Aspects of ontology visualization and integration

Dmitrieva, J.B.

Aspects of Ontology Visualization and Integration

Julia B. Dmitrieva

Acknowledgement Biorange NBIC
Drukker Ridderprint BV

Aspects of Ontology Visualization and Integration

PROEFSCHRIFT

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof. mr. P.F. van der Heijden,
volgens besluit van het College voor Promoties
te verdedigen op woensdag 14 September 2011
klokke 16.15 uur

door

Joelia Borisovna Dmitrieva

Geboren te Moskou, Rusland, in 1971

Promotiecommissie

Promotor

Prof. Dr. J.N. Kok

Co-promotor

Dr. Ir. F.J. Verbeek

Overige Leden

Prof. Dr. T. Bäck	Leiden University
Prof. Dr. F. de Boer	Leiden University
Prof. Dr. J.B.T.M. Roerdink	Groningen University

Contents

Contents iii

List of Tables vii

List of Figures ix

1 Introduction 1

1.1 Motivation 1

1.2 OWL and Description Logics 3

1.2.1 Description Logics Basics 4

1.2.2 Different Description Logics 7

1.2.3 Reasoning 9

1.3 Ontology Visualization 9

1.4 Ontology Modularity and Integration 13

1.4.1 E–connections 15

1.4.2 Distributed Description Logic 15

1.4.3 Integration using OWL:imports Construct 16

1.4.4 Modularity 17

1.4.5 Ontology Mapping 17

1.5 Structure of the Thesis 18

2 Ontological Context Visualization 21

2.1 Introduction 23

2.2 Background 24

2.2.1 Ontology Integration 24

2.2.2 Definitions 25

2.3 "ContextVis": Proof of Concept 26

2.4 Knowledge Acquisition 27

2.4.1 Global Ontological Context 28

2.4.2 Local Ontological Context 28

2.4.3 Reasoning 29

2.4.4 Database Representation 31

2.4.5 Relationships 32

2.4.6 Concept Expansion 33

2.5 Visualization 34

2.6 Implementation Details 34

2.7 Conclusion, Discussion and Further Work 37

3 Visualization of Ontology 39

3.1 Introduction 41

3.2 Related Work 43

3.2.1 Containment Methods 44

3.2.2 Node-Link Methods 48
3.3 Requirements 53
3.4 Transformation to Visual Representation 55
3.4.1 Graph Generation 56
3.4.2 Hierarchical Tree Representation 58
3.5 Representations of Views Based on Different Relations 59
3.6 Conclusions 60

4 Node-Link Methods in Ontology Visualization 67

4.1 Introduction 69
4.2 Related Work 70
4.3 GUI Aspects of Node-Link Visualization 75
4.4 Representations of Views Based on Different Geometry 76
4.4.1 Euclidean Views 76
4.4.2 Hyperbolic Views 77
4.4.3 Klein Model 79
4.4.4 Euclidean variant of H3 Layout 83
4.4.5 Poincaré Disk Model Description 85
4.4.6 Poincaré like layout 87
4.4.7 Spherical Geometry 90
4.4.8 Results with Node-Link Approach 94
4.5 Implementation Details 96
4.6 Conclusions 96

5 Ontology Visualization with Containment Method 99

5.1 Introduction . . . 101

5.2 Related Work . . . 101

5.3 Spherical Variant of Containment . . . 104

5.3.1 Description of Visualization Algorithm . . . 104

5.3.2 Semantic Zoom . . . 107

5.3.3 Results with Containment Approach . . . 107

5.4 Conclusions . . . 109

6 Integration of Modules 111

6.1 Introduction . . . 113

6.2 Related Work . . . 115

6.3 Module Extraction . . . 117

6.3.1 Modules from Enriched Signature . . . 118

6.3.2 Fixpoint Modules . . . 120

6.3.3 Properties of Fixpoint . . . 123

6.4 Ontology Mapping . . . 127

6.5 Integration Information from Ontologies . . . 129

6.5.1 Solving Unsatisfiable Classes in Merged Pairs . . . 129

6.5.2 Solving Unsatisfiable Classes in Integrated Ontology . . . 134

6.6 Conclusions . . . 137

7 Conclusions, Discussion and Future Directions 141

7.1 Ontological Context Visualization . . . 142

7.2 Representation of Properties in Ontology Visualization 143

7.3 Role of Different Geometries in Node-Link Ontology Visualization . . 144

7.4 Exploration of Ontology Hierarchy with Semantic Zoom 145

7.5 Building of a new Ontology 146

7.6 Ontology Integration 146

7.7 Conclusions 147

7.8 Future Directions 148

List of Tables

1.1 Syntax and semantics of concept and role-forming constructors 6

3.1 Evaluation of Ontology Visualization Methods 53

6.1 The size of the modules after reaching fixpoint 121

6.2 Number of unsatisfiable classes in the merged pairs of modules 130

List of Figures

1.1 The part of the RDF graph for the concept Organic_Cation_Transporter 12

2.1 Architecture of the "ContextVis" 27
2.2 Creation of the Global Ontological Context 29
2.3 Depth-First search traversal 30
2.4 Explanation with Protégé 4 31
2.5 Relational Database Schema for "ContextVis" 31
2.6 Global Context Around "Alzheimer" 35
2.7 Local Context Around ADP 36

3.1 CropCircles 45
3.2 Zooming with CropCircles 46
3.3 Jambalaya 47
3.4 OntoSphere 49
3.5 OntoRama 51
3.6 TGVizTab 52
3.7 Acute_Adult_T-Cell_Lymphoma_Leukemia Manchester syntax 57
3.8 Dolce-Lite with all properties 61

3.9 Dolce-Lite with sub/super class relations 62

3.10 Antigen Presenting Cell with all properties 63

3.11 Antigen Presenting Cell with sub/super class relations 64

4.1 H3Viewer 72

4.2 Hyperbolic Browser 73

4.3 Botanical Tree of Kleiberg et. al 74

4.4 Euclidean 2D and 3D Views 78

4.5 Klein model 80

4.6 Euclidean variant of the calculation of ϕ and θ 82

4.7 Transformations in the Klein model 84

4.8 Transformations in Poincaré model 88

4.9 Determining R and α 89

4.10 Poincaré and Poincaré-like layout 91

4.11 Stereographic layout 92

4.12 Mapping of the graph structure to sphere surface 93

4.13 Möbius Transformations in Stereographic View 95

4.14 Multi-View 97

5.1 TreeMaps 103

5.2 Procedure for determining node R 105

5.3 Semantic zoom with Onto-Earth 108

5.4 More detailed view Onto-Earth 108

5.5 Detailed view Onto-Earth 109

6.1 Visualization of the concept Toll-like receptor 114

6.2 Fixpoint 121

6.3 Algorithm that finds fixpoint modules for different ontologies 122

6.4 Chain of Fire for Multiple Ontologies 125

6.5 Explanation of Gene unsatisfiability 1 131

6.6 Explanation of Gene unsatisfiability 2 132

6.7 Explanation of Chromatin unsatisfiability 1 133

6.8 Explanation of Binding unsatisfiability 134

6.9 Explanation of Translation unsatisfiability 135

6.10 Explanation of Nucleic_Acid_Binding unsatisfiability 136

6.11 Explanation of Chromatin unsatisfiability 2 138

Chapter 1

Introduction

1.1 Motivation

The subject of this thesis is Ontology Visualization and Integration. Although these two topics contain more than enough challenges to be investigated separately, in this thesis they are combined, because ontology integration process could be triggered, influenced, improved and simplified by ontology visualization.

Ontology is a specification of the conceptualization of domain of discourse [67]. This specification provides a formal description of concepts and relations in the domain. The area of ontology application is very broad, it comprises industry [64], E-commerce [41], E-learning [38], law and legal reasoning [116, 117], geosciences [98], health care [82], astronomy [45], life sciences [87, 101], to mention a few. The domain that will be emphasized in this thesis is that of the life sciences and bioinformatics specifically. The reason for this choice is that currently biology, chemistry and medicine produce increasing body of data that has to be organized before it can be represented and consumed as knowledge; the use of ontologies seems to be necessary and unavoidable in these domains.

During the past decades, research in the life sciences has to deal with the phe-

nomenon of data explosion. Data from genomics and proteomics after the achievements in sequencing technology are becoming available for analysis in large quantities. However, without the application of an appropriate information management technology data are less valuable and more difficult to maintain over time. The Semantic Web [34] has emerged in response to needs of researchers and organizations from different areas with introduction of RDF [66], RDFS [24], OWL [19], SWRL [71] and other technologies which are indispensable in the modern data management systems. Therefore, the Semantic Web, including ontology technology, is broadly accepted by the life sciences community. The W3C recommendation for knowledge representation in the Semantic Web is Web Ontology Language (OWL).

Description Logics [33] (DLs) – which are the logical underpinning of OWL – are decidable fragments of the First Order Logic (FOL), hence, for a life scientist OWL is not easy to learn and understand. Therefore, it will be very helpful if an ontology could be represented in a more intuitive way than just collection of logical statements about a domain of discourse. For representation of ontology from the logical point of view ontology editing tools, like Protégé [21] or Swoop [78], can be used. However, when an abstraction of the ontology needs to be viewed, ontology visualization tools are more suitable. An ontology visualization tool does not need to show the logical representation of entities, such as concept definitions by means of DLs constructors. We argue that it has to be complementary to the ontology editing tools. It has to represent important parts of the domain knowledge in a simple way, without directly confronting the viewer with logics. The hierarchy of the ontology, the concepts in the domain, and the properties must be visible. A visualization should include interactions to allow expansion and navigation op concepts in the knowledge space.

In addition to visualization also integration of ontologies is an essential research area. The ontology integration plays important role in the integration of heterogeneous resources, such as databases which are spread on the Internet. Moreover, by means of integration the ontologies can be developed in an orthogonal way,

in the sense that one term is defined once and then reused in different ontologies. Related terms could be connected to each other through *bridges* [36] or *links* [83]. This is an ongoing field of research that embraces different areas at the same time, e.g. ontology mapping [76], ontology alignment [44], $\mathcal{E}$–connections [83], Distributed Description Logic [36], ontology modularity and segmentation [62, 102], and other related subjects.

The research questions to which this thesis is dedicated are the following:

- How an ontology can be visualized. The visualization has to be considered from perspective of the user, efficiency and implying restrictions, so that a practical visualization will be realized.
- How to create an ontological context around a specific term or concept and how to visualize this.
- How to create a new ontology on the basis of interesting terms or concepts from different ontologies.

In order to make this thesis complete and readable, in the remainder of this chapter, we will introduce backgrounds related to the fields of the OWL language, ontology visualization and integration.

1.2 OWL and Description Logics

Ontologies are used to describe a domain of discourse in an unambiguous and formal way. They play an important role in data representation and integration processes. The Web Ontology Language (OWL) [19] is a World Wide Web Consortium (W3C) recommendation for knowledge representation in the Semantic Web. There are three sublanguages (species) of OWL, namely OWL Lite, OWL-DL and OWL Full. These three sublanguages differ in their expressive power and reasoner support. Different

communities have accepted OWL-DL for reasons of language expressivity and decidability. OWL-DL provides high expressive power for knowledge representation, thus, very complex domains, like the biological domain [108], can be modeled with OWL-DL. At the same time it is decidable; that is according to Horrocks et al. [72], there exists an algorithm that could guarantee to determine whether or not one OWL ontology entails another. Consequently, algorithms can be developed in order to reason in the knowledge bases that are represented in OWL-DL. Moreover, through restricting the language to certain constructors, the efficiency of these algorithms can be guaranteed.

Description Logics [33] are indispensable for OWL, because DLs provide the logical underpinning and formal semantics for OWL. Without this semantics there will never be agreement between humans and/or computers regarding the precise meaning of statements in a knowledge base.

1.2.1 Description Logics Basics

A knowledge base in a DL consists of three parts that are referred in literature to as TBox (terminology), ABox (assertions) and RBox (role assertions). The vocabulary (signature) of a TBox is build from atomic concepts [1] (unary predicates) $A, B, \ldots$, denoting sets of individuals, and atomic roles (binary predicates) $R_1, R_2, \ldots$, denoting binary relationships between individuals. Furthermore, different constructors are available in order to define complex concepts.

The most simple logic is referred to as $\mathcal{EL}$, where a concept [2] can be defined

[1] Atomic concepts (atomic roles) are elementary descriptions which are used to build complex descriptions. They can be considered as terminal symbols in production rules of a formal grammar.

[2] The terms *concept* and *class* are used interchangeably by OWL as well as by DL community. In this thesis we will also not make a distinction between these two terms.

by means of the following constructors:

$\top$	(universal concept)
A	(atomic concept)
$C_1 \sqcap C_2$	(intersection)
$\exists R.C$	(existential restriction)

The meaning of a Description Logic knowledge base is specified via model-theoretic semantics which maps each individual, concept and relation to the elements of the domain $\Delta^{\mathcal{I}}$. This mapping is defined by means of an interpretation function denoted as $\cdot^{\mathcal{I}}$. This interpretation function assigns to each atomic concept A, an non-empty set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to each atomic role R, a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

In the consecutive chapters different DL constructors will be addressed. Therefore, to make this thesis self-contained, different constructors for making complex concepts and roles with the corresponding semantics are provided in Table 1.1.

Besides concept and role constructors a DL knowledge base consists of a set of axioms:

Generalized Concept Inclusion (GCI) Generalized Concept Inclusion asserts that one class C is a subclass of another class D ($C \sqsubseteq D$). The subset axiom $C \sqsubseteq D$ means that belonging to D is necessary condition to belong to C, e.g $Toll\_Like\_receptor \sqsubseteq Pattern\_Recognition\_Receptor$ [35] [3].

Equality The Equality axiom $C \equiv D$ can be rewritten as two GCI axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. The equivalence axiom means that belonging to D is necessary and sufficient condition to belong to C, e.g. $RNA \equiv Ribo\_Nucleic\_Acid$ or $Acute\_Leukemia \equiv Leukemia \sqcap$
$\forall\ Disease\_Excludes\_Finding.Chronic\_Clinical\_Course$ [28].

Concept Assertion With the Concept Assertion axiom $C(a)$ one can state that

[3] Toll-like receptors (TLRs) are responsible for recognition of pathogen molecules and activation of immune cells.

Constructor Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$
top	$\top$	$\triangle^{\mathcal{I}}$
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\triangle^{\mathcal{I}} \setminus C^{\mathcal{I}}$
universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$
existential quantification ($\mathcal{E}$)	$\exists R.C$	$\{d_1 \mid \exists d_2 : (d_1, d_2) \in R^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}$
number restriction ($\mathcal{N}$)	$(\geq nR)$	$\{d_1 \mid \sharp(d_1, d_2) \in R^{\mathcal{I}} \geq n\}$
	$(\leq nR)$	$\{d_1 \mid \sharp(d_1, d_2) \in R^{\mathcal{I}} \leq n\}$
collection of individuals $\mathcal{O}$	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$
role name	P	$P^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$
role conjunction	$Q \sqcap R$	$Q^{\mathcal{I}} \cap R^{\mathcal{I}}$

Table 1.1: Syntax and semantics of concept and role-forming constructors. A represents an atomic concept, C and D represent concept descriptions which can be built from other concepts by means of concept constructors. R is an atomic binary role, P and Q are complex roles.

one individual a belongs to a certain class C, for example with the statement $Prion(PrP^C)$ one can state that the prion [4] with the name PrP^C belongs to the class of prions $Prion$.

Role Assertion With the Role Assertion axiom one can state that an individual a is in relation R to an individual b $R(a, b)$, for example with the following statement $Infectious\_isoform\_of(PrP^{Sc}, PrP^C)$ we can define that the scrapie prion PrP^{Sc} is an infectious isoform of the normal prion protein PrP^C.

1.2.2 Different Description Logics

Dependent on the constructors used, different logics exist that vary in expressive power and reasoning support. The logic $\mathcal{ALC}$ ($\mathcal{AL}$ = attributive language, $\mathcal{C}$ = complement) extends the logic $\mathcal{EL}$ with the concept negation $\neg C$.

The $\mathcal{S}$ logic extends the $\mathcal{ALC}$ with the transitivity axiom for atomic roles $Trans(R)$. The transitive roles allow objects to be in relation to each other if they are in the transitive closure (R^+) of the role defined as the transitive. Example: by stating that the role has_Part is transitive, from the statements $DNA \sqsubseteq \exists has\_Part.Deoxynucleotide$ and $Deoxynucleotide \sqsubseteq \exists has\_Part.Deoxyribose$ it can be inferred that $DNA \sqsubseteq \exists has\_Part.Deoxyribose$.

The logic $\mathcal{SHOIQ}$ [73] extends the logic $\mathcal{S}$ with the role hierarchies $\mathcal{H}$ ($R_1 \sqsubseteq R_2$), with the role inversion $\mathcal{I}$ (r^-), with qualified number restrictions $\mathcal{Q}$ ($\geq nS.C$) and with nominals $\mathcal{O}$ ($\{a\}$). The nominals provide a possibility to make a concept from a singleton set $\{a\}$ containing an individual a. The qualified number restrictions allow to describe statements such as "protein has more than 50 signaling proteins as ligands" ($14\text{-}3\text{-}3 \sqsubseteq Protein \sqcap \geq 50\ has\_Ligand.\ Signaling\_Protein$). Logics which have support for datatypes and values (D) (e.g., integer, string, float datatypes, and values such as "35") provide a possibility to restrict the values of

[4] Prion is an infectious agent composed of a misfolded protein [96]

properties to some certain values: e.g. $Russian\_Alphabet \sqsubseteq has\_Letters.33$ or to subset of some datatype $Person \sqsubseteq hasSecurityNumber.string$.

The ontology language OWL-DL corresponds with $\mathcal{SHOIN}$ logic where $\mathcal{N}$ is a cardinality restriction ($\geq nR$). The cardinality restrictions allow to define concepts such as "protein has at least seven isoforms" (14-3-3 $\sqsubseteq Protein \sqcap \geq 7$ $has\_Isoform$).

The underlying description logic for the OWL 2 [43] language that has become a W3C recommendation in October 2009 is $\mathcal{SROIQ}$ [70]. This logic extends the $\mathcal{SHOIQ}$ Description Logic with the following axioms:

disjoint roles For example, the roles $part\_of$ and has_part should be defined as being disjoint.

reflexive roles The role $self\text{-}contradicts$ is an example of reflexive role.

irreflexive roles $proper\_subset\_of$ is an example of irreflexive role.

negated role assertions can be used to declare that one individual is not in relation with another individual. For example, we can state that malaria is not transmitted by yellow fever mosquito
$(malaria, Aedes\_aegypti) : \neg transmitted\_by$.

complex role inclusion axioms are the axioms of the form $\mathcal{R} \circ \mathcal{S} \sqsubseteq R$. This type of axioms allows to propagate one property along a chain of other properties, for example in the modeling of pathways we can state that if a molecule regulates a reaction that is part of a pathway then this molecule also regulates the pathway $regulates \circ part\_of \sqsubseteq regulates$.

local reflexivity $\mathcal{SROIQ}$ allows to declare concepts of the form $\exists R.Self$ that can be used to define a local reflexivity, with this expression, for instance, we can make a statement about prions that induce production of other prions [96]. $Prions \sqsubseteq \exists induce\_forming.Self$

1.2.3 Reasoning

The OWL language provides a toolset for modeling concepts and relationships. In order to query a model and to make implicit knowledge explicit, reasoning services are required. This reasoning support is provided through Description Logic reasoners, such as Pellet [17], FaCT++ [111], HermiT [5].

The main components of reasoning in ontologies are the following:

Consistency The TBox $\mathcal{T}$ is consistent if there is a non-empty model, or in other words: this model makes sense.

Classification The classifying TBox means finding all implicit sub/super class relationships that could be inferred from the given TBox.

Subsumption($C \sqsubseteq D$) determines whether the concept C is more specific than the concept D.

Concept Satisfiability is the problem of checking whether there exists an interpretation in which the given concept denotes a non-empty set.

Equivalence ($C \equiv D$) Two concepts C and D are equivalent if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of the given TBox $\mathcal{T}$.

Disjointness Two concepts C and D are disjoint if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model $\mathcal{I}$ of the TBox $\mathcal{T}$.

Realization Realization finds the most specific classes that the given individual belongs to.

1.3 Ontology Visualization

To date, the majority of research and development in ontology is devoted to fields such as OWL language, ontology editors, and DL reasoning. The field of ontology

visualization, however, does not get an equal amount of attention from the ontology community. This lack of interest can be explained from different points of view. First, the ontologies with their elaborate logical structure are not suitable for visualization. It is very difficult, a challenge in itself, to realize how a set of First Order Logic axioms can be represented in a visualization. Second, different users experience ontology in a different way. One person sees an ontology as a network of nodes which represent concepts and edges which represent relations, whereas another can consider an ontology as a hierarchical structure. Yet another user is interested in individuals only. A researcher from the RDF [66] community sees an ontology as a bunch of triples, and there are also users who envision an ontology as a sharable thesaurus or controlled vocabulary. Thus, this multiple perspective provides the difficulties for defining a requirements for a visualization approach.

Most visualization applications consider an ontology as a graph or a tree. The graph paradigm is used in the node-link approach. On the other hand, the tree paradigm is used in the containment approach. When represented as a graph/tree all graph/tree visualization techniques could be considered for application in ontology visualization. A graph, for example, can be visualized with a Force Directed Placement [52] algorithm, and a tree can be visualized with the Cone Tree method [99].

The motivation for the work represented in this thesis was to develop an ontology visualization that represents an ontology as an abstraction of the formal model. In this visualization one can be more interested in the domain knowledge than in the logical representation. Typical questions that could be asked about an ontology might be following:

- what are the classes in this ontology,
- what is the underlying hierarchical structure,
- what kind of relationships exist between classes.

This representation could be interesting for a person that is searching for an ontology

in order to reuse it in an application; this person does not want to be confronted with logical axioms at this stage of the development. The typical application area for this kind of "naive" ontology representation can be data annotation, e.g. annotation of biological experimental data such as genes, proteins, images [77] with ontology terms. The simple visual representation of ontology can also be useful in guiding a query process. For example, a user would like to make a query in a database, then the global view of the ontology can make the user knowledgeable about the terminology in the underlying database, whereas the local view on some concept with a context information, such as super/subclasses, can help the user to formulate a query in the correct way.

Before an ontology can be visualized it must be represented in an amenable way for visualization, i.e. a graph or a tree structure. An OWL ontology is serialized [5] in RDF/XML format. The RDF structure can be represented as a graph. This graph, however, reflects the syntactic features of the ontology, and has very little to do with the semantics of the ontology. A developer of a visualization method, however, has to consider a representation that as much as possible corresponds with the semantics of the ontology. For example, the concept $Organic\_Cation\_Transporter$ from the NCI Thesaurus [28] ontology can be represented by the RDF graph structure depicted in Figure 1.1. The problem with the RDF graph representation is that each statement needs to be converted to a set of triples. This inevitably leads to the explosion of the triples for one simple statement. If the application developer selects this representation for a visualization tool, the ontology will not be clear, will not be understandable, and will not represent implicit semantics as this will be hidden behind a chain of triples with lots of anonymous resources (*blank nodes*). Therefore, other ways for the graph ontology representation need to be explored.

For aforementioned reasons, in the visualization approach described in this thesis a model is chosen that is abstracted from each OWL language serialization, as well as abstracted from the axiomatic ontology model. Therefore, this model

[5] Serialization is a representation of a data structure in a format for storage.

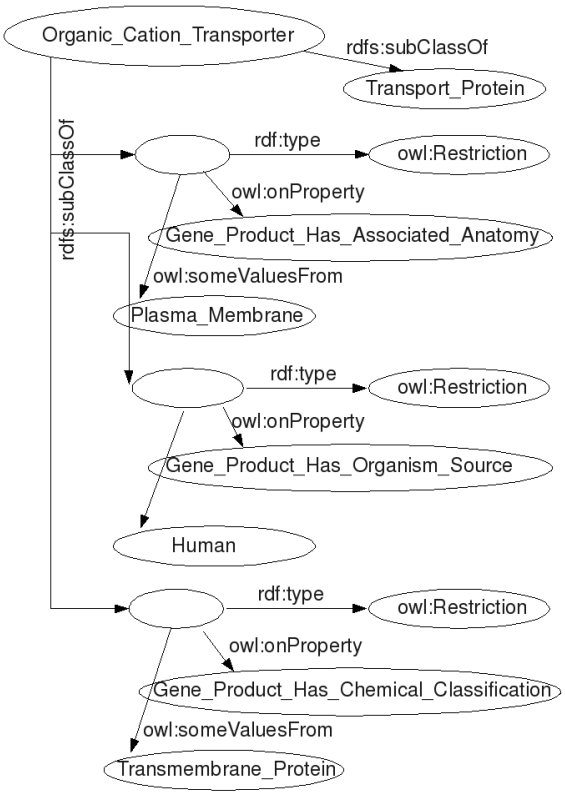

Figure 1.1: The part of the RDF graph for the concept Organic_Cation_Transporter

does not contain intersections, complements, union, and other logical statements. The model does contain class hierarchy and properties. In this representation an ontology is converted to a graph, where classes are connected to each other with links which represent either hierarchical property, or other domain dependent properties. In the literature this representation is referred to as the node-link method.

This approach has, however, a drawback; it represents a very simplified model of an ontology. Moreover, it seems not possible to visualize all properties for each class, because the lower the class is in the hierarchy the more properties it inherits from its ancestors. This results in an exponential explosion of nodes, which have to be cloned, or edges, which have to be added in the visualization. From this we can conclude that it is still not clear how to visualize an ontology and whether or not we need to represent other properties besides hierarchy. A more detailed discussion of advantages and disadvantages of properties in ontology visualization will be given in subsequent chapters of this thesis.

Another visualization approach described in this thesis builds on the basis of the ontology hierarchy. This representation elaborates on the containment methods, such as treemaps [104] and CropCircles [113]. The containment method uses the paradigm of *inclusion* that is realized by representing children entities inside the parental entities. The ontology hierarchy fits almost perfectly in this paradigm. A class in an ontology represents, namely, a set of things in a domain, and sublasses are subsets which are contained in the parental superset. This method, however, is not suitable for the representation of properties.

1.4 Ontology Modularity and Integration

Besides ontology visualization, the problem of ontology reuse and integration is considered in this thesis.

In the past decades, the life sciences are confronted with a proliferation and

expansion of biological and biomedical data that are represented in rather heterogeneous form. To deal with this problem data needs to be managed in an intelligible way. Frequently, data is spread between different resources, e.g. data bases, XML records, text files, images, etc. Ontologies provide the necessary technology for information management and integration. Here, the concept *integration* is used in a broad and an informal way: the meaning of this concept is the result of a process where heterogeneous resources are connected to each other and a query can be formulated in *one* language. The result of this query is generated on the basis of integrated information acquired from the different resources.

In the case of the integration of databases the following scenario can be used: each local database has a local model that could be described with a local ontology, while a global ontology describes the user's view on the domain. This global view is used in order to query the databases. The integration method [39] in this case describes each entity from the global model in the terms of the local models. This method is called *Global as View* (GAV). Another approach is to describe each entity of the local model in terms of the global model, this is known as *Local as View* (LAV) model. In addition, a hybrid method ($GLAV$) is introduced in which these two methods are combined.

An other application in the area of the ontology integration is *reuse* of a *foreign* ontology during the modeling process. For example, a biologist modeling the protein–protein interaction domain is not familiar with the chemicals which are defined in the ChEBI [6] ontology, however he/she would like to reuse entities from the ChEBI ontology. This approach is preferable above the reduplication because at first, the developer does not introduce the entities which already exist, and second the user does not need to model the knowledge where he/she is not a specialist in.

[6]Chemical Entities of Biological Interest (ChEBI): http://www.ebi.ac.uk/chebi/

1.4.1 E–connections

There exist different approaches devoted to ontology integration. One of them is the $\mathcal{E}$–connections [83]. This is a framework developed in order to combine different Abstract Description Systems, such as Description Logics, Modal Logics, and different logics of time and space. The $\mathcal{E}$–connections provide the possibility to connect ontologies by means of *link properties* which describe the relationships between connected knowledge bases. This formalism is suitable when different ontologies need to interoperate in one application. The ontologies in this case describe disjoint domains. The $\mathcal{E}$–connections are proposed as extension of the OWL and are integrated into the Swoop [78] ontology editor. In addition, the reasoning support is developed [63].

1.4.2 Distributed Description Logic

Although $\mathcal{E}$–connections is a very important formalism for the ontology integration, it is not suitable for the ontologies describing knowledge bases with overlapping domains. This is due to the fact that the *link properties* are not supposed to describe generalization or subsumption between concepts in different ontologies. However, it is exactly what we need when dealing with close and overlapping domains, where one ontology is used as a reference and an another ontology reuses some subset of entities from this particular reference ontology. To this respect one can think about the Foundational Model of Anatomy (FMA) [100] ontology as a reference, and some biomedical ontology as a specialization of some subset of concepts, such as heart, long, from FMA. For such particular requirements, the research community has provided an alternative mechanism, i.e. Distributed Description Logic (DDL) [36]. With this framework different ontologies can be interconnected by means of *bridge rules* which are used to express the generalization or equivalence relationships between entities in different ontologies. Besides the introduction of the *bridge rules* formalism, the reasoning support is provided by means of the *distributed*

reasoning services in which the standard Description Logics reasoning procedures as Satisfiability and Subsumption are realized for interlinked ontologies.

1.4.3 Integration using OWL:imports Construct

Despite the fact that DDL and $\mathcal{E}$–connections provide very good possibilities for ontology integration, they are not a standard for ontology integration on the Semantic Web. The standard way to integrate ontologies in OWL is through the usage of the $owl : imports$ construct. By means of this construct all axioms from the referenced ontology are imported in the original ontology. This method, however, suffers from a number of drawbacks. First, if ontology O_1 imports ontology O_2, and ontology O_2 imports ontologies O_3 and O_4 all ontologies from the transitive closure [7] need to be loaded in the reasoning tool, i.e. FaCT++ [111] or Pellet [105], in order to provide reasoning about entities in the integrated ontology. The consequence of this import will be high computation cost. Second, an ontology developer when using concepts from *foreign* ontology will be overwhelmed with the knowledge content where he is not a specialist in. Moreover, $owl : imports$ might damage an ontology [58]. For example, if ontology O_1 imports ontology O_2 and these two ontologies describe overlapping domains then new axioms about concepts in O_2 could be entailed from the merged ontology $O_1 \cup O_2$, including the fact that some classes become unsatisfiable. Whether these entailments are intended or not depends on the developer's choice, in either case the developer has to be aware of the consequences of the use of $owl : imports$.

[7] Ontologies could be considered as nodes in a directed graph where edges are determined by the relation $owl : imports$. Thus, the node O_i has directed edge to the node O_j if the ontology O_i imports the ontology O_j. The transitive closure of this graph will be the new graph where a node v has an edge to a node w if there is a directed path between these nodes.

1.4.4 Modularity

The usage of *owl* : *imports* construct could be problematic, we can, however, rely on it while combined with modularity. If the developer is only importing a subset of a *foreign* ontology, also referred to as a module, which is dedicated to concepts of specific interest only, then the modularity and *owl* : *imports* mechanisms can coexist. In order to realize this modularity approach the module has to be extracted. There are two trends in the research of modularity. The first one is the structural approach [90, 102], which considers an ontology as a graph. The module is then represented as a subgraph. The second one, is the logical approach [75], where a module is extracted on the basis of safety and locality principles. The second method is preferable from our point of view because it provides modules which are concise and logically correct, thus contain all necessary axioms and at the same time remain minimal.

1.4.5 Ontology Mapping

Independent of the reasons for the integration or the method that is chosen for the integration, the first step that has to be accomplished is to find mappings between entities in the ontologies that have to be merged. In a recent survey [76] Kalfoglou et al. describe the *state of the art* in the area of ontology mapping. They define the ontology mapping as a morphism, which consists of collection of functions assigning the symbols used in one vocabulary to symbols of the other. The closely related research domains that need to be mentioned in context of ontology mapping are ontology alignment and ontology articulation. Ontology alignment is defined by the authors as a task of establishing a collection of binary relations between the vocabularies of two ontologies. Although this research area is still in a state of infancy, its existence has already proven its importance by starting the Ontology Alignment Evaluation Initiative (OAEI) [8].

[8] http://oaei.ontologymatching.org

The alignment of ontologies is dedicated to the search for interrelationships between ontologies. Mostly these are equivalences or subsumptions between concepts from different ontologies. In order to find these interrelationships between entities different possibilities are provided. Some methods are dedicated to the linguistic similarities which are based on similarities between labels, descriptions, identifiers, synonyms, etc. Other methods are going a step further and augment the linguistic similarities with the structural similarity; in this case also the structural properties of an ontology, such as sub/superclasses and domain dependent properties, play an important role. In this context, ontology articulation is defined as a way in which the merging of ontologies has to be carried out.

