the on-line process of fact and dimension extraction proposed in this dissertation from ontological sources. The use case proposed for MD analysis of SW data is specified in Section 3.2.1 and is focused on the analysis of the efficacy of different drugs in patients diagnosed with inflammatory diseases. To facilitate the understanding of the proposed methods, Figure 5.1 shows a simplified graph-based representation of part of the ontology axioms from the previous use case. All the subsequent examples are based on this simplification.

5.2 Multidimensional query specification

In this section we present how the analyst information requirements are expressed in terms of a conceptual MD query. The analyst defines the MD elements of the query (i.e., the subject of analysis, the dimensions and measures) by selecting concepts and properties from the ontology. This can be easily

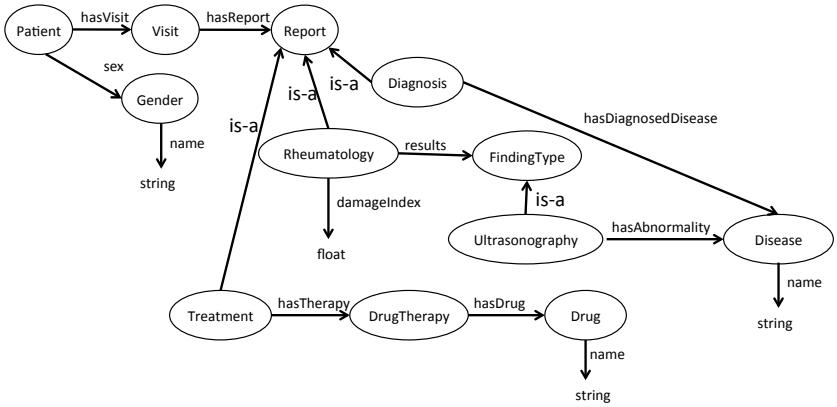

Figure 5.1: Graph-based representation of an excerpt of the use case ontology.

performed through a graphical front-end where the user selects the intended elements. Alternatively, the user can also express the MD elements in natural language text and, either through simple string matching mechanisms or more advance semantic annotation tools [17], these requirements can be mapped to ontological entities and compose a DL expression. We limit the expressivity of the DL expression to which MD elements are mapped to star-shaped conjunctive queries, where the central node is a named concept and role restrictions over this concept are applied. This restriction allows us to use the indexing mechanism defined in 4.2.2 to efficiently extract the instances that satisfy the DL expression from the ABox.

Let $\mathcal{O}$ be an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is a TBox expressing ontology axioms and $\mathcal{A}$ an ABox or instance store.

Definition 5.1. *A multidimensional query is a triple* $Q = (C_{SUB}, \{D_i\}_{i=1,...,n}, \{M_q\}_{q=1,...,t})$*, with* C_{SUB} *as the subject of analysis,* $\{D_i\}$ *the set of corresponding dimensions and* $\{M_q\}$ *the set of measures.*

We use the term *MD element* to refer to a dimension or a measure when we need to refer to one of them indistinctly. That is, $elem \in MD$ is an MD element where MD is an ordered tuple $MD = (D_1, ..., D_n, M_1, ..., M_t)$. Also, $MD(e)$ returns the MD element (i.e., dimension or measure) to which e is associated, being e an ontology entity, and MD_i refers to i^{th} MD element.

Next, we define each of the elements of a multidimensional query.

Definition 5.2. *The subject of analysis* C_{SUB} *is the concept that describes the entities around which the multidimensional analysis is performed.*

The subject of analysis is the focus of the analytical process and serves as starting point to derive multidimensional facts. For example, $C_{SUB} = Patient$

means the subject of analysis are instances of *Patient*. The concept expressing the subject of analysis can be restricted with roles, such that the facts will be extracted only from subject instances satisfying the restrictions. For example, the analyst may express in natural language text the subject of analysis as: "*patients of female sex with age under 16*". This text will be mapped to the DL expression: $C_{SUB} = Patient \sqcap \exists\ sex.Female \sqcap \leq hasAge.16$. Therefore, facts will describe only patients satisfying these restrictions.

Definition 5.3. *A dimension D is a tuple $(desc, proj)$, where $desc$ is the concept describing the semantics of the dimension and $proj$ is the datatype role over which the dimension values that compose the dimension $desc$ are projected*[1] *to obtain the corresponding dimension values.*

Given a dimension D_i, we use the functions $desc(D_i)$ and $proj(D_i)$ to access the conceptual description and the projection of the dimension, respectively. However, we use the term *dimension* in general to refer to its conceptual description.

Definition 5.4. *A dimension hierarchy H_D for a dimension D is a rooted directed acyclic graph of sub-concepts of D* [2]*.*

Intuitively, a dimension is something that characterizes the subject of analysis, therefore, there must be a relation between both. This relation will be investigated later on. The conceptual description of the dimension is expressed with an ontological concept, and it means that the values that compose the dimension hierarchy must be sub-concepts of the dimensional concept. The readable dimension values are obtained by projecting the ontological sub-concepts that compose the dimension hierarchy over the datatype role selected by the user.

Regarding the modeling of the dimension hierarchies, in traditional DW and OLAP scenarios it is known to be a laborious engineering process, where the DW designer carefully defines the different dimension categories with their associated dimension values and their partial order relations (i.e., hierarchies) to meet as much as possible the initial user requirements. Later on, OLAP users can express their queries based on the pre-defined dimensions. In our dynamic setting, we avoid any a priori calculation and let the system find appropriate dimension hierarchies starting from the extracted base dimension values that constitute the base facts. The relations among the dimension values that form the hierarchies will be automatically derived from the semantics encoded in

[1] We understand projection of a concept or instance c over a direct datatype role p, $\Pi_p(c)$ as in the relational algebra.

[2] We borrow the commonly used term "dimension hierarchy" from MD modeling to refer to a graph. However, a hierarchy is modeled mathematically as a rooted tree: the root of the tree forms the top level, and the children of a given vertex are at the same level, below their common parent.

the knowledge resources. This way, the analyst will be able to perform meaningful, semantic aggregations. As a result, the user is only concerned with the conceptual specification of the dimensions.

For example, through the dimension $D_1 = (Disease,\ name)$ the analyst requires instances of type *Disease* that related to the subject of analysis to be dimensional. The hierarchy for the *Disease* dimension is composed by sub-concepts of *Disease* and is extracted a posteriori in a bottom-up fashion starting from the *Disease* instances that compose the facts. The dimension values are obtained by projecting the sub-concepts composing the hierarchy over the role *name*. As with the subject of analysis, the specification of a dimension can be restricted with roles. For example, $D_2 = (Disease \sqcap \exists\, hasDiagnosedDisease^{-}.Diagnosis,\ name)$, where the analyst restricts the dimension to only *Disease* instances that are related to instances of *Diagnosis* through the inverse of the role *hasDiagnosedDisease*, in short, disease instances that appear in a *Diagnosis* report (i.e., the patient's diagnosis). As before, the role restrictions only act at the instance level when extracting the facts.

Finally, we can also consider a numerical value as dimension, such as the age of the patients. It would be expressed as $D_4 = (Patient,\ hasAge)$. Only *Patient* instances having the role *hasAge* are considered. These instances are projected over such role to obtain the dimension values. Discretization techniques can later be applied to these numerical dimensions in order to organize the values in a hierarchy. In any case, this is part of the modeling of the hierarchies and is dealt with in Section 5.6.

Definition 5.5. *A measure M is a triple $(desc, proj, aggFunction)$, where desc is the concept describing the semantics of the measure, proj is the datatype role over which desc is projected to obtain the corresponding measure values and aggFunction is the aggregation function to apply over the measure values.*

The specification and semantics of a measure are similar to the specification of a dimension except for the aggregation function. The user is responsible for assigning appropriate aggregation functions to the measures that she defines. Aggregation functions wrongly applied to certain data can lead to wrong results. We distinguish between three types of aggregate functions: $\Sigma = SUM, COUNT, AVG, MIN, MAX$, which are applicable to data that can be added together, $\Phi = COUNT, AVG, MIN, MAX$, which are applicable to data that can be used for average calculations, and $c = COUNT$, applicable to data that is constant. As the summarizability property only holds for distributive functions, the AVG can be calculated by keeping both SUM and $COUNT$.

For example, the measure $M_1 = (Thing,$
$damageIndex,\ AVG)$ specifies as potential instances satisfying the measure any instance that has an existential restriction over the role *damageIndex*. Then, these instances are projected over *damageIndex* to obtain the measure

values. The aggregate function AVG indicates that the measure values should be aggregated using the average function. A more specific measure is $M_2 = (Rheumatology,\ damageIndex,\ AVG)$, which aggregates any value related through the existential role $damageIndex$ to instances of $Rheumatology$ using the average function.

There is a special case when dealing with the aggregation function $COUNT$. In such case, we allow both the conceptual description of the measure and the projection to be empty. If both are left empty, it means that the analyst wants to keep track of the number of facts that contribute to the current aggregation. In such case, we attach to each generated fact a hidden field with the number one. When aggregating the facts, the measure is interpreted as a summation over this field. For example, the measure $M_3 = (-,\ -,\ COUNT)$ over the running use case will display the number of facts that contribute to each combination of dimension values. If only the projection is left empty, the count function is applied over the instances satisfying the measure. For example, the measure $M_3 = (Patient,\ -,\ COUNT)$ considers the instances of $Patient$ in the resulting facts and shows the number of different patients for a specific combination of dimension values.

Given a measure M_i, we use the functions $desc(M_i)$, $proj(M_i)$ and $agg(M_i)$ to access the conceptual description, the projection and the aggregation function of the measure, respectively. As with the dimensions, we use the term $measure$ in general to refer to its conceptual description.

The MD query specification proposed is thought to be simple and does not require the user to have an exact knowledge of the ontology axioms describing the data (i.e., the data schema). Simply by selecting concepts and roles from the ontology through a graphical interface, the user can quickly build an MD query that is automatically translated to DL expressions.

For the running use case, the MD query specified by the user is translated into the following DL expressions:

$$SUBJECT(Patient) = Patient$$
$$D_1(Disease) = (Disease\_or\_Syndrome,\ name)$$
$$D_2(Drug) = (Pharmacological\_Substance,\ name)$$
$$D_3(Gender) = (Gender \sqcap \exists sex^{-}.Patient,\ name)$$
$$M_1(AvgDIndex) = (Rheumatology,\ damageIndex,\ AVG)$$
$$M_2(NumPatients) = (Patient,\ -,\ COUNT)$$

From now on, we refer to the dimensions and measures of the running use case with the short names between parenthesis.

5.3 Aggregation paths

The MD query specified by the user maps to a series of conceptual descriptions, which are scattered in the ontology. The next step consists in finding the connections between the MD elements so that we can produce facts. That is, we must ensure that the subject of analysis is logically related to each dimension and measure. Traditional MD modeling restricts these connections to functional and inclusion dependencies in order to preserve summarizability. However, as discussed previously, huge amounts of data are complex by nature and their relationship escape the summarizability constraints [119]

In an attempt to capture and analyze complex data, we break the functional dependency constraint usually required between facts and dimensions and use the notion of aggregation path, which was initially introduced in [83].

Definition 5.6. *Given an ontology* $\mathcal{O} = (\mathcal{T}, \mathcal{A})$*, the expression* $(C_1, r_1, ..., C_n) \in Paths(C, C')$ *is an aggregation path from concept* C *to concept* C' *if* $C_1 = C$*,* $C_n = C'$*, and there is an interpretation* $I = (\Delta^{\mathcal{I}}, ^{\mathcal{I}})$ *of* $\mathcal{O}$ *and* $\exists x_i \in \Delta^{\mathcal{I}}, 1 \leq i \leq n$*, such that* $x_i \in C_i^{\mathcal{I}}$*,* $(x_i, x_{i+1}) \in r_i^{\mathcal{I}}$*, for* $1 \leq i < n$*. In such case, we say that concept* C' *is reachable from the concept* C*, written* $\mathcal{I} \models C \rightsquigarrow C'$

Notice that for an aggregation path between two concepts of an ontology to exist we only need an interpretation satisfying the ontology axioms and the conditions imposed by the definition of aggregation path. The length of an aggregation path is given by the number of roles. The aggregation paths of length one, (C_1, r_1, C_2), are called *direct* aggregation paths, $\mathcal{I} \models C \rightsquigarrow_D C'$. The subscript D means the aggregation path is direct. These paths are of special relevance, as they constitute the basis over which aggregation paths of any length are generated.

Therefore, a MD query can produce facts if there is at least one aggregation path between the subject of analysis and each dimension and measure. The following definition formalizes the necessary conditions that make an MD query valid.

Definition 5.7. *A MD query specification* $Q = (C_{SUB}, \{D_i\}_{i=1...n}, \{M_q\}_{q=1...t})$ *over an ontology* $\mathcal{O}$ *is a valid query if*

1. $\mathcal{O} \models C_{SUB} \not\sqsubseteq \bot$
2. $\forall C_i \in \{desc(elem_i)/elem_i \in MD\}$, $\mathcal{O} \models C_i \not\sqsubseteq \bot$
3. $\forall C_i \in \{desc(elem_i)/elem_i \in MD\}$, $\exists \mathcal{I}$ *of* $\mathcal{O}$ *such that* $\mathcal{I} \models C_{SUB} \rightsquigarrow C_i$

According to the previous definition, an MD query is valid if 1) C_{SUB} is satisfiable in $\mathcal{O}$, 2) all the dimensions and measures are satisfiable in $\mathcal{O}$ and 3) all the dimensions and measures selected by the user are reachable from the subject of analysis, that is, there is an interpretation where at least an aggregation path between the subject of analysis and each MD element exists.

Although finding one aggregation path between the subject of analysis and each MD element is enough to check the validity of an MD query, such aggregation path may not be the one with the user intended semantics, as there can be many distinct aggregation paths between a pair of concepts. Therefore, we consider interesting to find all the possible aggregation paths between the subject of analysis and each MD element, as they account for all the possible cases of reachability between a pair of concepts logically allowed by the ontology. Later, the user can select the intended ones.

Before shedding light on the algorithm that allows to capture all the different aggregation paths between two concepts, we reflect more on the importance of identifying the different aggregation paths. There exist several reasons why an MD element can have more than one aggregation path associated. The first is that the user's specification of the MD element is somehow general or unconstrained. In this case, there can be an interpretation where the concept can be reached from several different aggregation paths. For example, if the user specifies the dimension $D_1 = (Disease, name)$ with no further restrictions, any aggregation path starting from the subject of analysis that reaches an instance of $Disease$ qualifies. These paths provide different semantics to the specified MD element: the disease can be the patient's main diagnosis, some family member diagnosis, a collateral disease derived from the main disease detected through laboratory tests or rheumatic exams, among others. The second reason has to do with heterogeneity issues. It could be the case that the set of ontology axioms we are dealing with is the result of integrating several ontologies. Think, for example, of an analysis of patients coming from different hospitals, where each hospital uses a different application ontology for recording the information about the patients. The information recorded is basically the same except for some differences in naming conventions. An upper ontology covering the application ontologies of each hospital would solve the problem. However, the proposed method would still work without any integration layer because it would identify each different aggregation path to the MD element. In this case, the semantics provided by each aggregation path are the same and they differ in the structure. A third reason for having different aggregation paths associated to a concept is given by the model-theoretic semantics of the DLs. That is, if we have an aggregation path from C to C' that goes through concepts $\{C_i\}$, all the paths from C to C' that go through more specific or more general concepts of $\{C_i\}$ are also valid aggregation paths.

5.3.1 Basic algorithm for aggregation paths

All the possible aggregation paths from the subject of analysis to each MD element can be represented as a graph rooted in the subject of analysis, where vertices are concepts, edges are direct aggregation paths between concepts and leaf vertices correspond to the conceptual description of MD elements. This graph is called the *reachability graph* and we formalize it in the following way:

Definition 5.8. *Let $Q = (C_{SUB}, \{D_i\}_{i=1,...,n}, \{M_q\}_{q=1,...,t})$ be the MD query of the user over the ontology $\mathcal{O}$. We define the reachability graph associated to Q as a directed graph $G = (V, E)$, where V is a set of vertices that represent ontology concepts and E is a set of ordered pairs (v, v') of vertices of V, called edges, that represent direct aggregation paths from v to v'. The graph satisfies the following conditions:*

1. *The root vertex is C_{SUB}.*
2. *$\forall p = (C_0, r_0, ..., C_n) \in Paths(C_{SUB}, C_i)$ such that $C_i \in \{desc(elem_i) / elem_i \in MD\}$, there is one path $v_0, v_1, ..., v_n$ in the graph where $v_0 = C_{SUB}$, $v_n = C_i$ and $(v_j, v_{j+1}) = (C_j, C_{j+1})$, $0 \leq j < n$*

We are now interested in determining when a direct path (C, r_1, C') can be derived from the ontology, as they are the building block for general aggregation paths.

Proposition 5.1. *Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, two concepts C and C' and a role r_1, there is a direct aggregation path (C, r_1, C') from C to C' through the role r_1 iff $C \sqcap \{\exists, \geq n, \forall, \leq n\} r_1.C'$ is satisfiable in $\mathcal{O}$.*

Proof. The "if" direction is a direct consequence of the definition of aggregation path. There cannot exist a direct aggregation path (C, r_1, C') such that $(C \sqcap \exists r_1.C')$ is not satisfiable in $\mathcal{O}$ because the path (C, r_1, C') implies an interpretation $I = (\Delta^{\mathcal{I}}, ^{\mathcal{I}})$ of $\mathcal{O}$ where $\exists x, x' \in \Delta^{\mathcal{I}}$ such that $x \in C^{\mathcal{I}}$, $x' \in C'^{\mathcal{I}}$ and $(x, x') \in r_1^{\mathcal{I}}$. Therefore, $(C \sqcap \{\exists, \geq n, \forall, \leq n\} r_1.C')^{\mathcal{I}}$ is not empty, which implies that $(C \sqcap \{\exists, \geq n, \forall, \leq n\} r_1.C')$ is satisfiable in $\mathcal{O}$.

We proof the "only if" direction with each different restriction (i.e., $\exists$, $\geq n$, $\forall$ and $\leq n$). From the definition of satisfiability, we have a model $\mathcal{I}$ of $\mathcal{T}$ such that $(C \sqcap \exists r_1.C')^{\mathcal{I}}$ is non-empty. Now we have to prove that $\mathcal{I}$ implies an aggregation path. By interpreting the concept expression, we have that $\exists x_1 \in \Delta^{\mathcal{I}}$ such that $x_1 \in C^{\mathcal{I}} \wedge x_1 \in (\exists r_1.C')^{\mathcal{I}}$. By interpreting the existential restriction we have that $x_1 \in C^{\mathcal{I}} \wedge x_1 \in \{x \in \Delta^{\mathcal{I}} / \exists x_2.(x, x_2) \in r_1^{\mathcal{I}} \wedge x_2 \in C'^{\mathcal{I}}\}$. This means that $x_1 \in C^{\mathcal{I}} \wedge (x_1, x_2) \in r_1^{\mathcal{I}} \wedge x_2 \in C'^{\mathcal{I}}$, which implies a direct aggregation path (C, r_1, C'). The proof with the greater or equal cardinality restriction is analogous. By interpreting the $\geq n$ restriction we have that $\exists x_1 \in \Delta^{\mathcal{I}}$ such that $x_1 \in C^{\mathcal{I}} \wedge x_1 \in \{x \in \Delta^{\mathcal{I}} / |\{x_2 / (x, x_2) \in r_1^{\mathcal{I}} \wedge x_2 \in C'^{\mathcal{I}}\}| \geq n\}$. This implies that there are at least n x_1 objects such that $x_1 \in C^{\mathcal{I}} \wedge (x_1, x_2) \in r_1^{\mathcal{I}} \wedge x_2 \in C'^{\mathcal{I}}$, which satisfies the definition of direct aggregation path. The difference between the previous restrictions and $\forall$ and $\leq n$ is that for $\exists$ and $\geq n$, all the models $\mathcal{I}$ of $\mathcal{T}$ imply an aggregation path, whereas for $\forall$ and $\leq n$ we can prove there exists a model that implies an aggregation path but not all the models imply it. By interpreting the $\forall$ restriction we have that $\exists x_1 \in \Delta^{\mathcal{I}}$ such that $x_1 \in C^{\mathcal{I}} \wedge x_1 \in \{x \in \Delta^{\mathcal{I}} / \forall x_2.(x, x_2) \in r_1^{\mathcal{I}} \implies x_2 \in C'^{\mathcal{I}}\}$. If we consider a model where x_2 exists, then we have $x_1 \in C^{\mathcal{I}} \wedge (x_1, x_2) \in r_1^{\mathcal{I}} \wedge x_2 \in C'^{\mathcal{I}}$, which implies a direct aggregation path (C, r_1, C'). Finally, by interpreting the $\leq n$ restriction

we have that $\exists x_1 \in \Delta^{\mathcal{I}}$ such that $x_1 \in C^{\mathcal{I}} \ \wedge x_1 \in \{x \in \Delta^{\mathcal{I}} / |\{x_2/(x, x_2) \in r_1^{\mathcal{I}} \wedge x_2 \in C'^{\mathcal{I}}\}| \leq n\}$. If we consider a model where there is at least one x_2, this implies at least one x_1 object such that $x_1 \in C^{\mathcal{I}} \ \wedge (x_1, x_2) \in r_1^{\mathcal{I}} \wedge x_2 \in C'^{\mathcal{I}}$, which satisfies the definition of direct aggregation path (C, r_1, C'). □

The basic algorithm for capturing all possible aggregation paths is shown in Algorithm 7 and consists in constructing the reachability graph for the user's MD query. Here, we restrict ourselves to aggregation paths composed by named concepts and named roles, as individual and role assertions in the ABox are placed in these terms. The algorithm starts from the subject of analysis concept C and derives all direct aggregation paths from C, (C, r_1, C'), by applying Proposition 5.1 with each possible named role r_1 and named concept C'. The direct aggregation paths are added to the graph and the reached concept is marked only if it is an MD element. Then, the algorithm recursively finds direct aggregation paths from the newly reached concepts. If a concept has already been reached, it is not expanded again. The algorithm terminates when all the reached concepts cannot be further expanded, which is when all concepts have been visited in the worst case. After that, the graph needs to be pruned by removing all paths that do not lead to MD elements (i.e., marked nodes). By transitivity, we obtain the aggregation paths from the subject of analysis to each MD element by navigating the edges of the graph. In summary, the algorithm is based on the application of the following two rules:

$$\text{If } (C \sqcap \{\exists, \geq n, \forall, \leq n\} r_1.C')^{\mathcal{I}} \neq \emptyset, \text{ then } \mathcal{I} \models C \rightsquigarrow_D C' \tag{5.1}$$

$$\text{If } \mathcal{I} \models C \rightsquigarrow_D C' \text{ and } \mathcal{I} \models C' \rightsquigarrow_D C'', \text{ then } \mathcal{I} \models C \rightsquigarrow C'' \tag{5.2}$$

The algorithm is sound, since it computes direct aggregation paths according to the Rule 5.1 and propagates them according to the transitivity Rule 5.2. The algorithm is complete because of Proposition 5.1. Moreover, as the number of named concepts and roles over the ontology is finite, the generation of direct aggregation paths and propagation by transitivity clearly terminates.