1.5 Structure of the Thesis

All considered, in this thesis we will describe and discuss methodologies for ontology visualization and integration. Two visualization methods will be elaborated. In one method the ontology is visualized with the node-link technique, and with the other method the ontology is visualized with the containment technique. To that end, first a method is introduced that transforms an OWL ontology to a graph structure, and subsequently the requirements are discussed that have been used as a guidance during the development of the visualization approach. In the node-link technique, it is investigated how different geometrical representations can contribute to the presentation of an ontology. The geometrical representations are applied in such a way that one can switch between them. This requires the study of these transformations. The outcome makes the visualization more flexible as one is not restricted to one particular representation at a time. The containment technique has, to date, not been applied in 3D for ontology visualization. The 3D approach for the containment technique is elaborated and for the 3D context, projection of the ontology data on a sphere is developed. This requires development of transformations while well-known data interaction techniques, taken from a requirements

study, are embedded in the 3D setting. This spherical alternative to containment visualization is new to this visualization approach. Ultimately, using the transformation of the ontology to the graph visualization engines can be developed with which one can seamlessly switch from the node-link to the containment technique. The developments described in this thesis are a prelude to that. Two integration approaches will be introduced. The first one is devoted to visualization of ontological context generated from different ontologies on the basis of a concept of interest, and the second one is devoted to creation of a new ontology on the basis of modules extracted from different ontologies.

The structure of this thesis is as follows: In Chapter 2 an approach is presented where the *ontological context* for a concept can be created from different ontologies. In Chapter 3 the ontology visualization will be discussed and an approach for visualization of ontology with node-link and containment methods will be proposed. In Chapter 4 an approach for a representation of an ontology with different geometries is presented. To that end a number of other geometrical representations are introduced. In Chapter 5 the containment method is discussed and extensions for ontology visualization with this method are elaborated. Here the containment method is put in a 3D context and the sphere is used as the basis for the containment. In Chapter 6 a methodology is described where a new ontology is generated on the basis of the integrated modules. The general conclusion and discussion is represented in Chapter 7.

Chapter 2

Ontological Context Visualization

Based on:

"Ontological Context Visualization"

Julia Dmitrieva, Yun Bei and Fons J. Verbeek

published in proceedings of the Third International Workshop OWL: Experiences and Directions (OWLED 2007)

Abstract Ontologies are logical structures representing domain knowledge by means of modeling things in the domain (concepts and relations between these things). In life sciences domains frequently overlap. Hence, the concepts from this overlap can be used as a glue to integrate information from related domains. In support of our efforts to represent information on interesting concepts, we have developed a visualization engine "ContextVis". Meanwhile, this tool illustrates a proof of concept for a visual knowledge mining and inspection of ontology integration. In this chapter we present the Ontology Context Visualization with a case study on Alzheimer disease; a typical example in which contextual information on concepts allows collecting knowledge from different resources.

2.1 Introduction

In the life sciences much effort is put in accumulating all existing knowledge and creation of a domain model. This model might be represented in different ways, such as text, database, UML [31] diagrams, Petri-nets [93], Semantic networks [106], conceptual graphs [107], RDF [66] triples and ontology. Although there are so many possibilities for modeling, the preferable way for knowledge representation–when the reusability, sharing, human/machine understanding and reasoning are important–is by means of an ontology as represented in OWL [19]. The OWL-DL species [1] of the OWL language is sufficiently expressive to model different complex domains, such as life sciences, and at the same time it is decidable [2] from the perspective of the reasoning support. Hence, it is not strange at all that the OWL-DL was embraced by the knowledge engineers that are active in the life sciences [108].

The possibility to model biomedical knowledge by means of the OWL language has resulted in producing huge knowledge bases, i.e. GALEN [97] or NCI-Thesaurus [28]. These knowledge bases are difficult to process, because operations such as querying with SPARQL [27] and reasoning [23, 17, 111] require a lot of computational resources. In [110] Tobies has shown that the reasoning in $\mathcal{SHOIN}$ which corresponds with the ontology language OWL-DL is NExpTime–complete. This has the clear consequence that the reasoning in large ontologies can be time consuming. Although large ontologies have advantages as to completeness, yet they might be considered as a burden by the community. Therefore modularity [81, 61] and segmentation [102] approaches emerged in order to make such large ontologies manageable and useful for the community. Modularity allows for reusing parts of existing ontologies. We investigate this reuse by means of a visualization.

The motivation for the work presented in this chapter is to provide a straight-

[1] OWL specification includes 3 sublanguages: OWL Lite, OWL DL, OWL Full. These different OWLs are frequently referred to as species in W3C.

[2] $\mathcal{SHOIN}$ it the logic underlying OWL-DL. It was shown in [110] that the complexity for this logic is NExpTime–complete.

forward visualization of *ontological context* [47]. Therefore, we start by introducing the concept of *ontological context*.

Definition 2.1.1 (Ontological Context)**.** *Ontological Context is information extracted from different ontologies on the basis of a concept of interest. This information is represented as collection of subgraph structures generated from different ontologies.*

This particular approach can be considered as a step in the direction of combining ontology visualization and integration. We introduce a way to develop an intuitive understanding through a visual representation of extracts/contexts that are dedicated to a concept of interest. With the existing approaches this can not be realized, because ontology integration and visualization are different topics which are dedicated to its own area of research and not often combined in the same field of research and development.

2.2 Background

In the ontology community the concept of *ontology integration* is not unambiguously defined. Hence, a very broad range of topics are brought under the umbrella of ontology integration. Such areas as ontology mapping, merging, alignment, articulation [76], reusing, modularization are frequently mentioning in the context of ontology integration. Besides, other closely related formalisms, i.e. $\mathcal{E}$–connections [83] and Distributed Description Logic (DDL) [36] can be also considered in the context of ontology integration.

2.2.1 Ontology Integration

In order to put the discussion in the right perspective, in this thesis we will use the definition given by Pinto and Martins in [95]:

Definition 2.2.1 (Ontology Integration)**.** *Ontology Integration is the process of building an ontology in one subject reusing one or more ontologies in different subjects.*

It has already been recognized in [95] that ontology integration when building a new ontology is a difficult process that affects the different stages of the ontology development cycle. It is difficult to envision that this will be ever accomplished in a fully automated way, because it requires human intervention during choosing, mapping and reusing of ontologies. Although it should be acknowledged that there is progress in development of alignment approaches, see for instance Ontology Alignment Evaluation Initiative [3](OAEI), for most of them the user intervention is unavoidable; examples of such approaches are: SAMBO [84] and PROMPT [92].

The "ContextVis" approach presented here is related to ontology integration in the way that we are reusing different overlapping ontologies for creation of ontological context. However, it can not be considered as a pure integration method. Nevertheless, it can be seen as a step in direction of ontology integration. The research questions for the development of "ContextVis" were:

1. How to generate extracts, i.e. ontological contexts, dedicated to a concept of interest from different ontologies.

2. How to combine these extracts together in one structure that can be further visualized and explored.

2.2.2 Definitions

Here we will introduce the additional specific terminology that we will use in this chapter.

[3]http://oaei.ontologymatching.org/

Definition 2.2.2 (Interesting Concept)**.** *The Interesting Concept is the* seed *term that is used in order to generate the information from different ontologies.*

Examples of interesting concepts are: Alzheimer, Toll-like receptor, HOX gene, Heart, Lung.

In therms of ontological context we distinguish a global and a local context. These are defined as:

Definition 2.2.3 (Global Context)**.** *The Global Context is the set of concepts from different ontologies that have* similarity [4] *to the* ***interesting concept****.*

Definition 2.2.4 (Local Context)**.** *The Local Context is an extract of an ontology which is generated on the basis of "one" concept. This extract contains a subgraph of ontology based on the hierarchy and related concepts.*

2.3 "ContextVis": Proof of Concept

We wanted to investigate the possibility of visualizing context from different ontologies. This requires two steps, i.e. integration step and visualization, where the results are presented in a visualization. In Figure 2.1 a flow chart representing the underlying software architecture is depicted.

During the process of knowledge acquisition, the information from different ontologies is collected and stored in a database. This information is dedicated to the concept of interest (cf. Def. 2.2.2). The information is represented as a set of modules (*local contexts* cf. Def. 2.2.4) extracted on the basis of the concepts from different ontologies that are similar to the *interesting concept*. Hence, here we explore how the *interesting concept* is represented in different ontologies. The

[4] With the *similarity* here we mean the syntactic similarity, e.g. two concepts are similar when their syntactic features, such as label, definition or synonyms are similar.

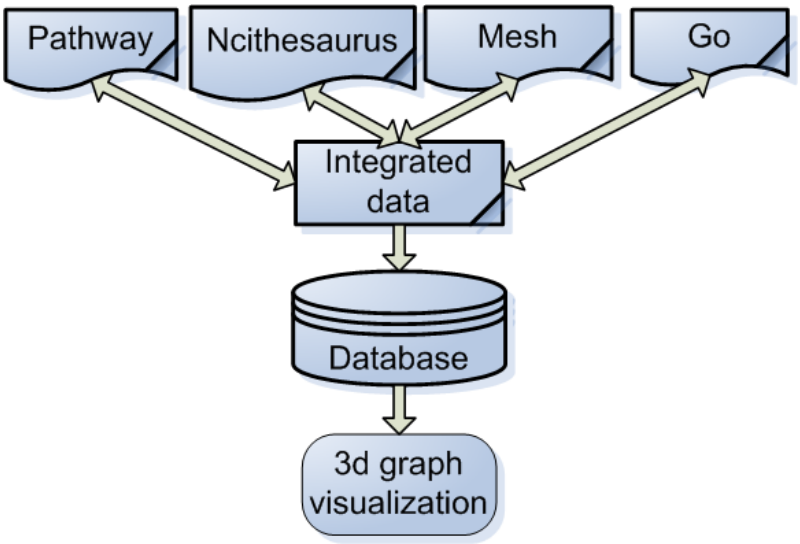

Figure 2.1: Architecture of the "ContextVis"

result of this representation is a graph structure that can be visualized for further exploration by a user; be it a domain specialist or a knowledge engineer.

2.4 Knowledge Acquisition

Data about the *interesting concept* are collected from different bio-ontologies, such as PATHWAY, MESH, NCI-THESAURUS and GO. These ontologies, except for NCI-THESAURUS, are obtained from the OBO FOUNDRY [16]. In order to meet the W3C standard and to make our method not specific for OBO ontologies, we have specifically used the OWL format. As a case study for our approach we will use Alzheimer Disease. This case study will be represented in the remainder of this chapter.

2.4.1 Global Ontological Context

The process of data acquisition starts with a search of concepts which, in their definitions, comments or labels, have words that are equal to the query term. To find these concepts we make use of SPARQL [27] queries in Jena [7]. First, the NCI-THESAURUS ontology is searched for relevant information. All retrieved concepts are saved in a temporary data structure and will be used later to trigger a similar query process in other ontologies. We have used the terms collected from NCI-THESAURUS besides the *interesting term* as *seed* terms for triggering the query process, because this ontology contains information about diseases, hence, in our particular case, i.e. *Alzheimer disease*, we will enrich the singleton set containing only *interesting concept* with the concepts that are related to *Alzheimer disease* and will probably match concepts from other ontologies. Subsequently, MESH ontology is searched and the concepts that contain the specified query term plus terms resulting from the search in NCI-THESAURUS are collected. Finally, in similar fashion, the PATHWAY and GO knowledge bases are processed. According to this strategy, we have obtained a set of concepts from different ontologies. In Figure 2.2 a schematic representation of the strategy for the global ontological context generation is depicted. This set is referred to as the *global ontological context*.

2.4.2 Local Ontological Context

Around each concept found in different ontologies a local ontological context is created from its native ontology. When creating a local ontological context we collect the information about related concepts using all available relationships, not just subclass/superclass. Examples are: *part_of* from PATHWAY or *Chemical_Or_Drug_Affects_Gene_Product* from NCI-THESAURUS.

While creating the *local context* around a class C, the ontology is traversed

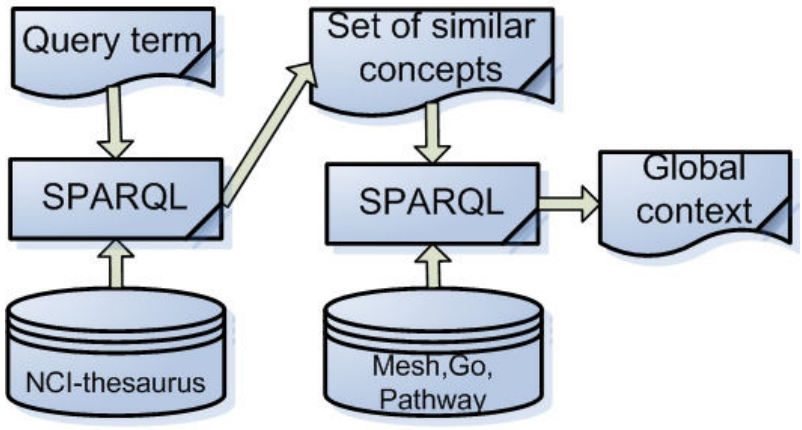

Figure 2.2: Creation of the Global Ontological Context

from this class along all relationships with other classes till certain level of depth [5] is reached. This means, that at the first level all sub/super classes and related classes of the class C are traversed and added to the database; at the second level all the children of the concepts from the first level are traversed, etc.

The algorithm that is used to traverse a graph is a depth-first search traversal. This algorithm is presented in pseudo-code in Figure 2.3.

2.4.3 Reasoning

It is important to realize that a DL reasoner, e.g. FaCT++ [111] or Pellet [105], will infer new relationships between concepts in an ontology. The model generated after ontology classifying procedure is called the *inferred model*. The classifying procedure generates new sub/super class structure of an ontology on the basis of new entailments.

The inferred model represent the ontology structure more correctly, because it makes the implicit relationships between entities explicit. In order to demonstrate

[5]The level of depth is the stop criterion for the traversal; we use 3 in our application. This is a trade-off between completeness and succinctness of the concept representation.

```
generates the local ontological context for the class A
Traversal(A){
    if(A.isVisited ||  A.level >= level)
        return
    A.isVisited = true
    children: generated_children_on_all_properties(A)
    for(each child B from children){
        create_link(A, prop, B)
        B.level = A.level + 1
        Traversal(B)
     } // for
} //Traversal
```

Figure 2.3: Depth-First search traversal

the advantages of inferred model over the asserted model we show an example from AMINO-ACID ontology [1] where the Pellet reasoner infers that histidine is an aromatic amino acid, while this information remained hidden in the asserted model. In Figure 2.4 a screendump of the Protégé Explanation plugin for an inference about the concept *histidine* is given.

From previous statements it becomes clear that the usage of inferred model is preferable in order to get implicit taxonomic and domain dependent relationships between concepts. Unfortunately, during development of the "ContextVis" we were not successful in classifying NCI-THESAURUS ontology, because it is simply too big. This was the reason that only asserted ontological model was used for ontological context visualization.

Figure 2.4: This is an explanation for H (*histidine*) from the AMINO-ACID ontology. In the asserted model H is a Specific Amino Acid, while in the inferred model it became to be Aromatic Amino Acid. This explanation is generated by the Explanation plugin available in Protégé 4 [21]

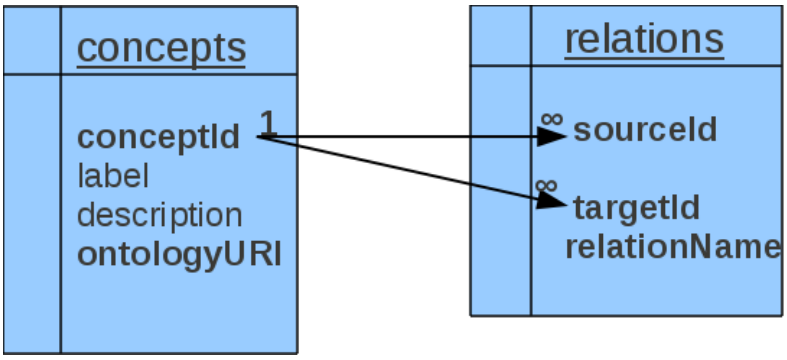

Figure 2.5: Relational Database Schema for "ContextVis"

2.4.4 Database Representation

The information about the *interesting concept* with surrounding global and local context is transferred to a special purpose database. In this database, initially, only two tables are used: one for concepts and the other for relations. The relational database schema is depicted in Figure 2.5. The field **conceptId** represents the unique identifier of a concept in an ontology, in most cases this identifier is automatically generated and not informative for the user. The fields **label** and **description** contain a label and description of the concept given in a human language. The field **ontologyURI** contains a URI of the ontology. The fields **conceptId** and **ontologyURI** define the primary key for the **concepts** table. The **relations** table

contains three fields **sourceId, targetId, relationName**, these fields determine the primary key.

2.4.5 Relationships

The skeleton of each OWL-DL ontology is a hierarchy. Hence, in the first place the concepts are related to each other via sub/super class relationships, e.g. *mercury dicyanide* $Hg(CN)_2$ *is_a mercury coordination entity*. Besides the skeleton, however, the concepts/classes in OWL could be defined as things that have restrictions on properties. Moreover, in biological ontologies from OBO FOUNDRY [16] it is normal to say that concepts are connected with each other via properties. For example the term *inflammatory response pathway* is represented in the following construct:

```
[Term]
id: PW:0000024
name: inflammatory response pathway
def: ...
relationship: part_of PW:0000234 ! innate immune response pathway
```

If we translate this statement in a human language it will mean that *inflammatory response pathway* is *part_of innate immune response pathway*, or that the first concept is related to the second via the *part_of* relation. It is reasonable to think about concepts in OBO ontologies as nodes that are connected with each other by means of relations. From this perspective, this representation is a graph structure. The OBO format for ontologies is, however, not the standard for ontology representation [6] on the web, therefore, "ContextVis" supports the ontologies in OWL format. It is possible to translate OBO to OWL, for example with OWL-API [18], the OWL format is also available from OBO Download Matrix [11].

[6]The OBO standard is more close to other formalisms, such as Conceptual graphs (http://www.jfsowa.com/cg/) and Semantic Networks. These two formalisms, however, lack a formal interpretation and expressive power.