Despite the theoretical importance underlying the previous algorithm, its direct application results inefficient, as it implies generating all the combinations of each possible named role and named concept to find direct aggregation paths. Notice that the algorithm generates all possible aggregation paths *allowed* by the ontology. This means that there is an interpretation satisfying the ontology (i.e., a model of the ontology) that allows instances of some concept C to reach instances of some other concept C' through binary relations. If the ontology is underspecified and not properly constrained, many of the generated aggregation paths, although valid, will be of no use to the analyst.

Moreover, the ultimate goal of the approach is to extract facts and dimensions from the ABox guided by the aggregation paths generated from the TBox. Therefore, aggregation paths that do not have a representation in the ABox are of no interest.

Algorithm 7 Aggregation paths (basic algorithm)

Procedure AggregationPaths(MDQ, G)
Input: MDQ, the multidimensional query
Output: G, reachability graph or empty graph

```
1: G ← ∅                                              ▷ Reachability graph
2: visited = ∅                                        ▷ List of visited concepts
3: AggregationPathsRec(MDQ.subject(),visited, G)
4: pruneReachabilityGraph(G)          ▷ Prune paths that do not end in an MD element
5: return G
```

Procedure aggregation_paths_rec(C,$visited$, G)
Input: C, $visited$, G
Output: G,

```
1: if C ∉ visited then
2:     add(visited, C)
3:     P = findDirectAggPaths(C)                      ▷ By Proposition 5.1
4:     for (r, C′) ∈ P do
5:         addEdge(G, (C, r, C′))
6:         checkForMDElement(C′)                      ▷ Mark C′ if it is an MD element
7:         getGraphDirectAggPaths(C′)
8: return
```

The following section elaborates on these ideas and makes a classification of the interesting aggregation paths to the user.

5.3.2 Interesting aggregation paths

According to the definition of Guarino [48] of an ontology, "an ontology is a logical theory which gives an explicit, partial account of a conceptualization". This definition captures the idea that an ontology has a formal specification and that an ontology cannot always be complete. This notion is illustrated in Figure 5.2. The conceptualization $\mathcal{C}$ allows the models of some language, $\mathcal{M}(\mathcal{L})$, to be constrained to a subset of intended models $\mathcal{I}_{\mathcal{K}}(\mathcal{L})$ due to a commitment $\mathcal{K}$ to a specific conceptualization. The ontology intends to constrain the possible interpretations of a language's vocabulary so that its logical models approximate as well as possible the set of intended models of a conceptualization of that domain.

According to the previous definition, we can categorize the quality of the ontologies in terms of how well they approximate the set of intended models of a conceptualization. This is shown in Figure 5.3. Good ontologies are those that cover the intended models with maximum precision. Maximum coverage should be a requirement, otherwise the ontology is not covering some intended models. The less precision they have, the more not intended models they allow. The most usual case is the latter, that is, the ontology is underspecified as it does not contain the necessary axioms to exclude unintended logical models. This situation has a direct impact on the generation of aggregation paths.

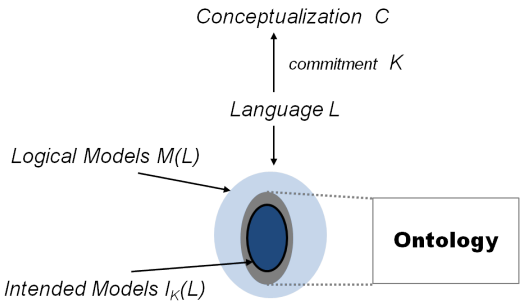

Figure 5.2: Relations between language (vocabulary), conceptualization, ontological commitment and ontology.

As a result, many aggregation paths will be generated because there is an interpretation that satisfies the definition (i.e., the ontology allows it), which is caused by the lack of further axioms that constrain the ontology models to the intended ones. These aggregation paths, although valid by definition, were not probably meant by the ontology engineer. Therefore, they are also of no use to an analyst whose aim is to perform an MD analysis over a repository of instances (ABox) and who expresses her requirements in terms of conceptual axioms of the TBox.

This situation also reflects on the ABox, as the usual process is to make assertions in the ABox using named concepts and roles specified in the TBox. A poor quality of the TBox can make the ABox to be consistent even if it contains assertions that were not meant by the ontology engineer.

From the previous reflection, we conclude that the number of possible aggregation paths in an ontology can be enormous and greatly depends on the quality of the TBox, which also directly affects the assertions in the ABox.

On the other hand, there is no doubt that the axioms that the ontology engineer explicitly creates when developing the ontology, are the ones intended to capture the conceptualization, even though these axioms may lead to unintended models as explained before. Similarly, as the analyst expresses her requirements in terms of the TBox, it is likely that the expected aggregation paths are the ones generated from the intended models of the conceptualization and not the ones generated by the unintended models that the ontology captures due to its poor precision.

Based on this assumption, we have decided to characterize different groups of aggregation paths based on the user's expectations. We also establish an order in their generation, giving priority to the most restrictive paths (Group 1), which are the ones that most exactly reflect the user's conceptualization, and opening the range to not so evident aggregation paths that may be also interesting for analysis (Groups 2 and 3). By establishing this priority, we

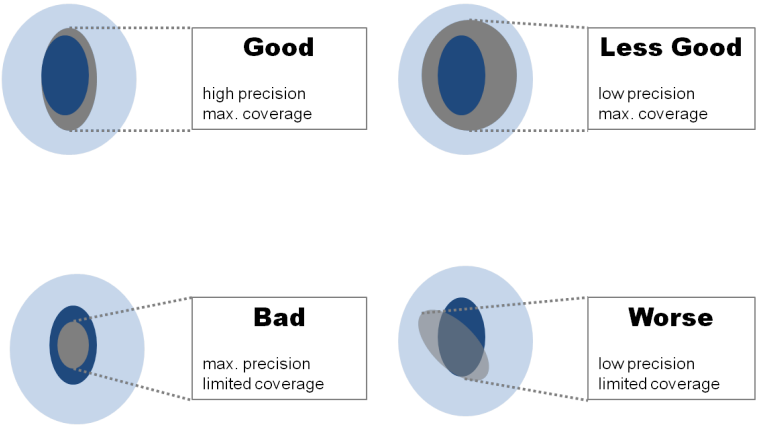

Figure 5.3: Ontology accuracy according to how well it approximates the intended models.

are cutting down the potential number of generated aggregation paths, which can be inadmissible if the ontology is underspecified. The different groups are not auto-contained and the aggregation paths can have an instantiation in the ABox regardless of the group, although the most common is to create ABox assertions that follow directly from the TBox axioms.

We provide an efficient implementation for the three groups of aggregation paths, which avoids trying any possible combination of role and concept. The different groups are generated by following the same exploratory-based approach as in the basic algorithm, that is, direct aggregation paths from the subject of analysis are firstly generated and then, this is recursively performed for each new reached concept. However, instead of applying Rule 5.1 which tries any possible combination of role and concept for generating direct aggregation paths, each group applies a smarter set of rules to derive direct aggregation paths that fit the group.

In the following sections we explain with more detail the characterization of each group of aggregation paths in order, from the most interesting to the analyst to the least.

5.3.2.1 Group 1: Strong aggregation paths

The most restrictive and interesting aggregation paths for the analyst are the ones that ensure the reachability from a concept C to C' in all models of the TBox. As opposed to the base definition of aggregation path in Definition 5.6, which only requires a model $\mathcal{I}$ of $\mathcal{O}$ to satisfy the conditions, the aggregation paths in this group are more restrictive as they are logically implied by the assertion of the TBox. Thus, we need to re-define the notion of aggregation

path in this group. We call these paths *strong aggregation paths.*

Definition 5.9. *Given an ontology* $\mathcal{O} = (\mathcal{T}, \mathcal{A})$*, the expression* $(C_1, r_1, ..., C_n) \in Paths(C, C')$ *is a strong aggregation path from concept* C *to concept* C' *if* $C_1 = C$*,* $C_n = C'$*, and for all interpretations* $I = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *of* $\mathcal{O}$ *and* $\forall x_i \in C_i^{\mathcal{I}}$*,* $(x_i, x_{i+1}) \in r_i^{\mathcal{I}}$*,* $1 \leq i < n$*.*

The previous definition requires that each individual of C is connected to at least one individual of C' by means of a chain of roles in $\mathcal{T}$. This definition is very restrictive compared to the definition of aggregation path in Definition 5.6, which does not always ensure connectivity in practice, as it only requires that there exists an interpretation $\mathcal{I}$ of $\mathcal{O}$ where at least an object of $C^{\mathcal{I}}$ is connected by a chain of roles to at least an object of $C'^{\mathcal{I}}$.

Notice that the functional dependency usually required between facts and dimensions in order to ensure summarizability is even more restrictive than the previous definition. Actually, all functional paths are captured in the definition of strong aggregation path.

By exploiting the model-theoretic semantics of DL, we show when a strong aggregation path can be inferred from an ontology $\mathcal{O}$.

Proposition 5.2. *Given an ontology* $\mathcal{O} = (\mathcal{T}, \mathcal{A})$*, the expression* $(C_1, r_1, ..., C_n) \in Paths(C, C')$ *is a strong aggregation path from concept* C *to concept* C' *iff* $C_1 = C$*,* $C_n = C'$*, and there is a role chain* $R = r_1 \circ ... \circ r_n$ *over* $\mathcal{T}$*,* $n > 0$*, such that* $\mathcal{T} \models C \sqsubseteq \{\exists, \geq n\}R.C'$

Proof. The "if" direction follows from Definition 5.9.

The "only-if" direction is proved by the canonical model property of the DL. □

According to the previous proposition we would have to check the implication $\mathcal{T} \models C \sqsubseteq \{\exists, \geq\}R.C'$ from the subject of analysis C to all concepts C' matching an MD element with all possible lengths n of the chain R, and all possible ways of composing R with named roles. As this is clearly inefficient, we follow the exploratory-based approach of the basic algorithm where the paths are composed by recursively discovering direct aggregation paths starting from the subject of analysis and the successive reached concepts, marking MD elements along the way. Thus, all the aggregation paths of the different groups are composed from direct aggregation paths thanks to the transitivity Rule 5.2.

The generation of strong direct aggregation paths is a direct consequence of Prop. 5.2.

Proposition 5.3. *Given an ontology* $\mathcal{O} = (\mathcal{T}, \mathcal{A})$*, the expression* (C, r_1, C') *is a strong direct aggregation path from concept* C *to* C' *through the role* r_1 *if* $\mathcal{T} \models C \sqsubseteq \{\exists, \geq n\}r_1.C'$ *or if* $\mathcal{T} \models C' \sqsubseteq \{\exists, \geq n\}r_1.C$ *and* r_1 *is an inverse functional role.*

Proof. The proof is the same as in Prop. 5.2 □

Therefore, the rules applied to generate direct strong aggregation paths are:

$$\text{If } \mathcal{T} \models C \sqsubseteq \{\exists, \geq n\} r_1.C', \text{ then } \forall \mathcal{I} \text{ of } \mathcal{O},\ \mathcal{I} \models C \rightsquigarrow_D C' \tag{5.3}$$

$$\text{If } \mathcal{T} \models C \sqsubseteq \{\exists, \geq n\} r_1.C' \text{ and } r_1 \text{ is inverse funct, then } \forall \mathcal{I} \text{ of } \mathcal{O},\ \mathcal{I} \models C' \rightsquigarrow_D C \tag{5.4}$$

These rules are efficiently implemented by using the $\mathcal{LS}^{-}$ index to propagate all the restrictions attached to a concept to its descendants (see Section 5.4.1).

An example of such kind of aggregation path in the running use case is the path
$(Patient, hasVisit, Visit, hasReport, Report)$, where each instance of $Patient$ is necessarily connected to at least one instance of $Report$.

5.3.2.2 Group 2: Aggregation paths through universal, less than cardinality restrictions, inverse roles, rdfs:domain and rdfs: range axioms.

Although interesting for the analyst, the previous group of aggregation paths is too restrictive and may not capture many aggregation paths interesting for analysis. This group of aggregation paths extends the previous group with three types of direct aggregation paths that, even though they are not satisfied in every model, they can be useful for analysis purposes.

The first type of direct aggregation paths that we cover in this group are the ones through $\forall$ and $\leq n$ restrictions. The use that the ontology engineers make of the different types of restrictions $(\exists, \geq n, \forall, \leq n)$ has a great impact on the logical aspects, although it is difficult to foresee it at first sight. The existential and the greater or equal cardinality restrictions used in the previous group (i.e., $\exists, \geq n$) ensure that the relation holds for all members of each concept in the path, whereas the universal and less or equal cardinality restrictions (i.e., $\forall, \leq n$) do not. Therefore, we extend the previous group of aggregation paths by also considering direct aggregation paths with the restrictions $\forall, \leq n$ to account for possible mistakes in the conceptualization of the ontology, even though at the logical level we know that an aggregation path found through a combination of these restrictions is not necessarily satisfied in every model of $\mathcal{T}$, thus, it is not a strong aggregation path anymore. Next, we show an example that illustrates this issue. The ontology developer may have correctly selected the $\forall$ restriction in the axiom $DiagnosisReport \sqsubseteq \forall hasDisease.Disease$ to constraint diagnosis reports to only be related to diseases. However, if this axiom is not accompanied by another one enforcing the relation (i.e., with $\exists$ restriction), the ontology is allowing models where diagnosis reports are not related to diseases, therefore, the aggregation paths derived from this axiom are not satisfied in every model.

The second type of direct aggregation paths that are included in this group are the ones derived from axioms involving inverse roles, in order to account for

different conceptualizations of the domain. We illustrate this with an example. Suppose that the ontology has an axiom $Visit \sqsubseteq \exists hasVisit^{-}.Patient$ to express that each $Visit$ is associated to a $Patient$. Even though we know that we cannot infer the direct strong aggregation path $(Patient, hasVisit, Visit)$, expressing that all patients will have an associated $Visit$ (unless $hasVisit$ is functional, and therefore, it would be an aggregation path of the Group 1), there are some interpretations where it is possible, thus, we allow these types of aggregation paths, $(Patient, hasVisit, Visit)$, to be inferred in this group.

The third type of direct aggregation paths regarded in this group are the ones derived from *rdfs:domain* and *rdfs:range* axioms. As many of the tested ontologies make use of these axioms to constrain the usage of the roles instead of using OWL restrictions, we are forced to consider these axioms so that our method to analyze such data can be applied. By translating these axioms into DL we obtain the following: R *rdfs:domain* C is equivalent to $\top \sqsubseteq \forall R^{-}.C$, from which we could derive the direct aggregation path $(C, R, \top)$, whereas R *rdfs:range* C' is equivalent to $\top \sqsubseteq \forall R.C'$, from with we could derive the direct aggregation path $(\top, R, C')$. However, these aggregation paths are dangerous in the sense that they leave one of the fillers for the role empty, meaning that any concept could be placed in that position and the aggregation path may grow in an uncontrolled way. To avoid this situation, we only allow to derive direct aggregation paths (C, R, C'), where both fillers are restricted to a concept. This happens when there is both the domain and range axioms defined for a role.

Consequently, we apply the following rules to generate direct aggregation paths in this group:

$$\text{If } \mathcal{T} \models C \sqsubseteq \{\exists, \geq n, \forall, \leq n\} r_1.C', \text{ then } \exists \mathcal{I} \text{ of } \mathcal{O}, \mathcal{I} \models C \rightsquigarrow_D C' \tag{5.5}$$

$$\text{If } \mathcal{T} \models C \sqsubseteq \{\exists, \geq n, \forall, \leq n\} r_1^{-}.C', \text{ then } \exists \mathcal{I} \text{ of } \mathcal{O}, \mathcal{I} \models C' \rightsquigarrow_D C \tag{5.6}$$

$$\text{If } \mathcal{T} \models \top \sqsubseteq \forall r_1^{-}.C \text{ and } \mathcal{T} \models \top \sqsubseteq \forall r_1.C', \text{ then } \exists \mathcal{I} \text{ of } \mathcal{O}, \mathcal{I} \models C \rightsquigarrow_D C' \tag{5.7}$$

5.3.2.3 Group 3: Aggregation paths through concept specializations

As mentioned, the previous aggregation paths may still be restrictive and may not capture some of the aggregation paths interesting for analysis. Actually, an interesting group of aggregation paths from the analyst's viewpoint is the one that extends the previous groups by including also aggregation paths that have been generated by navigating through roles to the sub-concepts of a reached concept. These aggregation paths do not satisfy either the definition of strong aggregation path.

We have realized that the ontology developing pattern of creating successively role restrictions over concept specializations is very extended and probably inherited from the development in RDFS. The running example illus-

trates this issue. Although there is the interesting aggregation path ($Patient$, $hasVisit$, $Visit$, $hasReport$, $DiagnosisReport$,$hasDiagnosis$,$Disease$) that allows patients to reach their diagnosed disease, this path is not captured in the previous two groups of paths, as the axioms $Visit \sqsubseteq \exists hasReport.Report$ and $DiagnosisReport \sqsubseteq Report$ do not logically imply that each $Visit$ is connected to some $DiagnosisReport$. In order to capture such aggregation paths, we extend the direct aggregation paths of the previous group by introducing a new rule that infers the direct aggregation path ($Visit$,$hasReport$,$DiagnosisReport$). The rule has the following shape:

$$\text{If } \mathcal{I} \models C \rightsquigarrow_D C' \text{ and } C''^{\mathcal{I}} \subseteq C'^{\mathcal{I}}, \text{ then } \mathcal{I} \models C \rightsquigarrow_D C'' \tag{5.8}$$

5.3.2.4 Composition of aggregation paths

The previous sections describe the types of direct aggregation paths that characterize each group. Here, we describe the types of direct aggregation paths that can compose aggregation paths of general length. Thus, aggregation paths belonging to Group 1 can only be composed by direct aggregation paths of type Group 1 (i.e., generated with Rules 5.3-5.4). Aggregation paths belonging to Group 2 can be composed by direct aggregation paths of Group 1 and Group 2 (i.e., generated with Rules 5.3-5.4 and 5.5-5.7. Aggregation paths belonging to G3 can be composed by direct aggregation paths of Group 1, Group 2 and Group 3 (i.e., generated with Rules 5.3-5.4, 5.5-5.7 and 5.8).

5.4 Implemented algorithm for aggregation paths

As discussed in the previous sections, the potential number of aggregation paths can be very large compared to the paths that are actually instantiated in the ABox. Therefore, generating all the aggregation paths can be both excessive and very expensive. On the other side, trying to reach MD elements from the subject of analysis by using the ABox without any guidance can also be very expensive with ABoxes of considerable size. Moreover, it could be the case that access to the ABox is only partial or restricted (e.g., via an SPARQL endpoint). In such case, the user has to exploit the TBox axioms to perform the MD analysis. Similarly, if the ABox inferences have not been fully materialized to keep the size of the ABox manageable, we need to resort to the TBox axioms to make ABox inferences on demand.

The adopted solution consists in progressively discovering aggregation paths from the previous groups in ascending order, from Group 1 to Group 3, combined with evidence from the ABox. We combine both the logical knowledge provided by the Tbox with evidence from the ABox to reach a compromise between logical consistency and efficiency. The evidence of the ABox is mainly used to prune potential aggregation paths that have not been instantiated.

This check will be performed by the function *hasABoxEvidence*, which by now, can be seen as a black box that returns the *true* value if the checked direct aggregation path has been instantiated in the ABox. Later in Section 5.4.3 we discuss the implementation of this function and the pre-processing required. As a result, the proposed algorithm obtains a *restricted reachability graph* that instead of containing all the potential aggregation paths to each MD element, contains only aggregation paths useful for the posterior MD analysis according to the previously defined groups.

We assume that the ontology has been indexed by applying the indexing mechanisms described in Chapter 4. Therefore, we have the indexes $\mathcal{LS}^-$ and $\mathcal{LS}^+$ for named concepts, the $\mathcal{LS}^-$ for roles and the tables T_C and T_R for the concept and role assertions available.