When using OWL we could express that one concept has the relation with another concept with the following OWL statement:

```
<owl:Class rdf:ID="PW_0000024">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#part_of"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#PW_0000234"/>
      </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

In Description Logic [33] this is represented as: $P_1 \sqsubseteq \exists part\_of.P_2$, where P_1 is the term $PW\_0000024$ (inflammatory response pathway) and P_2 is the term $PW\_0000234$ (innate immune response pathway). In standard DL interpretation this means that it is necessary for every individual from *inflammatory response pathway* to have at least one *part_of* relation with an individual from *innate immune response pathway*. In our approach we interpret this statement as a graph-triple with subject $PW\_0000024$, object $PW\_0000234$ and the predicate $part\_of$. This approach allows us to treat an ontology as a simple graph. We realize, however, that OWL-DL goes beyond this simple construct. In order to transform more complex DL constructs, like $C \sqsubseteq D \sqcap (\exists R_1.B_1 \sqcap \exists R_2.B_2)$ the reasoning tools are needed. In the next chapter (cf. Chapter 3) we will discuss these transformations in more detail.

2.4.6 Concept Expansion

Implicit in the ontology is the granularity, in a visualization we expect that users would like to expand some of the concepts, and explore them at a higher level of

granularity where the subclasses and related concepts for the class of choice are visualized. This implies that the ontological context also has to be defined for the global concepts that are surrounding the query term that we started from (cf. Figure 2.6). This can be easily achieved by extending our special purpose database with extra tables. For each concept from the global ontological context we create a concept table and a relation table.

2.5 Visualization

Given the number of relations and the aforementioned requirement for exploration we have chosen to do visualization in $3D$ instead of in a simple $2D$ representation. We argue that $3D$ implementation helps to utilize the space efficiently as well as to experience the ontological world created from the merging process in an intuitive way.

In "ContextVis" interaction is provided through using the mouse as pointing device. Relevant visualization functionalities such as zoom, pan and rotate are made available. As here is more information to present – the click and drag approach is exploited. A concept definition, if existing in the ontology, can be visualized by clicking the "right" mouse button. The nodes for which the concept and relation tables are created can be expanded and, at each expansion, the local context of concept will be represented. An example of the global and local context visualization is depicted in Figure 2.6 and in Figure 2.7 respectively.

2.6 Implementation Details

While creating ontological context we have to extract concepts and relationships. For this need we have used the Jena API [7]. In order to find the concepts which are similar to the concept of interest, the SPARQL [27] query language was used.

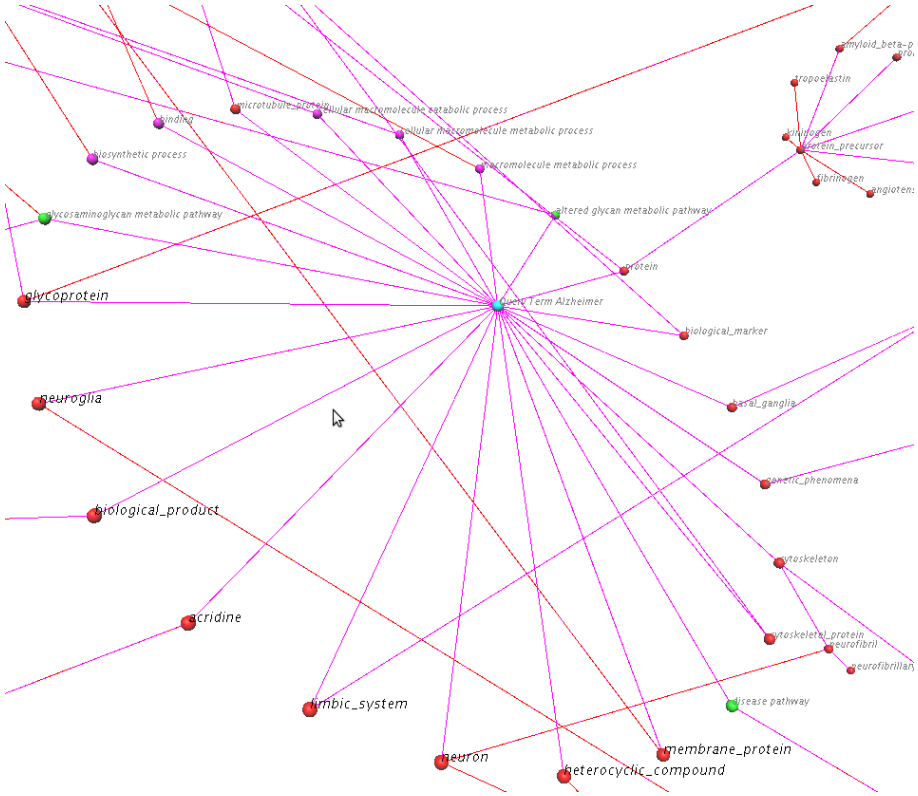

Figure 2.6: Visualization of global ontological context created around concept "Alzheimer". The concepts are collected from different ontologies and comprise the global ontological context. Red colored nodes represent concepts from the MeSH ontology, green colored nodes represent concepts from Pathway ontology, magenta colored nodes represent concepts from GO ontology.

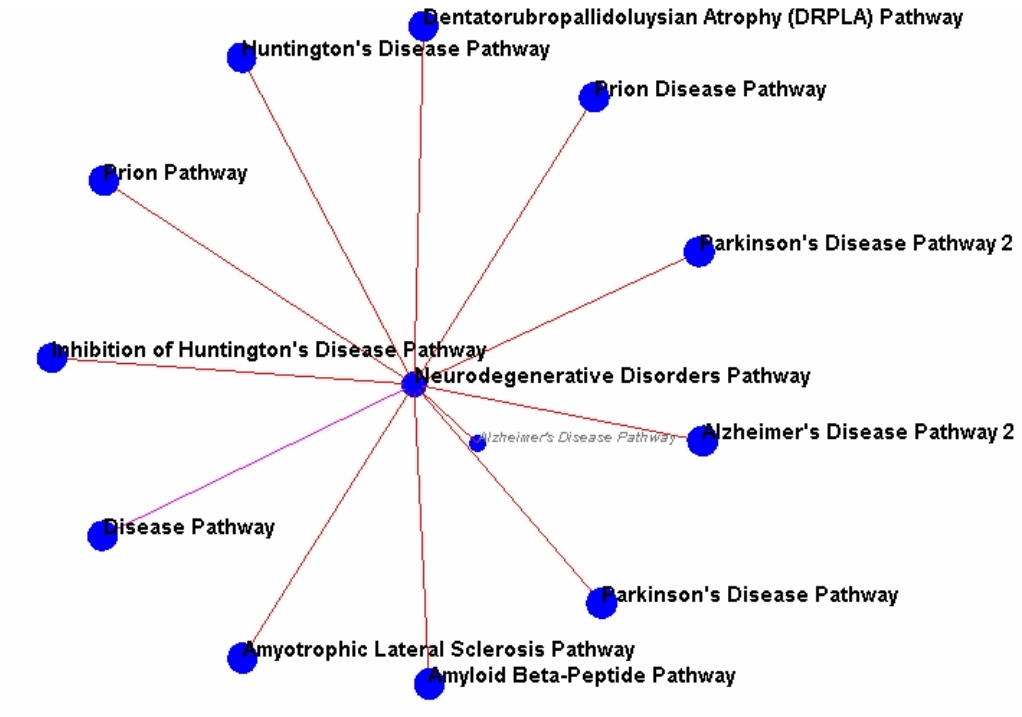

Figure 2.7: Visualization of local ontological context created around *Alzheimer's disease pathway* from NCI-THESAURUS ontology. The red edges represent "subclassOf" relation: *Prion Disease Pathway* $\sqsubseteq$ *Neurodegenerative Disorders Pathway*. Magenta edge represents "superclassOf" relation:
Disease Pathway $\sqsupseteq$ *Neurodegenerative Disorders Pathway*

We extract subsets of ontology, so called ontological contexts, around each concept that is retrieved; this context is saved in a dedicated MySql [10] database.

The visualization engine retrieves the data from the database and represents these as a graph in a 3D environment. The implementation of the graph visualization is realized with the Java3D API [6].

2.7 Conclusion, Discussion and Further Work

In this chapter we have described our efforts in developing a tool that can visualize the ontological context created around some *interesting concept*. With the "ContextVis" tool we can explore how one concept is represented in different ontologies. This brings about the possibility to connect knowledge from one domain with knowledge existing in other domains. Although our approach is not an ontology integration, it can be considered as a stage before the building of a new ontology from other available ontologies; e.g. a developer wants to understand what is known about the topic, how the topic is represented in other ontologies, which ontologies should be considered for reuse. Moreover, "ContextVis" can be of interest to a domain specialist in order to know what is already common knowledge in the domains of interest.

The important result of this work is that it shows a manner to combine the knowledge from different ontologies. This is, however, not sufficient, because the mapping of different concepts is not yet addressed. For example, a concept such as *protein_binding* co-occurs in different ontologies, and it seems therefore logical that this concept exists as unique node in the visualization. Hence, the further work could be in direction of real integration of these ontology parts. The straightforward approach could be merging of the similar nodes into one node, assembling all related concepts under the merged parent. Subsequently this will provide a clearer idea about the consequences of an integration. The idea of the integration of subparts of

ontologies, also referred to as *modules*, is elaborated further in Chapter 6.

Chapter 3

Visualization of Ontology

Based on:

"Multi-View Ontology Visualization"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of 11*th International Protégé Conference 2009*,

"Node-Link and Containment Methods in Ontology Visualization"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of 6*th International Workshop on OWL: Experiences and Directions (OWLED 2009)*

Abstract OWL language is accepted by the community as a formal representation of ontologies. This language has sufficient expressive power to model knowledge in different domains. This high expressivity, however, provides a challenge for the development of visualization tools. In this chapter we present our ontology visualization methodology. We investigate by what means an ontology can be represented as a graph, and what are advantages and disadvantages of such representation. In context of our visualization approach we discuss the concept of logical view.

3.1 Introduction

Ontologies can be considered from different perspectives. Therefore we will start this introduction by giving a representative but not exhaustive set of examples of the use of an ontology.

reasoning An ontology can be envisioned as a logical structure, that contains implicit model description. That model requires reasoners or rule engines in order to make inferences and make implicit knowledge explicit. In this case we could consider ontology as a knowledge base that is used for reasoning about content.

knowledge capture An ontology can be created in order to capture knowledge about some domain of discourse so as to represent this knowledge in an formal and unambiguous way and make it usable and available for a particular community.

data bases Use of ontology in an application, where an ontology is a knowledge layer between the user and heterogeneous and dispersed database resources. In this case the ontology is used to formulate queries by using concepts from the ontology.

data annotation Use of an ontology in the area of data annotation; in this case the ontology terms are used to annotate data collection, such as proteins, genes, biomedical images, 3D models of objects, photographs, textual information, etc. This provides the way for unambiguously labeling of data as well as brings data in the context of knowledge represented by the given ontology.

reuse Yet another application of ontologies is to reuse them in order to create a new ontology.

From the list above, we now see that an ontology could be developed, reasoned with, queried, reused in other ontologies, updated, inspected, etc. There are different

tools available for modeling and editing ontologies, e.g. Protégé [21], Swoop [25], OntoEdit [13]. Although a lot of effort has been put in the field of the ontology modeling, editing, and reasoning, the area of ontology visualization is still insufficiently investigated. We argue, however, that the ontology visualization is very important, especially at the stage of ontology reusing, at the moment a user has to make a choice between available ontologies. Besides, it can be also important for ontology evaluation, at the point when the developer needs an overview of a domain of discourse in order to better comprehend it.

From a survey [79] it follows that most visualization tools are devoted to a representation of the hierarchical part of ontology. We think that this is also the most clear and obvious representation of ontology. However, most domains are frequently not that straightforward so that they can be modeled as a taxonomy, because concepts in the domain can be related to each other by means of different kinds of relations, besides the sub/super class.

In this chapter we will analyze different ontology visualization methods. On the basis of this analysis we will formulate the requirements for ontology visualization methodology, that was leading us during development of our visualization approach. We will concentrate on *ontological* – thus not geometric – aspects of ontology visualization and introduce our ontology visualization approach [48, 49] where an ontology can be represented with two graph drawing techniques, i.e. node-link and containment. Both these techniques are based on the underlying graph structure. Most attention will be given to challenges of the generation of a graph from an ontology.

The remainder of this chapter is organized as follows: in Section 3.2 the short presentation of the state of the art in the area of ontology visualization is provided. Subsequently, in Section 3.3 we will describe the requirements, which we formulated as part of our design. In Section 3.4 we describe how to represent an ontology as a graph and a tree structure. Section 3.5 describes how ontological views can be generated. Finally, in Section 3.6 we present our conclusions.

3.2 Related Work

In describing related work we will restrict ourselves to the most important highlights in the field of ontology visualization. Ontology visualization is complex and there is more to it than just the representation of a network with nodes and edges. The complexity of Description Logic [33], underpinning the OWL language, contributes to the difficulties that arise in the process of ontology visualization.

In an exhaustive survey [79] ontology visualization methods are grouped in very fine-grained categories, such as:

1. Indented List
2. Node-Link and Tree
3. Zoomable
4. Space-filling
5. Focus+Context or Distortion
6. 3D Information Landscape

Moreover, the methods were further classified according to the dimensions used, i.e. 2D, 2.5D or 3D. We argue, that all these categories can be grouped in two global categories:

1. Node-Link representation
2. Containment representation

We choose for these two groups because they reflect two views on ontology in the best way. The containment methods represent an ontology as sets of classes, where the parental set includes children sets. The hierarchical structure of an ontology is suitable for this representation. The node-link methods represent an ontology

as a graph, where the nodes are concepts and the links are relations between the concepts. In addition to the hierarchical relations, this view allows other kinds of relations between the entities.

3.2.1 Containment Methods

In most visualization tools an ontology is represented as a hierarchical tree structure. The treemaps approach very clearly reflects the hierarchy of the classes in an ontology. The subclasses are contained inside the parental classes. In CropCircles [113], for example, the classes are represented as circles, the child circles are placed inside the parental circle and have the radius dependent on the size of the subtree, this is illustrated in Figure 3.1. From Figure 3.1 it follows that this method is not suitable for ontologies with deep hierarchies, because concepts at the deepest levels of the hierarchy will not be distinguishable from each other and it will cost many steps before the user can reach and explore these concepts using a zooming function. While zooming, however, the user will lose the overview of the ontology. An example of the consecutive zooming is depicted in Figure 3.2

The OWL ontology languages support multiple hierarchy, it means that some classes can have more than one parent. To deal with this problem the CropCircles method has chosen for multiple representation, a.k.a. cloning, of the classes if necessary. The introduction of clones of classes can be confusing to the user, because the visualization shows more classes than there are actually present in an ontology. This seems, however, unavoidable in this representation.

Jambalaya [109] is another ontology visualization tool. This tool is implemented as a plugin for Protégé [21] and combines different methods for ontology visualization. Besides a simple visualization of a hierarchical tree with a node-link paradigm, the tool provides also the containment representation, where classes are represented as rectangles, and subclasses are placed inside the parents (cf. Figure 3.3). It is clear that this method suffers from the same shortcomings as seen in

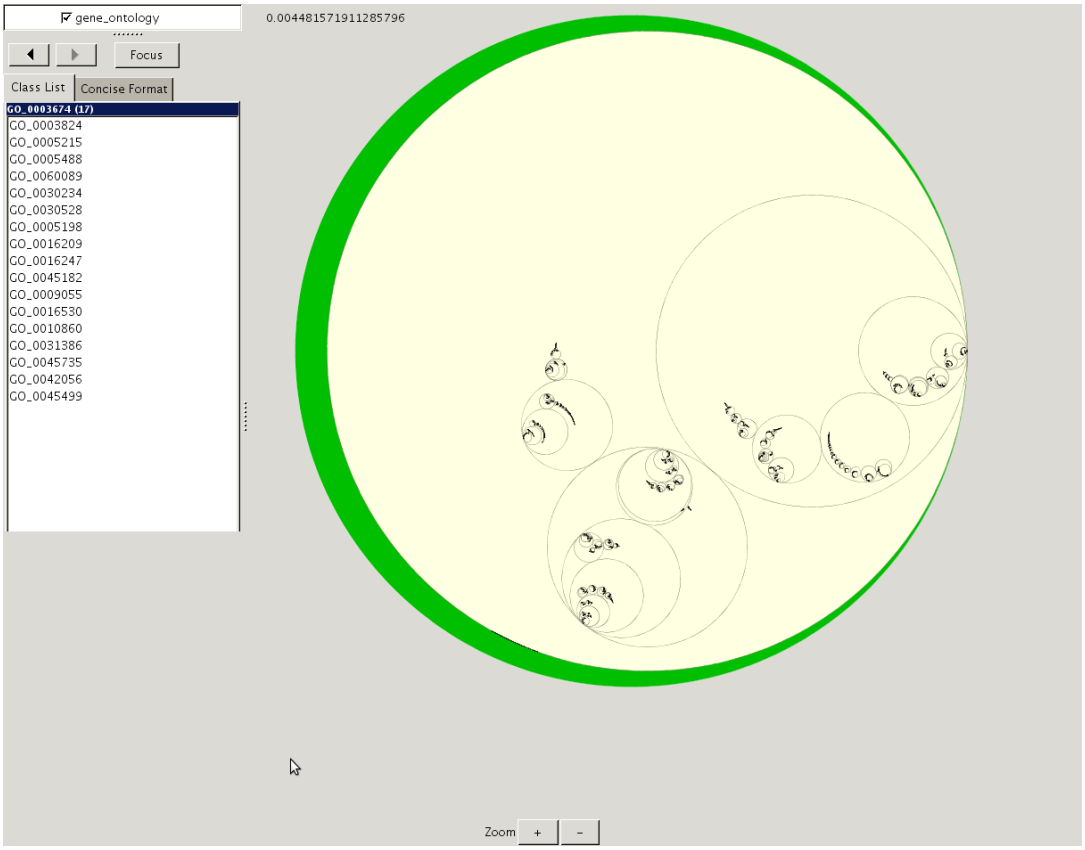

Figure 3.1: CropCircles layout for the MOLECULAR FUNCTION ontology (part of GO [40]). The most general concept is represented as a largest circle. The subclasses are represented as circles and are contained inside their parental circles. The size of the circle depends on the size of the subtree. The image is generated using the Swoop [25] ontology editor.

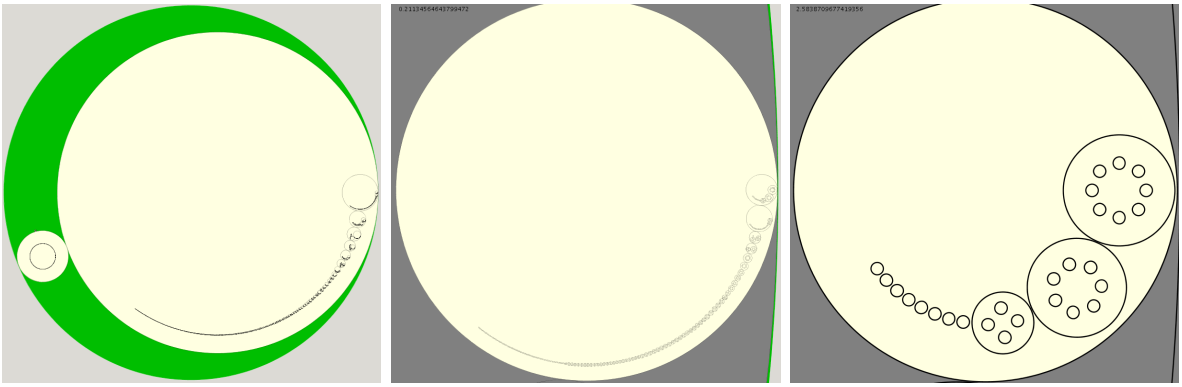

Figure 3.2: Three consecutive zooming-in steps generated for the PROTEIN ontology [22] using CropCircles. The mouse-over interaction triggers the display of the concept identifiers.

the CropCircles method; i.e. visualizing concepts at the deeper levels of the hierarchy is problematic. Besides the visualization of the hierarchy with the containment method, Jambalaya also supports visualization of relations between classes which are represented as arrows. However, the introduction of edges that are crossing the visualization window in all direction makes the visualization aesthetically unpleasant and also complicates the overview of the hierarchy.

We have evaluated a hierarchy representation with the containment method on the basis of two tools, i.e. CropCircles and Jambalaya. We conclude that it is not feasible to provide a visualization including an overview of the complete hierarchical structure. Moreover, the addition of extra relations to the containment method complicates the understanding of the ontology. For a good understanding of the ontology representation in the containment method, the Focus+Context technique [1] is indispensable. We therefore, have added this technique to the requirements for Ontology Visualization which we will describe later in the Section 3.3.

[1] http://www.infovis-wiki.net/index.php/Focus-plus-Context

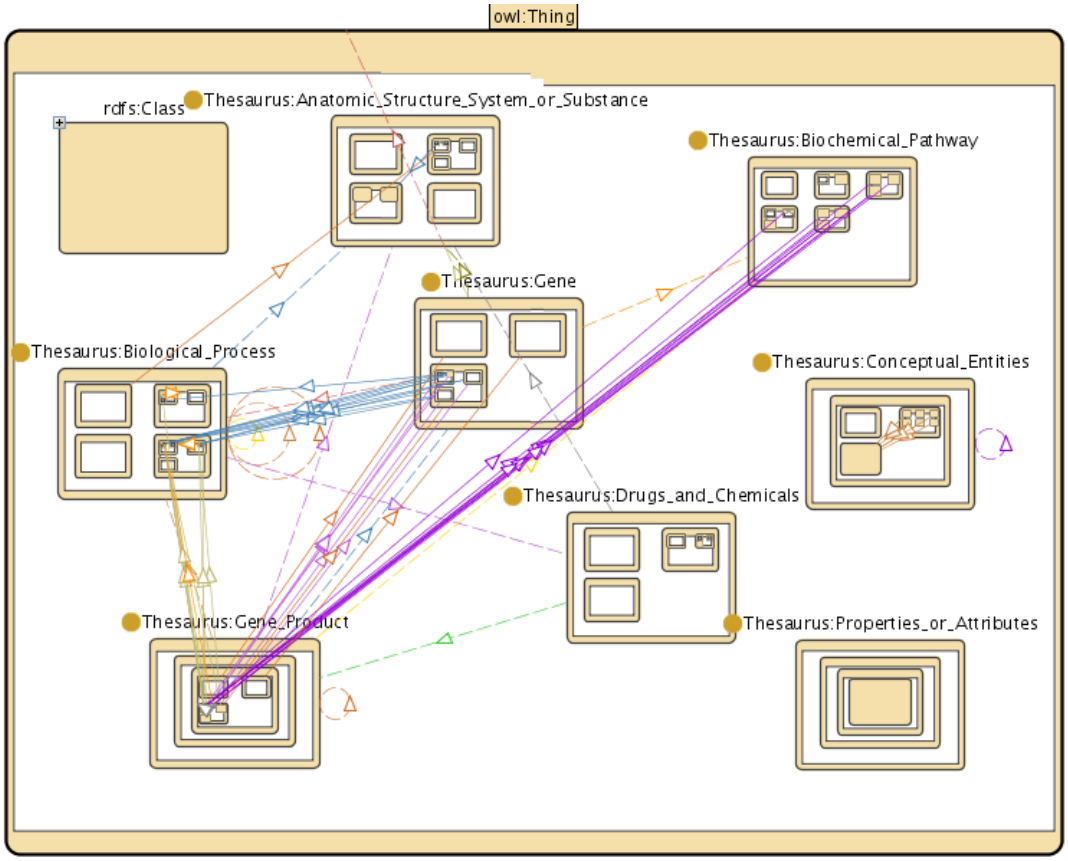

Figure 3.3: Layout for a module extracted from NCI THESAURUS ontology. The rectangles represent concepts, subclasses are represented as smaller rectangles embedded in parental rectangles. Properties are represented as arrows. The image is generated with the Jambalaya plugin for Protégé [21].

3.2.2 Node-Link Methods

In the node-link approach an ontology is represented as a graph, where the nodes are classes and the edges in most cases are the hierarchical relations. OntoSphere [15] proposes a node-link visualization build around a selected concept in 3-dimensional space, this is illustrated in Figure 3.4. The tool is available as a plugin for both Eclipse [2] and Protégé. We have experimented with both representations of OntoSphere. From testing and evaluation through cognitive walktrough [115] and using Nielsen heuristic rules [88] we concluded that the tool is good for the visualization of a local context of a concept. The tool represents both hierarchical as well as other domain dependent relations. It is, however, not possible to explore the neighborhood of a selected concept up to level of depth of the user's interest. Moreover, it is not possible to generate a kind of overview of the ontology. Another problem of this visualization approach is that it did not seem to be amenable for the most ontologies of our interest, we were only successful in visualizing rather small and simple ontologies, like Pizza [20] or Amino-acid [1].

OntoRama [53] is an RDF browser capable of visualizing an ontology as a tree that is laid out on a Poincaré-like Disk [54, 65]. An example of such hyperbolic view with OntoRama is depicted in Figure 3.5. The Poincaré layout uses the Poincaré projection of the hyperbolic plane; this representation will be further discussed and elaborated in Chapter 4. The basis for the layout is the tree structure extracted from the RDF graph. In order to represent the RDF graph as a tree and maintain the graph planar [2], the nodes with multiple incoming edges are cloned. The hyperbolic layouts allow to make efficient use of the available space. Before OntoRama, multiple variants of these layouts have been already successfully applied in multiple graph/tree visualizations [85, 89, 32]. It has been mentioned [113], that although hyperbolic layouts are very suitable for large hierarchies, interaction with

[2] A graph is planar when it can be laid out at the plane in such a way that the edges are intersect only in the vertices

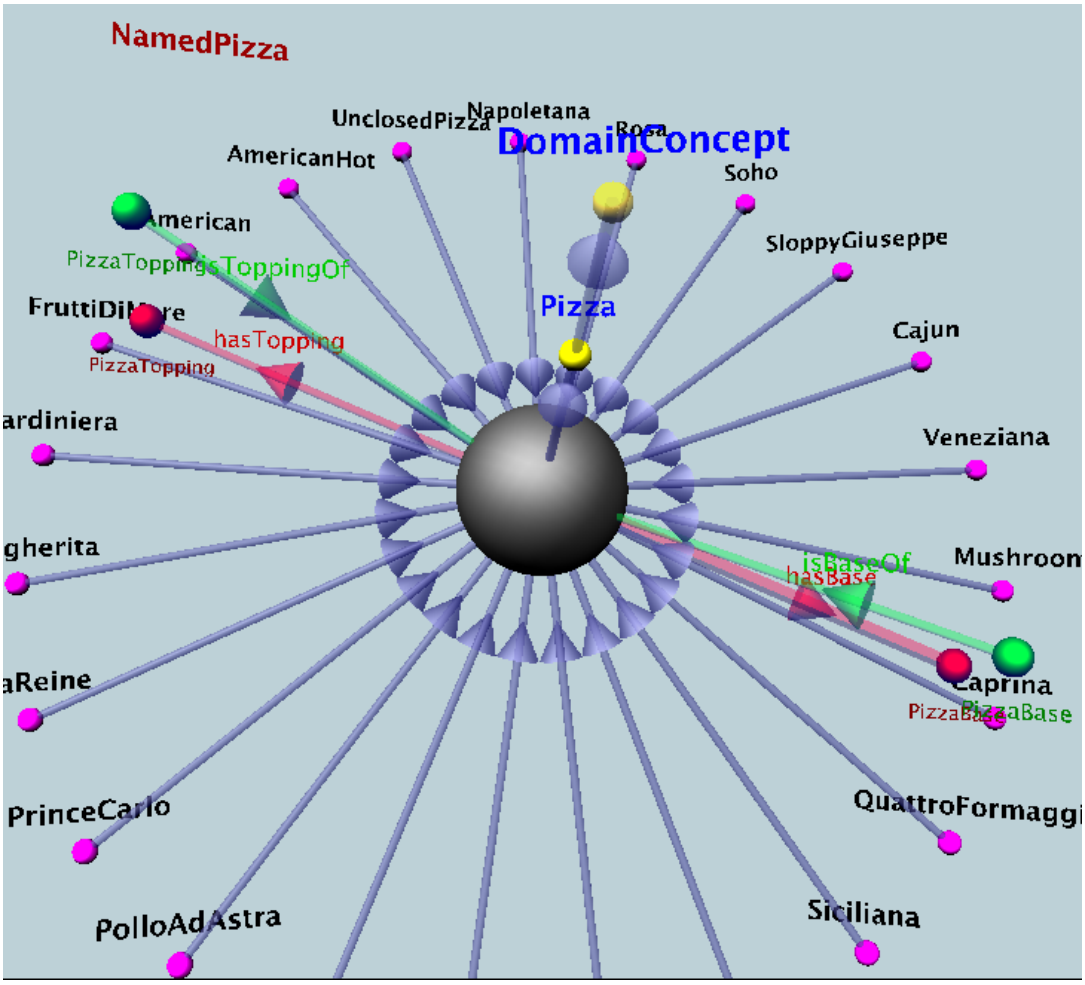

Figure 3.4: Pizza ontology visualized with OntoSphere. The Pizza concept is expanded: superclasses, subclasses, and individuals are visualized in 3–dimensional space. This image is generated with the OntoSphere plugin for Protégé [21]

the visualization is difficult for a user, because "constant relayout [3] makes it difficult to maintain a mental map of where the nodes are or what the structure is" (cf. [113]). It is possible that using of non-standard hyperbolic transformations is difficult for the unexperienced/novice user. The hyperbolic layout is a realization of the *focus+context* technique [85], bearing the consequence that the user will see a detailed view in the center of the visualization (*focus*), but, the more distant the entities are from the center, the less space is reserved for them in the visualization. However, by means of interactions the user can bring the interesting concept and its neighborhood to the focus.

TGVizTab [29] is another ontology visualization plugin for Protégé based on the TouchGraph [30] technology. The TouchGraph approach uses the spring-layout [55] technique in order to visualize a graph network. This tool displays the classes as well as individuals. Ontology relations, the hierarchical as well as other kind of relations, are represented in this visualization as graph edges, this is illustrated in Figure 3.6. The user can generate views at different levels of depth and expand the children of the nodes. In this tool an ontology is represented as a graph, consequently the visualization suffers from crossing edges that obscure the view.

An evaluation of the aforementioned methods is summarized in Table 3.1. From this table it follows that visualization of an ontology overview is still problematic and that ontology properties can be represented only by the node-link visualization. The multiple hierarchy or multiple parents are represented through cloning or by addition of extra edges.

[3]The principle of relayout refers to the dynamic redrawing of the hierarchy upon an event trigger, such as a mouse event.

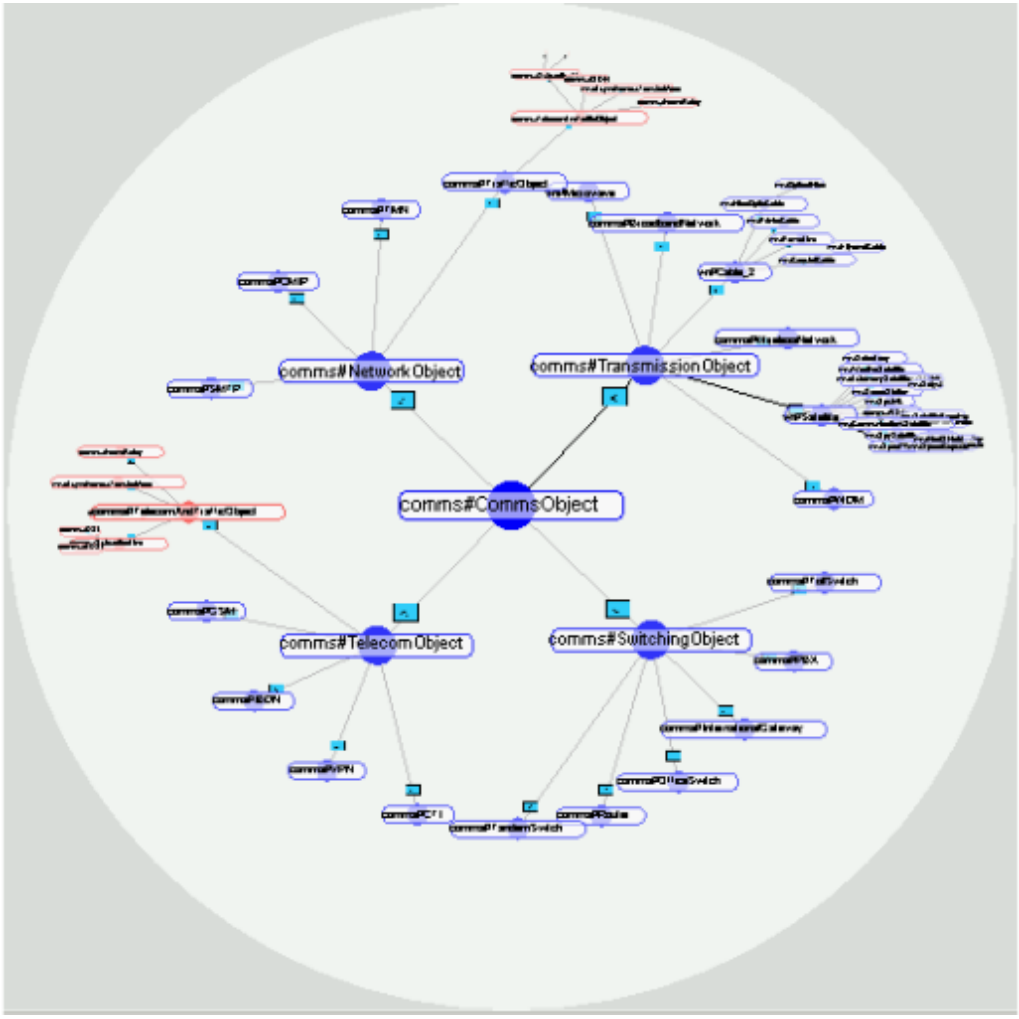

Figure 3.5: OntoRama hyperbolic-style view. The figure is copied from the paper of Peter Eklund et al. "OntoRama: Browsing RDF Ontologies using a Hyperbolic-style Browser" published in: Proceedings of the First International Symposium on Cyber Worlds 2002, with permission of IEEE (©[2002] IEEE).

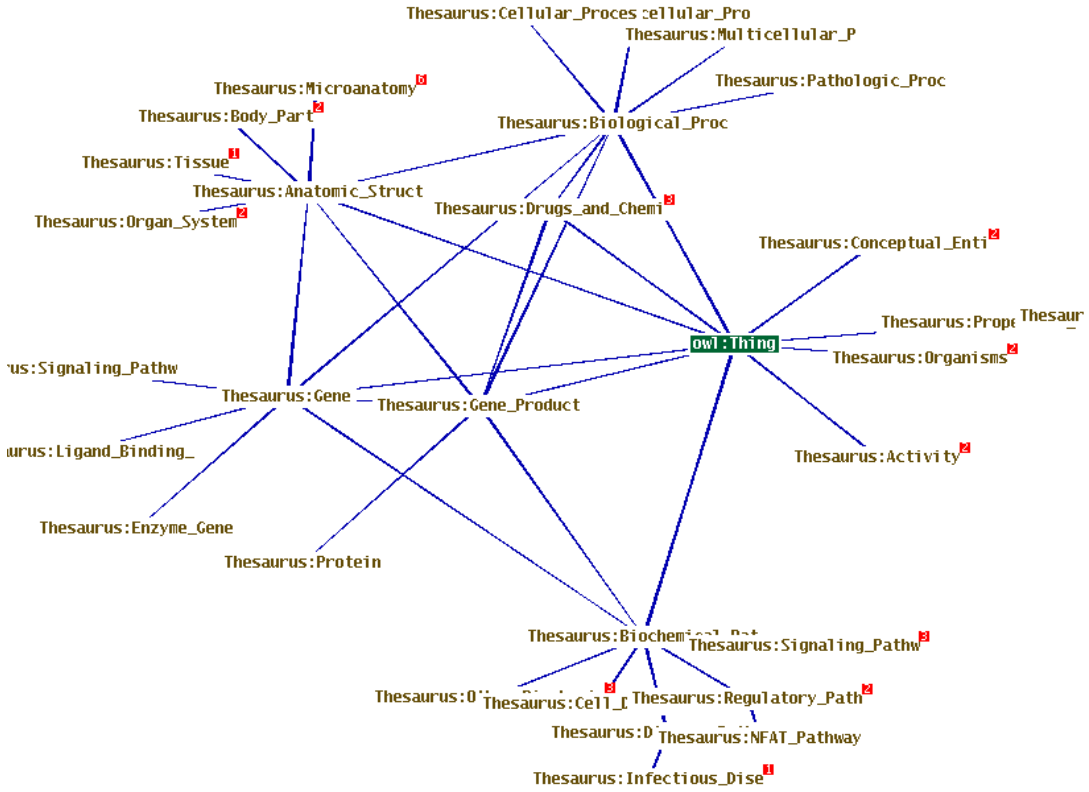

Figure 3.6: Layout for a module extracted from NCI THESAURUS ontology. The ontology is represented as a graph and visualized with the level of depth 3. Image is generated with TGVizTab plugin for Protégé [21]

Table 3.1: Evaluation of Ontology Visualization Methods. The label "no data" refers to our evaluation meaning that no data were available or could be extracted. The evaluation of text labels is not considered here.

Method	Hierarchy	Properties	Level of Depth	Overview	Cloning	Extra Edges
CropCircles	yes	no	yes	partly	yes	no
Jambalaya	yes	yes	yes	partly	no	yes
OntoSphere	no	yes	no	no	no	no
OntoRama	yes (RDF)	yes (RDF)	no data	no data	yes	no
TGVizTab	yes	yes	yes	no	no	yes

3.3 Requirements

From the analysis of different ontology visualization approaches we can formulate the important requirements. In this thesis we will discuss our visualization approach in context of these requirements.

1. **Hierarchy.** Because the backbone of an ontology is the hierarchy, a visualization tool should represent the hierarchical structure.

2. **Properties representation.** Most ontologies are more than just a hierarchy. Frequently the concepts are described by using restrictions on properties. For this reason it is important that an ontology visualization can represent the properties.

   Furthermore, in 3.4.1 we will describe how to retrieve the properties and discuss the difficulties with understanding the properties in context of classes.

3. **Level of detail.** It should be possible to select a level of detail in a visualization. Ontologies from medical and biological domains tend to be very large

and can require extensive computational and memory resources. Potentially, this can slow down the visualization application. Moreover, it can be difficult to comprehend the whole ontology shown in one visualization window.