The algorithm is shown in Algorithm 8 and can be split in several blocks. First, the satisfiability of the subject of analysis and the MD elements is checked (lines 5-9). Then, the different groups of interesting aggregation paths to the MD elements are explored in turn (procedure calls in line 10 for Group 1, line 12 for Group 2 and line 14 for Group 3). The main idea of these procedures is to only generate the most interesting aggregation paths for the user in descending order of priority according to the previous classification of aggregation paths in groups. If an aggregation path to an MD element has been discovered by one of these procedures, further aggregation paths for that MD element discovered by the subsequent procedures are not considered. The procedure $FindAggPaths$ serves to generate the paths and is divided in two phases. The first one is performed by the $CreateDataStructure$ procedure, which is in charge of creating an intermediate data structure that has a graph shape, where each node represents a concept and collects the necessary information to reconstruct the aggregation paths of the corresponding group, depending of the *mode* flag. Regardless of this, the second phase is analogous for the three groups of aggregation paths and is performed by the $GetPaths$ procedure, which reconstructs the aggregation paths starting from the found MD elements and going backwards to the subject of analysis through the previous data structure. Finally, if it is a valid graph (line 15), the user selects the intended aggregation paths for each MD element. This procedure is further detailed later. The algorithm terminates when: 1) all the MD elements have been reached or 2) when all the reached concepts are *leaves* (meaning that they cannot generate any aggregation path from them). During the exploration, the algorithm keeps track of the visited concepts in order to detect cycles. Already visited concepts are not further expanded.

Before going into the details of the algorithm, let us firstly clarify the data structures and the naming conventions adopted. We use the terms *ancestors* and *descendants* of a concept to refer to all the super-concepts and sub-concepts, respectively, logically inferred from the ontology. When dealing with graph-oriented data structures, if there is an edge from node v to u, then v is a *predecessor* of u and u is a *successor* of v. We also differentiate between the

Algorithm 8 Generation of aggregation paths

Require: MDQ: the multidimensional query
Ensure: G: restricted reachability graph or empty graph

```
1: function GenerateAggPaths(MDQ)
2:     S ← ∅                                              ▷ Intermediate data structure
3:     G ← ∅                                                          ▷ Reachability graph
4:     MD ← ∅                          ▷ List containing nodes that are MD elements
5:     if not IsSatisfiable(MDQ.subject()) then
6:         return G
7:     for elem ∈ MDQ do
8:         if not IsSatisfiable(elem) then
9:             return G
10:    FindAggPaths(S, MD, G, MDQ.subject(), mode = 1)
11:    if |MD|! = |MDQ.elements()| then
12:        FindAggPaths(S, MD, G, MDQ.subject(), mode = 2)
13:        if |MD|! = |MDQ.elements()| then
14:            FindAggPaths(S, MD, G, MDQ.subject(), mode = 3)
15:    if IsReachabilityGraph(G, MD) then
16:        UserVerification(G)
17:    else
18:        G ← ∅
19:    return G

20: function FindAggPaths(S, MD,G, c, mode)
21:    CreateDataStructure(S, MD, c, mode)
22:    GetPaths(MD, G)

23: function CreateDataStructure(S, MD, c, mode)
24:    n = S.createNode(c)
25:    CreateDataStructureRec(S, MD,n, mode)

26: function CreateDataStructureRec(S,MD, n, mode)
27:    if not n.isProcessed() then
28:        n.setProcessed()
29:        CheckMDElement(MD,n)        ▷ If n is an MD element, add it to the list MD
30:        aggPaths =GetDirectAggregations(n.concept(), mode)
31:        for aggPath ∈ aggPaths do
32:            if hasABoxEvidence(aggPath) then
33:                newNode = S.createNode(aggPath.range())
34:                n.addAgg(newNode)                          ▷ Extend aggregation path
35:                newNode.addPredecessor(n)
36:                CreateDataStructureRec(S,MD,newNode, mode)
37:    if mode == 3 and n ∉ n.getPredecessors().descendants() then
38:        descendants =GetDescendants(n.concept())
39:        for desc ∈ descendants do
40:            newNode = S.createNode(desc)
41:            n.addDescendant(newNode)              ▷ each node keeps the descendants
42:            newNode.addPredecessor(n)
43:            CreateDataStructureRec(S,MD, newNode, mode)
44:        for aggNode ∈ {n.aggregations()} do
45:            CreateDataStructureRec(S,MD, aggNode, mode)
```

```
46: function GetDirectAggregations(C, mode)
47:     if mode == 1 then
48:         aggPaths =GetRestrictions(C, G1)
49:     if mode == 2 then
50:         aggPaths =GetRestrictions(C, G2)
51:     return aggPaths

52: function GetPaths(MD, G)
53:     for node ∈ MD do
54:         if not node.expanded() then ▷ MD elements whose agg. paths have been
    expanded in previous steps are not expanded again, as the previous paths have priority
55:             GetPathsRec(G,node)
56:             node.expanded() = true

57: function GetPathsRec(G,node)
58:     vertex = createVertex(node) ▷ Creates a vertex and adds it to G
59:     G.add(vertex)
60:     descVertices = ∅ ▷ Keep current processed vertex and its descendants in this list to
    later add to them the corresponding agg. paths
61:     descVertices.add(vertex)
62:     for descNode ∈ node.descendants() do
63:         descVertex = createVertex(descNode)
64:         G.add(descVertex)
65:         descVertices.add(descVertex)
66:     for aggNode ∈ {node.aggregations()} do
67:         if aggNode ∈ G then ▷ Only add agg. path to already processed nodes
68:             vertexAgg = G.get(aggNode)
69:             for descVertex ∈ descVertices do
70:                 descVertex.addSuccessor(vertexAgg) ▷ Extend aggregation path
71:                 for descAggNode ∈ aggNode.descendants() do
72:                     descAggVertex = createVertex(descAggNode)
73:                     G.add(descAggVertex)
74:                     descVertex.addSuccessor(descAggVertex) ▷ Extend aggregation path
75:     directRec=0
76:     for predNode ∈ node.getPredecessors() do
77:         if node ∉ predNode.descendants() then ▷ Recursion only of nodes that are not
    ancestors
78:             directRec=1
79:             GetPathsRec(G, predNode)
80:     if not directRec then
81:         for predNode ∈ node.getPredecessors() do
82:             for predPredNode ∈ predNode.getPredecessors() do ▷ Skip ancestor nodes
83:                 GetPathsRec(G, predPredNode)
```

nodes of the intermediate structure, which we simply call *nodes*, and the nodes of the reachability graph, which we call *vertices*.

Now, we explain with more detail the execution flow to create aggregation paths of each group. The *CreateDataStructure* procedure (line 23) creates an intermediate data structure where nodes represent concepts and the arcs between nodes represent direct aggregation paths between concepts. This structure is kept in the S variable and is constructed by recursively processing each concept starting from the subject of analysis and following a depth-first search approach. During the exploration, the function *CheckMDElement* (line 29) checks if the concept that represents the current node is an MD element. In that case, the node is marked and added to the global list *MD*. Each node n keeps the concept that represents, which can be accessed with the *n.concept* method, along with references (i.e., arcs) to other nodes, which represent direct aggregation paths. To find direct aggregation paths of Group 1, the function *GetDirectAggregations* (line 30) with the mode flag set to 1 is called, which retrieves strong direct aggregation paths according to Rules 5.3 and 5.4. To find direct aggregation paths of Group 2, we set the flag to mode 2 and retrieve direct aggregation paths according to Rules 5.5 and 5.6. Both types of direct aggregation paths have been precomputed with the help of $\mathcal{LS}^-$ index (see Section 5.4.1) and are accessible via a simple index look-up. Only if these direct aggregation paths have ABox evidence, the algorithm builds an arc from node n representing C to node m representing C' and the recursion continues. To find direct aggregation paths of Group 3, the previously encountered paths need to be propagated to the sub-concepts of the destination node. For that, the S structure keeps also the descendants of a node and the recursion expands each of them. As a result, we obtain the data structure S where, for each concept (i.e., node), all its direct aggregation paths are made explicit through arcs. Notice that we only explore nodes that have not been processed (line 27).

Then, the procedure *GetPaths* is in charge of reconstructing all the aggregation paths from the previous data structure, starting from each node that is an MD element and going backwards through the predecessor arcs by using the method *node.getPredecessors*(). To build aggregation paths of Group 1, the algorithm traverses backwards only direct aggregation paths of Group 1. To build aggregation paths of Group 2, the algorithm can traverse backwards direct aggregation paths of Group 1 and Group 2. For the case of Group 3, as we are also considering aggregation paths through the sub-concepts, the direct aggregation paths of Group 1 and Group 2 associated to a node have to be created and propagated to all the descendants of such node (line 70). Each of the newly created direct aggregation paths triggers Rule 5.8 and creates a new direct aggregation path between the starting node of the aggregation path and each descendant of the destination node (line 74). Notice that when processing a node, we only create direct aggregation paths to other nodes that are already in the reachability graph (condition in line 67). This ensures that we are extending only the paths that have started from MD elements. Then,

the function is recursively applied over predecessor nodes of the current node but skipping nodes that are ancestors. The ancestors of a node should only be processed if they are reached from an MD element. In such case, these nodes will be reached when the reconstruction of the aggregation paths of such MD element begins. Therefore, we do not need to process them. The function finishes when we reach the subject of analysis node, which has no predecessors.

If the resulting graph contains at least one path from the subject of analysis to each MD element specified by the user, then the graph is a reachability graph (line 15). Otherwise, the empty graph is returned as the user's MD query cannot generate facts. The procedure $UserVerification$ is a user-assisted process where the user selects the aggregation paths of interest for each MD element. This process is explained later.

Figure 5.4 shows the *restricted reachability graph* obtained for the running use case before the user verification process. Shaded nodes represent the dimensions and measures. Notice that for all MD elements, there is only one aggregation path from the subject of analysis except for the dimension *Disease*, where two possible aggregation paths have been found.

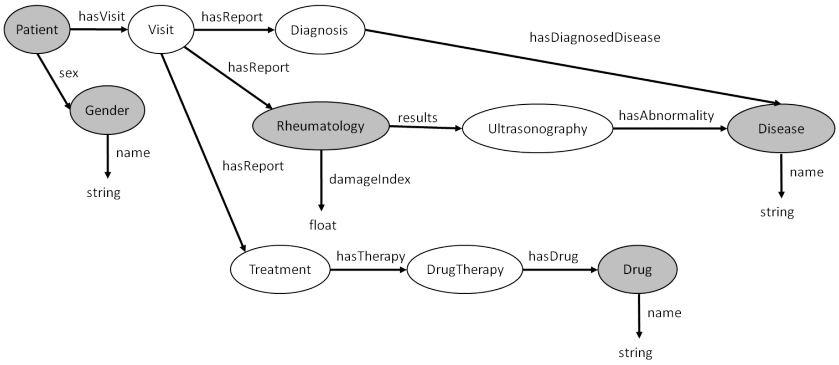

Figure 5.4: Restricted reachability graph of the running use case.

The following sections explain in detail some of the phases of the proposed algorithm. In particular, Section 5.4.1 shows an efficient method to precompute the different types of direct aggregation paths. Section 5.4.2 explains in more detail the user verification process by which (s)he selects the intended aggregation paths for each MD element. Finally, Section 5.4.3 proposes an efficient approach to query for ABox evidence during the construction of the graph.

5.4.1 Precomputation of direct aggregation paths

Direct aggregation paths are the building block to construct aggregation paths. In this section we show an efficient precomputation of all the possible direct

aggregation paths that makes use of the $\mathcal{LS}^-$ index of concepts. Notice that the precomputation has many benefits, as it is done only once per ontology. Then, different reachability graphs can be built from different MD queries very fast by accessing the precomputed direct aggregation paths. As a result, the function *GetDirectAggregations* (line 30 of Algorithm 8) returns the corresponding direct aggregation paths via a simple look-up. Algorithm 9 shows how the precomputation is performed. The direct aggregation paths are looked for in every named concept of the ontology. Thus, the function *GetRestrictions* looks into the definition of the concept for the associated restriction, according to the type of direct aggregation path that we are looking for (either Group 1 or Group 2), and returns aggregation paths of the required type. Then, these paths are propagated to the descendants, as any restriction associated to a concept is inherited by its descendants. Therefore, we create an index that holds, for each concept and type of aggregation path, the associated direct aggregation paths.

Algorithm 9 Precomputation of direct aggregation paths

Procedure PreDirectAggPaths(r)
Input: $\mathcal{LS}^+$, $N_{c'}$: named concepts
Output: I, an index

```
1: for C ∈ N_c' do
2:     if IsSatisfiable(C) then
3:         aggPaths1 =GetRestrictions(C, G1)    ▷ Set of (C, R, C') from Rules 5.3 and 5.4
4:         aggPaths2 =GetRestrictions(C, G2)    ▷ Set of (C, R, C') from Rules 5.5, 5.6 and 5.7
5:         descendants = expand(C^{LS^-})
6:         for desc ∈ descendants do
7:             for (C, R, C') ∈ aggPaths1 do
8:                 I[C][type1].add(C, R, C')
9:             for (C, R, C') ∈ aggPaths2 do
10:                I[C][type2].add(C, R, C')
11: return I
```

Notice that we do not precompute aggregation paths of Group 3, as it would be redundant. These direct aggregation paths correspond to any of the precomputed direct aggregation paths of Group 1 or Group 2, where the range is substituted by a sub-concept. These paths can immediately be obtained by querying the $\mathcal{LS}^-$ index for the sub-concepts.

Both the time and space complexity of the algorithm is $O(r \cdot n^2)$, where r is the number of roles and n the number of concepts. However, this upper bound is hardly achievable as not all the concepts have restrictions to all the roles and also, not all concepts are hierarchically related.

5.4.2 User Verification

The $UserVerification$ procedure shows the user the discovered aggregation paths for each MD element. If one MD element has more than one aggregation path associated, the user can choose between different options according to the intended semantics of the MD element. That is, the disambiguation process due to the poor specificity of the MD query is handled by the user. The options that the user can choose from are translated into transformations over the reachability graph. The options are:

1. To select only one intended meaning for the MD element by selecting only one path and discarding the rest. This option is shown in Figure 5.5a. In this case, two different aggregation paths have been found for the *Disease* dimension. However, the initial intended meaning for this dimension was the main diagnosed disease. Therefore, the user discards the path leading to secondary diseases or abnormalities found in ultrasonography reports.

2. To select a subset of paths for the MD element and discard the remaining ones. The user may be interested in selecting more than one path for one MD element to account for heterogeneity in the knowledge base. This option is shown in Figure 5.5b. Let us suppose that some patients have the diagnosed disease recorded at the visit level (i.e., in each visit) while some patients have it at the top level. In that case, the user will want to consider both paths for the dimension and discard the rest.

3. To split the initial MD element into as many MD elements as required, by associating to each new MD element the required paths. This option is shown in Figure 5.5c. Let us suppose the user has initially specified the *Disease* dimension to account for the patients' diagnosis. However, when the system shows the aggregation paths associated to disease, the user finds interesting to also analyze the patients according to the diseases found through the ultrasonographies. Therefore, she decides to add this new dimension that had not occurred to her in the first place. In this case, the dimension node is split in two to account for the two different dimensions, and any nodes under it are also replicated.

From now on, when talking about the reachability graph, we are referring to the verified reachability graph by the user, which is the result of applying the above-mentioned transformations over the restricted reachability graph. For the running use case, the verified reachability graph is shown in Figure 5.6, where the user has decided to split the aggregation paths to the *Disease* dimension and consider both as separate dimensions, *Disease* and *Secondary_Disease*.

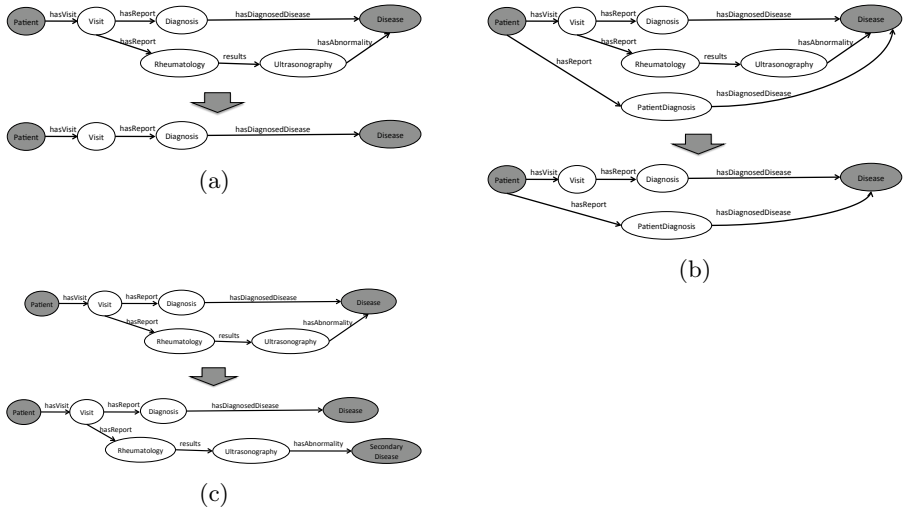

Figure 5.5: Transformations applied by the user to the reachability graph in order to select the intended aggregation paths for each MD element. The options are: a) to select only one path for the intended meaning, b) to select a subset of paths for the intended meaning and c) to split the path to account for two independent MD elements.

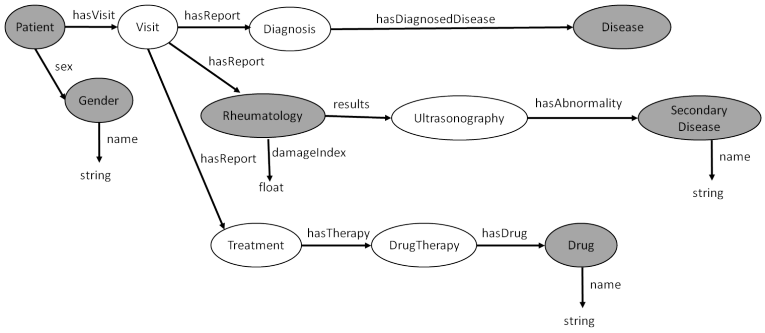

Figure 5.6: Verified reachability graph of the running use case. The disease dimension has been split in two different dimensions, *Disease* and *Secondary Disease*.

5.4.3 Finding ABox evidence efficiently

The implemented algorithm for finding aggregation paths between the subject of analysis and the MD elements combines the domain knowledge of the TBox with instance evidence from the ABox to build the aggregation paths, as the

final goal is to analyze instances.

Given a direct aggregation path (C, r_1, C'), the algorithm checks if such path has been instantiated in the ABox with the function *hasABoxEvidence*. Otherwise, it makes no sense to consider the path. This is equivalent to ask if the conjunctive query $q(x, y) := C(x) \wedge r_1(x, y) \wedge C'(y)$ is not empty.

Conjunctive query answering over DL knowledge bases has been subjected to research during the last years [28]. Actually, it is well-known that conjunctive query answering over expressive DLs is a difficult problem. However, our setting is much simpler as the distinguished variables are always individuals and the concepts C and C' that may appear in the query are restricted to be named concepts, and the role r_1 is a named role. This is a consequence of the construction of the aggregation paths in the previous algorithm.

In Section 4.2.2 we have developed a mechanism to store the ABox assertions together with the $\mathcal{LS}^-$ indexes so that we can respond to conjunctive queries about named concepts and roles efficiently. However, when constructing the reachability graph we do not require the answers but only knowing if there is any answer. Therefore, we have designed an index that is able to efficiently check if a conjunctive query has an answer. Each entry of the index is a pair (C, C') of named concepts, and has associated a list of roles $\{r_i\}$. An entry of the index indicates that the ABox contains individuals of type C that are related with individuals of type C' through the roles $\{r_i\}$.

The construction of the index is done once at the beginning by processing the ABox. The algorithm is shown in Algorithm 10. In the index, we only materialize the most specific named concepts of an individual. The algorithm processes each property assertion of the ABox and retrieves the concepts to which the subject and object of the assertions belong to. Given two sets of concepts and a role, the function *indexAux* fills the index. Then, possible inverse roles and role chains in the assertions of the ABox are handled. If a role r is defined as the inverse of another role r_1, the index also holds the entries with the order of the concepts switched and the role r_1. The processing for handling role chains is a bit more complicated as we need to temporarily keep all triples (C, r, C') where C and C' are concepts and r is part of a role chain. Then, these triples are joined with themselves and, if the result contains a chain of roles that implies some other role r_{chain}, then we must index the concept entries with the complex role r_{chain}[3].

Given a query $C(x) \wedge r_1(x, y) \wedge C'(y)$, we must check if there is an entry $e = (D, D')$ in the index such that $\mathcal{O} \models D \sqsubseteq C \wedge D' \sqsubseteq C'$. That is, we must check if any combination of all the sub-concepts of C and C' has an entry. In such case, we check if $\mathcal{O} \models r_i \sqsubseteq r_1$ with $r_i \in roles(D, D')$. That is, we check if any of the roles associated to the entry pair (D, D') is a sub-role of r_1. As both the inferred concept and property hierarchy have the $\mathcal{LS}^-$ index, the previous operations can be efficiently performed. These are equivalent to

[3]For brevity, we only show the procedure with role chains of size two, but the algorithm could be easily extended to role chains of size n

Algorithm 10 ABox evidence index

Procedure RETRIEVECONCEPT(C)
Input: T_C, T_R: ABox assertions
Output: I, an index

$I = \emptyset$
for $(a\ r\ b) \in T_R$ **do**
  $C_{set} = \Pi_{id}(\sigma_{x=a}(T_C))$
  $C'_{set} = \Pi_{id}(\sigma_{x=b}(T_C))$
  INDEXAUX(C_{set}, C'_{set}, r, I)
  if $isInverseOf(r)$ **then**
    $r_1 = removeInverse(r)$
    INDEXAUX($C'_{set}, C_{set}, r_1, I$)
if $isPartOfRoleChain(r)$ **then**
  for $C \in C_{set}$ **do**
    for $C' \in C'_{set}$ **do**
      $tmp+ = (C, r, C')$
$tmp = \Pi_{C_1, r_1, r_2, C'_2}(tmp \bowtie_{C'_1 = C_2} tmp)$
for $(C_1, r_1, r_2, C'_2) \in tmp$ **do**
  if $isRoleChain(r_1, r_2)$ **then**
    $r_{chain} = getRoleChain(r_1, r_2)$
    INDEXAUX($\{C_1\}, r_{chain}, \{C'_2\}, I$)

Procedure INDEXAUX(C_{set}, C'_{set}, r, I)
Input: C_{set}, C'_{set}, r, I
Output: I

for $C \in C_{set}$ **do**
  for $C' \in C'_{set}$ **do**
    $I[C, C'].add(r)$
return I

check if there is an index entry $e = (D, D')$ such that $id(D^{\mathcal{LS}^-}) \in int(C^{\mathcal{LS}^-})$ and $id(D'^{\mathcal{LS}^-}) \in int(C'^{\mathcal{LS}^-})$. In such case, we check if $id(r_i^{\mathcal{LS}^-}) \in int(r_1^{\mathcal{LS}^-})$ for each $r_i \in roles(D, D')$

5.5 Fact Extraction

In this section, we explain how SW data are extracted according to the obtained reachability graph and the posterior aggregation into facts. First, we present the foundations of the method and then, the implemented algorithms.