Therefore, in our approach we will provide the possibility to choose up to what level an ontology needs to be visualized (cf. 4.3).

4. **History.** Visualization is an interactive process and as such it has to support the possibility to go back to the concepts that were chosen in the previous steps.

   Therefore in the approach provided, the concepts chosen in the previous steps are saved. By means of mouse interaction the user can *step back* to the concepts chosen in the previous steps (cf. 4.3).

5. **Query support.** Ontologies could contain thousands of classes. It can take a lot of interaction steps to find the concept of interest. To that end, the visualization approach should provide a query support.

   Therefore in our approach a simple query interface is provided, so that the concepts which are similar to the given query term are retrieved. The similarity is based on the comparison of description, label and synonyms of ontology classes with the query term (cf. 4.3).

6. **Filtering.** Ontologies can contain hundreds of properties interconnecting the concepts, e.g. NCI Thesaurus [28], The user can be interested in only the subset of the ontology, based on the central concept and the properties of the user's choice (cf. 3.5).

   Therefore, in our approach the concept of logical view is supported. The logical view provides a view on a subgraph of ontology on the basis of a central concept and the properties as chosen by a user.

7. **Multiple geometrical views.** Although this requirement was not previously mentioned [79] in the context of the ontology visualization, we argue that the

representation of the graph in different geometrical models can help to better understand the structure of an ontology.

Therefore, we will comprehensively discuss the ontology visualization in different geometrical models in Chapter 4.

8. **Zoom semantic/geometric.** A visualization tool equipped with semantic and geometric zoom interactions can be supportive in seeing more or less details during ontology exploration. With the geometric zoom the visualized object is scaled when the user zooms in/out. The semantic zoom provides the possibility to see more/less details of the object by zooming in/out.

   In our containment approach the user can see the higher/lower levels of the hierarchy, when zooming in/out (cf. 5.3.2).

9. **Focus + Context** This technique allows to have the information of interest in the foreground (focus) while all the remaining information is visible in the background. In the design of our visualization methods we have also considered it (cf. Chapter 4).

Next we will describe our visualization approach in context of these requirements.

3.4 Transformation to Visual Representation

In order to represent an ontology in a structure feasible for a visual representation we need to transform it to either a graph or a tree data structure. OWL as such is not feasible for visualization. Therefore, in our approach, a graph structure is the basis for the node-link representation, and the hierarchical tree structure is the basis for the containment representation. In the next paragraphs we will discuss the transformation from OWL ontologies to the structures we will employ in our visualization method.

3.4.1 Graph Generation

The formal basis for OWL ontologies is provided by Description Logics constructors [33]. In the biomedical domain, e.g. NCI Thesaurus, ontology concepts are frequently defined with interchanging of boolean operators such as the union or the intersection. As an example, we present the concept *Acute_Adult_T-Cell_Lymphoma_Leukemia* from NCI Thesaurus ontology. The DL representation for this concept is given by the following logic formula:

$$
\begin{aligned}
(3.1)\ A \sqsubseteq B \sqcap C \sqcap \Big(& \big(\exists P_1.D_1 \sqcap \exists P_1.D_2\big) \sqcup \\
& \big(\exists P_2.D_3 \sqcap \exists P_2.D_4\big) \sqcup \big(\forall P_3.D_5 \sqcap \forall P_3.D_6\big) \sqcup \big(\forall P_3.D_7 \sqcap \forall P_3.D_8 \sqcap \forall P_3.D_9\big)\Big) \sqcap \\
& \exists P_4.D_{10} \sqcap \exists P_4.D_{11} \sqcap \exists P_4.D_{12} \sqcap \exists P_4.D_{13} \sqcap \exists P_2.D_{14} \sqcap \exists P_2.D_{15} \sqcap \exists P_2.D_{16} \sqcap \\
& \exists P_2.D_{17} \sqcap \forall P_5.D_{18} \sqcap \forall P_3.D_{19}.
\end{aligned}
$$

This DL expression corresponds with the representation in Manchester syntax [4] given in Figure 3.7 in such a way that
concept A represents *Acute_Adult_T-Cell_Lymphoma_Leukemia*,
concept B represents *Adult_T-Cell_Lymphoma_Leukemia*,
concept C represents *Leukemic_Phase_of_Lymphoma*, etc.
This example shows that ontologies in real biomedical domains could be very intricate. Because of the diversity and complexity of DL constructors, we need to find a way to represent both the taxonomy as well as other kinds of *connections* between the concepts. In the node-link approach we represent ontology as a graph structure. Reasoner services [17] are used to generate the inferred hierarchy and infer another relationships between concepts. The transformation to the graph structure is realized as follows:

1. **Subclasses** The reasoner finds all subclasses of the given concept C:
   If $B \sqsubseteq C$ then C is connected to B with a red edge.

[4] Manchester syntax [69] is a "less logician like" syntax for representing OWL ontologies developed to support users who are not DL experts.

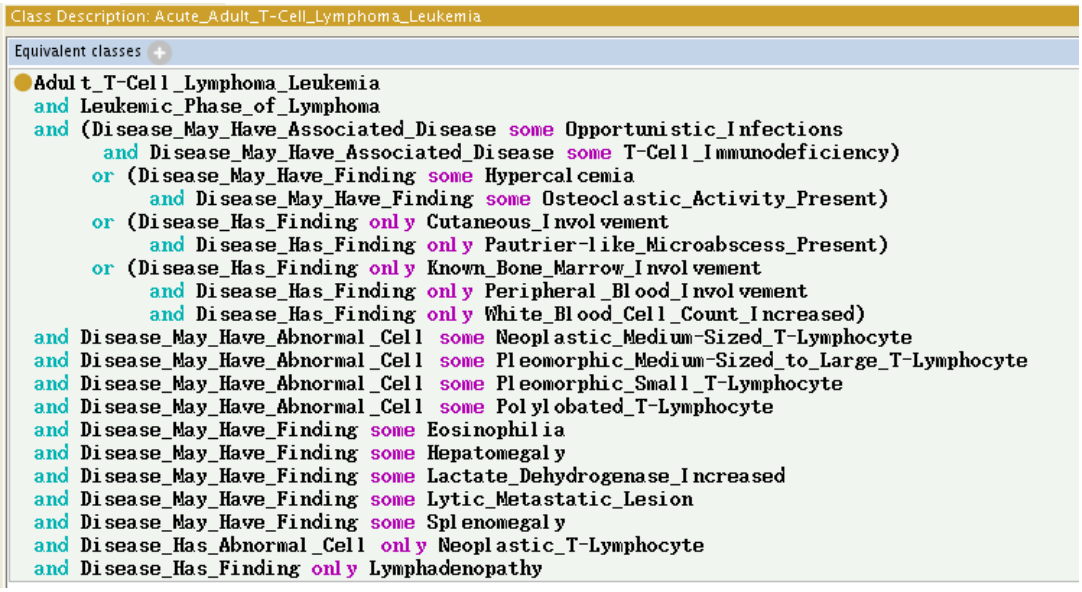

Figure 3.7: Definition of the concept $Acute\_Adult\_T-Cell\_Lymphoma\_Leukemia$ from NCI THESAURUS ontology given in Manchester syntax with Protégé [21]

2. **Superclasses** The reasoner finds all superclasses of the given concept C: If $C \sqsubseteq B$ then C is connected to B with a green edge.

3. **Properties/Roles** Inferring that the C and E are related to each other via R requires extraction of properties from axioms of the concept. For this requirement the axioms $C \sqsubseteq \exists R.E$ are searched. Subsequently, the reasoner can be asked whether the concept $C \sqcap \neg\exists R.E$ is satisfiable. If not, then it is necessary for the class C to have the property R with the value E on this property. This does not fully represent the properties of a class, because only the properties from the asserted model will be retrieved. Otherwise we have to check for each property P whether this is necessary for the given class to have P or not. We have, however, chosen for this (not complete) representation of asserted properties. If we attempt to represent all properties of the class, also all inherited properties have to be included and this would lead to an exponential explosion of edges or nodes in the graph.

In order to traverse an ontology and transform it to a graph, the depth-first search traversal was used [114]. The traversal procedure starts with the root concept *Thing*. The procedure calls itself on each concept that is related to the given concept. The relations could be hierarchical, i.e. subclass, superclass as well as other kind of *domain dependent* relations that we have retrieved with the aforementioned method (cf. 3.4.1). The procedure stops when given level of depth is reached. This level of depth corresponds to some parameter obtained from the user interface. When the graph traversal procedure reaches a concept that was previously visited from some other parent, the concept will not be explored for a second time, but the edge between the parent and the child will be created. Consequently, this concept has multiple parents.

Although the properties make sense in context of individuals, we use them to describe classes in an ontology. We can justify our method, because we choose only the properties that are necessary for the given class. The drawback of this approach is that we are not visualizing the inherited properties, because it will lead to the exponential explosion of the number of edges in the graph. To the best of our knowledge, in other tools, for example Jambalaya [109] and TGVizTab [29] the graph is also restricted to asserted properties. Moreover, the use of reasoner services in order to extract properties of a class have not been reported in available ontology visualization methods.

3.4.2 Hierarchical Tree Representation

The skeleton for the containment visualization approach is the hierarchy that represents subclass relationships between concepts in an ontology. This hierarchical graph can be generated from the *told hierarchy* or from the inferred hierarchy. Told hierarchy reasoning is a very simple form of reasoning, where the subsumption hierarchy is directly extracted from ontology axioms; such simple reasoner is part of the OWL API [18].

However, the hierarchy is not a tree but a directed acyclic graph (DAG), because classes in ontologies can have multiple parents. To adapt this graph structure to the tree representation, for our approach we clone each node when it has more than one parent. The generation of the tree begins with the root node (*Thing*). The procedure visits direct subclasses of the *Thing* and adds them to the children set of the root node. Subsequently, the procedure calls itself recursively on each concept. When this traversal reaches a concept that is already visited from some other parent, this concept will be cloned, but the subtree of this concept will not be visited/visualized for a second time.

3.5 Representations of Views Based on Different Relations

In this section we will elaborate on the *Filtering* requirement proposed and defined in 3.3. While considering the problem on how to represent properties in an ontology, we have to understand that concepts do not have properties, only individuals have. Concepts, however, are represented as classes that could be defined with restrictions on properties: $C \sqsubseteq \exists R.D$ (all individuals of the class C must have at least one relation R with an individual from the class D) or $C \sqsubseteq \forall R.D$ (if an individual from C has a relation R than this relation has to be with an individual from D). For ontology visualization, this means that concepts are interconnected to each other with different kinds of relations, therefore an ontology is transformed from a logical knowledge base to a kind of Semantic Network [5].

In Section 3.4 we have described how to extract properties for a class and generate a graph. Representation of all relations that are available in the graph can be too complicated for visualization, because the network of edges can occlude

[5]John F. Sowa:"A semantic network or net is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs"[106].

important information and it is unpleasant from aesthetic considerations. Moreover, one can be interested only in a particular part of the ontology where the concepts are related via a subset of the properties. This representation is referred to as a *logical view*, *Traversal View* or simple as a *view*.

The main idea of the view generation is based on the work of Noy and Musen [90], in which *Traversal View* was defined as a self-contained portion of ontology. We elaborate on this idea for our visualization, with that distinction that instead of extraction of a self-contained portion of an ontology we visualize a graph structure that represents this particular part of the ontology. An example of our visualization including the property selection is given for the DOLCE-LITE [6] ontology in Figure 3.8 and Figure 3.9. An example for an ontology from the biomedical domain (NCI THESAURUS module) is depicted in the Figure 3.10 and Figure 3.11.

In the current version of our approach we only provide the global level of traversal depth, meaning that all the properties will be traversed with the same level of depth. From the list of properties, a subset of interest can be selected. On the basis of this selection the graph structure will be generated for only a subset of the ontology. This subset contains the concepts that are reachable from the central concept by the defined set of properties and the defined depth of the traversal.

3.6 Conclusions

In this chapter we have discussed different methods of ontology visualization. We conclude that it is difficult to represent whole ontology with the containment method in one visualization, because the deeper layers of the hierarchy are not distinguishable. Therefore, another approach must be provided, in which the level of detail is a user defined parameter. A similar situation is recognized with the node-link method.

[6] DOLCE: a Descriptive Ontology for Linguistic and Cognitive Engineering (http://www.loa-cnr.it/DOLCE.html).

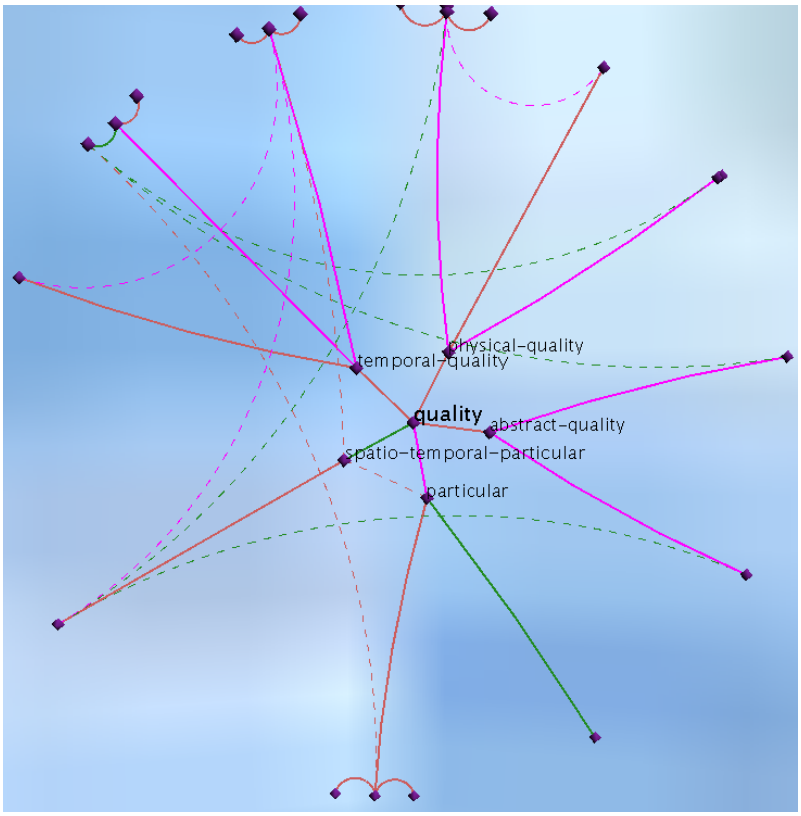

Figure 3.8: Visualization of the concept "quality" from the Dolce-Lite ontology with all properties. The subclass relations are red, superclass relations are green, other relations are magenta. The dashed lines represent non-spanning tree relations.

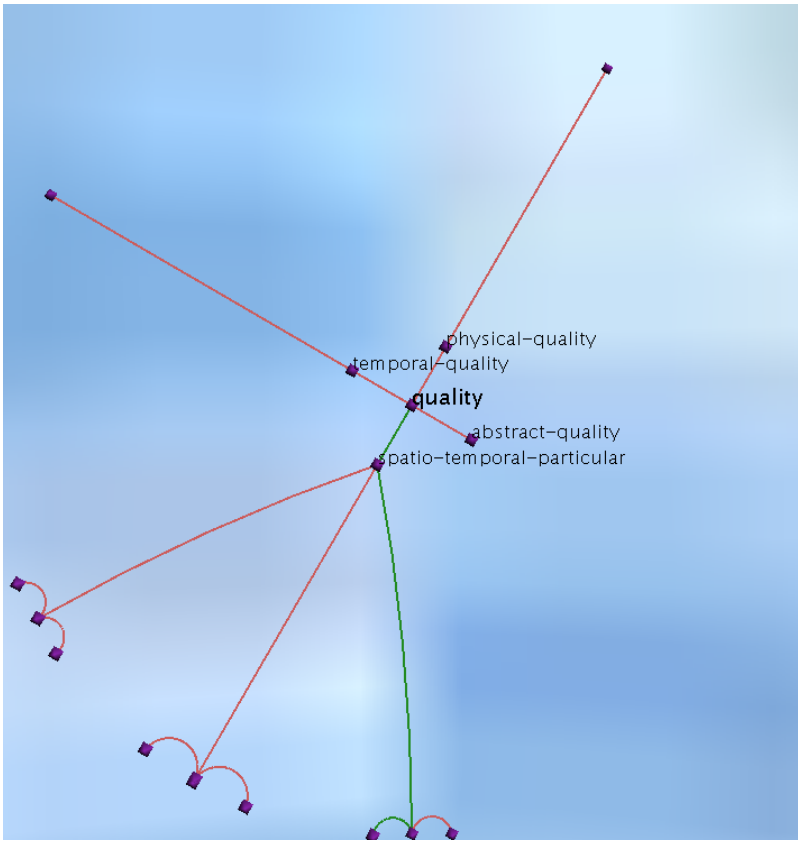

Figure 3.9: Visualization of the concept "quality" from Dolce-Lite ontology with sub/super class relations.

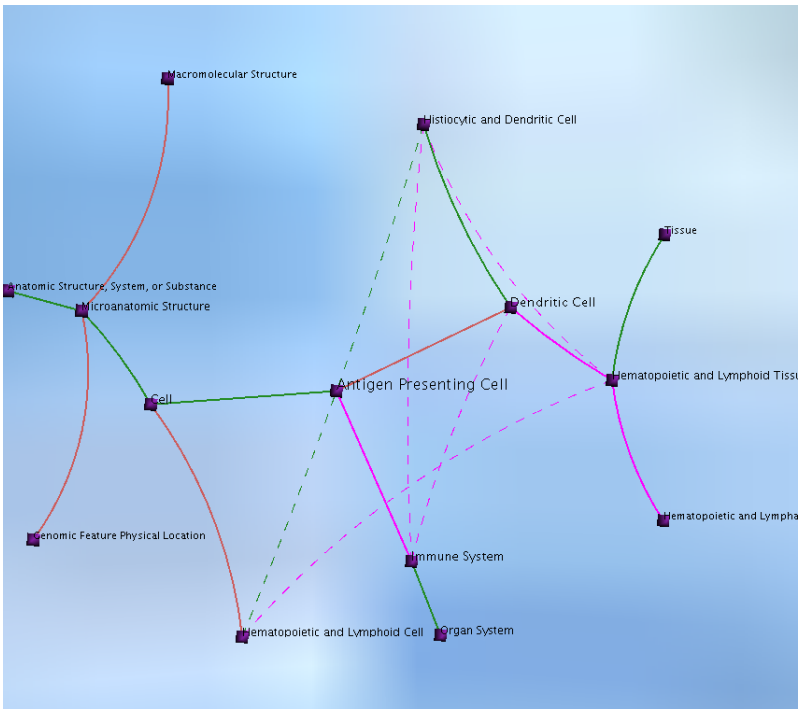

Figure 3.10: Visualization of the concept "Antigen Presenting Cell" from NCI THESAURUS ontology module with all properties. The subclass relations are red, superclass relations are green, other relations are magenta. The dashed lines represent non-spanning tree relations.

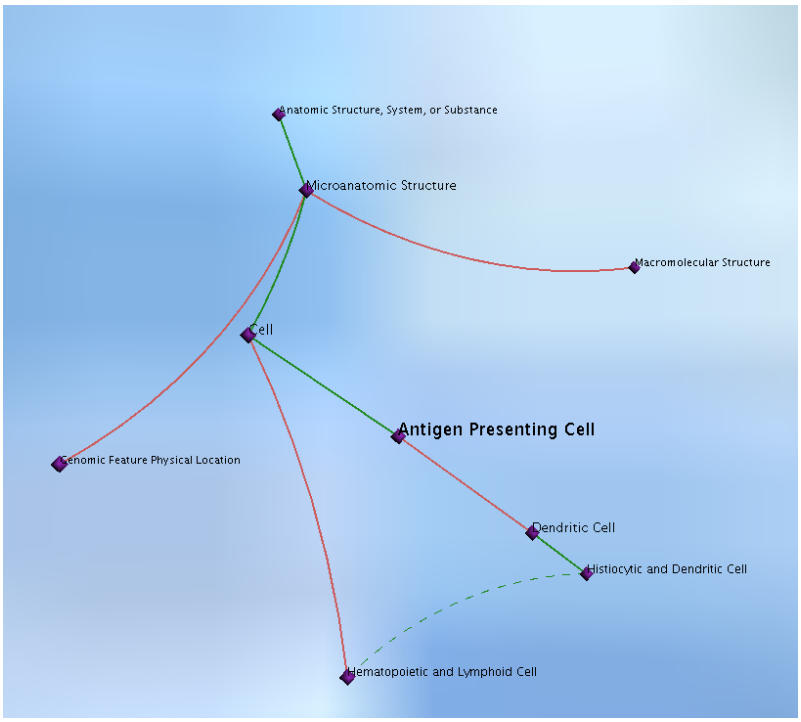

Figure 3.11: Visualization of the concept "Antigen Presenting Cell" from NCI Thesaurus ontology module with sub/super class relations.

Although, the hyperbolic approaches are able to visualize whole ontology, most space will be alloted to the concept in the center and its neighborhood, the rest of the structure, however, is represented in the area that exponentially reduces. In this manner the general idea of overview will be lost. The user needs to make multiple navigation steps in order to explore the ontology tree/graph. Hence, representation of ontology overview can be only realized with user interaction. For example, in the containment method it can be achieved with zooming and in node-link method with navigation.

Multiple hierarchy and/or multiple parents, while representing properties, violates the tree feature and requires the transformation of an ontology to a graph. Graph drawing problems that optimize several aesthetics are, however, known to be NP-complete [37], and most available algorithms provide either iterative solutions or are suitable only for visualization of small graphs. The problems with graph visualization can be avoided if a visualization algorithm is based on a tree-skeleton structure. This could be realized in ontology visualization through cloning the nodes with multiple parents or with introduction of extra edges. Unfortunately, both of these methods have drawbacks: the first method could create a wrong impression about a number of classes in an ontology, because it seems as if there are more classes, while the second method obscures the visualization with crossing edges. For our node-link approach described in Chapter 4 we have chosen for graph representation, where extra edges are introduced for the nodes with multiple parents. The containment method is based on a tree structure, hence, we have chosen for cloning in our containment representation, that will be further elaborated in Chapter 5.

From an ontology visualization approach it is not expected that it provides a logical view on the ontology. For this particular requirement, tools such as ontology editors must be used. Instead, an ontology visualization has to provide an abstract view on the domain of discourse, in which the structure of the ontology can be easily seen, in which the content of the ontology is clear, and in which neighborhood of a concept showing parents and children can be generated. In order to generate

this representation a reasoner is required that generates subsumption and extracts properties of classes. In this manner a graph or a tree structure can be extracted that can be further visualized with our approaches that will be described in Chapter 4 and in Chapter 5.

In our analysis we have focused on the requirements itself and not on the interactions in a given GUI - this usability study is important but considered outside of the scope of this thesis. The requirements for visualization, formulated in this chapter, have helped to determine a conversion of the ontology to a structure that can be visualized. This important finding is, as indicated, elaborated in the Chapters 4 and 5 in which this structure is used for developing ontology visualizations.

Chapter 4

Node-Link Methods in Ontology Visualization

Based on:

"Multi-View Ontology Visualization"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of 11*th International Protégé Conference 2009*,

"Node-Link and Containment Methods in Ontology Visualization"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of 6*th International Workshop on OWL: Experiences and Directions (OWLED 2009)*,

"Different Geometries in Ontology Visualization"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of SPIE, Visualization and Data Analysis conference (SPIE 2010).

Abstract In this chapter we introduce a method for visualization of ontologies based on a node-link approach using different geometrical representations. The basic component for the visualization is the spanning tree skeleton of a graph underlying the ontology and includes 5 different geometrical views, i.e. 2 Euclidean, 2 hyperbolic and 1 spherical. All the views are augmented with corresponding geometrical transformations so that interaction like pan, zoom and rotate can be invoked. Our approach provides the means to visualize an ontology from different perspectives and with different levels of detail, the interaction that is provided greatly enhances the user perception of otherwise complex information.

4.1 Introduction

In the previous chapters we have emphasized that ontologies can be used for knowledge modeling, representation, integration and sharing. In Chapter 2 we have discussed that ontologies could be reused in order to create an "ontological context". Part of this process is inspecting different ontologies in order to make a decision as to whether or not an ontology is relevant for the application. In support of this evaluation process specific tools for ontology visualization are needed.

In Chapter 3 we have described different methods for ontology visualization. In particular we have introduced our approach for the representation of an ontology as a graph structure. This graph structure is the skeleton for the node-link method. Although different ontology visualization methods are developed on the basis of node-link paradigm, e.g. OntoRama [53], OntoSphere [15], TGVizTab [29], no single method has applied different geometries in order to visualize an ontology. To the best of our knowledge, each approach is realized on the basis of just one geometrical model, and is implemented in one dimension, 2D or 3D.

In this chapter we will describe our approach for node-link ontology visualization based on different geometries [51]. To that end we have experimented with Euclidean, hyperbolic and spherical geometries in two- and three-dimensional space. We will describe different algorithms that we have developed, implemented and evaluated in order to visualize ontology graph structure. The analysis of the algorithms takes the requirements formulated in Chapter 3 into account.

The remainder of this chapter is organized as follows: in Section 4.2 a summary of graph visualization techniques that can be applied for node-link ontology visualization is presented. In Section 4.3 we will discuss our approach in context of requirements that are formulated in Chapter 3. Further, in Section 4.4 we will introduce different geometric models that we have selected and implemented. Section 4.5 describes the relevant implementation details. Finally, in Section 4.6 we present our conclusions.

4.2 Related Work

Because this chapter is more dedicated to graph visualization methods considered from geometric point of view, we will shift paradigm from the area of ontologies to the area of graph visualization and discuss approaches that can be applicable to ontology visualization.

We will not describe all available graph visualization techniques, a comprehensive survey devoted to this topic is available [68]. We will concentrate on most interesting highlights in this area and consider graph visualization from the perspective of applicability to node-link ontology visualization.

One of the common techniques used to visualize a graph is the force-directed placement [55]. This technique was successfully applied for the ontology visualization in TGVisTab [29]. It is based on the spring layout algorithm that attempts to find placement for nodes which are subject to attractive and repulsive forces. The physical model for this algorithm is based on Eades model [52], where nodes are considered as steel rings connected by springs. At initialization, the nodes are randomly placed. At each iteration the repulsive forces try to push the nodes away from each other, while the attractive forces attract the nodes if they are connected by links. Although this algorithm generates an aesthetically pleasing layout, because of its random basis, it produces different layouts for the same data set. From the perspective of a user, this visualization technique is potentially confusing, specially with a dataset such as an ontology, where the graph structure reflects the semantics of the data. Moreover, with larger graphs where the number of nodes exceeds a thousand, this algorithm converges poorly.

Frequently, a graph can be represented as a spanning tree when we take in account domain-specific knowledge [89]. To that end, the tree visualization algorithms can be applied to the visualization of graphs. The spanning tree is responsible for layout, while other non-tree edges could be visualized on request. Another possibility to represent a graph as a tree is the cloning of non-tree nodes; such has been

already described in Chapter 3, (cf. 3.4.2).

The H3Viewer is a typical example of a tree visualization with a non-Euclidean geometry. It was developed by Munzner [4, 89]. In this visualization the spanning tree is laid out at the hyperbolic space and the Klein [65] projection is used in order to project this tree structure on the unit sphere. An example of a tree structure visualization generated by H3Viewer is given in Figure 4.1. Another example of the use of the hyperbolic geometry was realized by Lamping and Rao in their Hyperbolic Browser [85]. In this approach the Poincaré model [54, 65] was used in order to project the graph structure laid out in an infinite hyperbolic plane to the unit disk. An example of a tree visualization with Poincaré model is given in Figure 4.2.

Applying hyperbolic geometry makes it possible to visualize very large tree structures. This is because the number of nodes in a tree grows exponentially with the level of depth. In the hyperbolic plane, the circumference and area of a circle grow exponentially with its radius. Hence, this feature of hyperbolic geometry could be utilized for the visualization of ontologies represented as graph structures.

In the work describing H3 method [89] the author was successful in visualizing trees with about $100,000$ nodes. In this particular application, the hyperbolic visualization is a realization of the *focus+context* technique, which uses the paradigm of emphasizing the information in the center (focus) while simultaneously providing the overview of the global tree structure (context). The combination of hyperbolic visualization with the *focus+context* technique could be utilized in ontology visualization. First, because ontologies can be very large and contain over thousands nodes, thus, hyperbolic plane/space with its exponential growing area/volume can be utilized. Second, because it is impossible to understand the whole ontology graph structure at a glance, the *focus+context* paradigm matches the best with the requirement on seeing the most important information magnified, and less important information reduced, whereas the information is still available and reachable during navigation.

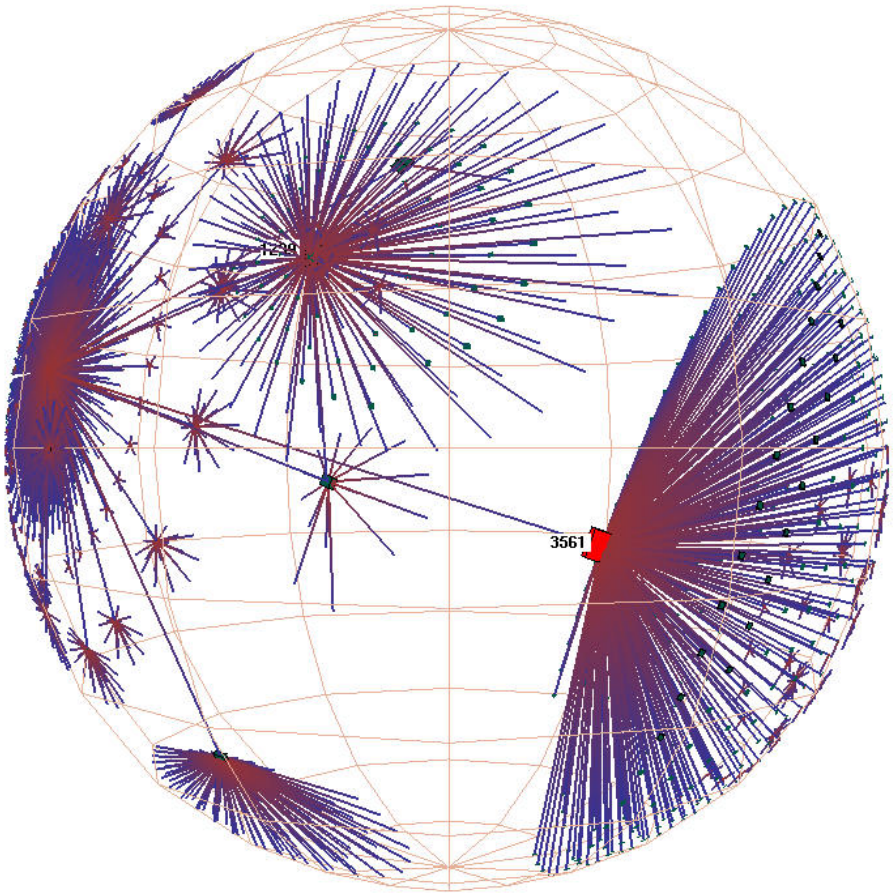

Figure 4.1: Hyperbolic tree visualization with H3Viewer. The image is copied from the paper of Tamara Munzner "H3: laying out large directed graphs in 3D hyperbolic space" published in: 1997 IEEE Symposium on Information Visualization, with permission of IEEE (©[1997] IEEE).

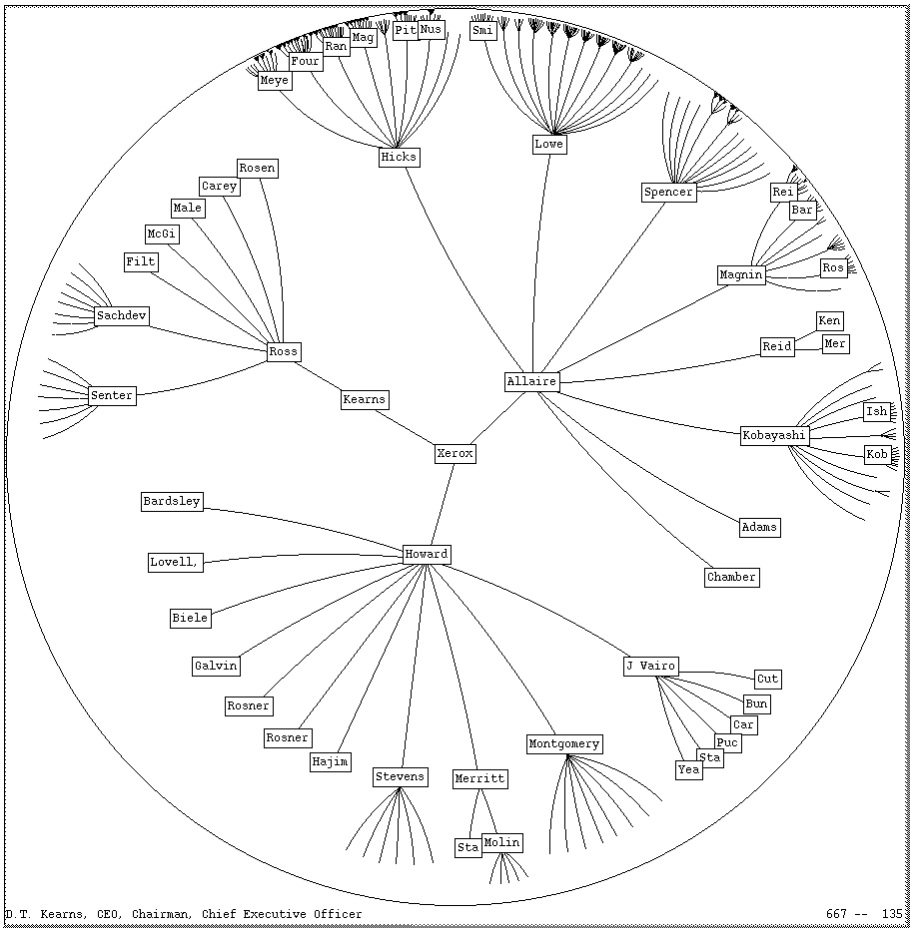

Figure 4.2: Visualization of tree with Hyperbolic Browser. The image is copied from the paper of John Lamping and Ramana Rao "The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies" published in: 1996, in Journal of Visual Languages & Computing, with permission of Elsevier (©[1996] Elsevier).

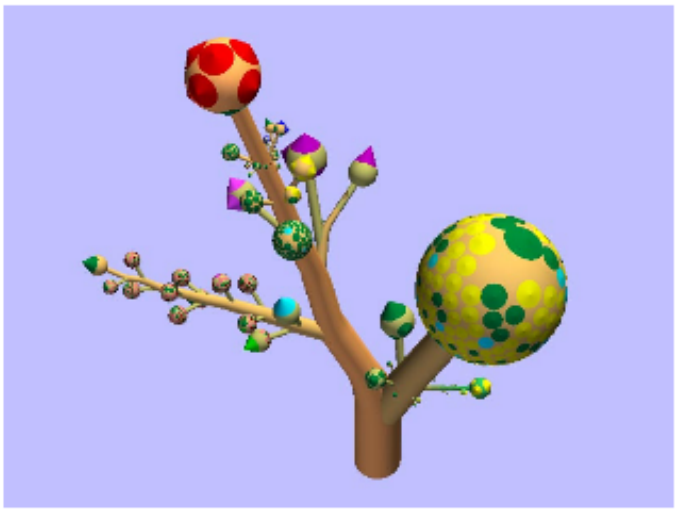

Figure 4.3: Visualization of tree with Botanical Tree technique of Kleiberg et. al [80]. (Courtesy of Jack (Jarke J.) van Wijk, Technische Universiteit Eindhoven).

In terms of another aesthetically pleasing tree visualization technique we would point to Kleiberg et al. [80]. The authors present a method where a hierarchical tree is visualized as a real tree. An example of a tree visualization with this approach is given in Figure 4.3. In this approach the non-leaf nodes are branches with the child nodes represented as subbranches and the leaf nodes are represented as fruits. This method can be applicable for ontology visualization, if an ontology is represented as a hierarchy. The representation of the leaf nodes, however, might be experienced as confusing. The leaf nodes will attract more attention, because they are represented differently. From the ontology representation point of view, however, the leaf nodes have the same status in an ontology as the non-leaf nodes; they are namely the sets representing "things" in the domain of discourse. The different representation of leaf and non-leaf nodes could be, however, exploited if ontology visualization will also support individuals. In this case, the individuals could be represented as fruits and this will distinguish them from ontology classes.

4.3 GUI Aspects of Node-Link Visualization

In Chapter 3 we have described requirements for an ontology visualization tool. Here, we will proceed with the description of our approach in context of these requirements.

We propose that levels of granularity in the data as well as interactivity should be supported. Therefore, in our approach the graph underlying the ontology can be derived for different levels of depth. In practice, this level will be a parameter that is a part of the user interface. A level 1 means that only the direct sub/superclasses and the concepts directly related to the root-concept are represented. A level 2 means that a graph is generated that also contains the children-concepts of the first level concepts.

In order to represent a complex dataset such as an ontology and make it more understandable for the user, an interactive visualization is required. Pointing devices such as the mouse can be used to navigate the graph. Each geometrical view in our approach is augmented with specific transformations, dependent on the geometry of the view. Besides these specific transformations the standard navigation techniques, i.e. zoom and rotate, are available.

It is important to see the local context for each concept. Therefore, each concept in the graph can be expanded. To that end, the mouse function is further extended by using the right mouse button to retrieve the definition of the concept. This definition is an annotation property and has no semantic meaning. However, as it contains the domain knowledge, it can be of interest to a specialist.

To fulfill the search requirement, a simple query interface has been designed and added. This query interface will be used for keyword (or concept) search. In the interface a query term is inserted and the ontology is subsequently explored for that particular term. The search is implemented as a string comparison between the query and annotated properties of all concepts in the ontology. The concepts

that comply with the query term are assembled and presented to the user as a list (list box element in the GUI). From this list a concept can be chosen (mouse interaction in the GUI) and the corresponding graph structure for this concept will be generated.

4.4 Representations of Views Based on Different Geometry

In this section we expand on the requirements for the visualization by comprehensive description of the multiple "geometrical views".

Representation of an ontology with different geometric views can be helpful in understanding of an ontology, because different views emphasize different parts of ontology structure. We can compare an inspection of an ontology by means of different geometries with an inspection of a multidimension dataset by means of different projections. Therefore, in the following sections we will discuss in more detail each of the geometrical views that we have included in our visualization approach.

4.4.1 Euclidean Views

Motivation For the completeness of the visualization approach and for comparison with other geometrical representations we have included two Euclidean views in our application.

Sphere view Our first Euclidean view we call the *Sphere* representation (cf. Figure 4.4). In this view the root concept is placed in the center. The children of this concept form the first layer $n = 1$ and are equally spread at the sphere of an empirically determined radius R. The children of the other layers $n > 1$ surround the parent node and spread at the parental hemisphere of the radius $R_n = R_{n-1}a$,

where a $(0 < a < 1)$ is an empirically determined constant.

Disk view Our second Euclidean view is a simple disk model (cf. Figure 4.4). Here children of the central concept are placed at circumference of the circle with radius R. The radius of the next layers is determined from the relation

$$R_{n+1} = R_n + ra^n, \tag{4.1}$$

where n is the layer number, r is the radius of the first layer, and a $(0 < a < 1)$ is an empirically determined constant. For each parental node an angular wedge is specified where the children of the next layer will be positioned. The size of the wedge depends on the number of the children of the given node.

Both Euclidean approaches do not produce an optimal layout of the ontology graph. This is due to the fact that the part of the circumference for disk model (hemisphere area for sphere model) that is determined at the layer n and assigned for the children at the next layer may not be sufficient to position all children. From our experience this layout can be used successfully only for ontologies with a low branching factor and up to level of depth $l < 4$.

Transformations The Euclidean views are augmented with standard transformations, e.g. zoom, pan and rotate.

4.4.2 Hyperbolic Views

Motivation Knowledge bases can be huge, e.g. the NCI-Thesaurus ontology comprises over 100 MB in OWL format. We consider the idea of "focus + context" most applicable for large knowledge bases. This visualization technique provides detailed information in the center, while at the same time the context information is conserved and presented around the central concept with a decreasing size. The hyperbolic geometry has shown to provide a very good ground for realization of this

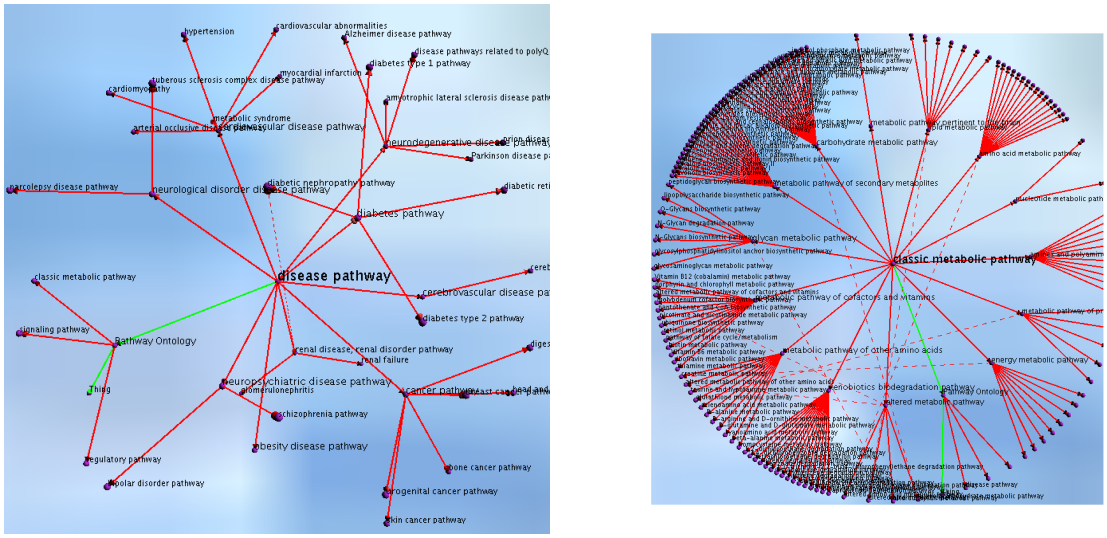

Figure 4.4: Visualization with our environment of the concept "Disease Pathway" with Euclidean 3D "Sphere view" (left) and with Euclidean 2D "Disk view" (right).

technique. The work of Munzner[89], applying the 3D Klein model in visualization of very big graphs, and work of Lamping and Rao[85], applying the Poincaré disk model for graph visualization, have been the source of inspiration for our visualization approach. Both are included and will be discussed next.

In order to make this chapter self-contained, we briefly introduce some important facts about hyperbolic geometry. The visualizations are then elaborated in the next paragraphs.

The Parallel Axiom After numerous unsuccessful attempts to deduce the fifth Euclid's postulate from other four postulates [65], the alternative consistent geometries were discovered. The postulate is given here:

Parallel Axiom 4.4.1. *Given a line l and a point $p \notin l$ there is a unique line m with $p \in m$ and $l \cap m = \emptyset$.*

It was proven that negating the "parallel axiom" does lead to geometries that

are as consistent as Euclidean geometry [65]. The parallel postulate can be negated in two ways. In spherical geometry it is not possible to draw a parallel line, given a line l and point p, because all lines intersect at the poles of the sphere. Alternatively, in hyperbolic geometry the parallel postulate can be formulated as follows:

Hyperbolic Parallel Axiom 4.4.2. *For every line l and every point $p \notin l$ there at least two distinct lines pass through p that are parallel to l.*

Models of Hyperbolic Geometry In principle, everybody is familiar with non-Euclidean geometry, because spherical objects are ubiquitous in our world. However, the objects of hyperbolic geometry are not so easy to find; lettuce leaves give some idea on how the hyperbolic plane looks like.

It is not possible to visualize the hyperbolic plane. We can only get a glimpse of the hyperbolic space by means of a projection to the Euclidean space. To that end different models exist: the Half-space model, the Poincaré or Interior of the disk model, the Jemisphere model, the Klein model, and the "Loid model". All of these projections can be mapped to each other and all mappings are described in literature[3].

4.4.3 Klein Model

There exists no isometric embedding of hyperbolic geometry in Euclidean space. Therefore, something must be sacrificed by a realization of such an embedding, either lines or angles. In the Klein model the entire hyperbolic plane is mapped to the unit disk. In this model the straightness of the lines is preserved, but angles and lengths are not preserved.

Algorithm for Graph Visualization The Klein Model was the basis for the H3 [89] graph layout in the 3D hyperbolic space. In this layout, objects in the hyperbolic volume are projected to objects in the volume of the unit sphere. The

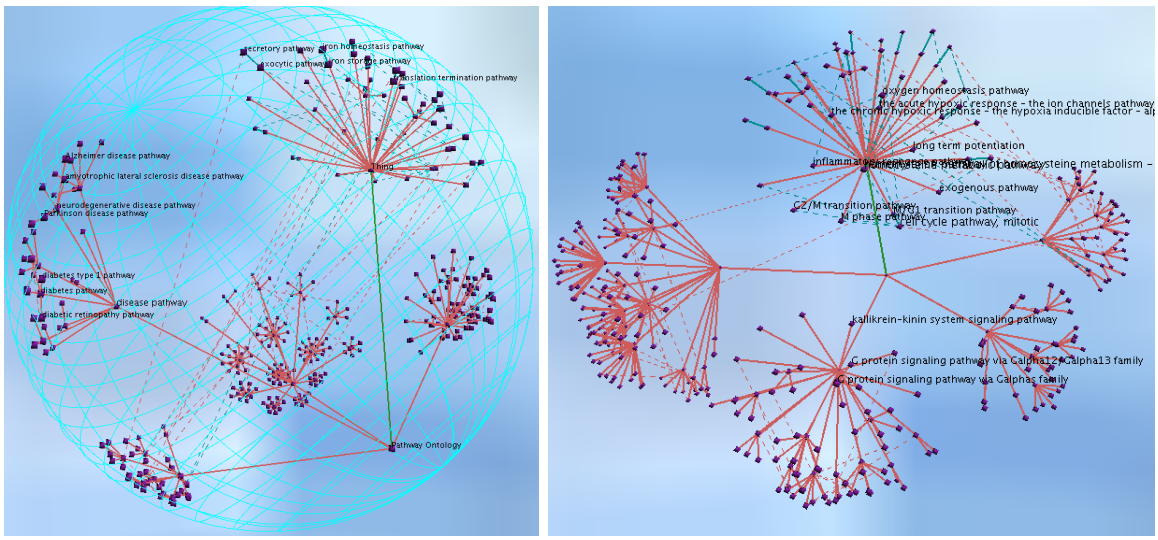

Figure 4.5: Visualization with our environment of Pathway ontology with Klein model (left). Euclidean variant of the H3 algorithm (right).

layout algorithm requires two passes. During the bottom-up pass the hemisphere radius R_h of the node is calculated. This radius is based on the area of the child disks at the bottom of the child hemisphere. During the top-down pass, based on the parental hemisphere radius R_h, the coordinates of the children are found. The calculation of the child coordinates (the θ and ϕ angles) is based on the sphere-packing technique [89]. The parent and child nodes are placed at hyperbolic distances as well as by the use of hyperbolic angles. The visualization with the Klein model is depicted in Figure 4.5. The two passes of the algorithm are refined as follows:

Bottom-up The bottom-up procedure firstly assigns the radii to the leaf nodes, then recursively moves up and calculates the hemisphere radii for the non-leaf nodes.

1. We represent the leaf nodes of the tree as disks, with a radius r. The hyperbolic area s for each leaf can be calculated with the equation:

$$s = 2\pi(\cosh(r) - 1). \tag{4.2}$$

2. Suppose, we want to find the hemisphere radius for some non-leaf node. If we know the r_i for each child i then we can calculate the area needed to place all children. This area S_h will be the sum of the child disks areas:

$$S_h = \sum_{i=1}^{n} 2\pi(\cosh(r_i) - 1). \tag{4.3}$$

Based on the formula for hyperbolic area of the hemisphere

$$S_h = 2\pi \sinh^2(r), \tag{4.4}$$

we can calculate the R_h based on the radii of the children:

$$R_h = \sinh^{-1}\sqrt{\frac{S_h}{2\pi}} = \sinh^{-1}\sqrt{\frac{\sum_{i=1}^{n} 2\pi(\cosh(r_i) - 1)}{2\pi}}. \tag{4.5}$$

Top-down During the top-down pass, on basis of the hemisphere radius R_h of the parent, the coordinates of the child nodes are calculated. These are the ϕ and θ angles, where the ϕ is the angle of the concentric band, and θ is the angle inside the concentric band. The calculations of ϕ and θ angles are based on the sphere packing algorithm. The Euclidean variant of the sphere packing algorithm with the disks of the same radius is presented in Figure 4.6.

1. The root node is placed in the origin.

2. For each non-leaf node we know the R_h, this determines the hemisphere where we need to place the children. The children are sorted in descending order. Now, the child with the largest subtree is placed in the center of the hemisphere. The rest of the children are placed in the concentric bands around the pole of the hemisphere.

   Angle ϕ_{j+1} of the band $j+1$ depends on the hemisphere radius R_h, the radius of the biggest child r_j at the band j and the radius of the biggest child r_{j+1} at the next band $j+1$. Then the $\delta\phi$ increment will be calculated corresponding the hyperbolic trigonometry by the following equation:

$$\delta\phi = \arctan\left(\frac{\tanh(r_j)}{\sinh(R_h)}\right) + \arctan\left(\frac{\tanh(r_{j+1})}{\sinh(R_h)}\right). \tag{4.6}$$

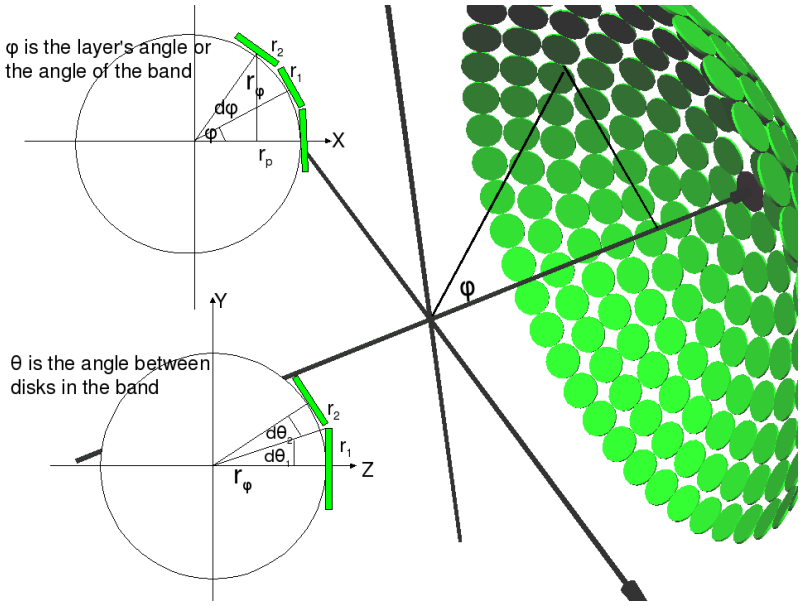

Figure 4.6: Euclidean variant of the calculation of ϕ and θ angles for the sphere packing algorithm. ϕ is the angle of a concentric band, and θ is the angle inside the concentric band. The Euclidean angle is calculated as follows:
$\delta\phi = \arctan \frac{r_i}{R_p} + \arctan \frac{r_{i+1}}{R_p}$, where r_i is the largest child in the layer i, and R_p is the radius of the sphere. The Euclidean angle $\delta\theta$ is calculated in the same way:
$\delta\theta = \arctan \frac{r_j}{r_\phi} + \arctan \frac{r_{j+1}}{r_\phi}$, where r_j and r_{j+i} are the radii of the consecutive circles placed in the layer ϕ and r_ϕ is the radius of this layer.

Given $\delta\phi$ and ϕ_j, then the angle ϕ_{j+1} will be $\phi_{j+1} = \phi_j + \delta\phi$. The angle θ_{i+1} inside the band j depends on the radius of the band R_ϕ, the radius r_i of the previous node in this band and r_{i+1} the radius of the node for which we want to calculate the angle. The radius R_ϕ depends on the ϕ_j and R_h. Correspondingly the hyperbolic right angle formula, the radius will be:

$$R_\phi = \sinh^{-1}(\sinh R_h \sin \phi_j). \tag{4.7}$$

After the radius of the band R_ϕ is known we can calculate the angle θ_{i+1} in the same way as we have calculated the angle ϕ_{j+1}.

$$\begin{aligned} \delta\theta &= \arctan\left(\frac{\tanh(r_i)}{\sinh(R_\phi)}\right) + \arctan\left(\frac{\tanh(r_{i+1})}{\sinh(R_\phi)}\right) \\ &= \arctan\left(\frac{\tanh(r_i)}{\sinh R_h \sin\phi_j}\right) + \arctan\left(\frac{\tanh(r_{i+1})}{\sinh R_h \sin\phi_j}\right). \end{aligned} \tag{4.8}$$

Transformations The hyperbolic transformations that are used for the navigation in the graph are represented as 4×4 matrices. These transformations are exhaustively described in the paper of Mark Phillips and Charlie Gunn [94]. The transformations in the Klein model applied for our approach are depicted in Figure 4.7.

4.4.4 Euclidean variant of H3 Layout

In order to support the idea that the hyperbolic 3D layout outperforms the Euclidean 3D layout we have implement the Euclidean version of this algorithm. In the Euclidean variant we have chosen for some empirically determined distance between parent node and child nodes. This is because if we calculate the distance based on the radius of the parental hemisphere, then the children will be very close to each other and the surface of this hemisphere will be covered with the labels of the child nodes. This results in an uninformative layout that looks unpleasant. Alternatively, if the children are more distant from each other, then the radius between parent

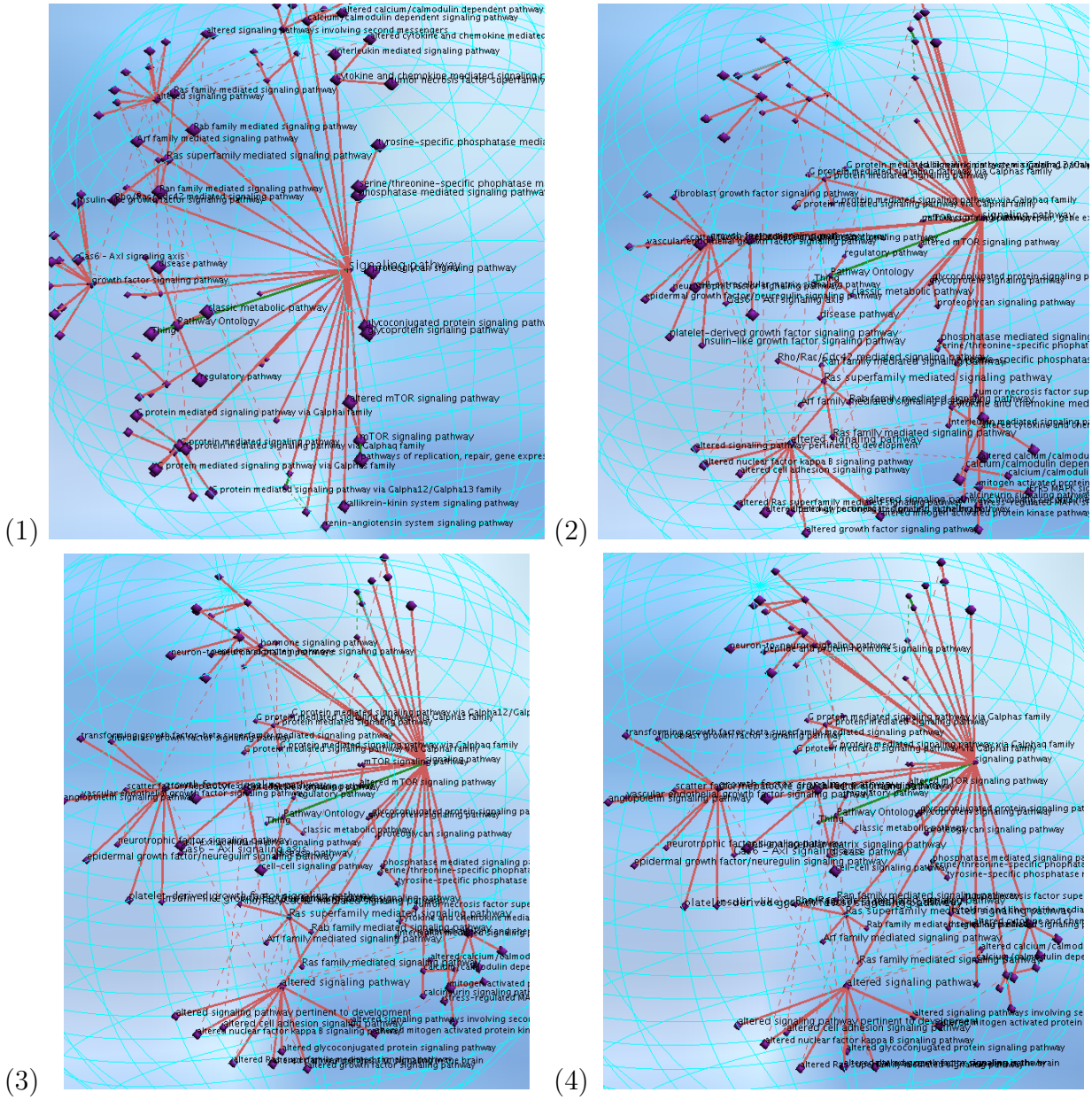

Figure 4.7: Hyperbolic transformations in the Klein model. The node "Signaling Pathway" is moved from left to right.

and child will be large, and this results in large trees, in which the children at the deepest layers are too far away from the central node. The Euclidean H3 layout is illustrated in Figure 4.5.

4.4.5 Poincaré Disk Model Description

Another hyperbolic model that we have applied for our visualization approach is the Poincaré model. In this model the infinite area of the hyperbolic plane is mapped entirely to the unit disk. All the lines become circle arc segments, orthogonal to the unit disk rim. In this non-isometric embedding, referred to as a conformal model, the angles are represented faithfully, while area and length relations are not preserved. In our approach the Poincaré Disk model [85] is applied for the graph visualization, see Figure 4.10.

Algorithm for Graph Visualization The algorithm is described in [85]. The distance d between a node at the ith layer and its child at the next layer $i+1$ is calculated by the formula

$$d = \sqrt{\left(\frac{(1-s^2)\sin(\alpha)}{2s}\right)^2 + 1} - \frac{(1-s^2)\sin(\alpha)}{2s}, \tag{4.9}$$

where α is the angle between the midline and the edge of the child's subwedge, and s is the desired distance between the child and the edge of its subwedge. Assume we know the number of the children n of the root node and distance d. If we translate each child node by the distance d and subsequently rotate it by the angle $2\pi i/n$, where i is child's index, we will get the child coordinates of the first layer of our graph. Then, for each node we need to calculate the mid line from which we will orient the children of the next layer, and the angle which will be used as the wedge where the next generation of the children will been placed.

1. Let us have node a with an angle α, and a mid line m.

2. Let this node has n children.

3. Transform child c_i to the distance d along the mid line, and rotate it by the angle $-\alpha/2 + \alpha/(2n) + i\alpha/n$, where $i \in 0 \dots (n-1)$ is the child's index.

4. For each child c_i the mid line coordinate is the point at infinity, that is the angle between c_i and the line $(0, 1)$ at the Poincaré disk

5. Calculate the angle for the child wedge. In the case that each child gets an equal sub wedge of the parental wedge the angle will be α/n. Alternatively, it is also possible to calculate for each child a weighted sub wedge, based on the formula

$$\alpha_i = \alpha w_i / w, \tag{4.10}$$

where w_i is the number of leaf nodes in the child subtree, and w number of leaf nodes in the parent subtree. This approach gives a more fair wedge assignment for the nodes, but requires one extra pass in order to calculate the number of leaf nodes for each subtree. We use the first approach where each child gets the equal subwedge angle from its parent, because of its simplicity and speed of layout generation.

Lines in Poincaré model Lines (geodesics) in the Poincaré model are the circular arcs orthogonal to the unit circle. How to build these lines with straightedge and compass is described in [59]. Assume we have two points A and B in the Poincaré disk. In order to build a hyperbolic geodesic, first we need to determine the inversion of the point A. Two points A and A^{-1} are inverse points with respect to circle C if $|OA||OA^{-1}| = r^2$, where the r is the radius of the circle and O is the center of the circle. The geodesic is the part of the circle, build through tree points A, B, A^{-1}, that connects points A and B.

Transformations The transformations on the Poincaré disk are represented as Möbius transformation.

$$z' = T(z; c, \theta) = (\theta z + c)/(\bar{c}\theta z + 1), \tag{4.11}$$

where the complex number θ describes the pure rotation in the Poincaré disk around the origin 0 and c describes the translation of the origin. By means of this transformation formula we can produce the layout of the nodes of our graph (represented as a tree). The transformations in the Poincaré model are shown in Figure 4.8.

4.4.6 Poincaré like layout

In addition, we have also investigated another algorithm that is not based on hyperbolic geometry, but generates a "focus+context" layout. The graph in this algorithm is laid out on the infinite plane and mapped to the unit disk. We call this layout *Poincaré like*. The algorithm for this approach requires three passes. These passes are:

Bottom-up During this pass we calculate the weight of each node. This weight is determined by the number of the leaf nodes, and it is calculated recursively.

Top-down During this pass we calculate on the basis of children and parent weight the angle α alloted to each node. The α is calculated in the same way as in formula 4.10. The angle α of the root node is equal to 2Π. For each node we calculate the radius R where the children have to be placed, given by:

$$R = \#Children \times (2r + \delta r)/\alpha, \tag{4.12}$$

where r is the node radius and δr is an extra wedge for each node. The procedure for calculating R and α for each node is given in pseudo-code representation in Figure 4.9. This procedure is called with the root node as parameter.

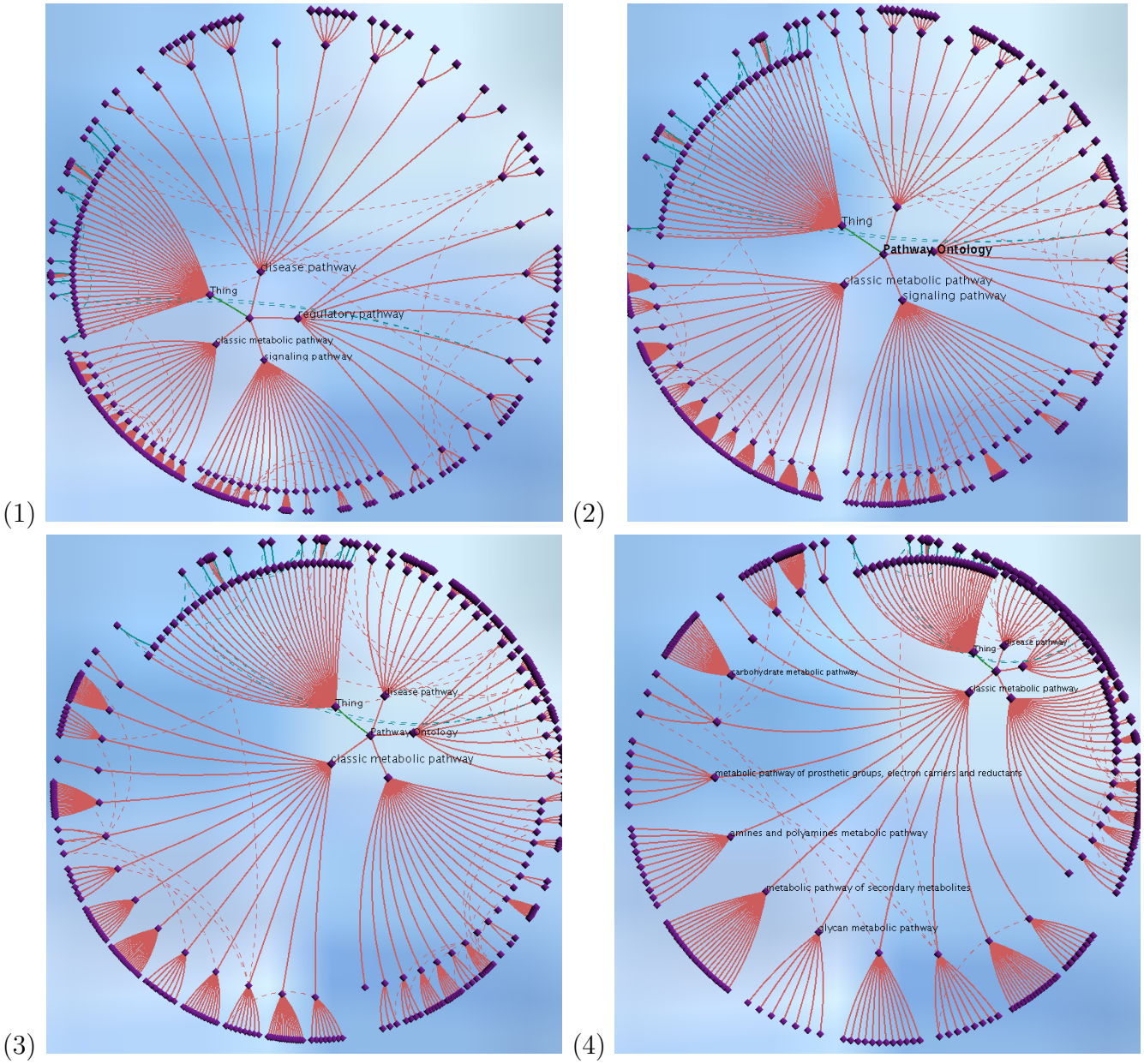

Figure 4.8: Hyperbolic transformations in the Poincaré disk model. The node "Pathway ontology" is moved from the South-West corner to the North-East corner.