5.5.1 Foundations

The fact extraction process is concerned with two main tasks: 1) the identification and their arrangement into tuples of instances from the ABox that are logically consistent with the obtained reachability graph derived from the query and 2) the posterior aggregation of these tuples into MD points characterized by the aggregated measure values. Therefore, the notion of fact is the same as

in traditional MD analysis. A *fact* is composed by a set of aggregated measures values, described by a MD point.

The difference relies on the type of data from which facts are extracted. Traditional approaches for MD modeling have focused on the search of functional dependencies between the facts and the dimensions. However, these approaches are not be able to find any facts in complex scenarios where the traditional MD integrity constraints such as the to-one relationship between the facts and the dimensions do not apply. In contrast, we are able to extract meaningful facts from data that are not MD by nature.

First, we describe the process of extracting instance and data tuples from SW data. Then, we show how to aggregate these tuples (i.e., the raw data) to produce meaningful facts. Moreover, we identify when a set of facts is summarizable, and can thus be pre-aggregated using conventional OLAP tools. Non-summarizable facts contain duplicated information that can lead to wrong results when aggregated. However, we show how to correctly calculate the measures in such case by handling duplicities in the data and thus, providing the user with meaningful results that cannot be otherwise obtained.

Figure 5.7 shows graphically an example of an instance of *Patient* from the ontology of the running use case. We use this example to illustrate the extraction of facts.

In the following, we introduce some definitions that contribute to the extraction of facts.

Definition 5.10. *The expression* $(i_0, r_0, i_1, r_1, ..., i_n) \in Paths(i, i')$ *with* $n \geq 1$, $i_i \in I$ *and* $r_i \in R$ *is an aggregation path from instance* i *to instance* i' *if* $i_0 = i$, $i_n = i'$ *and* $\mathcal{O} \models r_j(i_j, i_{j+1})$, $0 \leq j < n$.

Definition 5.11. *An aggregation path* $p_i = (i_0, r'_0, i_1, r'_1, ..., i_n)$ *between instances* i_0 *and* i_n *is consistent with an aggregation path* $p_c = (C_0, r_0, C_1, r_1, ..., C_n)$ *if* $O \models C_i(i_i) \wedge r'_i \sqsubseteq r_i$, $0 \leq i \leq n$.

For example, the instance path $p = (PTN\_XY21, hasVisit, VISIT1, hasReport, RHEX1) \in Paths(PTN\_XY21, RHEX1)$ is an aggregation path from instance PTN_XY21 to instance $RHEX1$ and is consistent with the aggregation path $p_c = (Patient, hasVisit, Visit, hasReport, Rheumatology) \in Paths(Patient, Rheumatology)$.

The previous definition of consistency between an instance path and an aggregation path is important because facts will be composed only from instance tuples whose values are reached by instance paths that are consistent with an aggregation path of the reachability graph. In fact, the reachability graph is used as a data guide to extract the instance tuples and the posterior facts.

Now we define the notion of context. Intuitively, given the subject of analysis C_{SUB} and a pair of MD elements C_1, C_2, the context is a concept of the reachability graph that lies in between C_{SUB} and C_1, C_2, and acts as the nearest common aggregator node for C_1, C_2.

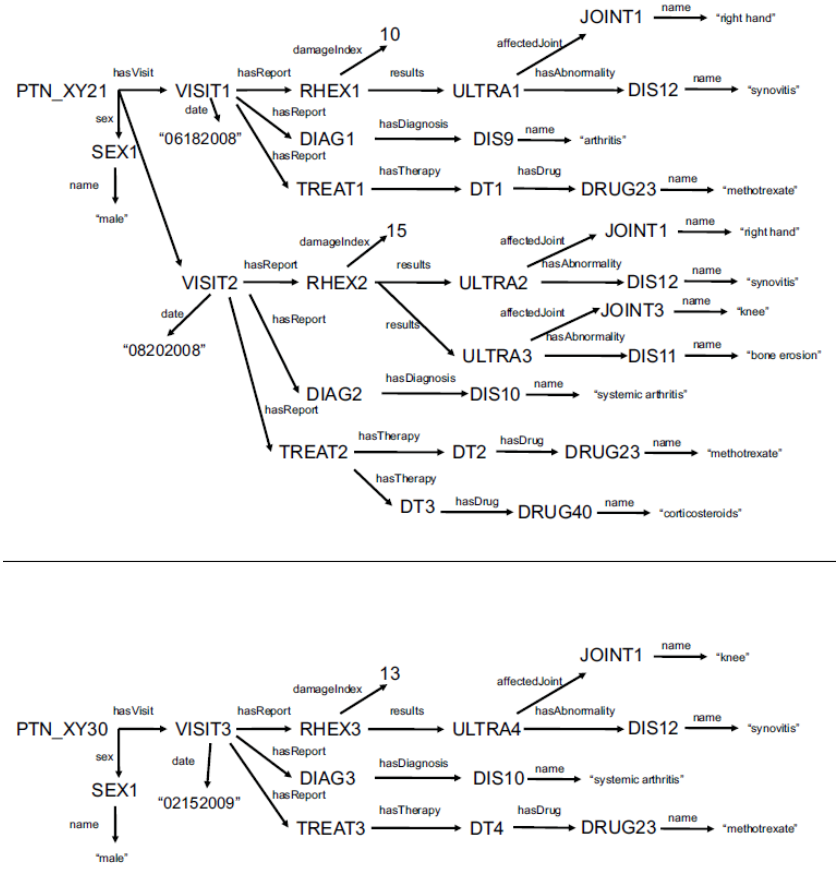

Figure 5.7: Example of *Patient* instance consistent with the ontology axioms of the running example.

Definition 5.12. *Let C_1, C_2 be concepts representing MD elements and C_{SUB} be the concept representing the subject of analysis.* $Contexts(C_1, C_2, C_{SUB}) = \bigcup_{\forall C' \in LCRC(C_1,C_2,C_{SUB})} \{C''/C'' \sqsubseteq C'\}$, *where the function $LCRC(C_1, C_2, C_{SUB})$ is the set of least common reachable concepts from C_1 and C_2 to C_{SUB} defined as follows: $C' \in LCRC(C_1, C_2, C_{SUB})$ if:*

1. *C' is common reachable concept (if one of the following condition applies):*
   - 1.1. $C' = C_{SUB} = C_1 = C_2$
   - 1.2. $C' = C_{SUB} = C_1 \wedge |Paths(C', C_2)| > 0$
   - 1.3. $C' = C_{SUB} = C_2 \wedge |Paths(C', C_1)| > 0$

1.4. $C' = C_1 = C_2 \wedge |Paths(C_{SUB}, C')| > 0$

1.5. $C' = C_{SUB} \wedge |Paths(C', C_1)| > 0 \wedge |Paths(C', C_2)| > 0$

1.6. $C' = C_1 \wedge |Paths(C_{SUB}, C')| > 0 \wedge |Paths(C', C_2)| > 0$

1.7. $C' = C_2 \wedge |Paths(C_{SUB}, C')| > 0 \wedge |Paths(C', C_1)| > 0$

1.8. $Paths(C_{SUB}, C')| > 0 \wedge |Paths(C', C_1)| > 0 \wedge |Paths(C', C_2)| > 0$

2. $\nexists E$ *that satisfying 1.,* $E \in LCRC(C_1, C_2, C_{SUB}) \wedge |Paths(C', E)| > 0$ *(*C' *is least)*

The first condition states that there must be at least one aggregation path connecting the subject of analysis C_{SUB} with the context C', another aggregation path connecting C' with C_1 and another aggregation path connecting C' with C_2. The eight cases account for the different configurations where the context coincides with some of the other concepts. The second condition states that the context concept must be the closest one in the aggregation paths to C_1 and C_2.

Identifying the different contexts of the MD elements at the conceptual level (i.e., in the TBox) is important for extracting the instance tuples correctly from the ABox and also to check the summarizability property of the derived facts. Specially important is the context that acts as aggregator node of all the MD elements. We name this context *parent context*. Figure 5.8 shows the reachability graph with the contexts for each pair of MD elements enclosed in boxes. The parent context is *Patient*, which coincides with the subject of analysis. The calculation of the contexts is trivial, as they correspond to the nearest common ancestors of the MD elements pairwise.

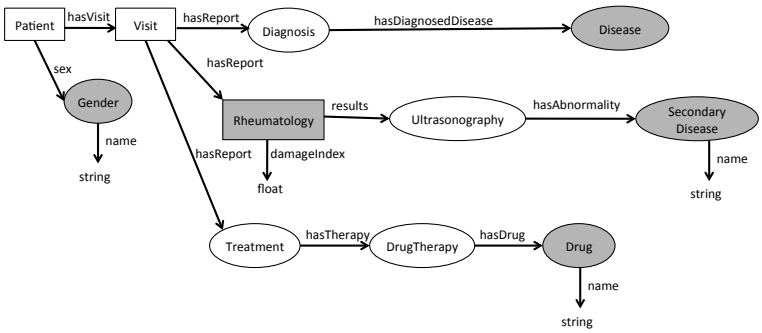

Figure 5.8: Verified reachability graph of the running use case with the context enclosed in a box.

The following example illustrates the semantics behind the notion of context. From the verified reachability graph in Figure 5.8, we see that the context

of $AvgDIndex$ and $Secondary\_Disease$ is $Rheumatology$. This means that a well-formed instance tuple should contain the measure value of the articular damage index together with a disease abnormality detected in an ultrasonography that has been taken in the same rheumatology report. It would not make sense to combine in the same tuple the value of the articular damage index taken in the first rheumatology report of the patient with a disease that has been detected and recorded in a different rheumatology report (e.g., a year later). From this reasoning, we conclude that the instances that compose an instance tuple should be extracted from the same context instances.

Now we formalize the notion of instance tuple.

Definition 5.13. *Let Q be a MD query having the associated reachability graph G. An instance tuple is a tuple of the form $it = (i_0, i_1, ..., i_n)$, $1 \leq j \leq n$, with $i_j \in I$, $n \geq |MD|$ such that:*

1. *i_0 is an instance of the parent context.*
2. *$\forall i_j \in it$, $O \models C_j(i_j) \wedge C_j \sqsubseteq D_j$, $D_j \in \{desc(elem_j)/elem_j \in MD\}$. We denote with $MD(i_j)$ to the MD element associated to the instance i_j by this condition.*
3. *$\forall i_j \in it$, $\exists p_1 \in Paths(i_{SUB}, i_j)$, $\exists p_2 \in Paths(C_{SUB}, C_j)$, $p_2 \in G$ and $C_j = desc(MD(i_j))$, such that p_1 is consistent with p_2 .*
4. *$\forall i_j, i_k \in it$, $j \neq k$, $\exists p_1 \in Paths(i_{SUB}, i_j)$, $\exists p_2 \in Paths(i_{SUB}, i_k)$ satisfying condition 2, such that if $\exists i_1 \in p_1$, $\exists i_2 \in p_2$ and $O \models C(i_1), C(i_2)$ and $C \in Contexts(C_{i_j}, C_{i_k}, C_{SUB})$, $C_{i_j} = desc(MD(i_j))$, $C_{i_k} = desc(MD(i_k))$, then $i_1 = i_2$.*

According to the previous definition, instance tuples satisfy the following conditions: 1) the parent context instance is the first element of the tuple, 2) the remaining instances in the tuple belong to a concept that is an MD element, 3) each instance of the tuple is reached by an instance path that is consistent with an aggregation path derived from the MD element associated to the instance and belongs to the reachability graph and 4) the instance paths associated to the instances in the tuples share pairwise the same context instances.

The instance tuples are the raw dimension and measure values extracted from SW data, were aggregations have not yet been performed, that is, measure values have not yet been calculated.

Table 5.1 shows the instance tuples generated for the running use case shown in Figure 5.7. The first column is the tuple id, only for reference purposes. The second column is the parent context instance. The following four columns are the dimensions and the last two columns are the extracted measure values previous to aggregation. As an example, we illustrate how the first tuple of the table satisfies all the requirements of the instance tuple definition. The first and second conditions are trivial and are satisfied. The third condition is also

t_id	Ctxt	Gender	Disease	Sec_Disease	Drug	ArtDamIndex	Patient
1	PTN_XY21	SEX1	DIS9	DIS12	DRUG23	RHEX1	PTN_XY21
2	PTN_XY21	SEX1	DIS10	DIS12	DRUG23	RHEX2	PTN_XY21
3	PTN_XY21	SEX1	DIS10	DIS12	DRUG40	RHEX2	PTN_XY21
4	PTN_XY21	SEX1	DIS10	DIS11	DRUG23	RHEX2	PTN_XY21
5	PTN_XY21	SEX1	DIS10	DIS11	DRUG40	RHEX2	PTN_XY21
6	PTN_XY30	SEX1	DIS10	DIS12	DRUG23	RHEX3	PTN_XY30

Table 5.1: Instance tuples generated for the running example in Figure 5.7.

satisfied and we show in Table 5.2, for each instance in the tuple, the instance path and the aggregation path belonging to the reachability graph to which the instance path is consistent. Finally, we check in the previous table the context instances pairwise and enclose them in boxes.

Instance MD element	Instance path Aggregation Path
SEX1 Gender	(PTN_XY21 , sex, SEX1) (Patient, sex, Gender)
DIS9 Disease	(PTN_XY21 , hasVisit, VISIT1 , hasReport, DIAG1, hasDiagnosedDisease, DIS9) (Patient, hasVisit, Visit, hasReport, Diagnosis, hasDiagnosedDisease, Disease)
DIS12 Sec. Disease	(PTN_XY21 , hasVisit, VISIT1 , hasReport, RHEX1 , results, ULTRA1, hasAb., DIS12) (Patient, hasVisit, Visit, hasReport, Rheumatology, results, Ultrasonography, hasAb., Disease)
DRUG23 Drug	(PTN_XY21 , hasVisit, VISIT1 , hasReport, TREAT1, hasTherapy, DT1, hasDrug, DRUG23) (Patient, hasVisit, Visit, hasReport, Treatment, hasTherapy, DrugTherapy, hasDrug, Drug)
RHEX1 ArtDamIndex	(PTN_XY21 , hasVisit, VISIT1 , hasReport, RHEX1) (Patient, hasVisit, Visit, hasReport, Rheumatology)
PTN_XY21 Patient	(PTN_XY21) (Patient)

Table 5.2: Instances of the first instance tuple in Table 5.1 with the respective instance paths and aggregation paths. The instances enclosed in boxes are the context instances.

Now we define the data tuples as the readable projection of the instance tuples.

Definition 5.14. *A data tuple associated to an instance tuple* $it = (i_0, i_1, ..., i_n)$ *is a tuple of the form* $dt = (i_0, s_1, ..., s_n)$, $1 \leq j \leq n$, *with* $i_0 \in I$ *and* $s_j \in$ *datatype such that:*

1. i_0 *is the parent context instance.*
2. $s_j = \Pi_p(i_j)$, $i_j \in it$ *if* $p = proj(MD(i_j))$ *is not empty.*
3. $s_j = i_j$, $i_j \in it$ *otherwise.*

A data tuple is the result of projecting the instances of an instance tuple over the datatype properties specified by the user for each MD element associated. In case the MD element has no projection associated, we keep the instance URI

in the data tuple. Following with the running use case, Table 5.3 shows the data tuples derived from the instance tuples in Table 5.1.

Notice that we take into consideration that the projection can be multivalued (i.e., more than one value can be accessed with the same data property in an instance). In this case multiple data tuples will result from one instance tuple.

t_id	Ctxt	Gender	Disease	Sec_Disease	Drug	DI	Patient
1	PTN_XY21	Male	arthritis	synovitis	methotrexate	10	PTN_XY21
2	PTN_XY21	Male	systemic arthritis	synovitis	methotrexate	15	PTN_XY21
3	PTN_XY21	Male	systemic arthritis	synovitis	corticosteroids	15	PTN_XY21
4	PTN_XY21	Male	systemic arthritis	bone erosion	methotrexate	15	PTN_XY21
5	PTN_XY21	Male	systemic arthritis	bone erosion	corticosteroids	15	PTN_XY21
6	PTN_XY30	Male	systemic arthritis	synovitis	methotrexate	13	PTN_XY30

Table 5.3: Data tuples generated from the instance tuples in Table 5.1 by following the running example in Figure 5.7.

Finally, facts are extracted from data tuples by collapsing the data tuples that share the same dimension values and applying the corresponding aggregation function to the measure values.

Definition 5.15. *A fact is a MD point* $f = (d_1, ..., d_k, m_{k+1}, ..., m_n)$ *where* d_i *are dimension values and* m_i *are aggregated measure values extracted from a set of data tuples* $s_f = \{(i_0^j, s_1^j, ..., s_n^j)\}_{1 \leq j \leq n}$ *that satisfies the following conditions:*

1. $d_i = s_i^j$, $1 \leq j \leq n$, $1 \leq i \leq k$
2. $m_i = AGG(\{s_i^j\}_{1 \leq j \leq n,\ k+1 \leq i \leq n})$ *such that* $AGG = agg(MD_{k+i})$

The first condition expresses the equality of the dimension values. The second condition expresses the aggregation of the measure values that share the same dimension values through the aggregation function defined for that measure, which corresponds to the aggregation function of the $k + i$ MD element. Notice that if the aggregation function of the previous definition is applied to non-summarizable data tuples, the resulting facts may contain wrong aggregations. Next, we define the summarizable property for the data tuples.

Property 5.1. *A set of data tuples gives rise to a set of summarizable facts iff there is a one-to-one relation between the parent context and the dimensions.*

The previous property means that each data tuple has to have associated a different parent context. In the example of Table 5.3 we observe that tuples 1-5 have the same parent context associated. Therefore, these tuples contain duplicated information and, when computing facts, correct aggregations cannot be ensured.

In spite of not being able to use pre-aggregation techniques over non-summarizable facts, we can still provide correctly aggregated facts to the user

by handling the duplicated information in the data tuples when calculating the aggregations. This depends, however, on the aggregation function applied. Next, we define how to correctly calculate aggregated data that is not summarizable by taking into account duplicated information.

Definition 5.16. *Given a fact f with $s_f = \{(i_0^j, s_1^j, ..., s_n^j)\}_{1 \leq j \leq n}$ being the data tuples associated, the different aggregation functions over the i^{th} measure are defined as follows:*

1. $SUM_i = \sum_{j=1}^{n} \frac{s_i^j}{COUNT(i_0^j, s_f)}$
2. $COUNT_i = \sum_{j=1}^{n} \frac{COUNT(s_i^j)}{COUNT(i_0^j, s_f)}$
3. $MIN_i = MIN(s_i^j),\ 1 \leq j \leq n$
4. $MAX_i = MAX(s_i^j),\ 1 \leq j \leq n$

Notice that for the aggregation functions SUM and $COUNT$ we have to apply a correction factor to each measure, which is given by the number of duplicated parent contexts in the set of measure values associated to the fact. This way, we can correctly deal with duplicated information and present correct results to the user. The AVG aggregation function can be easily derived by maintaining the SUM and $COUNT$. The MIN and MAX aggregation functions are not affected by duplicated information.

Next, we show how the data tuples in Table 5.4 are aggregated into facts.

t_id	Gender	Disease	Sec_Disease	Drug	ArtDamIndex					Patient
					SUM	C	AVG	MIN	MAX	COUNT
1	Male	arthritis	synovitis	metho	10	1	10	10	10	1
2,6	Male	sys arthritis	synovitis	metho	15+13	1+1	14	13	15	2
3	Male	sys arthritis	synovitis	cortico	15	1	15	15	15	1
4	Male	sys arthritis	bone erosion	metho	15	1	15	15	15	1
5	Male	sys arthritis	bone erosion	cortico	15	1	15	15	15	1

Table 5.4: Facts generated from the data tuples in Table 5.3 characterized by four dimensions.

From the resulting facts, we observe that the fact in the second row is the result of collapsing data tuples 2 and 6. As these two tuples have different parent contexts the aggregations are right. Next, we show another example where the data tuples are aggregated by *Gender*, *Disease* and *Sec_Disease*. That is, we remove the dimension *Drug*. Notice that we cannot use the pre-aggregated results of Table 5.4, as these facts are not summarizable. Therefore, we build the new facts from the base data tuples. Table 5.5 shows the results.

In this case, the second fact shows the aggregated results for data tuples 2, 3 and 6. However, tuples 2 and 3 share the same parent context, therefore, we must apply the correction factor to these measure values. The same occurs

t_id	Gender	Disease	Sec_Disease	ArtDamIndex					Patient
				SUM	COUNT	AVG	MIN	MAX	COUNT
1	Male	arthritis	synovitis	10	1	10	10	10	1
2,3,6	Male	sys arthritis	synovitis	15/2+15/2+13	1/2+1/2+1	14	13	15	2
4,5	Male	sys arthritis	bone erosion	15/2+15/2	1/2+1/2	15	15	15	1

Table 5.5: Facts generated from the data tuples in Table 5.3 characterized by three dimensions.

with the third fact, which is the result of aggregating data tuples 4 and 5 and these tuples have the same parent context.