```
generates alpha and R for each tree node
initialize: root.alpha = 2*Pi
initialize: R[0...n] = 0; // radius of each level is 0
determine_Alpha_R(node P){
    numberOfChildren = 0;
    for(each child C of node P){
        C.alpha = P.alpha * C.weight/P.weight;
        determine_Alpha_R(C);
        numberOfChildren++;
     } // for
     // determine the value of R required to place the children
     P.R = numberOfChildren * (2*r + delta_r)/P.alpha;
     if(P.R > R[P.level]) // update R for the given level
        R[P.level] = P.R
} //determineAlpha
```

Figure 4.9: Procedure for determining R and α in Poincaré like layout.

We need to update the radius of each level to the maximal value in this layer, because otherwise the nodes belonging to the same level will be at different distances from the center.

Mapping of infinite plane The values R and α determine the coordinates of the node at the infinite plane. In order to fit the infinite plane in the unit circle we apply the mapping realized by the following equation

$$d_u = \frac{d}{d+1}, \tag{4.13}$$

where d_u is the node distance from center in the unit circle, and d is the node distance from center in the infinite plane.

Compared to the Poincaré disk layout algorithm this algorithm needs one extra pass. However, from our point of view it generates Poincaré-like layout which is also aesthetically pleasing and informative. The results can be compared by examining Figure 4.10.

4.4.7 Spherical Geometry

To the best of our knowledge, there is no ontology visualization approaches in which spherical geometry is applied. To this end, we have elaborated on spherical geometry and implemented an another view to our panel of geometrical views, this view is referred to as Stereographic View (cf. Figure 4.11). In this view the graph is laid out on the surface of a sphere. In order to generate this view, first the graph is laid out in the infinite plane and then projected to the surface of the hemisphere.

The projection (cf. Figure 4.12) is realized as follows: the south pole of the unit sphere is placed at the distance D that is equal to the largest radius of the laid out graph. By this placement each node at the infinite plane will be projected at the northern hemisphere via the south pole, with the farthest node placed at the equator.

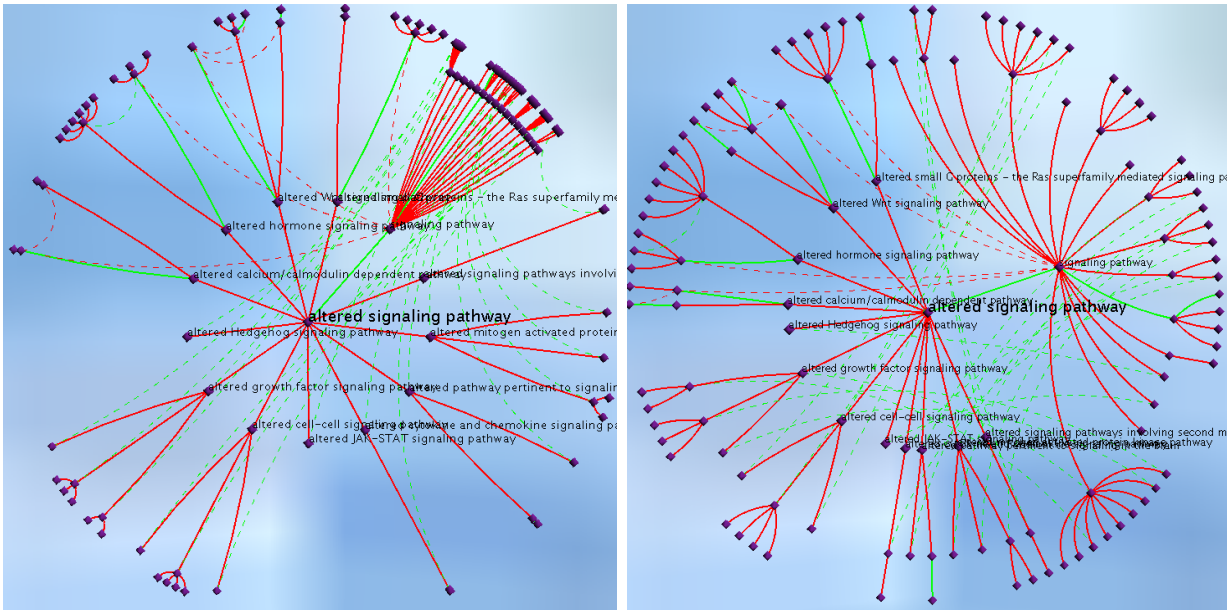

Figure 4.10: Visualization with our environment of a Pathway ontology concept *altered_signaling_pathway* with the Poincaré disk layout (left) and with the Poincaré-like disk layout (right).

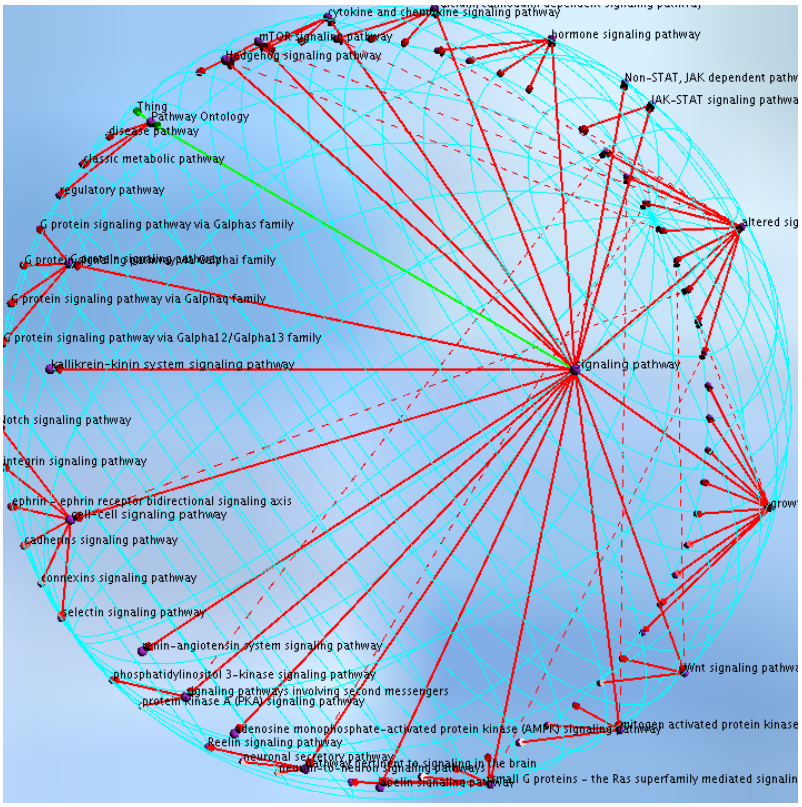

Figure 4.11: Visualization with our environment of a Pathway ontology concept *signaling_pathway* using the Stereographic layout.

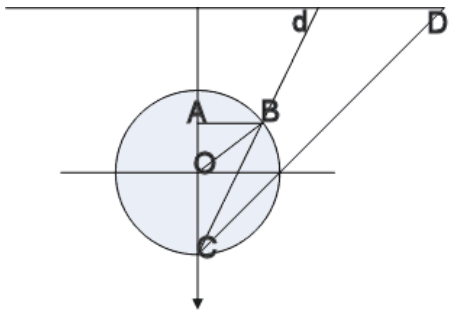

Figure 4.12: Mapping of the graph structure to sphere surface

The new coordinates (x', y', z') are calculated from the old (x, y) as follows:

$$\alpha = \arctan \frac{\sqrt{x^2 + y^2}}{D} \tag{4.14}$$

$$h = 2r\cos(\alpha) \tag{4.15}$$

$$d' = h\sin(\alpha) \tag{4.16}$$

$$x' = d'\frac{x}{\sqrt{x^2 + y^2}} \tag{4.17}$$

$$y' = d'\frac{y}{\sqrt{x^2 + y^2}} \tag{4.18}$$

$$z' = -(h\cos(\alpha) - r), \tag{4.19}$$

where D (cf. Figure 4.12) is the maximal distance from the center node in the plane to the furthest node, this distance is the same as the distance between the plane and the south pole of the sphere. Furthermore, d is the distance of some node which we map at the sphere, angle $\alpha = \angle OCB = \angle OBC$, hypotenuse $h = CB$, $d' = AB$, and sphere radius $r = OB = OC$.

Transformations The transformations used in this view require special attention. Standard transformations on the sphere will not provide any advantages above standard rotation, therefore we have explored whether the Möbius transformations [9] will give the user more insight in the ontology graph structure. We concluded that stereographic view augmented with Möbius transformations inherits both advantages from the Poincaré and spherical model, because the Möbius transformations

offer the "focus+context" insight in the structure and the spherical layout gives the 3D insight. In this manner, the structure can be explored from two different perspectives. Examples of transformations in Stereographic View are given in Figure 4.13.

4.4.8 Results with Node-Link Approach

Each geometrical representation has its benefits and shortcomings. Therefore, the question which view fits best for an ontology does have more than one answer. The global information, i.e. what a domain is about, what are concepts in this domain, can be visualized very well using the hyperbolic geometry. Alternatively, if we want to see the local context of the concept of interest, then the Euclidean views are more suitable.

In Euclidean geometry each graph node needs the equivalent space for its visual representation. However, the backbone of the graph is a spanning tree structure, and number of nodes increases exponentially in the tree. For this reason the Euclidean views are only sufficient for a small number of layers; from our empirical observations this should not exceed three.

The hyperbolic views offer a possibility to use more space, because the circumference of the circle in the hyperbolic plane increases exponentially with the radius. For this reason the hyperbolic geometry is more suitable for the representation of global visualization of larger structures.

In the spherical geometry we can benefit from 3D representation of the graph laid out at the plane, because the stereographic projection provides the possibility to map this graph on the surface of a hemisphere. This view, however, shares the disadvantages with the Euclidean view, because the individual nodes in the deeper layers are difficult to distinguish in a crowd of concepts.

In order to appreciate the differences between geometrical views, all views

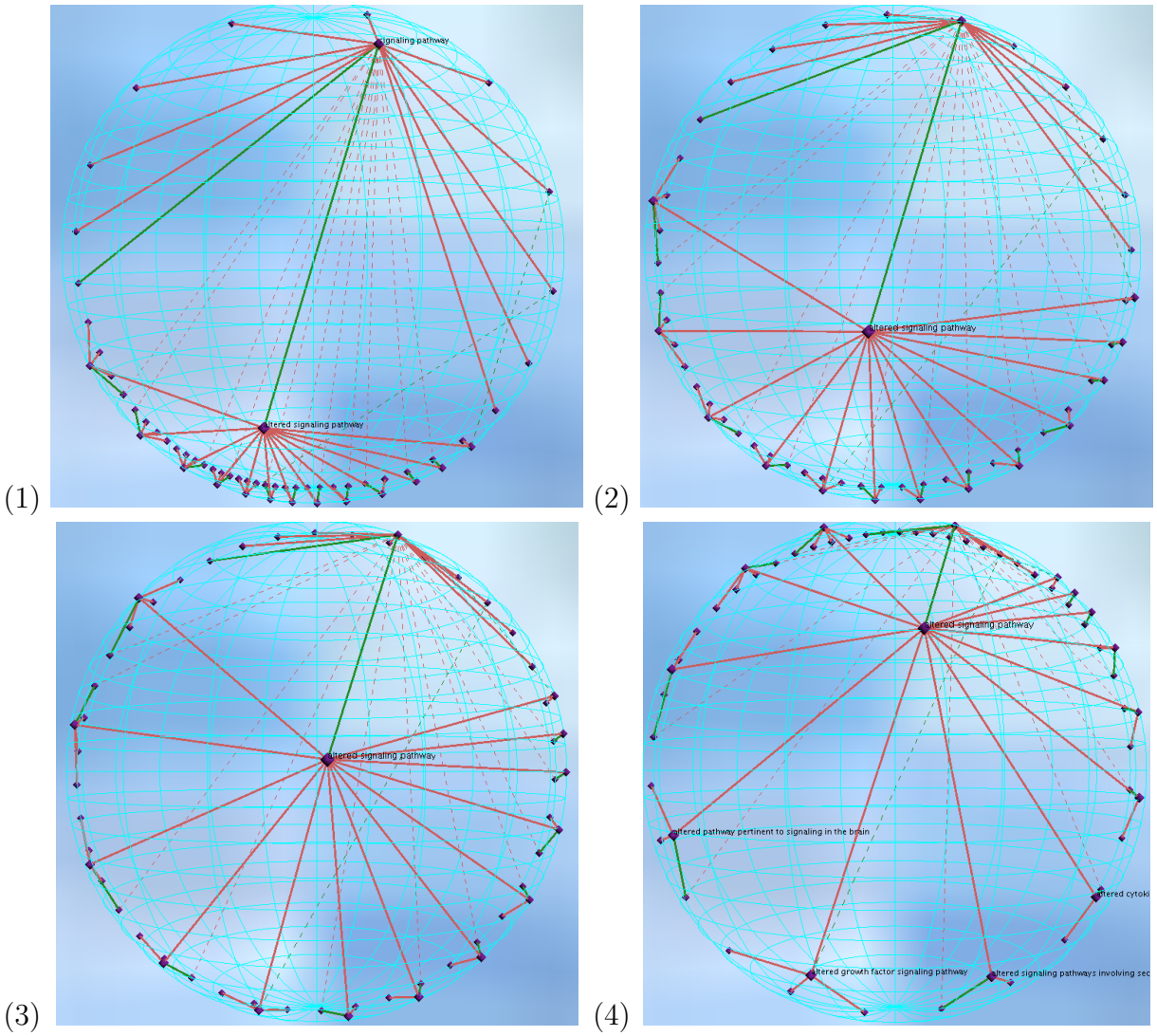

Figure 4.13: Möbius Transformations in the Stereographic View. The node "altered signaling pathway" is moved from a bottom position to a top position.

together of the concept *Regulatory Pathway* are depicted in Figure 4.14.

4.5 Implementation Details

All implementations were realized in the Java language. To generate the *virtual* graph structure with predefined level of depth and selected properties, we have used two API's. OWL-API[18] is used to parse an ontology to extract concept annotations and restriction axioms. The Pellet reasoner[17] is used to extract the inferred hierarchy, that defines the sub/super class relations in our graph and to verify properties of the given concept that are extracted from the axioms. The visualization component is developed on the basis of Java-3D API[6]. We have reused source code from the H3Viewer[4] and Walrus[32].

4.6 Conclusions

In this Chapter we have presented how different geometries can be used for node-link ontology visualization. Our node-link visualization approach is a combination of different geometrical views augmented with corresponding interactions. The user interactions depend on the geometry of the view, e.g. in the Euclidean views the Euclidean transformations are used, and in hyperbolic views the hyperbolic transformations. This multi-view concept can help the user to see different parts of the complex structure. 2D and 3D Euclidean views can be helpful to show the local context of some concept, both 2D and 3D hyperbolic views can be suitable to show the whole structure of the ontology, and at the same time these views give a clear representation of the information in the "focus". The Stereographic view inherits the transformations from Poincaré view, and at the same time provides the possibility to explore the graph in a 3D perspective, because the graph is laid out at the surface of a sphere.

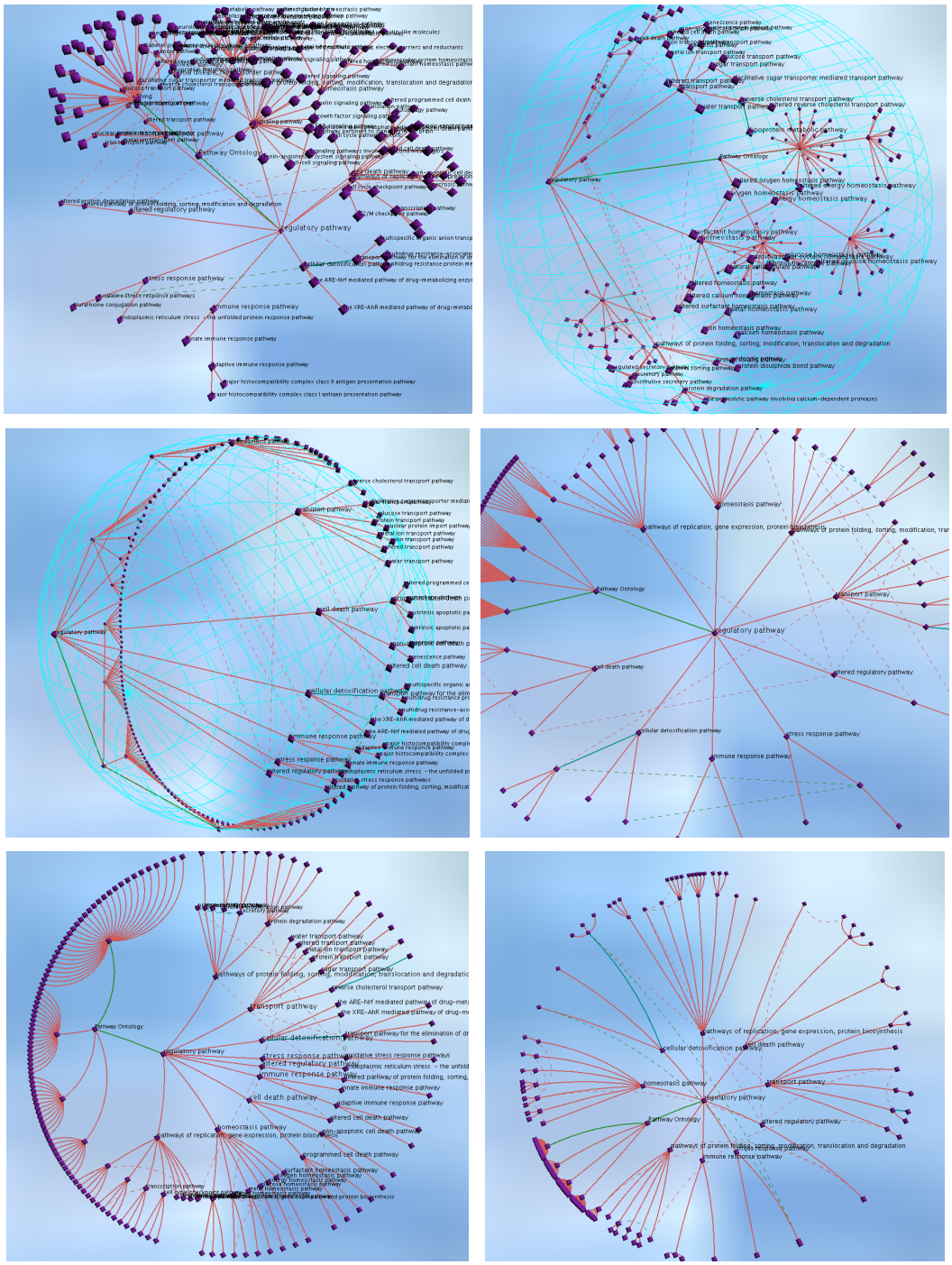

Figure 4.14: Visualization with our environment of all views together: Euclidean $3D$, Hyperbolic $3D$, Stereographic, Euclidean $2D$, Poincaré-like, Poincaré of the concept *Regulatory Pathway*.

The novelty of our methodology lies in combination of different geometries in the context of ontology visualization. There exists no visualization application in which Euclidean, hyperbolic and spherical geometry are integrated in one visualization suite. Having all geometries available, one can benefit from the advantages that each of these geometries offers in inspecting an ontology through visualization. Moreover, we consider the Stereographic view as provided in our visualization framework a useful new technique for ontology visualization.

The visualization methods that are presented in the visualization framework follow the requirements that are formulated in Chapter 3. For each of the methods a heuristic evaluation of the interaction was done and in the test environment the used interaction was included. Now that the visualization artefact is available further user studies can be done. The flexibility of the visualization framework provides users with new possibilities for the exploration of ontologies through the node-link approach. In the next chapter, in addition, the containment approach will be elaborated.

The visualization framework is available on-line [46] for probing and inspection as a Java Web Start application.

Chapter 5

Ontology Visualization with Containment Method

Based on:

"Node-Link and Containment Methods in Ontology Visualization"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of 6th International Workshop on OWL: Experiences and Directions (OWLED 2009),

"Different Geometries in Ontology Visualization"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of SPIE, Visualization and Data Analysis 2010 conference (SPIE 2010)

Abstract For the visualization of the ontology hierarchy we use the containment method. We were inspired by 2-dimensional circular variant of treemaps method used in the CropCircles approach. However, in our visualization approach we introduce a 3-dimensional ($3D$) spherical alternative. In this visualization nodes are represented as spherical caps and placed on the surface of a sphere. Each parental node contains its children, which are places on the surface of the parental cap. In addition, we augment our approach with concept of the semantic zoom, by means of which it is possible to choose a level of detail by using a simple mouse interaction.

5.1 Introduction

Containment methods use the paradigm of nested containers in order to represent hierarchical tree structures. This paradigm can be successfully applied to ontology visualization. An ontology contains classes which represent sets of things with particular properties. From this perspective, general classes contain more specific classes, like realm of wines comprises white, red and rosé wines. White wines include Chardonnay, Pinot Blanc, Sauvignon Blanc, White Burgundy, etc. This set of sets paradigm in ontology could be very well reflected by the containment method.

In our containment ontology visualization method [51] we were inspired by the CropCircles application [113]. We have, however, moved from the $2D$ circular packing idea of this method and introduced a 3D spherical alternative in which the most general concept *Thing* is represented as a sphere, and more specific concepts are represented as spherical caps. Children caps are placed above the parental caps. The placement of the children is realized by the sphere packing algorithm, the same algorithm was also the basis for the 3D hyperbolic layout in [89] described in Chapter 4. Furthermore, to increase interaction, we augment our approach with the semantic zoom technique. By using of geometric zoom, when zooming in or out, a visual object will be only rescaled, no more visual detail will be revealed. While with semantic zoom, the level of detail depends on the distance of the object from the viewer. In the case of ontology visualization, the in/out zooming will trigger the appearance/disappearance of children on the surface of their parent. In this way an uncluttered visualization can be presented while more detail can be obtained through semantic zooming.

5.2 Related Work

In the early stage of development of information visualization, the only available approaches for representation of hierarchical structures were command line listings

and directory tree outlines. In these representations each item of the hierarchy needs one line on a display, hence, only small hierarchies could be visualized. Later, different node-link approaches were developed. These approaches, however, are frequently criticized for inefficient use of display space. Indeed, the display in these methods contains more empty space in comparison to the space allocated to the nodes of the graph. To cope with this problem the containment methods have been developed [104].

The basis for the containment method in the field of information visualization was introduced by Shneiderman in [104]. In this treemaps approach the hierarchical tree structure is mapped to a rectangular window. This is realized by means of space-filling algorithm, where for each subtree, rectangular space is allocated with the area proportional to the weights of its children. The weight depends on the size of the subtree. With this method such structures as directory trees or organization hierarchies could be represented very well, see an example in Figure 5.1 The treemaps method has, however, a number of drawbacks. In treemaps representation the content of leaf nodes (size of files or *importance* of the smallest unit in an organization) is more important then the structural information. Moreover, for the balanced trees, where the nodes have the same size, it will be not possible to reveal the structural information because the treemaps will degenerate into a regular grid [112].

Compared to files in a directory, the content of ontologies is rather different. The leaf nodes in ontology hierarchy are not distinguishable from each other and have the same size. Moreover, from semantical point of view there is no difference between leaf and non-leaf nodes. The treemaps method, however, is giving more attention to leaf nodes. The algorithm strives to use a space in the most efficient way and provides only little space for the bounding box that comprises the children nodes, because the more space is given for the bounding box the less space is allowed for items representing the actual information, namely for files in the case of directory visualization.

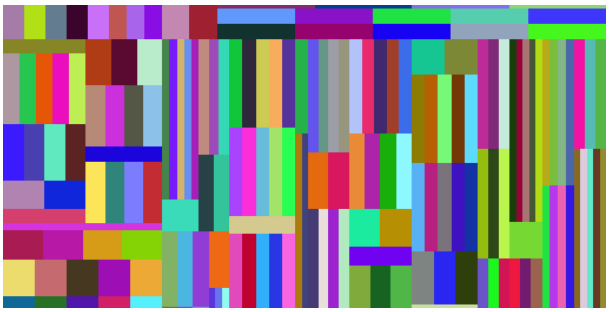

Figure 5.1: Visualization of a random generated tree structure with treemaps method.

To cope with the problem of indistinguishability of structural information in treemaps method, van Wijk and van de Wetering have introduced a variant of treemaps method (Cushion Treemaps) [112] that was successfully applied in directory visualization [26]. The authors augment the original treemaps algorithm with the shading effects. They use the differences in shade in order to represent entities as illuminated surfaces. With this method the structural information can be revealed much more clearly.

Although the Cushion treemaps method emphasizes the structure of the hierarchy, the use of shading effect is not enough in order to visualize a hierarchy such as an ontology. The reason is the same as in the case of the original treemaps method, namely, most space is allocated to the leaf nodes, but in an ontology the leaf nodes are not distinguishable from the non-leaf nodes, they are just a representation of a current knowledge about concepts in a domain. In later version of an ontology this knowledge may change, upon availability and acquisition of new information, then current leaf nodes could be populated with children.

In the ontology visualization, treemaps method has inspired Wang and Parsia in the creation of the CropCircles [113] visualization method. We have already introduced their method in Chapter 3. In this visualization the authors do not strive for space efficiency, but for a clear representation that supports understanding of

the ontology hierarchy. Hence, instead of rectangles that can be completely filled with other rectangles, circles are used. It is not possible to fill a circle with another smaller circles, and thus there will be empty space. This can be considered as one of the advantages for this method. The child nodes are better distinguishable from each other. Moreover, it is always clear who is the parent of a particular child.

5.3 Spherical Variant of Containment

In our spherical variant of the containment approach we replace the standard two-dimensional space-filling approaches[113, 109] with a sphere surface packing technique.

The skeleton of the ontology is the hierarchy, which represents subclass relationships between parent and child. We start at the root node, in the case of ontologies it will be the most general concept *Thing*. Then, the direct subclasses of the *Thing* will form the first level of our visualization. The nodes are placed on the surface of the sphere in layers. The second layer is formed with the children of the first layer children. Each node is represented as a spherical cap. The children caps are nested in the parental cap, and are placed at the surface of the bigger sphere to give the impression of a relief in the visualization. We can compare the nodes of the first layer with continents on the surface of the Earth. The nodes of the second layers then will be the countries inside the continents. The nodes of the third layer can be compared to the provinces or counties.

5.3.1 Description of Visualization Algorithm

In order to be able to place the spherical caps, which represent the nodes of the ontology, on the surface of the sphere, the radius of the sphere R needs to be determined. This is done on the basis of the recursive algorithm presented here in pseudo-code (cf. Figure 5.2).

```
determineRadius(node){
if (node is leaf) node.radius = r // constant radius of leaf node
else {
    area = 0
    for (each child c){
         determineRadius(c);
         area += Pi * (c.radius + extra)^2;
    }
    if(node is root)
        node.radius = sqrt(area/4PI) // determine the sphere radius
    else
        node.radius = sqrt(area/PI) // determine the circle radius

}
```

Figure 5.2: Procedure for determining node R in the Containment method.

The radius R of the root node is taken as the radius of the sphere on which the child nodes will be positioned in layers. In the computational procedure, the area calculated for each node is just an approximation of the spherical cap area. This spherical cap area can only be determined if the radius of the sphere on which the nodes are placed is known. However, that particular sphere radius required the calculation of the area of the caps. This "chicken and egg problem" is bypassed by approximating the spherical cap area with the area of the circle.

Layered Placement of the Nodes Each node will be placed as a spherical cap on top of the parent. The nodes then are sorted on the basis of their radius, the largest radius has the node with the largest number of leaf nodes. The placement algorithm calculates spherical coordinates of a node (ϕ, θ angles), and places the node over the sphere surface, subsequently the algorithm recurses on the each child of the node.

On the basis of their approximated area, the algorithm first calculates the angles that the children caps will have.

$$\alpha = \arccos(1 - \frac{S}{2\pi R^2}), \tag{5.1}$$

where S is the node's area, and R is the sphere radius. After that, the children will be placed in layers on the basis of the sphere packing algorithm (as described in [89] and depicted in Figure 4.6). The ϕ angle of the $(i+1)$th layer depends on angle α_i of the previous layer i and angle α_{i+1} of the first node in the layer $i+1$. When all the children nodes are placed, we can measure how much the ϕ angle of the last layer exceeds the α angle of the parental node. When ϕ exceeds the allowed value of α, we calculate the coefficient

$$k = \frac{\alpha}{\sum_{i=1}^{n} \delta\phi_i}, \tag{5.2}$$

where n is the number of layers and $\delta\phi_i$ is angle of the biggest child in layer i. Then, for each child node we can calculate the correct angle α_{new} for the spherical cap

$\alpha_{new} = \alpha k$. After this postprocessing step the children nodes will fit in the parental nodes.

Now the children nodes are places over the parental spherical caps on the surface of the sphere with the radius $(1 + \delta)R$, where $\delta < 1$. This is applied to accomplish an effect of a relief in the visualization.

5.3.2 Semantic Zoom

Through standard zooming the user can only observe the different geometrical scales of an object. Through semantic zooming, when zooming in/out, the user can control the level of detail in the visualized object. Node placement in layers is very suitable for implementation of semantic zoom. The (dis)-appearance of the levels is controlled by the distance of the sphere with which the user interacts. When the sphere is far away only the lowest layers are visualized, such as the direct subclasses of the *Thing* concept. When the user zooms in, direct children of the nodes of the first layer appear above the spherical caps of their parents. The result of semantic zoom is depicted in Figure 5.3, Figure 5.4, and in Figure 5.5.

5.3.3 Results with Containment Approach

We argue that our spherical containment method is very suitable for the representation of the hierarchical structure of an ontology. We assert this on the basis of the fact that other containment ontology visualization approaches, like Jambalaya[109] are popular in ontology community. The Jambalaya tool, however, only shows part of the hierarchy, and it requires multiple interaction steps in order to comprehend the hierarchy. In contrast, with our approach, exploration of the hierarchy of an ontology can proceed more intuitively, by using a standard zoom functionality, which is further enhanced with semantic zoom technique.

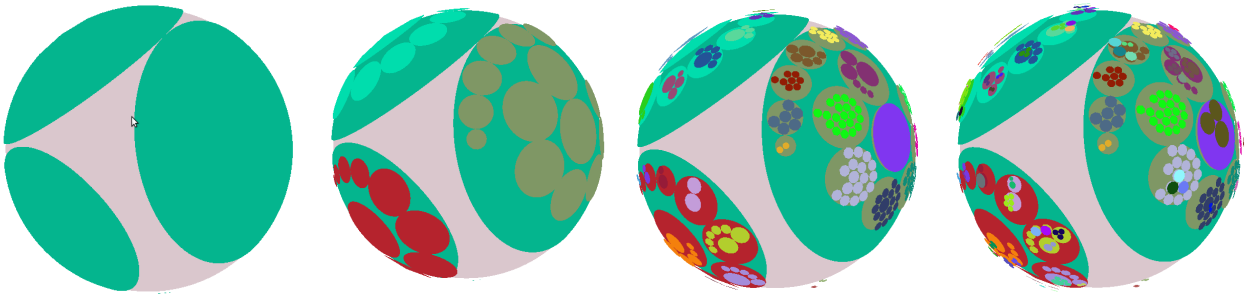

Figure 5.3: Consecutive changes of the visualization of the levels of ontology as result of zooming. Most left is a visualization of direct children of *Thing*. The more to the right, the more levels appear and more detail is visualized. The children concept caps can be observed as being *above* the parent caps. The labels of nodes can be retrieved on demand.

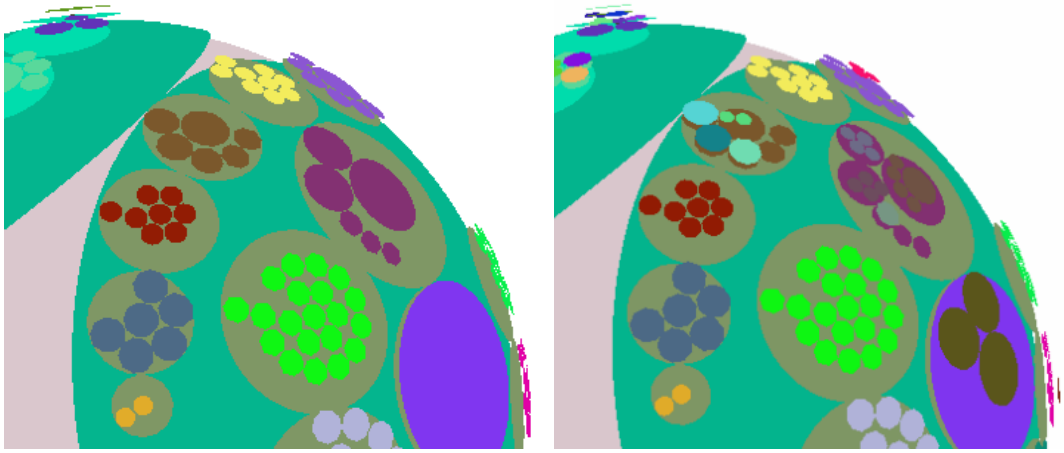

Figure 5.4: Detail from the right-most pictures of Figure 5.3. The effect of the semantic zoom can be appreciated by inspection the appearance of new child concepts in the right figure as more zoom has been applied.

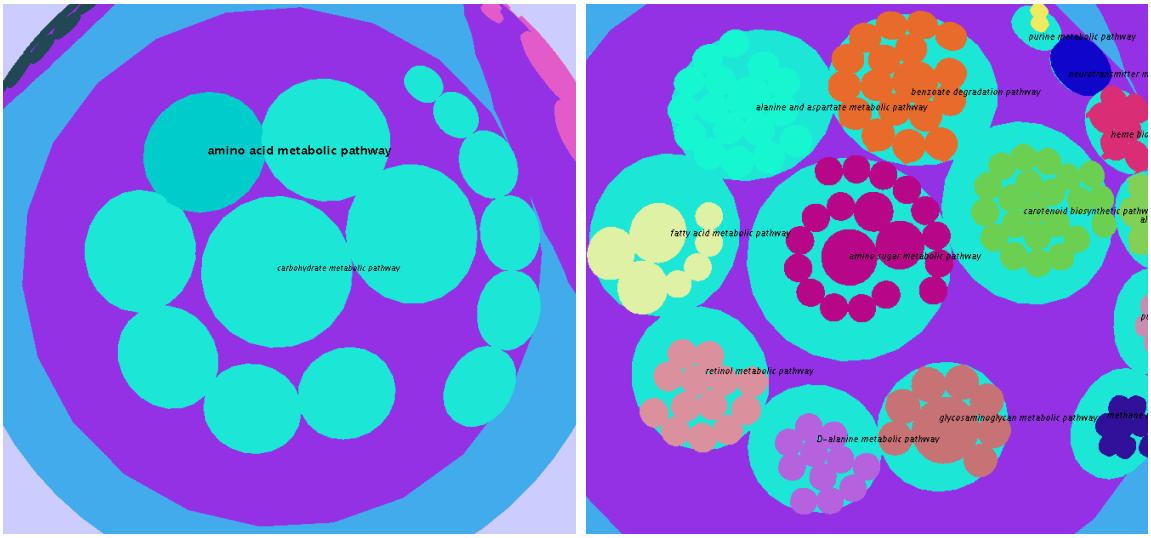

Figure 5.5: Visualization of the Pathway ontology. Detailed view generated when semantic zoom is applied from the level 4 to the level 5.

5.4 Conclusions

In this chapter we have introduced an alternative of the containment method for ontology visualization based on the sphere surface packing technique. This method is inspired by the CropCircles visualization, but instead of a 2D circle-filling approach, we have chosen for a non-standard 3D alternative representation. The nodes are placed as spherical caps on the top of the parental caps in different layers. This layered placement technique turns out to be very applicable in combination with semantic zooming. Therefore, we have augmented our approach with semantic zoom interaction. This combination provides a possibility to control the level of detail in addition to the standard geometric zoom functionality. This new method is suitable for the representation of the hierarchical ontology structure and should also be considered as a visualization method for other tree structures.

This approach for visualization satisfies the requirements formulated in Chapter 3. The 3D containment method is intrinsically interactive and this interactivity

helps in the exploration of an ontology. The application of the semantic zooming technique within this visualization method makes it easier to address the deeper layers, i.e. the detail, of the ontology while the notion of the sphere maintains the relation with the bigger picture, i.e. the overview. The intuitiveness of this method is very suitable for further user centered studies that are focused on strategies for information retrieval from ontologies. The containment visualization approach is a useful tool for such studies.

Chapter 6

Integration of Modules

Based on:

"Creating a new Ontology: a Modular Approach"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of Semantic Web Applications and Tools for Life Sciences (SWAT4LS 2010)

"Modular Approach for a new Ontology"

Julia Dmitrieva and Fons J. Verbeek

published in proceedings of 5th International Workshop on Modular Ontologies ESSLLI 2011 (WoMO 2011)

Abstract In the life sciences, researchers are working in the data rich area where different domains frequently overlap. Overlapping information can be utilized at the moment the domains are integrated. A typical case is the drug discovery process; the information from different domains, e.g. diseases, proteins, pathways, drugs, etc. need to be integrated in order to connect a disease with genes, pathways and find a potential chemical compound that can be active as a drug. However, information from different domains is often available in different ontologies. In order to combine these data an ontology integration approach is required.

In this chapter we demonstrate an approach in which a new integrated ontology is created from modules that are extracted from different ontologies. First, we extract small modules from ontologies. Module extraction is based on well defined notions of modularity, locality and conservative extensions. The signature of the modules is based on symbols of the user interest. Subsequently, the mappings between the similar concepts are generated. Finally, on the basis of these mappings we integrate modules in one ontology.

6.1 Introduction

In life sciences, ontologies, in particular available in the OBO Foundry [16] and BioPortal [91] repositories contain information about species, proteins, chemicals, genomes, pathways, diseases, etc. Information in these ontologies might overlap, and it is possible that a certain concept is defined in different ontologies from a different point of view and at different level of granularity. Therefore, the combination of information from different ontologies is useful for the creation of a new ontology.

Case Study The integration will be illustrated with a case study on Toll-like receptors. As it is known, Toll-like receptors are important in immune response, they recognize molecules specific for pathogens and activate immune system. In order to create an ontology about Toll-like receptors it is important to use ontologies which can provide information about immune response, cell, cell membrane, proteins, biological pathway, biological process, diseases, etc. Therefore, we have chosen for ontologies in the biomedical domain provided by OBO Foundry [16].

If we want to investigate what kind of information about Toll-like receptors is available in the Molecule Role Ontology (MoleculeRoleOntlogy) [16], then we will learn that Toll-like receptors are defined as pattern recognition receptors (see Figure 6.1). In the Biological Process Ontology (part of GO) [16] the Toll-like receptors are described in the context of a signaling pathway and are subsumed by the *pattern recognition receptor signaling pathway* concept. In Protein ontology [16], a Toll-like receptor is just a protein (see Figure 6.1). In NCI_Thesaurus ontology [28], Toll-like receptors are defined as *Cell Surface Receptors*. From these examples follows that multiple ontologies model different aspects of the same concept and the combination of the available information provides more knowledge about concepts that are of potential interest to ontology developer.

A number of methods have been developed for ontology integration. Some of these methods are dedicated to reuse parts of ontology, such as the modular approach

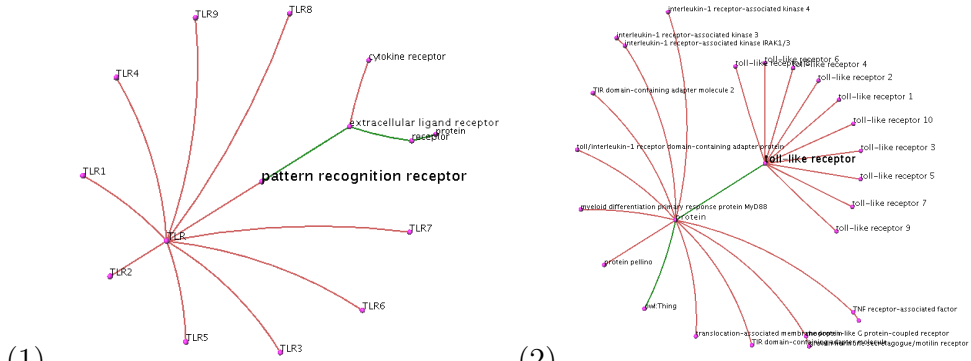

Figure 6.1: Visualization of (1) the concept TLR from Molecule Role Ontology and (2) the concept Toll-like receptor from Protein ontology.

[62] and the *MIREOT* approach [42], other are devoted to connect concepts from different ontologies with each other, such as Distributed Description Logics (DDL) [36] and $\mathcal{E}-$connections [83]. We are interested in the creation of a new ontology about a topic of user interest. Therefore, we introduce an approach for generating a new ontology from different ontologies obtained from the OBO Foundry repository/library [50]. In this method first, we extract modules from these ontologies, on the basis of the well defined modularity approach [62]. As a signature for the modules we are using the symbols that match the terms of interest. In our case study we create an ontology about Toll-like receptors, therefore we use two *seed* terms (Toll, TLR). Subsequently, we create mappings between concepts in the modules. It has already been shown [56] that the simple similarity algorithms outperform structural similarity algorithms in biomedical ontologies. Therefore, we have based our mappings on the similarity distance [86] between labels and synonyms of classes in the modules. Finally, a new ontology is created where the mappings are represented by means of the owl:equivalentClass axiom and small concise modules are imported.

6.2 Related Work

The current mechanism of integration OWL ontologies is based on the OWL:IMPORTS axiom. This mechanism, however, contains several drawbacks. First, ontologies can be very large, and including a foreign ontology then leads to importing all ontologies in the transitive closure. The price for the import can be very high, because processing, querying and reasoning in the ontology tend to be time consuming operations. Second, developers are not necessarily experts in the domain of the ontology to be included, i.e. he or she can be only interested in a small subset of the domain. Furthermore, the import can damage the consistency of the including and included ontology, because the original concepts can be defined with the foreign concepts, and the other way around, the concepts from foreign ontology can be redefined in the including ontology [57].

Because of the disadvantages of the OWL:IMPORTS mechanism, there is a need for alternative strategies for ontology integration. To that end, there is ongoing research in the area of ontology modularity and integration.

In [75], for example, the authors provide a mechanism and a tool support for reusing of ontology. They describe a method where a module of ontology can be extracted and reused in *a safe and economic way.* Based on the undecidability results for such problems as whether $\mathcal{Q}_1$ is an S module in $\mathcal{Q}$ [62], they have based a module extraction on the notion of *locality* [62].

An alternative direction for reuse of parts of ontology, however, without preserving the logical structure and inferences, was chosen by developers of OBI ontology [14] in their approach referred to as Minimum Information to Reference an External Ontology Term (MIREOT) [42].

MIREOT MIREOT is a set of guidelines which are used in order to create the Ontology of Biomedical Investigations (OBI [14]). These guidelines are based on importing of parts of foreign ontologies into the ontology under development. The

imported part is simply the class of interest and its superclass with annotations (label, comment, definition). Although this method guarantees the minimal reuse, the logical inferences about the reused class, in the first place, are not complete, because no axioms about this class are imported. In the second place, reuse of this class can lead to unintended inferences. We are, however, interested in a methodology that does not just reuse classes of interest, but also guarantees that logical inferences about these classes are conserved.

Along the same lines of ontology reuse and integration the related theoretical formalisms, e.g. $\mathcal{E}$-connections [83] and Distributed Description Logics [36], can be considered. We will, therefore, shortly discuss these formalisms.

Distributed Description Logics In Distributed Description Logics (DDL) different knowledge systems are combined by means of a new set of axioms, so called bridge rules [60]. The DDL formalism with *bridge rules* is based on the idea of distributed and independent ontologies, which can be linked together. Next to the mechanism for ontology interconnection, there is also reasoning support available that provides the possibility to reason in distributed ontologies [103].

$\mathcal{E}-$connections A formalism related to DDL is the $\mathcal{E}-$connections. This technique can be used to combine different knowledge bases expressed in different languages. This technique provides a possibility to connect different ontologies by means of *Link Properties*, which are defined as properties between classes of source ontology and classes of some foreign ontology. If, for example, the source ontology is from the domain of chemical substances, and a foreign ontology is from the domain of drugs with the classes *chem:acetyl_salicilyc_acid* and *drug:aspirin* respectively, then we can define a *LinkProperty* $Is\_Used\_In\_Drug$ connecting two given classes.

In our work we are not using DDL and $\mathcal{E}-$connections. First, because these formalisms are currently in development and not standardized yet. Second, both these formalisms require the detailed knowledge of the ontologies reused. The de-

veloper needs to be expert in all domains described in reused ontologies in order to interlink them whether with *bridge rules* or with *Link Properties*. Our methodology is, however, dedicated to the user which is not a specialist in biomedical domain. Finally, DDL and $\mathcal{E}-$connections methods are developed for interlinking ontologies or for connecting autonomous ontological modules obtained from large ontologies. We are, however, interested in a method with which we can create a new ontology on the basis of terms of user interest and from ontologies which are available (on the Internet).

6.3 Module Extraction

There are two main directions in the area of module extraction. One of them is based on structural approaches [102], [90]. A typical example of the structural approach is provided in [90] where the authors describe a method which can be used to extract self-contained parts (*traversal views*) from ontologies. In their research a *traversal views* is based on the set of concepts and relations of the user's choice and the ontology graph generated on the basis of the hierarchy and chosen relations. As pointed out in [75] the structural approaches are not providing a guarantee that such module is complete and logically correct. Moreover, from studies [62] it follows that modules generated by structural methods are not minimal. Therefore, we have based our module extraction on the well defined notions of locality, conservative extensions and modularity as described in [62] and [75].

In [62] the concept of a *module* is introduced where a module captures the meaning of a given set of terms; in addition the algorithm for module computing is presented. In this thesis we use the definition of the *signature* and *module* as given in [62]. The signature is defined as follows:

Definition 6.3.1 (Signature)**.** A signature Sig of a DL is the disjoint union of sets C of atomic concepts $(A, B, \ldots)$ representing sets of elements, R of atomic roles

$(r, s, \ldots)$ representing binary relations, and I of individuals $(a, b, c, \ldots)$ representing constants.

In other words, the signature of an ontology is a vocabulary or a set of symbols used in the knowledge base. The module is defined as follows:

Definition 6.3.2 (Module)**.** Let L be a description logic, $\mathcal{Q}_1 \subseteq \mathcal{Q}$ be two ontologies expressed in L and S be a signature. $\mathcal{Q}_1$ is an S–module in $\mathcal{Q}$ w.r.t. L, if for every ontology $\mathcal{P}$ and every axiom α expressed in L with $Sig(\mathcal{P} \cup \alpha) \cap Sig(\mathcal{Q}) \subseteq S$, we have $\mathcal{P} \cup \mathcal{Q} \models \alpha$ iff $\mathcal{P} \cup \mathcal{Q}_1 \models \alpha$.

This means that all inferences that could be realized about symbols over signature S after whole ontology $\mathcal{Q}$ is imported in ontology $\mathcal{P}$ are the inferences that could be done after importing module $\mathcal{Q}_1$ in $\mathcal{P}$. Hence, the import $\mathcal{Q}$ in $\mathcal{P}$ will not add extra information about the symbols from signature S compared to importing $\mathcal{Q}_1$ in $\mathcal{P}$.

In [75] the authors describe a system that, in a *safe and economic* way, can extract relevant parts from ontologies that can be further used in the development of a new ontology. We elaborate on these methods for our module extraction, because, as it seems, these approaches allow to generate small and logically correct modules.

6.3.1 Modules from Enriched Signature

For our case study we are interested in the creation of an ontology concerning *Toll-Like* receptors in an automated way. Therefore, we have used the following biomedical ontologies most of which are obtained from the OBO Foundry:

- National Cancer Institute Ontology (NCI_Thesaurus [28])
- GO Ontology (GO)
- Protein Ontology (PRO)

- Dendritic Cell ontology (DENDRITIC_CELL)
- Pathway ontology (PATHWAY)
- Molecule Role Ontology (MOLECULEROLEONTOLOGY)
- Gene Regulation Ontology (GENE_REGULATION)
- Medical Subject Heading ontology (MESH [8])
- Chemical Entities of Biological Interest (CHEBI)

The Medical Subject Heading Ontology (MESH), currently not a member of OBO FOUNDRY, was fetched from the resource [12].

A module comprises knowledge of a part of the domain that is dedicated to a set of terms of interest (*seed terms*). Let T_1 be this set. In our case study we have used two terms *Toll* and *TLR*. Let S_1 be a set of terms (signature) from the ontology O_1 that represents the classes whose labels, descriptions, ID, or other annotation properties contain the symbols from T_1. The first module that we have extracted is the module from NCI_THESAURUS M_1. This is chosen because it is the largest ontology and it is expected to contain most matches with the *seed terms*. In order to generate a signature for the next ontology O_2, we are using not only the terms from T_1 but we enrich this set with the terms from the module M_1. Consequently, the set of terms for the generation of the second module M_2 will be the collection of two sets $T_2 = Sig(M_1) \cup T_1$. The same procedure is applied to the rest of the ontologies, namely module M_i is extracted on the basis of the terms $T_i = Sig(M_{i-1}) \cup T_{i-1}$. Hence, during the module extraction more and more symbols are collected that can be matched and used for the signature extraction from a following ontology. This method, however, has two drawbacks. First, it depends on the order of ontologies. The symbols that are matched in ontology O_i are based on the terms from previously generated modules $\cup_{k=1}^{i-1} Sig(M_k) \cup T_1$. So, if for example, an ontology O_{k+1} contains symbols that have match only with the symbols from the module extracted from

ontology O_k then these symbols will be never discovered if O_{k+1} will be processed before O_k during the module extraction process. This is because the module from O_{k+1} will be empty. Second, with the generation of the new module M_i new symbols can be introduced that will match symbols from ontologies used in previous steps. In order to overcome these drawbacks we will introduce the generation of the fixpoint. This will be discussed in section 6.3.2.

The reason that we have used the symbols from the modules $M_1, M_2, \ldots, M_{i-1}$ created in previous steps and not only symbols from T_1 is that not every ontology contains entities that have matches with T_1, thus, only few ontologies can be considered for module extraction. All ontologies used in this experiment, however, contain overlapping symbols, thus the symbols from module extracted in previous step can match the symbols from ontology used in the next step.

6.3.2 Fixpoint Modules

We have investigated whether or not we will find a fixpoint [1] with our method for module extraction. The schematic representation of fixpoint finding procedure is depicted in Figure 6.2.

The fixpoint is reached at the moment the set of terms T which is used in order to generate modules during step t_i does not change any more after another run with all ontologies. This can be written as $\cup_{k=1}^{n} Sig(M_{k,i}) = \cup_{k=1}^{n} Sig(M_{k,i+1})$, where $M_{k,i}$ is the module k created during the step t_i. It can be formulated in a "fixpoint-like" way $Mod(T) = T$. The algorithm (cf. Figure 6.3) is used to find a fixpoint.

In Table 6.1 the sizes of the modules with which the fixpoint was reached are depicted.

[1] In mathematical terms, a fixpoint (fixed point) is a point that is mapped to itself by the function ($f(x) = x$).

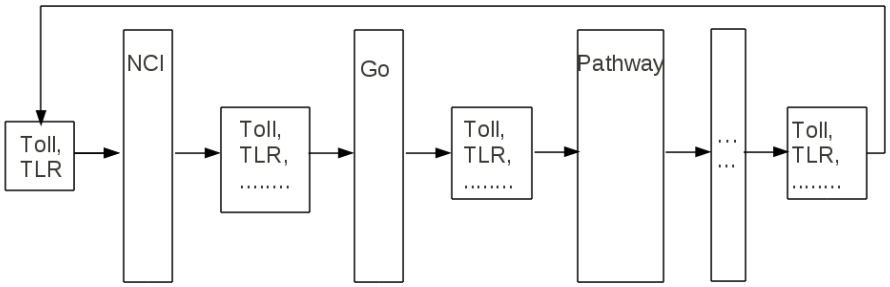

Figure 6.2: Schematic representation of fixpoint finding procedure. First, the set of terms $T = TOLL \cup TLR$ is used to find matches in NCI_Thesaurus ontology. Subsequently, for the extraction of the module M_{i+1} the set of terms T is enriched with the symbols from the module M_i extracted in the previous step. The fixpoint is reached when no new symbols are introduced in the set T.

Table 6.1: The size of the modules created on the basis of seed terms (*Toll, TLR*) after reaching fixpoint.

module	size in KB	original $\mathcal{O}$ size in KB
Module_from_gene_regulation	88.7	418.8
Module_from_protein	23.4	9900
Module_from_chebi	218.6	17800
Module_from_mesh	59.2	6200
Module_from_dendritic_cell	4.2	57.5
Module_from_pathway	4.1	360.5
Module_from_cellular_component	35.4	18400
Module_from_molecular_function	11.4	18400
Module_from_MoleculeRoleOntology	46.9	4100
Module_from_biological_process	221.1	18400
Module_from_Thesaurus	802.1	154700