The notion of duplicate facts has been previously investigated in other scenarios. For example, the work in [116] acknowledges the semantic problems that may arise when integrating a new, external dimension into an OLAP cube. The authors devise three solutions when the external dimension returns more than one value to "decorate" the cube: 1) use an arbitrary value, 2) concatenate all different values or 3) use all the values thereby creating duplicated facts. Our approach resembles the third option but we do handle duplicated information when aggregating data so that correct results are obtained.

5.5.2 Implemented algorithms

The process to extract facts is composed by two main steps: instance and data tuples extraction, and aggregation of data tuples into facts. Here, we present the developed algorithms for these tasks.

To obtain instance and data tuples we process the reachability graph. Recall that each edge of the reachability graph represents a direct aggregation path (C, r_1, C'). In order to retrieve triples that instantiate such path we need to ask the conjunctive query $q(x, y) := C(x) \wedge r_1(x, y) \wedge C'(y)$ over the ontology. Therefore, the algorithm processes the reachability graph in depth-first order and joins the instantiated triples of the successive direct aggregation paths. The process is shown in Algorithm 11. We make use of the efficient ABox querying mechanism presented in Section 4.2.2. The extracted triples, are recursively joined until the graph is completely processed. The obtained tuples are then projected over the attributes that represent MD elements, obtaining this way the instance tuples. Finally, the obtained tuples can be also projected over the projection attributes specified by the user for each MD element, $proj(MD_i)$, obtaining this way the data tuples.

Recall that the expressivity of the DL expression to which the user MD elements are mapped is that of a star-shaped conjunctive query, where the central node is a named concept and role restrictions over this concept are applied. Therefore, we can efficiently extract only the tuples whose MD elements (i.e., subject of analysis, dimensions and measures) satisfy the restrictions stated by the user. For simplicity of exposition we use $retrieve(C)$ in the algorithm but C can be a star-shaped conjunctive query, which would imply to expand C to

its components and process each of them with the provided functions for query answering.

Algorithm 11 Tuples extraction

Procedure TuplesExtraction(G)
Input: G, reachability graph
Output: T, table of data tuples
1: $T = \emptyset$
2: $C = G.rootNode()$
3: $T = retrieve(C)$
4: $T =$TuplesExtractionRec(T, C, G)
5: $IT = \Pi_{(desc(MD_1),...,desc(MD_n))_{1 \leq i \leq |MD|}}(T)$
6: **return** $\Pi_{(proj(MD_1),...,proj(MD_n)_{1 \leq i \leq |MD|})}(T)$ ▷ assuming $proj(c)$ is in T

Procedure TuplesExtractionRec(T, C, G)
Input: T, C, G
Output: T
1: $edges = G.outGoingEdges(C)$
2: **for** $(C, R, C') \in edges$ **do**
3: $T = T \bowtie retrieve(R)$
4: $T = T \bowtie retrieve(C')$
5: $T =$TuplesExtractionRec(T, C', G)
return T

The second task consists in aggregating data tuples into facts. We show the algorithm expressed in relational algebra in Algorithm 12. The extraction of facts consists in grouping the data tuples by the dimensions and aggregating the measure values. As we also deal with non-summarizable facts, the aggregation functions applied to the measures are those specified by the user for each measure but they are applied according to the Definition 5.16 to take into account duplicities.

Algorithm 12 Fact extraction

Procedure FactExtraction(T)
Input: T, table of data tuples, $MD = (D_1, ..., D_n, M_1, ..., M_t)$
Output: F, facts
1: **return** $D_1, ..., D_n\ G_{f_1(M_1),...,f_n(M_t)}(T)$ where $f_i = agg(MD_{n+i})$

All the relational algebra operations that appear in the algorithms for fact extraction have been implemented using MySQL as relational back-end, assuming that the data does not fit in memory. Nevertheless, for small data sets, these algorithms could be easily implemented using main memory data structures.

5.6 Dimension Extraction

In traditional DW, the schema of dimension hierarchies is usually driven by the schema of the data sources used to populate the DW. Typical dimension

hierarchies are usually static, fixed and pre-defined, for example, the *time*, *place* or *product category*. These dimensions usually have few and well-established dimension levels, and the possibility of dynamically modifying the hierarchies is limited or none.

The SW scenario offers new possibilities where richer dimension hierarchies can be constructed ad hoc by taking into account the user requirements and from truly semantic relations found among the ontology concepts. This richness of knowledge introduces, however, new challenges regarding the construction of the hierarchies, as extracting dimension hierarchies without any restrictions could lead to too large and overwhelming hierarchies that are not suitable for MD analysis.

Our approach aims at extracting dimension hierarchies from ontologies in an automatic way, releasing the user from the task of having to specify each dimension level and category. The dimension hierarchies are roughly specified by the user in terms of an ontological concept. After that, the hierarchical dimensions are gathered from the concept hierarchy inferred from the ontologies at hand.

Traditionally, the dimension hierarchies have been restricted to a determined shape in order to perform properly OLAP operations. That is, extracted hierarchies comply with a series of constraints to ensure summarizability [54]. However, we model a dimension hierarchy as a directed acyclic graph of nodes, where nodes are sub-concepts of the user conceptual description for the dimension, and the edges correspond to the semantic relations (e.g., "is-a" relationships) between the nodes. This decision has been taken in attempt to capture the rich hierarchies underlying SW data that cannot be otherwise used. Therefore, we do not transform the dimension hierarchies into summarizable ones but only re-shape the dimensions to favor both dense regions and good aggregation nodes, while preserving the semantics as much as possible. If summarizability is a must, there are several other approaches focused on obtaining summarizable hierarchies [119, 7]. We focus on the hierarchical relationships although other kind of relationships could also be used (e.g., transitive properties, property compositions, etc.) and have actually been treated in the literature [126].

The following sections explain the process of extracting hierarchical dimensions from the knowledge encoded in the ontologies that fit the base dimension values of the facts and favor aggregation. In Section 5.6.1, we take advantage of the modularity techniques described in Chapter 4 to select the concepts that are candidates to be part of the dimension hierarchies. Then, in Section 5.6.2, we present two alternative algorithms to select concepts for aggregation based on measures that favor a good classification and distribution of the base dimension values.

5.6.1 Dimension Modules

The first step to extract each hierarchical dimension D_i consists of selecting the part of the ontology that can be involved in it. Let $Sig(D_i)$ be the set of most specific concepts of the instances participating in the instance tuples for dimension D_i (see Definition 5.13).

We define the dimension module $M_{D_i} \subseteq \mathcal{O}$ as the upper module of the ontology $\mathcal{O}$ for the signature $Sig(D_i)$. We define upper modules in the same way as in [60],that is, by applying the notion of conservative extension. Thus, an upper module M of $\mathcal{O}$ for the signature Sig is a sub-ontology of $\mathcal{O}$ such that it preserves all the entailments over the symbols of Sig expressed in a language $\mathcal{L}$, that is, $O \models \alpha$ with $Sig(\alpha) \subseteq Sig$ and $\alpha \in \mathcal{L}$ iff $M_{D_i} \models \alpha$. For hierarchical dimensions we only need to preserve entailments of the form $C \sqsubseteq D$ and C *disjoint* D, C and D being named concepts. This kind of upper modules can be extracted very efficiently over very large ontologies by using any of the modularity approaches proposed in Chapter 4.

Let $TAX(M_{D_i})$ be the inferred taxonomy for the extracted module M_{D_i}. This taxonomy can be represented as the DAG (V, E), where V contains one node for each concept in M_{D_i}, and $c_i \rightarrow c_j \in E$ if c_i is a direct ancestor of c_j. This taxonomy could be directly used as dimension hierarchy to aggregate the dimension values. However, $TAX(M_{D_i})$ is usually an irregular, unbalanced and non-onto hierarchy, which makes it not suitable for OLAP operations. Therefore, we transform it into a more regular structure. However, this transformation is also required to preserve as much as possible the original semantics of the concepts as well as to minimize the loss of information (e.g., under-classified concepts).

Several approaches have been proposed in the literature to transform hierarchies in order to satisfy the desirable properties for OLAP aggregations. Former work about transforming OLAP hierarchies [119] proposed the inclusion of fake nodes and roll-up relationships to avoid incomplete levels and double counting issues. Normalization is also proposed as a way to solve non-onto hierarchies [81]. These strategies however are not directly applicable to ontology taxonomies for two reasons: the number of added elements (i.e., fake nodes or intermediate normalized tables) can overwhelm the size of the original taxonomy and, the semantics of concepts can be altered by these new elements. The work in [29] proposes a clustering-based approach to reduce taxonomies for improving data tables summaries. This method transforms the original taxonomy to a new one by grouping those nodes with similar structures and usage in the tuples. However, this method also alters the semantics of the symbols as new ones are created by grouping original ones. The recent approach in [7] extends OLAP operations in order to be able to incorporate semantic dimensions while preserving summarizability.

In the next section, we propose two algorithms to select concepts of the hierarchy that better classify and distribute the base dimension values.

5.6.2 Generation of hierarchies

Before going into the details of the proposed algorithms, we analyze why $TAX(M_{D_i})$ presents such an irregular structure that makes it necessary to devise new methods to either transform or select parts of the hierarchy that are more suitable for OLAP analysis. The usage of symbols in $Sig(D_i)$ can be very irregular due to the "popularity" of some symbols (i.e., Zipf law), which implies that few symbols are used very frequently whereas most of them are used few times. As a result, some parts of the taxonomy are more used than others, affecting to both the density (few dense parts and many sparse parts) and the depth of the taxonomy (few deep parts). A direct consequence is that some concepts in $Sig(D_i)$ are covered by many spurious concepts which are only used once, and therefore are useless for aggregation purposes. So, our main goals should be to identify dense regions of the taxonomy and to select nodes that best classify the concepts in $Sig(D_i)$. For this goal, we propose a series of measures to decide which nodes deserve to participate in the final hierarchy.

The first measure we propose to rank the concepts in $TAX(M_{D_i})$ is the *share*:

$$share(n) = \prod_{n_i \in ancs(n)} \frac{1}{|children(n_i)|} \tag{5.9}$$

where $ancs(n)$ is the set of ancestors of n in $TAX(M_{D_i})$, and $children(n)$ is the set of direct successors of n in the taxonomy.

The idea behind the *share* is to measure the number of partitions produced from the root till the node n. The smaller the share the more dense is the hierarchy above the node. In a regular balanced taxonomy the ideal share is $S(n)^{depth(n)}$, where S is the mean number of children that the ancestor nodes of n have. We can then estimate the ratio between the ideal share and the actual one as follows:

$$ratio(n) = \frac{S(n)^{depth(n)}}{share(n)} \tag{5.10}$$

Thus, the greater the ratio, the better the hierarchy above the node is.

The second ranking measure we propose is the *entropy*, which is defined as follows:

$$entropy(n) = \Sigma_{n_i \in children(n)} P_{sig}(n, n_i) \cdot log(P_{sig}(n, n_i)) \tag{5.11}$$

$$P_{sig}(n, n_i) = \frac{coveredSig(n_i)}{coveredSig(n)} \tag{5.12}$$

where $coveredSig(n)$ is the subset of $Sig(D_i)$ whose members are descendants of n.

The idea behind the entropy is that good classification nodes are those that better distribute the signature symbols among its children. Similarly to decision trees and clustering quality measures, we use the entropy of the groups derived from a node as the measure of their quality.

In order to combine both measures we just take the product of both measures:

$$score(n) = entropy(n) \cdot ratio(n) \tag{5.13}$$

Both algorithms proposed to generate dimension hierarchies consist of selecting a set of "good" nodes from the taxonomy based on the previous measures, and then re-constructing the hierarchy by applying the transitivity property of the subsumption relationship between concepts. Algorithm 13 presents a *global* approach, which consists of selecting nodes from the nodes ranking ordered by the score until either all signature concepts are covered or there are no more concepts with a score greater than a given threshold (usually zero). Alternatively, we propose a second approach in Algorithm 14, the *local* approach, which selects the best ancestors of each concept leaf of the taxonomy. In both approaches, the final hierarchy is obtained by extracting the *spanning tree* that maximizes the number of ancestors of each node from the resulting reduced taxonomy.

One advantage of the local approach is that we can further select the number of levels up to each signature concept, defining so the hierarchical categories for the dimensions. However, this process is not trivial and we decided to leave it for future work. Each method favors different properties of the generated hierarchy. If the user wants to obtain a richer view of the hierarchy, she must select the global one. Instead, if the user wants a more compact hierarchy (e.g., few levels) then she must select the local one.

It is worth mentioning that the step for reconstructing the taxonomy is efficiently performed by using the ontology index and the modularity strategies proposed in Chapter 4.

Algorithm 13 Global approach for dimension hierarchy generation

Procedure DimExtractionGlobal(M_{D_i},$Sig(D_i)$)
Input: M_{D_i}, the upper module, $Sig(D_i)$, the signature for dimension D_i.
Output: H_i, a hierarchy for dimension D_i

1: Let L_{rank} be the list of concepts in M_{D_i} ordered by $score(n)$ (highest to lowest);
2: Let $Fragment = \emptyset$ be the nodes set of the fragment to be built.
3: **repeat**
4: pop a node n from L_{rank}
5: add it to $Fragment$
6: **until** $score(n) \leq 0$ or $\bigcup_{n \in Fragment} coveredSig(n) == Sig(D_i)$
7: Let $NewTax$ be the reconstructed taxonomy for the signature $Fragment$
8: **return** $spanningTree(NewTax)$

Algorithm 14 Local approach for dimension hierarchy generation

Procedure DimExtractionLocal(M_{D_i},$Sig(D_i)$)
Input: M_{D_i}, the upper module, $Sig(D_i)$, the signature for dimension D_i.
Output: H_i, a hierarchy for dimension D_i

1: Let L_{leaves} be the list of leaf concepts in M_{D_i} ordered by $score(n)$ (highest to lowest);
2: Let $Fragment = \emptyset$ be the nodes set of the fragment to be built.
3: **for** $c \in L_{leaves}$ **do**
4: Set up $L_{ancs}(c)$ with the ancestors of c ordered by $score(n)$
5: **for** $n_a \in L_{ancs}(c)$ **do**
6: **if** there is no node $n_2 \in L_{ancs}(c)$ such that $score(n_2) \leq score(n_a)$ and $order(n_2) \leq order(n_a)$ **then**
7: **if** $n_a \notin Fragment$ **then**
8: add n_a to $Fragment$
9: Let $NewTax$ be the reconstructed taxonomy for the signature $Fragment$
10: **return** $spanningTree(NewTax)$

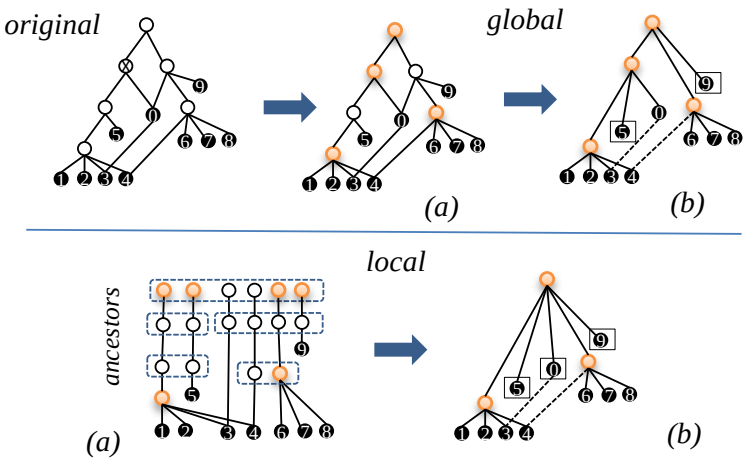

Figure 5.9: Example of local and global methods: (a) node selection and (b) hierarchy reconstruction. Dashed edges in (b) are removed in the final spanning tree. Nodes inside squares in (b) are those that change its parent in the resulting dimension.

5.7 Evaluation

The experimental evaluation performed in this section has three main objectives. First, we are concerned with demonstrating the viability and scalability of the algorithm for generating aggregation paths from the user MD query over different ontologies. Then, we perform experiments to demonstrate the scalability of the fact extraction process. Finally, we evaluate the two proposed

methods for generating dimensions from the ontology knowledge by measuring the quality of the resulting hierarchies.

The experiments were performed on a Linux server with 8 1.86GHz Intel(R) Xenon(R) processors, 33GB of RAM, Ubuntu 10.04.4, Kernel Linux 2.6.32-45.

5.7.1 Datasets

The experiments have been performed over several SW data sets that have different features regarding aspects such as size, structure and expressivity, in order to subject our methods to this diversity. Table 5.6 shows the main features of the selected ontologies.

HeCOnto: We have generated this OWL dataset synthetically from the features identified from a set of real patients. The Tbox template has been carefully designed following the structure of the medical protocols defined in the HeC project for rheumatic patients. Moreover, domain concepts are taken from UMLS. With the previous setup, we are able to generate synthetic instance data of any size and with the intended structural variations to account for heterogeneity and optional values of semantic annotations.

BioPax: BioPAX (Biological Pathway Exchange) is a RDF/OWL-based ontology to represent biological pathways at the molecular and cellular level. Its major use is to facilitate the exchange of pathway data. Through BioPAX, millions of interactions organized into thousands of pathways across many organisms, from a growing number of sources, are available. Thus, large amounts of pathway data are available in a computable form to support visualization, analysis and biological discovery.

LUBM (10 univ): LUBM (Lehigh University Benchmark) is probably the best known synthetic framework for evaluating SW systems. It provides an OWL ontology about the university domain (i.e., universities, staff, students and so forth) and code for populating variable-sized corpora of instance data using this terminology.

SwetoDBLP: SwetoDblp is a large-size ontology focused on bibliography data of Computer Science publications where the main data source is DBLP. The ontology is composed mainly by RDFS axioms and huge bulks of instance data are available for download.

Ontology	# C	# OP	# DP	# I	Expressivity
HeCOnto	1779	124	12	605,420	$\mathcal{ALCHI}(\mathcal{D})$
BioPax	41	33	37	62,021	$\mathcal{ALCHN}(\mathcal{D})$
LUMB (10 univ)	43	25	7	$\simeq$ 1.3M	$\mathcal{ALEHI}(\mathcal{D})$
SwetoDBLP	60	16	30	$\simeq$ 1.5M	$\mathcal{ALH}(\mathcal{D})$

Table 5.6: Ontologies and their features.

For the experiments, it was hard to find SW data with both large TBox

and ABox. Instance data subject to analysis usually have a small TBox that models the domain. Ontologies with very large TBoxes do not usually have ABox. This is the case, for example, of NCI Thesaurus[4], which has an OWL version over which we have applied our methods. However, these kinds of large ontologies are not suitable for the MD analysis method that we propose because they have the ABox internalized into the TBox. An interesting line of research, which is out of the scope of this thesis, is to devise how to separate the instance data from an ontology such as NCI so that our MD analysis methods can be applied.

5.7.2 Aggregation Paths

The experiments performed in this section are concerned with demonstrating the feasibility of extracting aggregation paths and, in particular, the importance of restricting the general aggregation paths into the the three types of aggregation paths defined. Recall that an aggregation path exists between a pair of concepts (C, C') if there exists at least one interpretation where at least one object of C is connected to one object of C' by roles. According to this definition, there can exist thousands of allowed aggregation paths between a pair of concepts, specially, if the ontology is poorly defined. The three types of aggregation paths defined in this thesis are meant to constrain the general definition of aggregation path so that the user is presented with a selection of interesting aggregation paths for analysis purposes.

The algorithm for extracting the reachability graph that suits an MD query is based on the extraction of direct aggregation paths. In Section 5.4.1 we show an algorithm to precompute the direct aggregation paths of Group 1 and Group 2. The cost of this algorithm is $O(rn^2)$, as the direct aggregation paths are extracted from each concept definition and are propagated to the descendants. Table 5.7 shows the number of direct aggregation paths precomputed for the selected ontologies. Even though in the worst case this precomputation could lead to a high amount of direct aggregation paths, the empirical evaluation demonstrates that this amount is usually very low for different ontologies.

Ontology	**Group 1**	**Group 2**
HeCOnto	5875	0
BioPax	42	228
LUBM (10 univ)	26	38
SwetoDLBLP	0	475

Table 5.7: Number of direct aggregation paths of Group 1 and Group 2

Now, we are concerned with the potential number of aggregation paths of

[4]NCI Thesaurus: `http://ncit.nci.nih.gov/`

each group that can be generated from the previous direct aggregation paths. Given the user MD query, which is expressed in terms of the ontology, the aim is to discover aggregation paths from the subject of analysis to the dimensions and measures. Therefore, the number and type of potential aggregation paths depends greatly on the structure underlying the ontology axioms and the MD query expressed by the user. In the following experiment, we simulate the user MD queries by taking as potential subject of analysis each concept from the ontologies and extract all possible aggregation paths from there, that is, we consider that each reached concept is a potential dimension.

The results of this experiment are shown in Table 5.8, where the objective is to count the number of aggregation paths of each group that can be discovered from a concept. The first column shows the average. We calculate it by adding the number of aggregation paths only from subject concepts that have connectivity greater than zero and dividing by the number of such subject concepts. This is done to avoid biased results in ontologies where there are a great number of leaf concepts (i.e., concepts with no connectivity). The second column shows the maximum number of aggregation paths from a subject concept and the third column shows the concept that holds this maximum. Only for aggregation paths of Group 3 we use the ABox evidence to prune the expansion of aggregation paths through sub-concepts, as the number of aggregation paths of the other groups is already low without this pruning.

	Group 1			**Group 2**			**Group 3**		
Ontology	Avg.	Max.	Max. C	Avg.	Max.	Max. C	Avg.	Max.	Max. C
HeCOnto	3.848	14	Patient	3.848	14	Patient	35.972	2971	Patient
BioPax	1.863	3	Pathway	8.531	46	Pathway	59.233	932	Pathway
$LUBM_{10}$	1.176	2	Chair	1.45	6	Chair	7.15	16	Chair
Sweto DBLP	0	0	-	8.93	18	Inbook	8.93	18	Inbook

Table 5.8: Average and maximum **number** of aggregation paths from a subject concept.

Regardless of the ontology, we observe that the average number of aggregation paths of Group 1 is very low, which proves that this type of aggregation path is too restrictive and hardly found in the ontologies. The number of aggregation paths on average of Group 2 is specially higher in ontologies with RDFS constructs (i.e. BioPax and SwetoDBLP) but still low. Finally, the average number of aggregation paths of Group 3 varies from one ontology to another. While for LUBM and SwetoDBLP ontologies the number is manageable, for BioPax and HeCOnto the number of aggregation paths on average is a bit high to be shown to the user. This is mainly due to the big number of specialization relationships, as in these type of paths the properties are propagated not only to the range concept but also to every sub-concept, making each sub-concept susceptible of expanding a new branch.

The concepts whose number of aggregation paths is maximum are good candidates to be subject of analysis, as they present relations with many other

concepts through aggregation paths, and that makes them good aggregators.

Table 5.9 shows the results from another experiment, where we measure the average depth of the aggregation paths. As previously, for each type of aggregation path, the first column shows the average depth, the second one the maximum depth and the concept that hold this maximum is shown in the third column.

	Group 1			Group 2			Group 3		
Ontology	Avg.	Max.	Max. C	Avg.	Max.	Max. C	Avg.	Max.	Max. C
HeCOnto	1.174	12	Patient	1.174	12	Patient	1.194	14	Patient
BioPax	4.777	15	Modulation	6.133	21	Modulation	16.033	199	Pathway
$LUBM_{10}$	1	1	Chair	1.15	3	ResearchAss.	2.15	3	Dean
SwetoDBLP	0	0	-	5.677	12	Inbook	5.677	12	Inbook

Table 5.9: Average and maximum **depth** of aggregation paths from a subject concept.

The depth of the aggregation paths of each type shows a similar pattern as previously. For Group 1 and Group 2, the average depth of the paths is relatively low in all the ontologies, whereas the depth of paths of Group 3 becomes unmanageable for BioPax. The TBox of BioPax is specially complex in term of semantics, with concepts related by many properties and with cycles, which explains an average depth so high.

Both the number and the depth of the aggregation paths have been calculated by obviating cycles in the ontologies, as a cycle implies that both the number and depth of aggregation paths are infinite. Therefore, when there is a cycle, both the number and depth of paths of that concept is updated but the concept is not again explored.

5.7.3 Fact extraction

The evaluation of the fact extraction process is mainly concerned with time complexity issues regarding the generation of the facts from the obtained reachability graph. In particular, we are interested in measuring how both the number of elements of the user's MD query and the size of the instance store affect the fact table generation process. To perform these experiments, we have selected the HeCOnto ontology. Figure 5.10 presents how the number of elements in the user's MD query (i.e., number of dimensions and measures) affects the time performance to generate the fact table. For the experiment setup we have preselected a set of eleven candidate dimensions and measures from the ontology and have computed all the reachability graphs and derived fact tables that can be generated from all subsets of these eleven MD elements. The total number of possible fact tables is $2^{11} = 2048$. Then, we have organized the fact tables according to the number of MD elements (i.e., number of dimensions and measures), from two MD elements to eleven in the x axis. Axis y shows the time performance in seconds. Each boxplot in the figure shows the variance in time

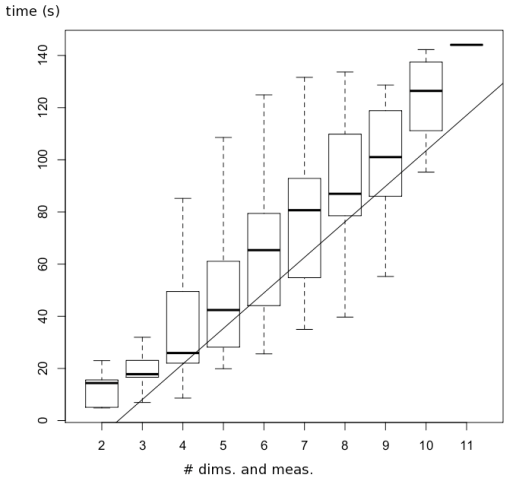

Figure 5.10: Fact table generation performance w.r.t. the number of dimensions and measures involved in the user's MD query.

between fact tables having the same number of MD elements. The explanation for this variance is that different MD configurations of the same size may obtain very different reachability graphs depending on their structural dependencies. Therefore, the number of instances to be processed and the number of joins are different, resulting in different processing times. However, we observe that the time complexity increases linearly w.r.t. the number of dimensions and measures of the user's MD query, which proves the scalability and efficiency of the approach.

On the other hand, we are also concerned with how the size of the instance store affects the generation of the fact tables. To evaluate this, we have selected from the previous experiment one of the smallest (i.e., two elements) and the largest (i.e., eleven elements) user's MD query. For these two MD queries we measure the required time to create the respective fact tables with instance stores of different sizes, ranging from 100 to 3,000 complex instances of type *Patient*. Figure 5.11 illustrates the results. Notice that axis x measures the number of subject instances, although the number of total instances in the store ranges from a thousand to more than half million instances. Axis y shows the time performance in seconds. For both MD queries, the time performance is linear w.r.t. the size of the instance store, which means the proposed method is scalable.

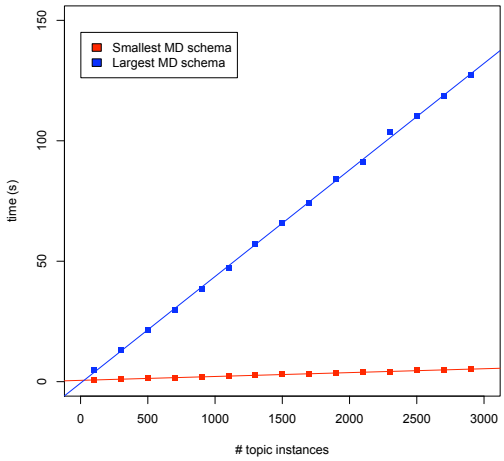

Figure 5.11: Fact table generation performance w.r.t. the size of the instance store.

5.7.4 Dimension extraction

The objective of this section is to evaluate the quality of the dimension hierarchies extracted by the methods proposed in Section 5.6 (i.e., global vs. local). Each of the dimension hierarchies is generated from the conceptual description of each dimension in the user's MD query. Recall that the process is automatic and the hierarchical relations between the dimension values that compose the hierarchy are truly semantic relations encoded in the ontologies. Our goal is to generate dimension hierarchies that have good aggregation power and also preserve as much as possible the original semantics of the involved concepts. In order to measure the quality of the dimension hierarchies, we have adapted the measures proposed in [29] for evaluating table summarization, which also aims at minimizing the information loss due to the reduction in details. These quality measures are *dilution* and *diversity*.

$$dilution(D_i, Hierarchy_{D_i}, T) = \frac{1}{|T|} \sum_{t \in T} \Delta_O(parent_O(t[D_i]), parent_{Hierarchy_{D_i}}(t[D_i])) \tag{5.14}$$

$$diversity(D_i, Hierarchy_{D_i}, T) = \frac{2}{(|T|^2 - |T|)} \sum_{t_1, t_2 \in T, t_1 \neq t_2} \Delta_{Hierarchy_{D_i}}(t_1[D_i], t_2[D_i]) \tag{5.15}$$

Measure	Global	Local
Reduction (%)	72.5 - 75.9	76.5 - 79.8
Sig. loss (%)	7.08-12.12	3.5-7.8
Dilution	0.367-0.473	0.342-0.429
Diversity	9.08-10.79	7.72-9.46

Table 5.10: Results for global an local dimension extraction methods. Value ranges represent the 0.75 confidence intervals of the results for all the signatures.

where T is the fact table, $parent_O(n)$[5] is the parent of n in O, and Δ_O is the taxonomic distance between two nodes in O. $t[D_i]$ represents the concept assigned to the fact t for the dimension D_i.

Dilution measures the weighted average distance in the original taxonomy between the new and original parents of the signature concepts from dimension D_i. The weight of each signature concept corresponds to its relative frequency in the fact table. The smaller the dilution, less semantic changes have been produced in the reduced taxonomy. Diversity measures the weighted average distance in the reduced taxonomy of any pair of concepts from dimension D_i used in the fact table. The greater the diversity, the better taxonomies are obtained for aggregation purposes. A very low diversity value usually indicates that most concepts are directly placed under the top concept.

In order to test the quality of the dimension hierarchies generated by the methods in Section 5.6, we have set up 25 signatures for 14 dimensions of the HeCOnto dataset. The size of these signatures ranges from 4 to 162 concepts (60 on average). The corresponding upper-modules are extracted from UMLS following the method proposed in [90]. The size of these modules ranges from 40 to 911 concepts (404 on average). Then, both the global and local method for selecting concepts that will participate in the dimension hierarchies are applied. The inferred taxonomy of the dimension hierarchies obtained present between 8 and 23 levels (17 on average). Table 5.10 shows the results with both the global and local method. Apart from the dilution and diversity of the generated dimension hierarchies, we also measure the reduction in size of the dimension hierarchy w.r.t. the upper module and the signature loss, that is, the signature concepts that do not appear in the dimension hierarchy. From these results we observe that the local method generates smaller dimension hierarchies and also implies less signature loss. The dilution values of both methods are not statistically different. However, diversity is usually greater in the global method (i.e., richer hierarchies are generated). To sum up, each method optimizes different quality parameters, and therefore, their application will depend on the user requirements.

[5]Indeed, we assume that the parent of n in O is the parent in the extracted spanning tree of O.

5.7.5 Implemented use case

We use MySQL database as back-end to store the domain and application ontologies (i.e., TBox), the TBox indexes of Section 4.2.1 (i.e., the $\mathcal{LS}^-$ and $\mathcal{LS}^+$ for concepts and the $\mathcal{LS}^-$ for roles) and the ABox assertions (i.e. tables described in Section 4.2.2. On the other hand, we use the *Business Intelligence* tool of *Microsoft SQL Server 2008*[6] to instantiate the MD query designed by the user and create cubes. Our method is completely independent of any data management system. We simply need to create an API to the back-end where the information is stored and the populated MD query is delivered as a series of tables that can be fed into any off the shelf analysis tool.

In Figure 5.12 we show the result of one of the MD queries proposed for the use case in Section 3.2.1. In this use case, the user is interested in analyzing the efficacy of different drugs w.r.t. a series of dimensions, such as the disease diagnosed, the patient's age, gender, etc. The method first generates the fact table according to the conceptual MD query proposed by the analyst and then, for each dimension, a dimension hierarchy is extracted using the *global* approach. The result (i.e., the populated MD query) has been fed to SQL Server and the BI tool allows the analyst to create cubes and navigate through them. In particular, Figure 5.12 shows the cube generated by averaging the *damageIndex* measure by *disease* (rows) and *drug* (columns). As shown, the user can navigate through the different levels of the dimension hierarchies and the measures are automatically aggregated. It is worth mentioning the added value that provides the semantics involved in the aggregations, since the dimension hierarchies express conceptual relations extracted from a domain ontology.

[6] SQL Server 2008: http://www.microsoft.com/sqlserver/2008

			Level 02 ▾ Level 03			
			⊞ C0003209=anti-inflammatory	⊞ C0021054=Factors,Immunologic	⊟ C2170827=tumornecrosisfactoralphablockers	
					⊞ C0666743=INFLIX	⊞ C1122087=tumornecrosis
Level 02 ▾	**Level 03**	**Level 04**	Avg damage Index	Avg damage Index	Avg damage Index	Avg damage Index
⊟ C0003873=Arthritisorpol	⊞ C0409651=rheumatoid/seropositivearthritides		26,10167	26,14971	26,16316	26,50977
	⊞ C0427391=Rheumatoidfactornegative		26,64191	26,93696	27,21934	27,03086
	⊟ C0553662=IdiopathicArth	⊞ C1384600=Juvenilearthritiswithsyste	26,18157	26,48039	26,67122	27,00051
		⊞ C1444840=Juvenileseronegativepoly	26,77114	26,97851	27,29854	27,24215
		⊞ C1444841=Juvenileidiopathicarthritis	27,21066	26,63931	26,25423	26,79112
		⊞ C1444844=juvenileidiopathicarthritid	27,22942	27,60329	26,49961	26,82511
		Total	26,83747	26,92224	26,67816	26,96601
	Total		26,68180	26,79518	26,69152	26,89885
Total general			26,68180	26,79518	26,69152	26,89885

Figure 5.12: Example of MD cube created by averaging the *damageIndex* measure by *disease* (rows) and *drug* (columns).

5.8 Discussion

This chapter has presented a method to enable scalable MD analysis over SW data. In particular, we are able to extract facts and dimensions from SW data overcoming several challenges such as the reasoning scalability issues and the complex nature of the relations that underlie the graph structure of the data. This is performed by taking into account the user requirements, which are expressed at the conceptual level.

Our notion of fact is more general than those proposed by other approaches that rely on functional dependencies [128]. This enables us to analyze data that are intrinsically complex and do not meet the traditional MD constraints. Moreover, we are able to still produce correct results when the extracted facts are not summarizable by managing duplicated information.

A fact is characterized by dimension and measure values that are reachable from the subject of analysis by means of an aggregation path. The notion of aggregation path was firstly introduced in [83]. This work proposes to extract the aggregation paths of a concept from the completion trees of the tableau algorithm. To that end, some new rules are introduced and others modified in the tableau algorithm. Although this work is also focused on MD analysis over SW data, the adopted approach differs from the method presented in this thesis in several aspects: 1) the aggregation paths are calculated from the tableau algorithm, which ties the method to a specific reasoner, 2) the complexity of the method corresponds to the complexity of the tableau algorithm for $\mathcal{SHOIQ}$, which is ExpTime-complete and 3) the method provides a summary of the ontology, that is, aggregations are performed inside the ontology by rolling-up instances.

In contrast, we propose a classification of the aggregation paths in three groups and an incremental algorithm that generates aggregation paths by interestingness order. Moreover, we meet the scalability challenge introduced by the reasoning systems by performing the reasoning only once off-line and pre-computing direct aggregation paths with the aid of the indexes proposed in Chapter 4. Also, our method is independent of any reasoning algorithm (e.g., tableau, hypertableau, rule-based, etc).

Facts are extracted from the ABox by processing the reachability graph, which is composed by aggregation paths from the subject of analysis to the dimension and measure user's conceptual specification. The reachability graph is translated into a series of conjunctive queries that are efficiently performed thanks to the indexes and algorithms described in Chapter 4.

Dimension hierarchies are dynamically extracted from semantic sources to better suit the base dimension values that compose the facts, in contrast to traditional MD scenarios, where the dimensions are static and pre-defined. In an attempt to capture as much semantics as possible for the later aggregation process, we model dimension hierarchies as directed acyclic graphs where nodes are sub-concepts of the conceptual description of the dimensions and the relations between nodes correspond to concept subsumption relations. However, in order to ease to posterior use of OLAP applications, which usually require well-shaped hierarchies, we propose a method to re-shape the extracted dimension hierarchies by identifying dense regions and selecting nodes with maximum aggregation power.

The performed experiments confirm the viability of the method and its usefulness in scenarios where data have complex relations and traditional MD analysis tools fall

short of the user's analysis requirements.

Chapter 6

Conclusions

The last chapter presents the main results of the thesis and outlines the future research lines. The chapter concludes by listing the publications resulted from this thesis work. Section 6.1 summarizes the results of the thesis. Section 6.2 discusses future work. Section 6.3 lists the main published contributions of the thesis.

6.1 Summary of the Thesis

SW data is currently being heavily used as a data representation format in scientific communities, social networks, business companies, news portals and other domains. The irruption and availability of SW data is demanding new methods and tools to efficiently analyze them and provide richer insights into the current business processes. Although there exist some applications that make use of SW data, advanced analytical tools are still lacking, preventing the user from exploiting the attached semantics. The success of the well-known discipline of MD analysis over traditional and structured data sources has prompted us to investigate the application of such techniques to more open and semi-structured scenarios such as the SW.

This thesis has reviewed the evolution of MD analysis techniques over different types of data, from the analysis of static, structured data residing in relational tables, to the analysis of external and semi-structured data coming from the Web (mainly XML), and more recently, the few approaches aimed at performing light analytical tasks over SW data. As far as we know, none of the approaches have tackled the problem of MD analysis of SW data in all its complexity. The thesis also summarizes the main advances on ontology modularization, as it is an intrinsic challenge that must be overcome when dealing with large amounts of SW data.

The goal of this thesis is to provide a formal framework that enables MD analysis of SW data in an efficient and scalable manner. We base our research on the hypothesis that SW data is an emerging knowledge resource worth exploiting and that the knowledge encoded in SW data can be leveraged to perform an efficient, scalable and full-fledged MD analysis of such data. Therefore, we have tackled research problems that are related to the manipulation and extraction of specific subsets of SW data, in order to arrange them in terms of facts and dimensions, which is the typical MD

structure. However, the full exploitation and manipulation of SW data efficiently and at large-scale imposes several challenges that we try to overcome during this dissertation.

SW data are generally based on formal descriptions, which enable to derive new logical consequences through the process of reasoning. However, reasoning techniques over large datasets are computationally expensive. In order to deal with the implicit semantics in a scalable manner we have devised an ontology indexing model that is applied to the inferred ontology hierarchy. The OIM is based on an interval labeling scheme that encodes information about ancestors and descendants of concepts in a compact format. Moreover, we have devised an interval algebra that allows to operate directly over the indexes and provides fast responses to queries about semantic relationships between concepts. Through this OIM, we minimize the expensive use of a reasoner. Going a step further, we have developed a DL query module that gives sound and complete answers to a restricted set of DL queries, which are responded with the indexes. The index is applied to the ABox in order to efficiently handle conjunctive queries over instance data.

A MD analysis task emerges from a specific user's informational need, usually expressed in terms of an MD query. Therefore, we are confronted by another challenge that concerns the extraction of only the required subsets of SW data that are useful for the MD analysis in a scalable manner. Several modularization approaches have been proposed to extract specific subsets from ontologies. However, we have identified a series of requirements on the extracted modules imposed by the analytical applications that the existing modularization approaches fail to achieve. These are summarized in the following sentence: the *user analysis requirements*, expressed as a signature, should drive the extraction of modules of *reduced size* that *preserve the semantics* of the signature elements to some extent while providing a suitable context and *structure* for such elements, all this remaining *scalable*. The lack of modularity approaches aimed at covering all the previous requirements has prompted the development of the OMETs, which are based on the OIM. In particular, we have developed four methods that offer a good trade-off among the previous requirements. Obviously, more semantics preservation leads to larger module sizes, whereas more compact modules leads to small sizes but less semantic preservation. The user is responsible for selecting the appropriate modularization method.

The previous indexing and modularization methods assist in the analytical tasks by enabling the construction of the MD schema based on the user query. That is, they are used to make the extraction of facts and dimensions from SW data efficient and scalable. However, there is still a mismatch between the graph model that underlies SW data and the cube-oriented MD model, which is expressed in terms of facts and dimensions. Identifying facts, dimensions, measures and well-shaped dimension hierarchies from the graph structure that underlies SW data is a big challenge. Also, as the web is continuously changing and growing, the developed methods should take into account the user requirements while being as automatic as possible so that the results can be reproduced to reflect changes in the data. To meet the previous challenges, the notions of fact and dimension are revisited in the SW context and both facts and dimensions are defined from a logical viewpoint. Facts are formally defined as MD points (a dimension value for each dimension) and quantified by measure values. However, we relax the functional dependency restriction usually imposed by traditional MD approaches and search for dimension

and measure values that are reachable from the subject of analysis by means of an aggregation path. To that end, we identify different types of aggregation paths. As a result, we obtain a reachability graph, which is a mapping of the user MD query to the TBox. Facts are extracted by aggregating the data tuples obtained by processing the reachability graph. Moreover, as we do not restrict the type of relations between facts and dimensions, we are able to detect the summarizability property of the extracted facts and still produce correctly aggregated facts when the facts are not summarizable. By relaxing the functional dependency between facts and dimensions, we are able to extract meaningful facts from data that are not MD by nature and cannot be otherwise analyzed. Dimensions are defined as directed acyclic graphs of nodes that are semantically related and adhere to the conceptual specification of the user. That is, nodes are sub-concepts of the conceptual specification of the dimension and edges correspond to subsumption relations. Two alternative methods allow the extraction of dimension hierarchies suitably shaped to perform MD analysis and that preserve the semantics of the dimension values.

The experimental evaluation performed on each of the components of the framework demonstrates that the proposed analysis framework scales to large ontological resources. Likewise, the developed use cases show the potential and usefulness of the MD analysis of SW data.

6.2 Future Work

A number of directions for further research have been pointed out throughout the thesis, which we summarize here. First, we point out to specific limitations of the current developed methods and suggest further improvements. Then, we refer to more general research lines that have emerged from this thesis.

Currently, the OIM does not give support for updates. When an ontology is updated (a new entity or axiom is added to the TBox), the indexes must be re-built. The OIM could be revised to include efficient change support. However, this is still an emerging area of research, as even the most recent state-of-the-art indexing techniques over large graph need to still face incremental updates [151].

The developed OMETs are mainly oriented to the extraction of subsumption relationships between concepts, whereas the properties are relegated to a secondary level. This decision was made to meet all the analysis requirements. Even though some existing modularization techniques consider properties as first-class citizens, the extracted modules are much larger, which implies a big processing overload at the same time that re-usability decreases. In fact, properties are very important for analytical queries, as they are a means to gather together the dimension values that constitute a fact. However, this thesis approaches this issue from a different perspective and considers properties by means of the aggregation paths. In any case, it would be interesting to research to which extent properties can be included as first-class citizens in the construction of the modules without sacrificing the size or the compactness.

The fact extractor is based on the reachability of the dimensions and measures from the subject of analysis. This reachability notion is formally defined by means of the aggregation paths. Even though we restrict the possible aggregation paths by making a classification that limits the search space, it would be interesting to

further devise and characterize new groups of aggregation paths that suit the analytic requirements of the user.

The methods for extracting hierarchical dimensions from ontologies have been devised to keep a suitable shape for aggregations. However, automatic stratification of the dimension hierarchies into levels would be interesting and has not yet been addressed.

Currently, the fact and dimension extractor are not sensitive to the changes that may occur in the data (both at the TBox and ABox level). If changes at the TBox level that invalidate the current pre-computation of aggregation paths occur, these paths need to be re-computed. Similarly, if changes occurs at the ABox level, the reachability graph corresponding to the MD query is still valid but the facts need to be extracted again. It would be interesting to extend the current methods so that when changes happen, the minimum amount of information must be re-calculated.

Another issue that has not been tackled in this dissertation and needs further research is the suitability of massive parallel processing to implement the developed framework, as these approaches are gaining momentum for processing large analytical workloads.

The thesis demonstrates the initial hypothesis that the semantic annotations attached to the data can be leveraged to provide efficient and scalable MD analysis. This positive result encourages us to widen the analysis perspectives on SW data to other kinds of analysis that aid decision making, specially data mining techniques. Just the same way that facts and dimensions are the building block for MD analysis, are transactions for data mining. Thus, we conceive data mining on SW data as the problem of extracting suitable transactions leveraged by the semantics attached to the data. Research in this direction is very promising and the first results have been published in [93, 100]. In particular, the aim of these papers is to extract association rules from SW data. To that end, we let the user express a query to conceptually identify the items and granularity of the transactions and then, we run a traditional association rules algorithm over the extracted transactions. The results are very encouraging, as the semantics attached to the transaction items, and thus, to the generated rules is very valuable and can be used to enhance the mining process itself. For example, a simple but smart extension is to filter and prune the large amount of discovered rules using this knowledge. In the book chapter [96], we discuss the main data mining approaches proposed so far to mine SW data as well as those that have taken into account semantic resources and tools to define semantics-aware methods.

Another research line that has come up during the development of this thesis is related to the field of IE. Although the amount of published SW data is continuously increasing, there are still huge amounts of implicit knowledge hidden in natural language texts. IE tools are widely known to automatically extract structured information from unstructured text. However, we go a step further and advocate the use of semantic annotation inside the IE process as a means to extract structured information in a SW format (i.e., information already linked to ontologies). In [95] we propose an unsupervised method to extract semantic relations from biomedical texts based on semantic annotation. As a result, we are able to extract triples $(subject, predicate, object)$, where both the *subject* and *object* have a well-defined meaning and the *predicate* expresses a semantic relation between the *subject* and *object*. This way, we are making available SW data, which can be used as input data set for the MD analysis framework developed in this thesis.

6.3 List of Publications

This section enumerates the publications concerning this thesis. We have grouped the publications by topics and point out the chapters that mainly influenced them.

Former research on ontology fragmentation was published in [61]. The work in [89] and [91] contains the main foundations discussed in Chapter 4 about indexing and modularization of ontologies.

> [61] Ernesto Jiménez-Ruiz, Rafael Berlanga Llavori, *Victoria Nebot*, and Ismael Sanz. Ontopath: A language for retrieving ontology fragments. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 4803 of *Lecture Notes in Computer Science*, pages 897–914. Springer, 2007. ISBN 978-3-540-76846-3.
>
> [89] *Victoria Nebot* and Rafael Berlanga Llavori. Efficient retrieval of ontology fragments using an interval labeling schema. In Sistedes, editor, *JISBD*, 2008. ISBN 978-84-612-5820-8.
>
> [91] *Victoria Nebot* and Rafael Berlanga. Efficient retrieval of ontology fragments using an interval labeling schema. *Inf. Sci.*, 179(24):4151–4173, 2009.

The indexing and modularization approaches presented in this thesis have given support to many applications. In [90, 16, 70] custom ontology fragments need to be extracted from large ontologies. The work in [13, 14] utilizes the methods to build conceptual spaces that semantically guide the exploration of unstructured sources.

> [90] *Victoria Nebot* and Rafael Berlanga. Building Tailored Ontologies from very large Knowledge Resources. In *ICEIS Conference Proceedings*, volume 2, pages 144–151. ICEIS, May 2009.
>
> [16] Rafael Berlanga, Ernesto Jiménez-Ruiz, *Victoria Nebot*, and Ismael Sanz. FAETON: Form analysis and extraction tool for ontology construction. *IJCAT*, 39(4):224–233, 2010.
>
> [70] Shahad Kudama, Rafael Berlanga, Lisette García-Moya, *Victoria Nebot*, and María José Aramburu. Towards tailored semantic annotation systems from wikipedia. In Franck Morvan, A Min Tjoa, and Roland Wagner, editors, *DEXA Workshops*, pages 478–482. IEEE Computer Society, 2011. ISBN 978-1-4577-0982-1.
>
> [13] Rafael Berlanga, Ernesto Jiménez-Ruiz, and *Victoria Nebot*. Building conceptual spaces for exploring and linking biomedical resources. In Albert Burger, M. Scott Marshall, Paolo Romano, Adrian Paschke, and Andrea Splendiani, editors, *SWAT4LS*, volume 698 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
>
> [14] Rafael Berlanga, Ernesto Jiménez-Ruiz, and *Victoria Nebot*. Exploring and linking biomedical resources through multidimensional semantic spaces. *BMC bioinformatics*, 13 Suppl 1 (Suppl 1): S6+, January 2012. ISSN 1471-2105. DOI 10.1186/1471-2105-13-S1-S6. URL `http://dx.doi.org/10.1186/1471-2105-13-S1-S6`.

The work in [92, 94, 98] contains the first steps towards the MD analysis of SW data and the work in [99] contains the foundations for MD analysis developed in Chapter 5. In the paper [101], an alternative to analyze SW data is explored and the book chapter in [18] contains an extensive review of the synergies between the fields of BI and the SW.

> [92] *Victoria Nebot* and Rafael Berlanga. Populating data warehouses with semantic data. In Antonio Vallecillo and Goiuria Sagardui, editors, *JISBD*, pages 303–314, 2009.
>
> [94] Victoria Nebot and Rafael Berlanga. Populating data warehouses with semantic data. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 8(2):150 –157, april 2010.
>
> [98] *Victoria Nebot* and Rafael Berlanga Llavori. Building data warehouses with semantic data. In Florian Daniel, Lois M. L. Delcambre, Farshad Fotouhi, Irene Garrigós, Giovanna Guerrini, Jose-Norberto Mazón, Marco Mesiti, Sascha Müller-Feuerstein, Juan Trujillo, Traian Marius Truta, Bernhard Volz, Emmanuel Waller, Li Xiong, and Esteban Zimányi, editors, *EDBT/ICDT Workshops*, ACM International Conference Proceeding Series. ACM, 2010.
>
> [99] *Victoria Nebot* and Rafael Berlanga. Building data warehouses with semantic web data. *Decision Support Systems*, 52(4):853–868, 2012.
>
> [101] *Victoria Nebot*, Rafael Berlanga, Juan Manuel Pérez-Martínez, María José Aramburu, and Torben Bach Pedersen. Multidimensional integrated ontologies: A framework for designing semantic data warehouses. *Journal on Data Semantics XIII*, 5530:1–36, 2009.
>
> [18] Rafael Berlanga, Oscar Romero, Alkis Simitsis, *Victoria Nebot*, Torben Bach Pedersen, Alberto Abell, and Mara Jos Aramburu. Semantic Web Technologies for Business Intelligence. In *Business Intelligence Applications and the Web: Models, Systems and Technologies*, pages 310–339. IGI Global, 2012.

During the research concerning this thesis, alternative approaches to MD modeling for analyzing SW data have been explored, with special emphasis on data mining techniques. This has given rise to the publications [93, 100], which explore association rule mining over SW data, and the book chapter [96], which reviews and discusses the main data mining approaches proposed so far to mine SW data as well as those that have taken into account semantic resources and tools to define semantics-aware methods.

> [93] *Victoria Nebot* and Rafael Berlanga. Mining association rules from semantic web data. In Nicolás García-Pedrajas, Francisco Herrera, Colin Fyfe, José Manuel Benítez, and Moonis Ali, editors, *IEA/AIE (2)*, volume 6097 of *Lecture Notes in Computer Science*, pages 504–513. Springer, 2010.
>
> [100] *Victoria Nebot* and Rafael Berlanga Llavori. Finding association rules in semantic web data. *Knowl.-Based Syst.*, 25(1):51–62, 2012.
>
> [96] *Victoria Nebot* and Rafael Berlanga. XML Mining for Semantic Web. In *XML Data Mining: Models, Methods, and Applications*, pages 317–342. IGI Global, 2012.

Semantic annotation is paramount for the development of the SW. The early work in [15] demonstrates the role of semantic annotation to integrate heterogeneous data. The tool described in [17] is a dictionary-based semantic annotation tool that scales to large ontologies and large texts.

> [15] Rafael Berlanga, Ernesto Jiménez-Ruiz, *Victoria Nebot*, David Manset, Andrew Branson, Tamas Hauer, Richard McClatchey, Dmitri Rogulin, Jetendr Shamdasani, Sonja Zillner, and Joerg Freund. Medical data integration and the semantic annotation of medical protocols. In *CBMS*, pages 644–649, 2008.
>
> [17] Rafael Berlanga, *Victoria Nebot*, and Ernesto Jiménez-Ruiz. Semantic annotation of biomedical texts through concept retrieval. *Procesamiento del lenguaje natural*, 45:247–250, 2010.

The work in [97, 102, 95] has emerged from this thesis as a related and interesting research line. The work has as common denominator to uncover implicit information hidden in natural language texts in an attempt to enrich the SW with new, semi-structured and semantically-enriched data.

> [97] *Victoria Nebot*, Shahad Kudama, and Rafael Berlanga. Towards the discovery of semantic relations in large biomedical annotated corpora. In Morvan et al. *2011 Database and Expert Systems Applications, DEXA, International Workshops, Toulouse, France, August 29 - Sept. 2, 2011*. IEEE Computer Society, 2011, pages 465–469.
>
> [102] *Victoria Nebot*, Min Ye, Mario Albrecht, Jae-Hong Eom, and Gerhard Weikum. DIDO: a disease-determinants ontology from web sources. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *WWW (Companion Volume)*, pages 237–240. ACM, 2011.
>
> [95] *Victoria Nebot* and Rafael Berlanga. Semantics-aware open information extraction in the biomedical domain. In Adrian Paschke, Albert Burger, Paolo Romano, M. Scott Marshall, and Andrea Splendiani, editors, *SWAT4LS*, pages 84–91. ACM, 2011.
>
> *Victoria Nebot* and Rafael Berlanga. Exploiting Semantic Annotations for Open Information Extraction: an experience in the biomedical domain. Accepted for publication in *Knowledge and Information Systems*, 2012.

Bibliography

[1] MEDLINE: National Library of Medicine's database. `http://www.nlm.nih.gov/databases/databases_medline.html`.

[2] Daniel J. Abadi, Adam Marcus, Samuel R. Madden, and Kate Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very Large Data Bases*, VLDB '07, pages 411–422. VLDB Endowment, 2007.

[3] Alberto Abelló, José Samos, and Fèlix Saltor. YAM2: a multidimensional conceptual model extending UML. *Inf. Syst.*, 31(6):541–567, September 2006.

[4] Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel Abadi, Avi Silberschatz, and Alexander Rasin. HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. *Proc. VLDB Endow.*, 2(1):922–933, August 2009.

[5] Foto N. Afrati and Jeffrey D. Ullman. Optimizing joins in a map-reduce environment. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 99–110, New York, NY, USA, 2010. ACM.

[6] Rakesh Agrawal, Alex Borgida, and H. V. Jagadish. Efficient management of transitive relationships in large data and knowledge bases. In *Proceedings of the 1989 ACM SIGMOD international conference on Management of Data*, SIGMOD '89, pages 253–262, New York, NY, USA, 1989. ACM.

[7] Stefan Anderlik, Bernd Neumayr, and Michael Schrefl. Using Domain Ontologies as Semantic Dimensions in Data Warehouses. In Paolo Atzeni, David Cheung, and Sudha Ram, editors, *Conceptual Modeling*, volume 7532 of *Lecture Notes in Computer Science*, pages 88–101. Springer Berlin Heidelberg, 2012.

[8] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite Family and Relations. *J. Artif. Intell. Res. (JAIR)*, 36:1–69, 2009.

[9] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 364–369, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[10] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[11] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL - A Polynomial-Time Reasoner for Life Science Ontologies. In *Third International Joint Conference on Automated Reasoning, IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 287–291. Springer, 2006.

[12] Jie Bao, Doina Caragea, and Vasant G. Honavar. Package-based description logics - preliminary results. In *Proceedings of the 5th International Semantic Web Conference*, ISWC'06, pages 967–969, Berlin, Heidelberg, 2006. Springer-Verlag.

[13] Rafael Berlanga, Ernesto Jiménez-Ruiz, and Victoria Nebot. Building Conceptual Spaces for Exploring and Linking Biomedical Resources. In Albert Burger, M. Scott Marshall, Paolo Romano, Adrian Paschke, and Andrea Splendiani, editors, *Proceedings of the Workshop on Semantic Web Applications and Tools for Life Sciences, SWAT4LS, Berlin, Germany, December 10, 2010*, volume 698 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[14] Rafael Berlanga, Ernesto Jiménez-Ruiz, and Victoria Nebot. Exploring and linking biomedical resources through multidimensional semantic spaces. *BMC bioinformatics*, 13 Suppl 1(Suppl 1):S6+, January 2012.

[15] Rafael Berlanga, Ernesto Jimenez-Ruiz, Victoria Nebot, David Manset, Andrew Branson, Tamas Hauer, Richard McClatchey, Dmitry Rogulin, Jetendr Shamdasani, Sonja Zillner, and Joerg Freund. Medical Data Integration and the Semantic Annotation of Medical Protocols. In *Proceedings of the 2008 21st IEEE International Symposium on Computer-Based Medical Systems*, CBMS '08, pages 644–649, Washington, DC, USA, 2008. IEEE Computer Society.

[16] Rafael Berlanga, Ernesto Jiménez-Ruiz, Victoria Nebot, and Ismael Sanz. FAETON: Form Analysis and Extraction Tool for ONtology construction. *Int. J. Comput. Appl. Technol.*, 39(4):224–233, October 2010.

[17] Rafael Berlanga, Victoria Nebot, and Ernesto Jiménez-Ruiz. Semantic annotation of biomedical texts through concept retrieval. *Procesamiento del Lenguaje Natural*, 45:247–250, 2010.

[18] Rafael Berlanga, Oscar Romero, Alkis Simitsis, Victoria Nebot, Torben Bach Pedersen, Alberto Abelló, and María José Aramburu. Semantic Web Technologies for Business Intelligence. In *Business Intelligence Applications and the Web: Models, Systems and Technologies*, pages 310–339. IGI Global, 2012.

[19] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[20] Olivier Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic acids research*, 32(Database issue), January 2004.

[21] Olivier Bodenreider. Biomedical ontologies in action: role in knowledge management, data integration and decision support. *Yearbook of medical informatics*, pages 67–79, 2008.

[22] Philip Bohannon, Juliana Freire, Prasan Roy, and Jérôme Siméon. From XML Schema to Relations: A Cost-Based Approach to XML Storage. In *Proceedings of the 18th International Conference on Data Engineering, 26 February - 1 March 2002, San Jose, CA*, pages 64–75. IEEE Computer Society, 2002.

[23] Alexander Borgida and Luciano Serafini. Distributed Description Logics: Assimilating Information from Peer Sources. *J. Data Semantics*, 1:153–184, 2003.

[24] Paolo Bouquet, Fausto Giunchiglia, Frank Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-OWL: Contextualizing Ontologies. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proceedings of the Second International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer Berlin Heidelberg, 2003.

[25] Dan Brickley and Ramanathan V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, 2 2004.

[26] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *Proceedings of the First International Semantic Web Conference*, Lecture Notes in Computer Science, pages 54–68, London, UK, 2002. Springer-Verlag.

[27] Luca Cabibbo and Riccardo Torlone. A Logical Approach to Multidimensional Databases. In *Proceedings of the 6th International Conference on Extending Database Technology*, EDBT '98, pages 183–197, London, UK, 1998. Springer-Verlag.

[28] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *Journal of Automated Reasoning*, 39:385–429, 2007. 10.1007/s10817-007-9078-x.

[29] K. Selçuk Candan, Mario Cataldi, and Maria Luisa Sapino. Reducing metadata complexity for faster table summarization. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 240–251, New York, NY, USA, 2010. ACM.

[30] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference*, WWW '04, pages 74–83, New York, NY, USA, 2004. ACM.

[31] Malú Castellanos, Umeshwar Dayal, and Meichun Hsu. Live Business Intelligence for the Real-Time Enterprise. In *From Active Data Management to Event-Based Systems and More*, pages 325–336, 2010.

[32] Vassilis Christophides, Dimitris Plexousakis, Michel Scholl, and Sotirios Tourtounis. On labeling schemes for the semantic web. In *Proceedings of the 12th international World Wide Web conference*, WWW '03, pages 544–555, New York, NY, USA, 2003. ACM.

[33] Edgar F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User Analysts: An IT Mandate. E. F. Codd and Ass., 1993.

[34] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Just the right amount: extracting modules from ontologies. In *Proceedings of the 16th international World Wide Web conference*, WWW '07, pages 717–726, New York, NY, USA, 2007. ACM.

[35] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. A logical framework for modularity of ontologies. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI'07, pages 298–303, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[36] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: theory and practice. *J. Artif. Int. Res.*, 31:273–318, February 2008.

[37] Bernardo Cuenca Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Automatic Partitioning of OWL Ontologies Using *E*-Connections. In *Description Logics*, 2005.

[38] Roxana Dánger and Rafael Berlanga. A Semantic Web Approach for Ontological Instances Analysis. In Joaquim Filipe, Boris Shishkov, Markus Helfert, and Leszek A. Maciaszek, editors, *Software and Data Technologies*, volume 22 of *Communications in Computer and Information Science*, pages 269–282. Springer Berlin Heidelberg, 2009.

[39] Mathieu d'Aquin, Marta Sabou, and Enrico Motta. Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components. In *WoMO*, 2006.

[40] Mathieu D'Aquin, Anne Schlicht, Heiner Stuckenschmidt, and Marta Sabou. Modular ontologies. In Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors, *Criteria and Evaluation for Ontology Modularization Techniques*, pages 67–89. Springer-Verlag, Berlin, Heidelberg, 2009.

[41] Jeffrey Dean and Sanjay Ghemawat. MapReduce: a flexible data processing tool. *Commun. ACM*, 53(1):72–77, January 2010.

[42] Jens Dittrich, Jorge-Arnulfo Quiané-Ruiz, Alekh Jindal, Yagiz Kargin, Vinay Setty, and Jörg Schad. Hadoop++: making a yellow elephant run like a cheetah (without it even noticing). *Proc. VLDB Endow.*, 3(1-2):515–529, September 2010.

[43] Paul Doran, Valentina Tamma, and Luigi Iannone. Ontology module extraction for ontology reuse: an ontology engineering perspective. In *Proceedings of the 16th Conference on Information and Knowledge Management*, CIKM '07, pages 61–70, New York, USA, 2007. ACM.

[44] Daniela Florescu and Donald Kossmann. Storing and Querying XML Data using an RDMBS. *IEEE Data Eng. Bull.*, 22(3):27–34, 1999.

[45] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. The Dimensional Fact Model: A Conceptual Model for Data Warehouses. *Int. J. Cooperative Inf. Syst.*, 7(2-3):215–247, 1998.

[46] Matteo Golfarelli, Stefano Rizzi, and Boris Vrdoljak. Data warehouse design from XML sources. In *Proceedings of the 4th ACM international workshop on Data warehousing and OLAP*, DOLAP '01, pages 40–47, New York, USA, 2001. ACM.

[47] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[48] N. Guarino. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy.* IOS Press, Amsterdam, The Netherlands, The Netherlands, 1st edition, 1998.

[49] Andreas Harth, Jürgen Umbrich, Aidan Hogan, and Stefan Decker. YARS2: a federated repository for querying graph structured data from the web. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 211–224, Berlin, Heidelberg, 2007. Springer-Verlag.

[50] Jan Hidders and Jan Paredaens. XPath/XQuery. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 3659–3665. Springer US, 2009.

[51] Pascal Hitzler and Denny Vrandečić. Resolution-Based approximate reasoning for OWL DL. In *Proceedings of the 4th International Semantic Web Conference*, ISWC'05, pages 383–397, Berlin, Heidelberg, 2005. Springer-Verlag.

[52] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *KR*, pages 57–67. AAAI Press, 2006.

[53] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The Instance Store: DL Reasoning with Large Numbers of Individuals. In Volker Haarslev and Ralf Möller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*, pages 31–40. CEUR-WS.org, 2004.

[54] Carlos A. Hurtado, Claudio Gutierrez, and Alberto O. Mendelzon. Capturing summarizability with integrity constraints in OLAP. *ACM Trans. Database Syst.*, 30(3):854–886, 2005.

[55] William H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, Inc., New York, NY, USA, 4rth edition, 2005.

[56] William H. Inmon, Derek Strauss, and Genia Neushloss. *DW 2.0: The Architecture for the Next Generation of Data Warehousing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[57] Mikael R. Jensen, Thomas Holmgren, and Torben Bach Pedersen. Discovering Multidimensional Structure in Relational Data. In Yahiko Kambayashi, Mukesh K. Mohania, and Wolfram Wöß, editors, *Proceedings of the 6th international conference on Data Warehousing and Knowledge discovery, DaWaK*, volume 3181 of *Lecture Notes in Computer Science*, pages 138–148. Springer, 2004.

[58] Mikael R. Jensen, Thomas H. Møller, and Torben Bach Pedersen. Specifying OLAP Cubes on XML Data. *J. Intell. Inf. Syst.*, 17(2-3):255–280, December 2001.

[59] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: logic-based and scalable ontology matching. In *Proceedings of the 10th International Semantic Web Conference*, ISWC'11, pages 273–288, Berlin, Heidelberg, 2011. Springer-Verlag.

[60] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ulrike Sattler, Thomas Schneider, and Rafael Berlanga. Safe and economic re-use of ontologies: a logic-based methodology and tool support. In *Proceedings of the 5th European Semantic Web Conference*, ESWC'08, pages 185–199, Berlin, Heidelberg, 2008. Springer-Verlag.

[61] Ernesto Jiménez-Ruiz, Rafael Berlanga Llavori, Victoria Nebot, and Ismael Sanz. OntoPath: A Language for Retrieving Ontology Fragments. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 4803 of *Lecture Notes in Computer Science*, pages 897–914. Springer, 2007.

[62] Antonio Jimeno-Yepes, Ernesto Jiménez-Ruiz, Rafael Berlanga, and Dietrich Rebholz-Schuhmann. Reuse of terminological resources for efficient ontological engineering in Life Sciences. *BMC Bioinformatics*, 10(Suppl 10):S4, 2009.

[63] Benedikt Kämpgen and Andreas Harth. Transforming statistical linked data for use in OLAP systems. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, ACM International Conference Proceeding Series, pages 33–40. ACM, 2011.

[64] Benedikt Kämpgen, Sean O'Riain, and Andreas Harth. Interacting with Statistical Linked Data via OLAP Operations. In *Proceedings of the International Workshop on Interacting with Linked Data (ILD 2012), Extended Semantic Web Conference (ESWC)*. CEUR-WS.org, Mai 2012.

[65] Selma Khouri and Ladjel Bellatreche. A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In *Proceedings of the 13th international workshop on Data warehousing and OLAP*, DOLAP 2010, pages 19–24. ACM, 2010.

[66] Ralph Kimball. *The data warehouse toolkit: practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[67] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

[68] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. CEX and MEX: Logical Diff and logic-based module extraction in a fragment of OWL. In *OWL: Experiences and Directions (OWLED)*, page online, 2008.

[69] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Semantic Modularity and Module Extraction in Description Logics. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI08)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 55–59. IOS Press, 2008.

[70] Shahad Kudama, Rafael Berlanga Llavori, Lisette García-Moya, Victoria Nebot, and María José Aramburu Cabo. Towards Tailored Semantic Annotation Systems from Wikipedia. In *Proceedings of 22nd International Workshop on Database and Expert Systems Applications*, DEXA '11, pages 478–482, Washington, DC, USA, 2011. IEEE Computer Society.

[71] Oliver Kutz, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1 – 73, 2004.

[72] Julien Leblay. SPARQL query answering with bitmap indexes. In *Proceedings of the 4th International Workshop on Semantic Web Information Management*, SWIM '12, pages 9:1–9:4, New York, NY, USA, 2012. ACM.

[73] Hans-Joachim Lenz and Arie Shoshani. Summarizability in OLAP and Statistical Data Bases. In Yannis E. Ioannidis and David M. Hansen, editors, *SSDBM*, pages 132–143. IEEE Computer Society, 1997.

[74] Yunyao Li, Cong Yu, and H. V. Jagadish. Schema-Free XQuery. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *VLDB*, pages 72–83. Morgan Kaufmann, 2004.

[75] Jimmy Lin and Chris Dyer. *Data-Intensive Text Processing with MapReduce*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers, 2010.

[76] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen. 3XL: Supporting efficient operations on very large OWL Lite triple-stores. *Inf. Syst.*, 36(4):765–781, June 2011.

[77] Alexander Löser, Fabian Hueske, and Volker Markl. Situational Business Intelligence. In Malu Castellanos, Umesh Dayal, Timos Sellis, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, and Clemens Szyperski, editors, *Business Intelligence for the Real-Time Enterprise*, volume 27 of *Lecture Notes in Business Information Processing*, pages 1–11. Springer Berlin Heidelberg, 2009. 10.1007/978-3-642-03422-0_1.

[78] Jing Lu, Li Ma, Lei Zhang, Jean-Sébastien Brunner, Chen Wang, Yue Pan, and Yong Yu. SOR: a practical system for ontology storage, reasoning and search. In *Proceedings of the 33rd international conference on Very Large Data Bases*, VLDB '07, pages 1402–1405. VLDB Endowment, 2007.

[79] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI'07, pages 453–458, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[80] Miguel Martínez-Prieto, Mario Arias Gallego, and Javier Fernández. Exchange and Consumption of Huge RDF Data. In Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti, editors, *The Semantic Web: Research and Applications*, volume 7295 of *Lecture Notes in Computer Science*, pages 437–452. Springer Berlin / Heidelberg, 2012.

[81] Jose-Norberto Mazón, Jens Lechtenbörger, and Juan Trujillo. A survey on summarizability issues in multidimensional modeling. *Data Knowl. Eng.*, 68(12):1452–1469, 2009.

[82] Jose-Norberto Mazón, Juan Trujillo, and Jens Lechtenbörger. Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. *Data Knowl. Eng.*, 63(3):725–751, December 2007.

[83] Roxana María Dánger Mercaderes. *Extracción y análisis de información desde la perspectiva de la Web Semántica*. PhD thesis, Universitat Jaume I, Castellón, Spain, 2007.

[84] Daniel L. Moody and Mark A. R. Kortink. From enterprise models to dimensional models: a methodology for data warehouse and data mart design. In Manfred A. Jeusfeld, Hua Shu, Martin Staudt, and Gottfried Vossen, editors,

Proceedings of the Second Intl. Workshop on Design and Management of Data Warehouses, DMDW 2000, Stockholm, Sweden, June 5-6, 2000, volume 28 of *CEUR Workshop Proceedings*, page 5. CEUR-WS.org, 2000.

[85] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language: Profiles. *W3C Recommendation*, 2009. `http://www.w3.org/TR/owl2-profiles/`.

[86] Boris Motik, Peter F. Patel-Schneider, and Bernardo Cuenca-Grau. OWL 2 Web Ontology Language Direct Semantics. *W3C Recommendation*, 2009. `http://www.w3.org/TR/owl2-semantics/`.

[87] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research (JAIR)*, 36:165–228, 2009.

[88] Turkka Näppilä, Kalervo Järvelin, and Timo Niemi. A tool for data cube construction from structurally heterogeneous XML documents. *J. Am. Soc. Inf. Sci. Technol.*, 59(3):435–449, February 2008.

[89] Victoria Nebot and Rafael Berlanga. Efficient retrieval of ontology fragments using an interval labeling scheme. In Sistedes, editor, *XIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2008), Gijón, Spain, 2008*, 2008.

[90] Victoria Nebot and Rafael Berlanga. Building Tailored Ontologies from Very Large Knowledge Resources. In José Cordeiro and Joaquim Filipe, editors, *Proceedings of the 11th International Conference on Enterprise Information Systems, ICEIS 2, Volume AIDSS, Milan, Italy, May 6-10, 2009*, pages 144–151, 2009.

[91] Victoria Nebot and Rafael Berlanga. Efficient Retrieval of Ontology Fragments using an Interval Labeling Scheme. *Inf. Sci.*, 179(24):4151–4173, December 2009.

[92] Victoria Nebot and Rafael Berlanga. Populating Data Warehouses with Semantic Data. In Antonio Vallecillo and Goiuria Sagardui, editors, *XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2009), San Sebastián, Spain, September 8-11, 2009*, pages 303–314, 2009.

[93] Victoria Nebot and Rafael Berlanga. Mining Association Rules from Semantic Web Data. In Nicolás García-Pedrajas, Francisco Herrera, Colin Fyfe, José Manuel Benítez, and Moonis Ali, editors, *23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2010*, volume 6097 of *Lecture Notes in Computer Science*, pages 504–513. Springer, 2010.

[94] Victoria Nebot and Rafael Berlanga. Populating Data Warehouses with Semantic Data. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 8(2):150 –157, april 2010.

[95] Victoria Nebot and Rafael Berlanga. Semantics-aware open information extraction in the biomedical domain. In Adrian Paschke, Albert Burger, Paolo Romano, M. Scott Marshall, and Andrea Splendiani, editors, *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences, SWAT4LS 2011, London, United Kingdom, December 07-09, 2011*, pages 84–91. ACM, 2011.

[96] Victoria Nebot and Rafael Berlanga. XML Mining for Semantic Web. In *XML Data Mining: Models, Methods, and Applications*, pages 317–342. IGI Global, 2012.

[97] Victoria Nebot, Shahad Kudama, and Rafael Berlanga. Towards the Discovery of Semantic Relations in Large Biomedical Annotated Corpora. In *Proceedings of the 2011 22nd International Workshop on Database and Expert Systems Applications*, DEXA '11, pages 465–469, Washington, DC, USA, 2011. IEEE Computer Society.

[98] Victoria Nebot and Rafael Berlanga Llavori. Building data warehouses with semantic data. In Florian Daniel, Lois M. L. Delcambre, Farshad Fotouhi, Irene Garrigós, Giovanna Guerrini, Jose-Norberto Mazón, Marco Mesiti, Sascha Müller-Feuerstein, Juan Trujillo, Traian Marius Truta, Bernhard Volz, Emmanuel Waller, Li Xiong, and Esteban Zimányi, editors, *Proceedings of the 13th International Conference on Extending Database Technology, EDBT*, ACM International Conference Proceeding Series. ACM, 2010.

[99] Victoria Nebot and Rafael Berlanga Llavori. Building data warehouses with semantic web data. *Decision Support Systems*, 52(4):853–868, 2012.

[100] Victoria Nebot and Rafael Berlanga Llavori. Finding association rules in semantic web data. *Knowl.-Based Syst.*, 25(1):51–62, 2012.

[101] Victoria Nebot, Rafael Berlanga Llavori, Juan Manuel Pérez-Martínez, María José Aramburu, and Torben Bach Pedersen. Multidimensional Integrated Ontologies: A Framework for Designing Semantic Data Warehouses. *Journal on Data Semantics XIII*, 5530:1–36, 2009.

[102] Victoria Nebot, Min Ye, Mario Albrecht, Jae-Hong Eom, and Gehard Weikum. DIDO: a disease-determinants ontology from web sources. In *Proceedings of the 20th international World Wide Web conference*, WWW '11, pages 237–240, New York, NY, USA, 2011. ACM.

[103] Thomas Neumann and Gerhard Weikum. RDF-3X: a RISC-style engine for RDF. *Proc. VLDB Endow.*, 1(1):647–659, August 2008.

[104] Bernd Neumayr, Michael Schrefl, and Konrad Linner. Semantic cockpit: an ontology-driven, interactive business intelligence tool for comparative data analysis. In *Proceedings of the 30th international conference on Advances in conceptual modeling: recent developments and new directions*, ER'11, pages 55–64. Springer-Verlag, 2011.

[105] Tapio Niemi, Santtu Toivonen, Marko Niinimäki, and Jyrki Nummenmaa. Ontologies with Semantic Web/Grid in Data Integration for OLAP. *Int. J. Semantic Web Inf. Syst.*, 3(4):25–49, 2007.

[106] Timo Niemi, Turkka Näppilä, and Kalervo Järvelin. A relational data harmonization approach to XML. *J. Inf. Sci.*, 35(5):571–601, 2009.

[107] Marko Niinimäki and Tapio Niemi. An ETL Process for OLAP Using RDF/OWL Ontologies. *Journal on Data Semantics XIII*, 5530:97–119, 2009.

[108] Natalya F. Noy and Mark A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *Int. J. Hum.-Comput. Stud.*, 59:983–1024, December 2003.

[109] Natalya Fridman Noy and Mark A. Musen. Specifying Ontology Views by Traversal. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004*, volume 3298 of *Lecture Notes in Computer Science*, pages 713–725. Springer, 2004.

[110] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1099–1110, New York, USA, 2008. ACM.

[111] Ignazio Palmisano, Valentina Tamma, Terry Payne, and Paul Doran. Task Oriented Evaluation of Module Extraction Techniques. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 130–145, Berlin, Heidelberg, 2009. Springer-Verlag.

[112] Zhengxiang Pan, Xingjian Zhang, and Jeff Heflin. DLDB2: A Scalable Multi-perspective Semantic Web Repository. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '08, pages 489–495, Washington, DC, USA, 2008. IEEE Computer Society.

[113] Byung-Kwon Park, Hyoil Han, and Il-Yeol Song. XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses. In A. Min Tjoa and Juan Trujillo, editors, *Proceedings of the 7th international conference on Data Warehousing and Knowledge discovery, DaWaK*, volume 3589 of *Lecture Notes in Computer Science*, pages 32–42. Springer, 2005.

[114] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. Web Ontology Language OWL Abstract Syntax and Semantics. *W3C Recommendation*, 2004. `http://www.w3.org/TR/owl-semantics/`.

[115] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker. A comparison of approaches to large-scale data analysis. In *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, pages 165–178, New York, NY, USA, 2009. ACM.

[116] Dennis Pedersen, Torben Bach Pedersen, and Karsten Riis. The Decoration Operator: A Foundation for On-Line Dimensional Data Integration. In *8th International Database Engineering and Applications Symposium (IDEAS 2004), 7-9 July 2004, Coimbra, Portugal*, pages 357–366. IEEE Computer Society, 2004.

[117] Dennis Pedersen, Karsten Riis, and Torben Bach Pedersen. XML-Extended OLAP Querying. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, SSDBM '02, pages 195–206, Washington, DC, USA, 2002. IEEE Computer Society.

[118] Torben Bach Pedersen and Christian S. Jensen. Multidimensional Data Modeling for Complex Data. In *Proceedings of the 15th International Conference on Data Engineering*, ICDE '99, pages 336–345, Washington, DC, USA, 1999. IEEE Computer Society.

[119] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Inf. Syst.*, 26(5):383–423, 2001.

[120] Juan Manuel Perez Martínez, Rafael Berlanga, María José Aramburu, and Torben Bach Pedersen. Integrating Data Warehouses with Web Data: A Survey. *IEEE Trans. on Knowl. and Data Eng.*, 20(7):940–955, July 2008.

[121] Cassandra Phipps and Karen C. Davis. Automating data warehouse conceptual schema design and evaluation. In Laks V. S. Lakshmanan, editor, *Proceedings of the 4th Intl. Workshop on Design and Management of Data Warehouses 2002, Toronto, Canada, May 27, 2002*, volume 58 of *CEUR Workshop Proceedings*, pages 23–32. CEUR-WS.org, 2002.

[122] Jaroslav Pokorny. Modelling stars using XML. In *Proceedings of the 4th ACM international workshop on Data warehousing and OLAP*, DOLAP '01, pages 24–31, New York, NY, USA, 2001. ACM.

[123] Padmashree Ravindra, Vikas V. Deshpande, and Kemafor Anyanwu. Towards scalable RDF graph analytics on MapReduce. In *Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud*, MDAC '10, pages 5:1–5:6, New York, NY, USA, 2010. ACM.

[124] Alan L. Rector and Jeremy Rogers. Ontological and Practical Issues in Using a Description Logic to Represent Medical Concept Systems: Experience from GALEN. In Pedro Barahona, François Bry, Enrico Franconi, Nicola Henze, and Ulrike Sattler, editors, *Reasoning Web, Second International Summer School 2006, Lisbon, Portugal, September 4-8, 2006, Tutorial Lectures*, volume 4126 of *Lecture Notes in Computer Science*, pages 197–231. Springer, 2006.

[125] María del Mar Roldán-García and Jose F. Aldana-Montes. DBOWL: Towards a Scalable and Persistent OWL Reasoner. In *Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*, ICIW '08, pages 174–179, Washington, DC, USA, 2008. IEEE Computer Society.

[126] Oscar Romero and Alberto Abelló. Automating multidimensional design from ontologies. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, DOLAP '07, pages 1–8, New York, NY, USA, 2007. ACM.

[127] Oscar Romero and Alberto Abelló. A survey of multidimensional modeling methodologies. *International Journal of Data Warehousing and Mining (IJDWM)*, 5(2):1–23, 2009.

[128] Oscar Romero and Alberto Abelló. A framework for multidimensional design of data warehouses from ontologies. *Data Knowledge Engineering*, 69(11):1138–1157, 2010.

[129] Oscar Romero, Diego Calvanese, Alberto Abelló, and Mariano Rodríguez-Muro. Discovering functional dependencies for multidimensional design. In *Proceedings of the ACM 12th international workshop on Data warehousing and OLAP*, DOLAP '09, pages 1–8, New York, USA, 2009. ACM.

[130] Oscar Romero, Alkis Simitsis, and Alberto Abelló. GEM: Requirement-Driven Generation of ETL and Multidimensional Conceptual Designs. In *Proceedings*

of the 13th International Conference on Data Warehousing and Knowledge Discovery, DaWaK 2011, Toulouse, France, August 29-September 2,2011. Proceedings, volume 6862 of *Lecture Notes in Computer Science*, pages 80–95. Springer, 2011.

[131] Lenhart K. Schubert, Mary Angela Papalaskaris, and Jay Taugher. Determining Type, Part, Color and Time Relationships. *IEEE Computer*, 16(10):53–60, 1983.

[132] Julian Seidenberg and Alan Rector. Web ontology segmentation: analysis, classification and use. In *Proceedings of the 15th international World Wide Web conference*, WWW '06, pages 13–22, New York, NY, USA, 2006. ACM.

[133] Denilson Sell, Dhiogo Cardoso da Silva, Fabiano Duarte Beppler, Marcio Napoli, Fernando Benedet Ghisi, Roberto C. S. Pacheco, and Jos Leomar Todesco. SBI: a semantic framework to support business intelligence. In *Proceedings of the first international workshop on Ontology-Supported Business Intelligence*, pages 1–11, Karlsruhe, Germany, 2008. ACM.

[134] Luciano Serafini, Alex Borgida, and Andrei Tamilin. Aspects of distributed and modular ontology reasoning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 570–575, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[135] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt, and Jeffrey F. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 302–314, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[136] Alkis Simitsis, Dimitrios Skoutas, and Malú Castellanos. Representation of conceptual ETL designs in natural language using Semantic Web technology. *Data Knowl. Eng.*, 69(1):96–115, 2010.

[137] K. Skaburskas, F. Estrella, J. Shade, D. Manset, J. Revillard, A. Rios, A. Anjum, A. Branson, P. Bloodsworth, T. Hauer, R. McClatchey, and D. Rogulin. Health-e-child: A grid platform for european paediatrics. *Journal of Physics: Conference Series*, 119, 2008.

[138] Dimitrios Skoutas and Alkis Simitsis. Designing ETL processes using semantic web technologies. In *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, DOLAP '06, pages 67–74, New York, NY, USA, 2006. ACM.

[139] Dimitrios Skoutas and Alkis Simitsis. Ontology-Based Conceptual Design of ETL Processes for Both Structured and Semi-Structured Data. *Int. J. Semantic Web Inf. Syst.*, 3(4):1–24, 2007.

[140] Dimitrios Skoutas, Alkis Simitsis, and Timos K. Sellis. Ontology-Driven Conceptual Design of ETL Processes Using Graph Transformations. *Journal on Data Semantics XIII*, 5530:120–146, 2009.

[141] Michael Spahn, Joachim Kleb, Stephan Grimm, and Stefan Scheidl. Supporting business intelligence by providing ontology-based end-user information self-service. In *Proceedings of the first international workshop on Ontology-supported business intelligence*, pages 1–12, Karlsruhe, Germany, 2008. ACM.

[142] Radhika Sridhar, Padmashree Ravindra, and Kemafor Anyanwu. RAPID: Enabling Scalable Ad-Hoc Analytics on the Semantic Web. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, pages 715–730, Berlin, Heidelberg, 2009. Springer-Verlag.

[143] Heiner Stuckenschmidt and Michel C. A. Klein. Structure-Based Partitioning of Large Concept Hierarchies. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004*, volume 3298 of *Lecture Notes in Computer Science*, pages 289–303. Springer, 2004.

[144] Heiner Stuckenschmidt and Michel C. A. Klein. Reasoning and change management in modular ontologies. *Data Knowl. Eng.*, 63(2):200–223, 2007.

[145] Jacopo Urbani, Spyros Kotoulas, Jason Maassen, Frank van Harmelen, and Henri E. Bal. WebPIE: A Web-scale Parallel Inference Engine using MapReduce. *J. Web Sem.*, 10:59–75, 2012.

[146] Panos Vassiliadis. Modeling Multidimensional Databases, Cubes and Cube Operations. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, SSDBM '98, pages 53–62, Washington, DC, USA, 1998. IEEE Computer Society.

[147] Boris Vrdoljak, Marko Banek, and Stefano Rizzi. Designing web warehouses from xml schemas. In Yahiko Kambayashi, Mukesh K. Mohania, and Wolfram Wöß, editors, *Proceedings of the 5th international workshop on Data Warehousing and Knowledge Discovery, DaWaK, Prague, Czech Republic, September 3-5,2003, Proceedings*, volume 2737 of *Lecture Notes in Computer Science*, pages 89–98. Springer, 2003.

[148] Cathrin Weiss, Panagiotis Karras, and Abraham Bernstein. Hexastore: sextuple indexing for semantic web data management. *Proc. VLDB Endow.*, 1(1):1008–1019, August 2008.

[149] Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, and Dave Reynolds. Efficient RDF Storage and Retrieval in Jena2. In Isabel F. Cruz, Vipul Kashyap, Stefan Decker, and Rainer Eckstein, editors, *Proceedings if the first International Workshop on Semantic Web and Databases, SWDB, Co-located with VLDB 2003, Humboldt-Universität, Berlin, Germany, September 7-8, 2003*, pages 131–150, 2003.

[150] Yu Xu and Yannis Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 527–538, New York, NY, USA, 2005. ACM.

[151] Hilmi Yildirim, Vineet Chaoji, and Mohammed J. Zaki. GRAIL: scalable reachability index for large graphs. *Proc. VLDB Endow.*, 3(1-2):276–284, September 2010.