```
generate the set of modules M_1, M_2, ... , M_n
from ontologies O_1, O_2, ... , O_n
T = {"Toll", "TLR"}; // the set of seed terms
T_is_changed = true;
while(T_is_changed){
   T_is_changed = false;
   for(each ontology O_i){
      Sig_i = find_matched_entities(O_i, T);
      M_i = extract_module(Sig_i);
      for(each term t from M_i){
         if(t is not in T){
            add t to T
            T_is_changed = true;
         }// if
      }// for
   }// for
}// while
```

Figure 6.3: Algorithm that finds fixpoint modules for different ontologies

6.3.3 Properties of Fixpoint

In order to show that the fixpoint algorithm (cf. Fig. 6.3) finds a fixpoint we have to assert:

- the algorithm terminates after reaching of the fixpoint,
- the fixpoint is unique and does not depend on the order of processing of the ontologies.

Theorem 6.3.1 (Fixpoint Algorithm Terminates)**.** *Fixpoint algorithm (cf. Fig. 6.3) terminates when a fixpoint is reached.*

Proof. Each module extraction procedure can be represented as a set of rules. Without loss of generality, we will consider the simplest case with only 2 ontologies O_1 and O_2. Let Σ be a signature, and let the following rule set R_1 correspond with the generation of new symbols by the module extraction procedure from the ontology O_1:

$$\begin{aligned} \Sigma &\rightarrow A \\ B &\rightarrow C \\ D &\rightarrow E \\ G &\rightarrow H \end{aligned} \tag{6.1}$$

Then, the signature Σ will generate the set of symbols A. In the same manner, the set of symbols B will generate C and the set of symbols D will generate the set E. Regarding the modularity approach [62], the following property holds $Mod_{\Sigma \cup \mathcal{M}}(O) = \mathcal{M}$, hence module extraction procedure over the symbols of module $\mathcal{M}$ will generate the same module. Thus, we can assert that there is no rule corresponding with the symbols generated on the right hand side (RHS) (A, C, E, H) for rule set R_1. In other words, insertion of a symbol that was previously generated does not introduce new symbols.

Let the following set of rules R_2 correspond with the generation of new symbols by the module extraction procedure from the ontology O_2:

$$\begin{aligned} \Sigma &\rightarrow F \\ A &\rightarrow B \\ C &\rightarrow D \\ K &\rightarrow L \end{aligned} \tag{6.2}$$

The fixpoint algorithm (cf. Fig. 6.3) iteratively generates new symbols, which are used as input for the module extraction in each iteration of the *while loop*. This procedure terminates because both sets of rules are finite. It follows from the fact that the module extraction procedure ($Mod_S(O) = \mathcal{M}$) at each step generates a subset of an ontology $\mathcal{M} \subseteq O$ that itself contains a finite set of axioms/symbols. Hence, only a finite set of symbols can be used in the rule set. When all rules have fired no new symbols will be introduced, stop condition will be satisfied, and the algorithm terminates. □

In the worst case, when each symbol on the right hand side (RHS) of one rule set will have corresponding symbol on the left hand side of another rule set, the algorithm requires $n + m$ steps, where n and m are the sizes of rule sets. It leads to extraction of whole ontology as a module. This situation is, however, not realistic because it is not possible that all symbols on RHS of O_i will match symbols on LHS of O_j.

It is important to prove that the fixpoint found by the fixpoint algorithm (cf. Fig. 6.3) is unique and independent of the order of processing of the ontologies in the module extraction process. Therefore, we introduce the following definition:

Definition 6.3.2 (Chain of Fire)**.** A *chain of fire* $CF_\Sigma(O)$ (in short CF) from an ontology O over a signature Σ is the sequence of rules triggered by signature Σ started from ontology O.

For example, for the rules 6.1 and 6.2 introduced in Theorem 6.3.1 the *chain*

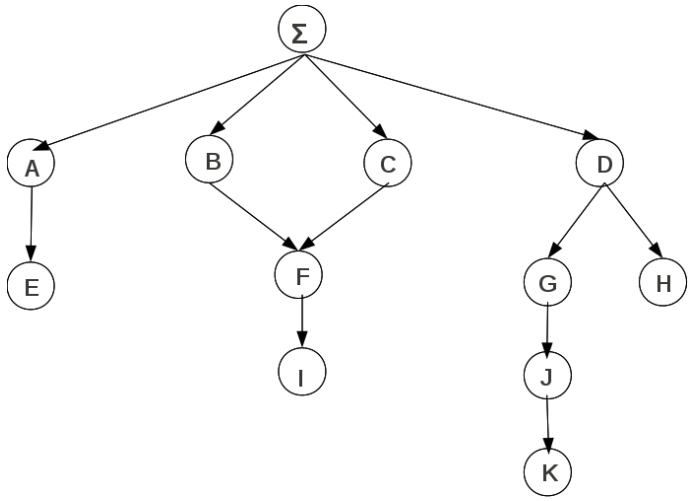

Figure 6.4: *Chains of Fire* for ontologies O_1, O_2, O_3, O_4 corresponding with the rule sets given in 6.5.

of fire started from ontology O_1 is determined as follows:

$$CF_\Sigma(O_1) = \Sigma \to A \to B \to C \to D \to E, \tag{6.3}$$

and started from ontology O_2 is determined as follows:

$$CF_\Sigma(O_2) = \Sigma \to F. \tag{6.4}$$

It is easy to see that each *chain of fire* is uniquely determined by the given set of rules and Σ. When the number of ontologies is more than two ($n > 2$) we will have n rule sets. In this case Σ triggers n rules of the type $\Sigma \to S$.

In Figure 6.4 different *chains of fire* (CFs) for ontologies O_1, O_2, O_3, O_4 are represented. These CFs are constructed on the basis of the following rule sets:

$$\begin{array}{cccc} O_1 & O_2 & O_3 & O_4 \\ \Sigma \to A & \Sigma \to B & \Sigma \to C & \Sigma \to D \\ D \to G & F \to I & A \to E & B \to F \\ C \to F & D \to H & G \to J & \\ J \to K & & & \end{array} \tag{6.5}$$

This example shows that there can be more than one CF generated from one ontology, because one symbol can match LHS of more than one rule. In this case CF will split in different subchains, for example in Figure 6.4, after generation of D, two rules $D \to G$ and $D \to H$ will be triggered and CF will split it two subchains. It is also possible that different CFs at some point generate the same symbol and then proceed with firing along the same chain. This case is represented in Figure 6.4 for the symbols B and C that both generate F.

Theorem 6.3.3 (Independence of Ontology Order)**.** *The fixpoint generated by algorithm (cf. Fig. 6.3) is uniquely determined by the rule set* $\{R_1, R_2, \ldots, R_n\}$ *and independent of the order of ontologies* $\{O_1, \ldots, O_n\}$.

Proof. During the first iteration, the algorithm (cf. Fig. 6.3) triggers all rules of the type $\Sigma \to S$, where S is a set generated after module extraction procedure from ontology O over signature Σ. Obviously, each CF will fire because Σ triggers each chain from each ontology.

It is obvious that every rule in each CF will fire. Suppose that it is not true, thus the fixpoint was reached and there is a rule $S_i \to S_j$ that was not triggered. This is possible only when S_i was not generated. It can happen only when all rules in the path $\Sigma \to S_1 \to \ldots \to S_i$ were also not processed, thus the given chain was not triggered at all. It contradicts the fact that Σ triggers each chain.

Because CF is uniquely determined by the given rule set R and each CF will fire, fixpoint will be uniquely determined by the rule set. Moreover, because each rule in each chain will fire, the order of rules is not important, thus the fixpoint (or modules generated at fixpoint) is independent from the order of ontologies. □

6.4 Ontology Mapping

In this thesis we use a less stringent definition of the concept *mapping* compared with the definition given in [76] in which mapping is a morphism and determined in the following way:

Definition 6.4.1 (Total Mapping)**.** A total ontology mapping from $O_1 = (S_1, A_1)$ to $O_2 = (S_2, A_2)$ is a morphism $f : S_1 \rightarrow S_2$ of ontological signatures, such that, $A_2 \models f(A_1)$, where A_i is a set of axioms in ontology.

In this thesis *mapping* is not considered as a morphism, but as a partial function that maps from subset $S_1 \subseteq Sig(O_1)$ to subset $S_2 \subseteq Sig(O_2)$. We deliberatively reject the morphism requirement, thus, the structural dependencies will not be preserved after mapping. This is because we are interested in consequents of this mapping to the original ontologies, namely, whether and how the structural dependences will be broken.

For our experimental prototype system we use our own mappings. Experiments with available alignment tools, such as Alignment API [2] do not give satisfactory results. Simply, we did not succeed to find mappings in ontologies, although the similar concepts were present. The reason for this setback can be found in the fact that the Alignment API and corresponding alignment algorithms are more directed to find the structural similarities, and not the syntactic similarities that are of interest to us. In order to find similar concepts we apply the string similarity. It has been already shown [56] that in the case of biomedical ontologies simple mapping methods are sufficient and outperform more complex methods.

Our mapping algorithm is based on the Levenshtein Distance [86]. Here we will discuss how this algorithm works.

Let us have a set of concepts from ontology O_1. For each concept we extract a number of characteristics. These characteristics will be further used in the

[2]Ontology alignment API and implementation, http://alignapi.gforge.inria.fr

calculation of the similarity, and these are:

Label The label of the concept. This is a name of the concept.

Synonyms Collection of names for the same term as present in different vocabularies.

ID The concept identifier. In some ontologies this is a unique string generated during serialization of the ontology with a dedicated tool. In other ontologies concept ID can be the same as the name of the concept. Because this property depends on how an ontology is serialized, it may get only a little weight during the calculation of the similarity.

We compare these characteristics for all classes from ontology O_1 with the same characteristics for all classes from ontology O_2. The comparison is based on the Levenshtein distance algorithm [86]. In order to have a metric that is independent of the length of the string and a metric which is normalized (in the range $[0 \dots 1]$), we have used the Levenshtein distance algorithm and introduce the similarity metric Lev.

For two strings A and B the metric Lev is calculated by the following equations:

$$A\&B = L_{max} - L_d \tag{6.6}$$

$$A \setminus B = L_A - A\&B \tag{6.7}$$

$$B \setminus A = L_B - A\&B \tag{6.8}$$

$$Lev = \frac{A\&B}{(A \setminus B + B \setminus A + A\&B)}, \tag{6.9}$$

where $A\&B$ is the common substring for A and B, L_{max} is the length of the longest string, L_d is the Levenshtein distance, $A \setminus B$ is the length of A without B, $B \setminus A$ is the length of B without A, L_A and L_B are the lengths of A and B respectively, and Lev is a new metric satisfying our constrains. When two classes C_i and C_j from ontologies O_i and O_j are compared respectively, the metrics of label Lev_{label}, synonyms Lev_{syns}

and ID Lev_{id} can be combined to one metric $Lev = w_1 Lev_{label} + w_2 Lev_{syns} + w_3 Lev_{id}$, and used for the comparison. In this new metric a w_i is a weight that determines the importance of the Lev_i and $\sum_i w_i = 1$. Two classes C_i and C_j are considered to be similar if they have the maximum value for Lev metric and if this value is also higher than the threshold t. In our experiments we have used an empirically determined threshold value $t = 0.95$; the lower values of t generate less precise mappings.

6.5 Integration Information from Ontologies

The final step of the ontology creation is the integration of the modules into one ontology. This is done on the basis of mappings. If there is a mapping found between two classes C_i and C_j, from the modules M_i and M_j respectively, we add the equivalence relation OWL:EQUIVALENTCLASS between these classes in the new ontology. Besides the equivalence relationships the new ontology contains the OWL:IMPORTS axioms, in which all the created modules are imported.

So far, this all seems rather straightforward. However, the problem with such integrated ontology $O_{1...n}$ is that it contains a lot of unsatisfiable [3] classes. In order to understand the reason of this unsatisfiability we have applied different experiments. First, we have merged all pairs of the modules, namely $\forall_{i \neq j} O_{i,j} \equiv M_i \cup M_j$. For each merged ontology $O_{i,j}$ we have checked for unsatisfiable classes. It was the case that already at this stage of integration different merged pairs contain unsatisfiable classes. To this end we have used the *explanation* functionality of the Pellet [105] reasoner in order to reveal the reasons of unsatisfiability.

6.5.1 Solving Unsatisfiable Classes in Merged Pairs

Here we describe how we have solved unsatisfiability in merged pairs of ontologies.

[3] From logical point of view a concept C is satisfiable w.r.t. a knowledge base $\mathcal{K}$ iff there is an interpretation $\mathcal{I}$ with $C^{\mathcal{I}} \neq \emptyset$ that satisfies $\mathcal{K}$ and is unsatisfiable otherwise.

Table 6.2: Number of unsatisfiable classes in the merged pairs of modules

	M_{greg}	M_{prot}	M_{chebi}	M_{mesh}	M_{denc}	M_{pw}	$M_{cellcom}$	M_{molfun}	$M_{MolRole}$	M_{bioPr}	M_{Thes}
M_{greg}	na										
M_{prot}	0	na									
M_{chebi}	41	0	na								
M_{mesh}	54	0	0	na							
M_{denc}	0	0	0	0	na						
M_{pw}	0	0	0	0	0	na					
$M_{cellcom}$	0	0	0	0	0	0	na				
M_{molfun}	0	0	0	0	0	0	0	na			
$M_{MolRole}$	2	0	0	0	0	0	0	0	na		
M_{bioPr}	0	0	0	0	0	0	0	0	0	na	
M_{Thes}	271	0	0	189	79	0	119	7	50	35	na

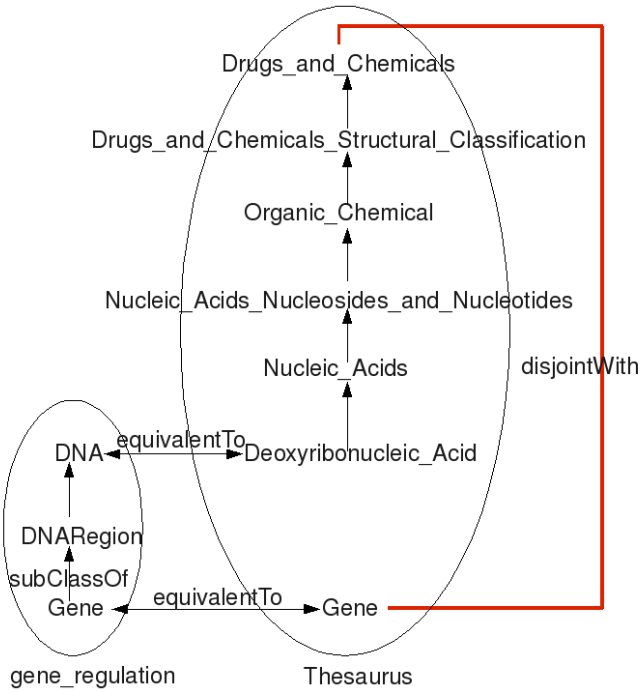

Figure 6.5: Explanation for unsatisfiability of the class Gene for the module $O_{Thes \cup greg}$. The reason is that *Thes:Gene* and *Thes:Drugs_and_Chemicals* are disjoint.

From Table 6.2 it is clear that the module created from NCI_THESAURUS and GENE_REGULATION $O_{Thes \cup greg}$ contains the largest amount of unsatisfiable classes.

In Figure 6.5 explanations for unsatisfiability of the class *Gene* from the merged ontology $O_{Thes \cup greg}$ are depicted. From this figure follows that the class *Thes:Gene* $\equiv$ *greg:Gene* is subsumed via the chain of subclass relationships and via the equivalence *Thes: Deoxyribonucleic_Acid* $\equiv$ *greg:DNA* by the class *Thes:Drugs_and_Chemicals*, but the classes *Thes:Gene* and *Thes:Drugs_and_Chemicals* are disjoint. This shows that the NCI_THESAURUS ontology appears to be too restrictive. To repair this flaw we therefore remove this restriction from the module O_{Thes}.

After removing this restriction from O_{Thes} the class *Gene* was still unsat-

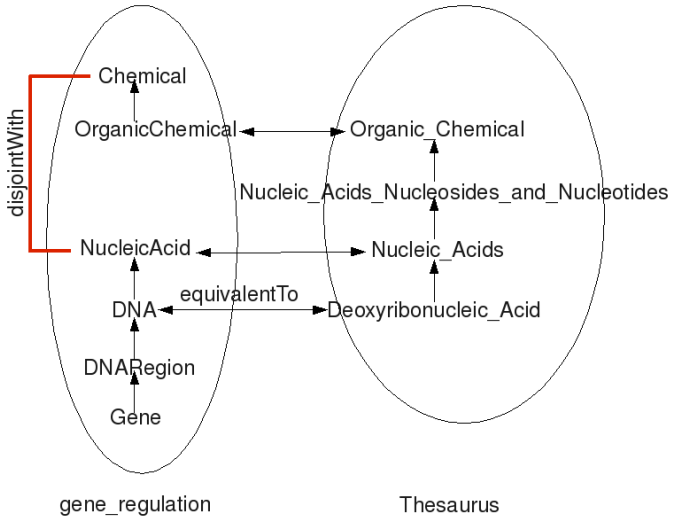

Figure 6.6: Explanation for unsatisfiability of the class Gene for the module $O_{Thes \cup greg}$. The reason is that *greg:Chemical* and *greg:NucleicAcid* are disjoint.

isfiable. In this case the reason was the statement *greg:Chemical disjointWith greg:NucleicAcid*, but via equivalences with classes in NCI_THESAURUS, the *greg:NucleicAcid* was subsumed by *greg:Chemical*, see Figure 6.6 for an explanation. This points to a too restrictive modeling in the GENE_REGULATION ontology or potential design errors.

In Figure 6.7 the explanation for unsatisfiability of the concept *Chromatin* from $O_{Thes \cup greg}$ is presented. Here *Chromatin* becomes subsumed under *Thes:Anatomic_Structure_System_or_Substance* and *Thes:Drugs_and_Chemicals* which are disjoint in NCI_THESAURUS. Moreover, the unsatisfiabilities of the class *Binding* is caused by disjointness, see Figure 6.8. For the explanation for unsatisfiability of *Translation*, see Figure 6.9.

After removing erroneous equivalences, such as *Thes:Normal_Tissue equivalentTo greg:Tissue*, and a number of disjointness axioms there are no unsatisfiable classes left in the integrated ontology $O_{Thes \cup greg}$.

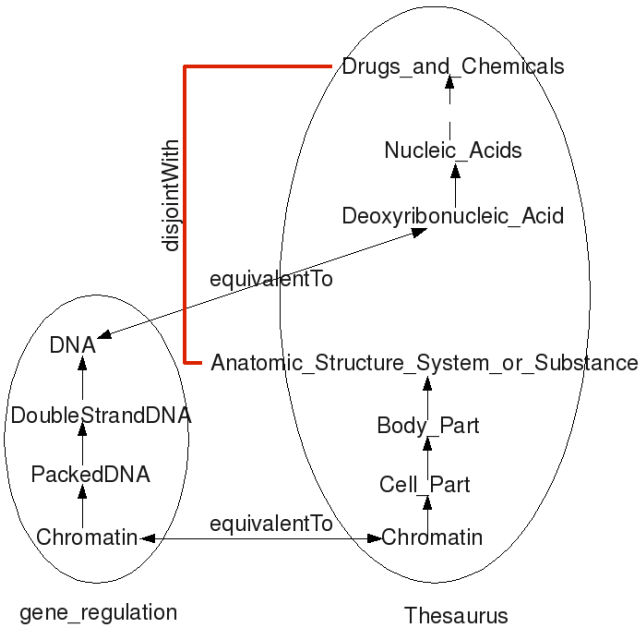

Figure 6.7: Explanation for unsatisfiability of the class *Chromatin* for the module $O_{Thes \cup greg}$.

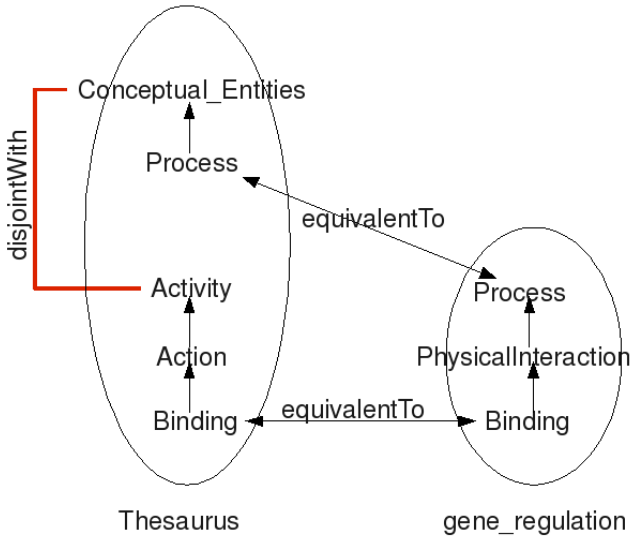

Figure 6.8: Explanation for unsatisfiability of the class *Binding* from the integrated pair $O_{Thes \cup greg}$.

Another interesting unsatisfiability pattern emerges from the integration of NCI_THESAURUS ontology with MOLECULAR_FUNCTION ontology. The explanation for this unsatisfiability is depicted in Figure 6.10. The class *Nucleic_Acid_Binding* was subsumed by *Activity* and *Biological_Process* via equivalence with *GO_0005488* (binding) concept. These classes are disjoint in NCI_THESAURUS, however this disjointness is not correct, because the *Nucleic_Acid_Binding* is a binding, is a process and is an activity at the same time. This reveals potential modeling errors in NCI_THESAURUS ontology.

6.5.2 Solving Unsatisfiable Classes in Integrated Ontology

After we have repaired unsatisfiable classes in the merged pairs of ontologies $O_{i,j}$ we had to check satisfiability of the integrated ontology. There were still 46 unsatisfiable classes. First we needed to remove the wrong assigned mapping *Cell equivalentTo*

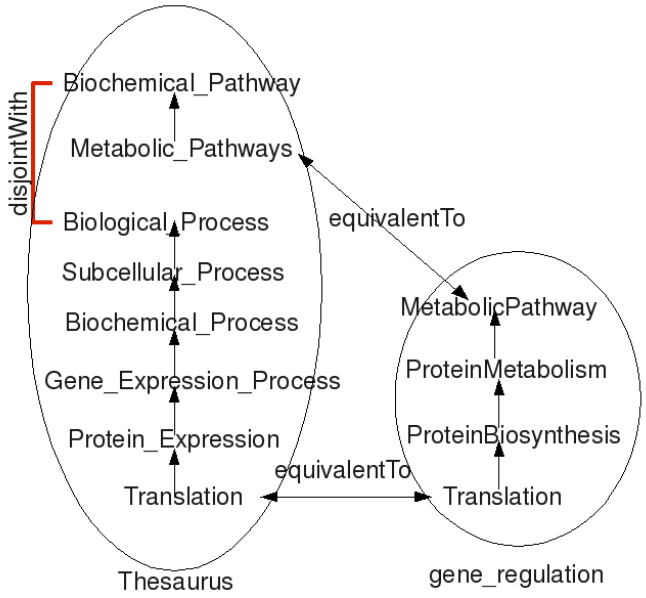

Figure 6.9: Explanation for unsatisfiability of the class *Translation* from the integrated pair $O_{Thes \cup greg}$.

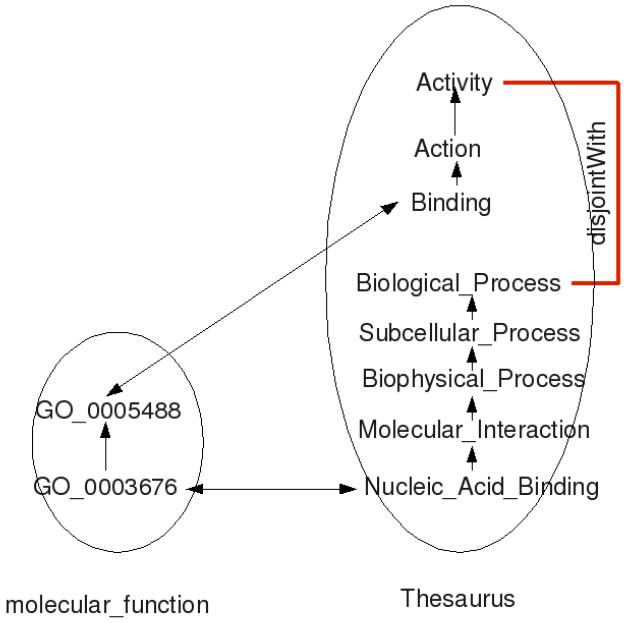

Figure 6.10: Explanation for unsatisfiability of the class *Nucleic_Acid_Binding* for the integrated pair $O_{Thes \cup molRole}$.

Cell_Space. After this, the class *Chromatin* was still unsatisfiable. The explanation is given in Figure 6.11. The concept *Chromatin* here is not consistent as a result of the fact that the concept *Nucleus* is not consistent. The reason for unsatisfiability of *Nucleus* is that after the integration it is subsumed by *greg:CellComponent* and *greg:Cell* at the same time, but these two classes are disjoint in GENE_REGULATION ontology. This strange behavior is caused by modeling errors in MeSH ontology, where the concept *organelle* is a *cellular_structure* which is subsumed by the concept *cell*. This means that in MeSH the $part\_of$ relationship is mixed with the is_a relationship. At this point, we do not have the intention to repair the wrong subsumption in MeSH ontology, the easiest way in this case is just remove the disjointness between *Cell* and *CellComponent* in GENE_REGULATION ontology. After removing other disjointness axioms of a similar kind, the final integrated ontology contains only satisfiable classes.

6.6 Conclusions

For our case study we have shown that the procedure of integration works and a new ontology can be developed on the basis of methods described in this chapter. Fixpoint and unsatisfiability have been solved in a more formal way, yet the whole procedure needs to be scaled in order to be generally applicable. Developing such procedure is the next step in the application of the integration process along the lines sketched in this chapter.

We have described a method to generate a new ontology on the basis of the ontologies available from the Internet. We have shown how to create modules on the basis of the terms of interest. The signature for the module extraction is enriched by the symbols from other modules with the fixpoint as a stop criterion. We have integrated modules on the basis of mappings created using Levenshtein distance similarity.

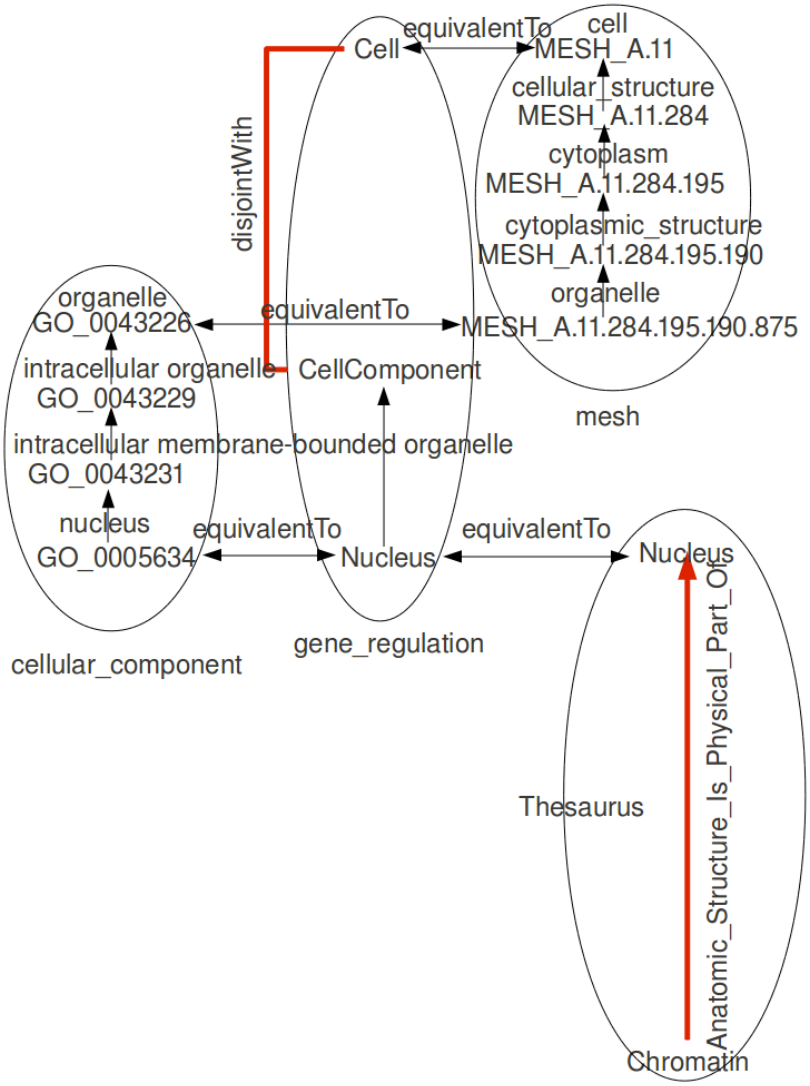

Figure 6.11: Explanation for unsatisfiability of the class *Chromatin* in the integrated ontology

We have investigated how to solve unsatisfiable classes that appear as a result of the integration of the modules. Although the number of unsatisfiable classes was large, it was possible to solve unsatisfiabilities with the help of explanations provided by the Pellet reasoner. We have investigated the unsatisfiability patterns and concluded that the most frequent reason for the unsatisfiability of the integrated ontologies is the disjointness of the concepts in NCI_Thesaurus and gene_regulation ontologies. This may indicate that these ontologies are modeled too strictly. Moreover, we can conclude that in the MeSH ontology the *part_of* and *is_a* relationships are mixed.

With this case study we have shown that the modularity and simple mappings provide a good foundation for the creation of a new ontology in an pseudo-automated way. This method can be used when an ontology engineer does not want to create a new ontology from scratch, but wants to reuse knowledge already presented in other ontologies. Moreover, this is a strategy that should be preferred and has to be adapted more often as ontologies gain importance in life sciences.

Chapter 7

Conclusions, Discussion and Future Directions

7.1 Ontological Context Visualization

We have proposed a methodology to visualize an ontological context and provided a proof of concept with the "ContextVis" approach. This methodology makes it possible to integrate small *extracts* from different ontologies and represent these in "one" visualization. This method is specifically useful for an ontology developer interested in knowledge about some *interesting concept* that is available in different ontologies. Moreover, we think that this approach can simplify a data annotation process for the life scientist. In tools used for data annotation, such as NCBO BioPortal [91], after the querying for the term of interest, a list of concepts from different ontologies is provided from which the user can choose the most corresponding concept. This requires exploration of each concept separately. With our approach, however, the user will immediately comprehend how a corresponding concept is represented in different ontologies in "one" visualization. The concepts are presented in their ontological context from which the most relevant concept can be found quicker and easier.

This method has, however, drawbacks. First, the *extracts* are generated on the basis of a graph representing a subset of the ontology. This graph representation is not complete and therefore is not correct, partly as this graph is emerged from asserted hierarchy and partly because of the problems with properties as discussed in section 7.2. The use of an inferred hierarchy in combination with properties retrieval approach, that we have developed in follow up studies (cf. 3.4.1), will make the graph representation more reliable and is presented as an important improvement of this approach.

In our initial prototype application of the context visualization the implementation of the database was not ideal. The apparent drawback of the database representation is that for each concept the local and global context has to be created and inserted to the database. Consequently, the database will contain a huge amount of duplicated concepts and relations, and therefore the database will be unnecessary

large. This problem can be solved if all ontologies after a reasoning process are inserted in the database. An improved version of the database requires only two tables, e.g. CONCEPTS and RELATIONS just with the same schema, as described in section 2.4.4. While the database is represented in this way, a *global context* can be generated by the searching the database on the fields *label* and *description* for the *interesting term*. Subsequently, for each concept that matches this term, the subgraph can be generated directly from the database and not from the ontology. This will make the process of the generation of the local context more efficient, because the subgraph will be generated directly from the database. Moreover, this will give an obvious advantage in computation time required to create the local context, because, in this case, the application does not need to load the ontology, extract the subgraph and load it to the database.

7.2 Representation of Properties in Ontology Visualization

When an ontology is transformed to a graph we were confronted with problems concerning the representation of properties.

From the experiences gained in the research presented in this thesis, we argue that the only clear and correct ontology visualization is the visualization of the inferred hierarchy. The representation of an ontology hierarchy enriched with properties could be easily misleading. There is confusion between the modeling in a *semantic network*–like style and Ontological modeling. The users of ontologies frequently erroneously think that classes in an ontology have properties. This is, however, not the case, only individuals have properties, and most ontologies do not contain individuals at all. The classes could be defined with restrictions on properties. There are also two types of restrictions, i.e. existential $\exists R$ and universal $\forall R$. It is not clear which kind of restriction we need to consider to be a *property*

of the class. From the reasoning perspective the universal restriction must not be interpreted as obligatory, because the meaning of the statement $C \sqsubseteq \forall R.D$ is the following: if an individual a from class C **has a relation** R then this relation must be only with individuals from class D. This statement, however, does not express that class C **must have** the relation R with class D. In the case of existential restriction the statement $C \sqsubseteq \exists R.D$ expresses that an individual from class C must have at least one relation R with an individual from class D. This statement, however, does not say anything about another properties and *fillers* on these properties. In this case, the class C could have other properties, which are not *attached* to this class in the ontology. A logical suggestion on how to deal with properties – when an ontology visualization approach with property representation is considered – might be the following: check for each property whether for the given class this property is obligatory or not. This bears, however, the danger of the exponential explosion of cloned nodes or edges in the graph/tree. This is a result of the fact that classes in the lower levels of the hierarchy inherit all the properties of their ancestors.

Another possibility is to represent only the properties which are *asserted* in the class axiom, meaning that the inherited properties will not be shown. Unfortunately, this minimalistic approach is rather confusing for the domain specialist. The classes which are presented in lower levels of the hierarchy inherit all the characteristics of the parents, however the user will see only the specific properties which are defined for this class. This will lead to an incomplete logical representation of the ontology.

7.3 Role of Different Geometries in Node-Link Ontology Visualization

In Chapter 4 we have introduced our node-link visualization methodology in which an ontology was represented in different geometric models. We have used *multi-view* paradigm in which different geometrical views could be chosen and, in that manner,

an ontology can be explored in many different ways. Ontologies could be either large or small, classes in one ontology could be populated with hundreds subclasses and have multiple parents. Classes in another ontology could have a couple of children while multiple inheritance is not allowed at all, to summarize, ontologies are different. Therefore, the possibility to choose a view that corresponds most with the ontology and with user's aesthetic preferences can be supportive during exploration of the ontology.

In Chapter 4, five different geometric views have been proposed: two Euclidean, two hyperbolic and one spherical. On the basis of our analysis we can conclude that the Euclidean views can be used if an ontology is visualized up to a level of depth no more then 3. Whereas the hyperbolic views are more suitable for visualization of large tree structures. Hyperbolic views allocate most of screen space for the concept in the center as well as to its context. Different interaction steps are required to explore structures in deeper layers. We argue that hyperbolic visualization will solve the problem of visualization of complete hierarchies, the possibility to choose the level of depth, however, is still needed.

7.4 Exploration of Ontology Hierarchy with Semantic Zoom

In Chapter 5 we have proposed a containment approach in which an ontology is visualized on the basis of the sphere packing algorithm. The children represented as spherical caps are placed on top of the spherical caps of their parents. This layered approach is very amenable for the realization of a semantic zoom. Appearance or disappearance of layers depends on the distance of the sphere to the viewer. The hierarchy can be easily explored by means of a pointing device such as a mouse. Hence, with this method the user can easily control the level of detail.

7.5 Building of a new Ontology

At the moment the ontology developer is interested in the development of a new ontology about a particular topic, he/she can start from scratch, or make use of already available resources. With the method proposed in this thesis it has been shown that it is possible to create an ontology in a semi-automated way on the basis of the modularity approach [62]. A module created with the modularity approach is logically correct, nevertheless, it will be interesting to investigate whether an ontology developer will be more satisfied with a new ontology created with the modularity method or will prefer a new ontology created with the structural method [102].

The problem with this methodology is that after the integration of modules in one ontology, a lot of classes become unsatisfiable. The unsatisfiabilities have to be resolved with the help of the reasoner. We argue that it is not possible to complete this process in a completely automated way. Nevertheless, for the modeler it can be very important to know the consequents of the ontology integration. Decisions concerning whether an axiom has to be changed in the original ontology, or whether two classes are indeed equivalent or not have to be taken by a domain specialist and not by an automatic agent.

7.6 Ontology Integration

As stated, we belief that the integration of ontologies can not be achieved in an automated way. Most ontology alignment methods need some form of user intervention, specifically at the point when the user has to agree with the suggestions provided by a method. From our point of view it is also not possible to provide an alignment method completely based on the semantic similarity. Such a method must rely on the structure of the ontology, however, this structure depends on the ontology development process and is a subject of domain granularity and ontology developer preferences. If this structure is represented as a graph, the ontology alignment

migrates to the research area of graph matching [74]. It is worth of investigating whether ontology alignment will gain from the graph matching algorithms.

7.7 Conclusions

The general conclusion of this thesis is that it is not possible, therefore, should not be striven at, to make a visualization method that is able to support whole spectrum of ontology users, from model developers to users involved in data annotation process. The ontology visualization has to be restricted to a specific part of the community. There are two main views on ontology: one is logic-based and other is graph-based. We have chosen to support the graph-based view for the development of our visualization approach, because a graph representation is most feasible for visualization. The proposed approach is complementary to ontology development tools, such as Protégé, that support the logic representation of an ontology. Our visualization approach provides an abstract view on an ontology where the user is not confronted with building blocks of a Description Logic. The node-link representation reflects a most intuitive view on an ontology: set of concepts and relations between them. The containment method represent the hierarchy of an ontology.

The visualization tools can also be used for ontology integration. To support this process we have investigated how an ontological context can be generated and visualized. Our "ContexVis" approach could be used in order to analyze how an concept is represented in different ontologies and could help in an ontology integration process and in creation of a new ontology on the basis of this concept.

The idea about ontological context was further elaborated in the approach that we have provided in order to generate a new ontology on the basis of modules. We have shown that ontology can be generated in a semi-automated fashion from the modules dedicated to some concept of interest.

7.8 Future Directions

Ontology visualization is a new research topic. A few usability experiments have been performed in order to investigate how user is interacting with an ontology [113]. It will be very interesting to investigate how the user experiences the paradigm of different geometric views representation; do different views support the user in ontology exploration process? The other interesting question will be, whether the representation of whole ontology in hyperbolic plane/space is preferred over the representation in which a different level of depth can be chosen.

The approach provided to containment method at this moment is applicable only for small ontologies. More research is required to investigate how it can be applied to large ontologies. A possible suggestion for implementations could be: represent hierarchy up to a certain level, when the user would like to zoom in one concept, expand this concept and make a new subtree on top of the parental spherical cap.

In our analysis user evaluation has not played a central role; we have used proven heuristics to evaluate existing methods and assess our own developments. The focus has been on the development of the paradigm, now that the artefact is available a more thorough study of the user interaction and the user interface can be conducted. In such study user centered usability evaluations are pivotal.

In the method (cf. Chapter 6) proposed for ontology creation, we have used modules that are complete with respect to reaching a fixpoint in their signatures. The extracted modules, however, contain too much general terms which are not important for a domain specialist. It will be interesting to investigate how to reach a balance between completeness of modules and the specificity of the domain.

Bibliography

[1] Amino-acid ontology. http://www.co-ode.org/ontologies/amino-acid/2009/02/16/.

[2] Eclipse. http://www.eclipse.org.

[3] Flavors of Geometry. http://www.msri.org/communications/books/Book31/contents.html.

[4] H3Viewer. http://www.graphics.stanford.edu/~munzner/h3/.

[5] Hermit OWL Reasoner. http://hermit-reasoner.com/.

[6] Java 3D API. http://java.sun.com/products/java-media/3D/.

[7] Jena A Semantic Web Framework for Java. http://jena.sourceforge.net/.

[8] Medical Subject Headings. http://www.nlm.nih.gov/mesh/introduction.html.

[9] Möbius Transformation. http://en.wikipedia.org/wiki/M%C3%B6bius_transformation.

[10] MySql The world's most popular open source database. http://www.mysql.com/.

[11] OBO Download Matrix. http://www.berkeleybop.org/ontologies/.

[12] The OBO Foundry Ontologies. http://berkeleybop.org/cgi-bin/obofoundry/table.cgi.

[13] OntoEdit an Ontology Engineering Environment. http://www.ontoknowledge.org/tools/ontoedit.shtml.

[14] The Ontology for Biomedical Investigations. http://obi-ontology.org/page/Main_Page.

[15] ONTOSPHERE. http://ontosphere3d.sourceforge.net/.

[16] The Open Biomedical Ontologies. http://www.obofoundry.org/.

[17] The Open Source OWL Reasoner. http://pellet.owldl.com/.

[18] The OWL-API. http://owlapi.sourceforge.net/index.html.

[19] OWL Web Ontology Language Overview. http://www.w3.org/TR/owl-features/.

[20] Pizza Ontology. http://www.co-ode.org/ontologies/pizza/2007/02/12/pizza.owl.

[21] Protégé, ontology editor and knowledge-base framework. http://protege.stanford.edu/.

[22] Protein Information Resource. http://pir.georgetown.edu/pro/pro.shtml.

[23] RacerPro an OWL reasoner. http://www.racer-systems.com/.

[24] RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[25] Semantic Web Ontology Editor. http://code.google.com/p/swoop/.

[26] SequoiaView, Visualize Hard Drive Space Usage. http://www.snapfiles.com/get/sequoia.html.

[27] SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[28] Terminology Resources: NCI Enterprise Vocabulary Services (EVS), Dictionaries, FedMed, FDA, CDISC, and NCPDP terminology.

[29] A TouchGraph Visualization Tab for Protégé 2000. http://users.ecs.soton.ac.uk/ha/TGVizTab/.

[30] ToughGraph Navigator. http://www.touchgraph.com/navigator.html.

[31] Unified Modeling Language. http://www.uml.org/.

[32] Walrus - Graph Visualization Tool. http://www.caida.org/tools/visualization/walrus/.

[33] 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, New York, NY, USA, 2003.

[34] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[35] Judith A. Blake and Midori A. Harris. *The Gene Ontology (GO) Project: Structured Vocabularies for Molecular Biology and Their Application to Genome and Expression Analysis*. John Wiley & Sons, Inc., 2002.

[36] Alex Borgida and Luciano Serafini. Distributed Description Logics: Assimilating Information from Peer Sources. *Journal of Data Semantics*, 1:2003, 2003.

[37] Franz-Josef Brandenburg. Nice Drawings of Graphs are Computationally Hard. In *Selected Contributions from the on 7th Interdisciplinary Workshop on Informatics and Psychology*, pages 1–15, London, UK, 1990. Springer-Verlag.

[38] Susanne Busse. Interoperable E-Learning Ontologies Using Model Correspondences. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, volume 3762 of *Lecture Notes in Computer Science*, pages 1179–1189. Springer Berlin / Heidelberg, 2005.

[39] Diego Calvanese. Data Integration: A Logic-Based Perspective. *AI Magazine*, 26:59–70, 2005.

[40] The Gene Ontology Consortium. The Gene Ontology project in 2008. *Nucl. Acids Res.*, 36(suppl_1):D440–444, January 2008.

[41] O. Corcho, A. Gomez-Perez, A. Leger, C. Rey, and F. Toumani. An Ontology-Based Mediation Architecture for E-Commerce Applications. In *Proceedings of Intelligent Information Systems*, pages 477–486. Springer, 2003.

[42] Mélanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. MIREOT: the Minimum Information to Reference an External Ontology Term. *Applied Ontology*, 6(1):23–33, January 2011.

[43] Bernardo Cuenca, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Web Semantics*, 6(4):309–322, 2008.

[44] Jos de Bruijn, Marc Ehrig, Cristina Feier, Francisco Martíns-Recuerda, François Scharffe, and Moritz Weiten. *Ontology Mediation, Merging, and Aligning*. July 2006.

[45] Sebastian Derriere, André Richard, and Andrea Preite-Martinez. An Ontology of Astronomical Object Types for the Virtual Observatory. *Proceedings of the International Astronomical Union*, 2(Highlights of Astronomy 14):603–603, 2006.

[46] Julia Dmitrieva. Multi-View Ontology Visualization. `http://www.liacs.nl/~jdmitrie/ontVis/OntologyVisualization.html`.

[47] Julia Dmitrieva, Yun Bei, and Fons J. Verbeek. Ontological Context Visualization. In *OWLED*, 2007.

[48] Julia Dmitrieva and Fons J. Verbeek. Multi-View Ontology Visualization. In *Proceedings of the 11th International Protégé Conference, June 23-26, 2009 - Amsterdam.*

[49] Julia Dmitrieva and Fons J. Verbeek. Node-Link and Containment Methods in Ontology Visualization. In *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, volume 529, 2009.

[50] Julia Dmitrieva and Fons J. Verbeek. Creating a new Ontology: a Modular Approach. *Proceedings of the 3rd International Workshop on Semantic Web Applications and Tools for the Life Sciences Berlin, Germany, December,* 2010 *(SWAT4LS)*, pages 1–4, 2010.

[51] Julia Dmitrieva and Fons J. Verbeek. Different Geometries in Ontology Visualization. In *Proceedings SPIE EI 2010, Visualization and Data Analysis*, volume 7530, 2010.

[52] P. Eades. A Heuristic for Graph Drawing. *Congressus Numerantium*, 42:149–160, 1984.

[53] Peter Eklund, Nataliya Roberts, and Steve Green. OntoRama: Browsing RDF Ontologies using a Hyperbolic-style Browser. *Proceedings of the First International Symposium on Cyber Worlds*, 2002.

[54] M. C. Escher. Circle Limit IV (Heaven and Hell). Woodcut in black and ocre. `http://www.mcescher.com/Gallery/recogn-bmp/LW436.jpg`, 1960.

[55] Thomas M. J. Fruchterman and Edward M. Reingold. Graph Drawing by Force-Directed Placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991.

[56] Amir Ghazvinian, Natalya F. Noy, and Mark A. Musen. Creating Mappings for Ontologies in Biomedicine: Simple Methods Work. In *AMIA 2009 Symposium Proceedings*, 2009.

[57] S. Ghilardi, C. Lutz, and F. Wolter. Did I Damage my Ontology? A Case for Conservative Extensions in Description Logics. In Patrick Doherty, John Mylopoulos, and Christopher Welty, editors, *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 187–197. AAAI Press, 2006.

[58] Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did I Damage my Ontology? a Case for Conservative Extensions in Description Logics. In *Proc. of KR2006*, pages 187–197. AAAI Press, 2006.

[59] Chaim Goodman-Strauss. Compass and Straightedge in the Poincaré Disk. *The Mathematical Assosciation of America*, 108(1):38–49, 2001.

[60] Bernardo C. Grau, Bijan Parsia, and Evren Sirin. Working with Multiple Ontologies on the Semantic Web. pages 620–634. 2004.

[61] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. A Logical Framework for Modularity of Ontologies. In *Proc. IJCAI-2007*, pages 298–304. AAAI, 2007.

[62] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Extracting Modules from Ontologies: A Logic-Based Approach. In *Modular Ontologies*, pages 159–186. 2009.

[63] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Tableau Algorithms for $\mathcal{E}$-connections of Description Logics. Technical report, 2004.

[64] Henson Graves. Ontology Engineering for Product Development. In *OWLED*, 2007.

[65] Marvin Jay Greenberg. *Euclidean and Non-Euclidean Geometries: Development and History.*

[66] RDF Working Group. Resource Description Framework (RDF). `http://www.w3.org/RDF/`.

[67] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowl. Acquis.*, 5(2):199–220, 1993.

[68] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[69] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, and Hai H Wang. The Manchester OWL Syntax. In *Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED 2006)*, 2006.

[70] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The Even More Irresistible SROIQ. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*, pages 57–67. AAAI Press, June 2006.

[71] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language. Combining OWL and RuleML. `http://www.w3.org/Submission/SWRL/`.

[72] Ian Horrocks, Peter F. Patel-Schneider, and Frank Van Harmelen. From SHIQ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1:2003, 2003.

[73] Ian Horrocks and Ulrike Sattler. A Tableau Decision Procedure for SHOIQ. *J. Autom. Reason.*, 39(3):249–276, 2007.

[74] Wei Hu, Ningsheng Jian, Yuzhong Qu, and Yanbing Wang. GMO: A Graph Matching for Ontologies. In *Integrating Ontologies*, 2005.

[75] Ernesto Jiménez-Ruiz, Bernardo Grau, Ulrike Sattler, Thomas Schneider, and Rafael Berlanga. Safe and Economic Re-Use of Ontologies: A Logic-Based Methodology and Tool Support. In *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 185–199. 2008.

[76] Yannis Kalfoglou and W. Marco Schorlemmer. Ontology Mapping: The State of the Art. In *Semantic Interoperability and Integration*, 2005.

[77] Amalia Kallergi, Yun Bei, and Fons J. Verbeek. The Cyttron Scientific Image Database for Exchange, 2009.

[78] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and James A. Hendler. Swoop: A Web Ontology Editing Browser. *J. Web Sem.*, 4(2):144–153, 2006.

[79] Akrivi Katifori and Constantin Halatsis. Ontology Visualization Methods - A Survey. *AMC Computing Surveys*, 39(4), 2007.

[80] Ernst Kleiberg, Huub van de Wetering, and Jarke J. Van Wijk. Botanical Visualization of Huge Hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, pages 87–94, Washington, DC, USA, 2001. IEEE Computer Society.

[81] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Semantic Modularity and Module Extraction in Description Logics. In *Proceedings of ECAI'08*, pages 55–59. IOS Press, 2008.

[82] Anand Kumar, Barry Smith, Domenico M. Pisanelli, Aldo Gangemi, and Mario Stefanelli. An Ontological Framework for the Implementation of Clinical Guidelines in Health Care Organizations. In *Studies In Health Technology And Informatics, 2004*, 2004.

[83] Oliver Kutz, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. $\mathcal{E}$-connections of Abstract Description Systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[84] Patrick Lambrix and He Tan. SAMBO - a System for Aligning and Merging Biomedical Ontologies. *Journal of Web Semantics*, 4:206, 2006.

[85] John Lamping and Ramana Rao. The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies. *Journal of Visual Languages and Computing*, 7:33–35, 1996.

[86] Vladimir Levenshtein. Binary Codes capable of Correcting, Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):845–848, August 1965.

[87] Robin McEntire. Ontologies in the Life Sciences. *Knowl. Eng. Rev.*, 17(1):77–80, 2002.

[88] Rolf Molich and Jakob Nielsen. Improving a Human-Computer Dialogue. *Commun. ACM*, 33(3):338–348, March 1990.

[89] Tamara Munzner. H3: Laying out Large Directed Graphs in 3D Hyperbolic Space. *IEEE Computer Graphics and Applications*, 18:18–23, 1998.

[90] Natalia F. Noy and Mark A. Musen. Specifying Ontology Views by Traversal. *LNCS*, 3298:713–725, 2004.

[91] Natalya F. Noy, Nigam H. Shah, Patricia L. Whetzel, Benjamin Dai, Michael Dorf, Nicholas Griffith, Clement Jonquet, Daniel L. Rubin, Margaret-Anne Storey, Christopher G. Chute, and Mark A. Musen. BioPortal: Ontologies and

Integrated Data Resources at the Click of a Mouse. *Nucleic Acids Research*, 37:W170–W173, 2009.

[92] Natalya Fridman Noy and Mark A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455. AAAI Press, 2000.

[93] Carl A. Petri. *Kommunikation mit Automaten*. PhD thesis, Universitiy of Bonn, 1962.

[94] Mark Phillips and Charlie Gunn. Visualizing Hyperbolic Space: Unusual Uses of 4×4 Matrices. 1991.

[95] Helena Sofia Pinto and João P. Martins. Ontology Integration: How to Perform the Process.

[96] Stanley B. Prusiner. Prions. *Proceedings of the National Academy of Sciences of the United States of America*, 95(23):13363–13383, 1998.

[97] Alan Rector and Jeremy Rogers. Ontological Issues in Using a Description Logic to Represent Medical Concepts: Experience from GALEN. In *IMIA Working Group 6 Workshop*, 2002.

[98] Femke Reitsma and Jochen Albrecht. Modeling with the Semantic Web in the Geosciences. *IEEE Intelligent Systems*, 20:86–88, 2005.

[99] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 189–194, New York, NY, USA, 1991. ACM.

[100] Cornelius Rosse and José L. V. Mejino, Jr. A Reference Ontology for Biomedical Informatics: the Foundational Model of Anatomy. *J. of Biomedical Informatics*, 36(6):478–500, 2003.

[101] Nadine Schuurman and Agnieszka Leszczynski. Ontologies for Bioinformatics. *Bioinformatics and Biology Insights*, 2:187–200, March 2008.

[102] Julian Seidenberg and Alan Rector. Web Ontology Segmentation: Analysis, Classification and Use. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 13–22, New York, NY, USA, 2006. ACM.

[103] L. Serafini and A. Tamilin. DRAGO: Distributed Reasoning Architecture for the Semantic Web. *Proc. of the Second European Semantic Web Conference (ESWC'05)*, pages 361–376, 2005.

[104] Ben Shneiderman. Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[105] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL Reasoner. *J. Web Sem.*, 5(2):51–53, 2007.

[106] John F. Sowa. Semantic Networks. `http://www.jfsowa.com/pubs/semnet.htm`.

[107] John F. Sowa. *Handbook of Knowledge Representation (Foundations of Artificial Intelligence)*, chapter Conceptual Graphs. Elsevier Science, 2007.

[108] Robert Stevens, Mikel Egaña Aranguren, Katy Wolstencroft, Ulrike Sattler, Nick Drummond, Matthew Horridge, and Alan Rector. Using OWL to Model Biological Knowledge. *Int. J. Hum.-Comput. Stud.*, 65(7):583–594, 2007.

[109] Margaret-Anne Storey, Mark Musen John Silva, Neil Ernst, Ray Fergerson, and Natasha Noy. Jambalaya: Interactive Visualization to Enhance Ontology

Authoring and Knowledge Acquisition in Protégé. In *Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001)*, 2001.

[110] Stephan Tobies. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *Journal of Artificial Intelligence Research*, 12:2000, 2000.

[111] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

[112] Jarke J. Van Wijk and Huub van de Wetering. Cushion Treemaps: Visualization of Hierarchical Information. In *INFOVIS '99: Proceedings of the 1999 IEEE Symposium on Information Visualization*, page 73, Washington, DC, USA, 1999. IEEE Computer Society.

[113] Taowei David Wang and Bijan Parsia. CropCircles: Topology Sensitive Visualization of OWL Class Hierarchies. *LNCS*, 4273:695–708, 2006.

[114] Mark Allen Weiss. *Data Structures & Algorithm Analysis in C++*. 1999.

[115] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. The Cognitive Walkthrough Method: a Practitioner's Guide. pages 105–140, 1994.

[116] Adam Wyner. An Ontology in OWL for Legal Case-based Reasoning. *Artif. Intell. Law*, 16(4):361–387, 2008.

[117] Adam Wyner and Rinke Hoekstra. A Legal Case OWL Ontology with an Instantiation of Popov v. Hayashi. *The Knowledge Engineering Review,Special Issue on Case Based Reasoning*, 2010.

Samenvatting

Een ontologie is een begrippenkader waarmee de concepten en de relaties tussen bepaalde concepten in een domein in kaart kunnen worden gebracht. De afgelopen decennium is er een toenemende belangstelling voor het gebruiken van ontologieën in verschillende gebieden van de wetenschap. Deze belangstelling is onder meer gelegen in het feit dat een ontologie veel mogelijkheden biedt voor het modelleren van een bepaald onderzoeksdomein als ook de integratie van data binnen dat onderzoeksdomein. Voor het ondersteunen van het maken van een ontologie zijn, voor elk van de typerende stadia in het proces, verschillende technologieën ontwikkeld. Zo zijn er speciale editors voor ontologieën, toepassingen om met een ontologie te redeneren, zg. reasoners, speciale zoekmachines voor ontologieën en toepassingen om de concepten in de verschillende ontologieën naast elkaar te zetten en te integreren, zg. alignment. Dit proefschrift gaat in het bijzonder in op het werken met de inhoud van bestaande ontologieën door visualisatie en integratie.

Een ontologie moet ondubbelzinnig kunnen worden geïnterpreteerd door mens en computer, dit is de belangrijkste factor voor de technologie die nodig is om een bepaald domein met behulp van een ontologie te kunnen modelleren. Om dit te kunnen realiseren moet er een taal beschikbaar zijn die aan een aantal belangrijke eisen moet voldoen. Zij moet beschikken over voldoende expressieve uitdrukkingsmogelijkheden en tegelijkertijd vanuit het perspectief van de algoritmiek beslisbaar zijn. Dit laatste wil zeggen dat er algoritmen ontwikkeld kunnen worden die in een eindige tijd een probleem kunnen oplossen door berekening. Er zijn verschillende

talen ontwikkeld en voorgesteld die aan deze eisen zouden kunnen voldoen: OIL, DAML+OIL, RDF, RDFS, OWL. De W3C community heeft uiteindelijk OWL (dit staat voor Web Ontologie Language en wordt uitgesproken als o-w-l) geselecteerd als de taal die gebruikt wordt voor het representeren van een ontologie binnen het zogenaamde semantische netwerk (Semantic Web).

De logica onderliggend aan OWL bestaat uit verschillende zogenaamde "Description Logics" (DLs). Description Logics zijn berekenbare fragmenten uit de eerste orde logica. Juist daarom is deze taal aan de ene kant interessant voor theoretici en ontwikkelaars van "redeneer" algoritmen terwijl het aan de andere kant een nuttig gereedschap is voor onderzoekers die in de dagelijkse praktijk bezig zijn met het proces van het ontwikkelen van een model/ontologie.

Een ontologie die geschreven is in OWL is in feite een logische structuur en bestaat uit een verzameling axiomas met daarnaast als bouwstenen verschillende constructies in Description Logics. De axiomas zijn verantwoordelijk voor de relaties tussen de objecten in het domein. De DL constructies worden gebruikt om, op basis van eenvoudige objecten, complexe objecten in het domein te kunnen bouwen.

Zoals aangegeven, in dit proefschrift ligt de nadruk op de visualisatie en integratie van ontologieën. De logische structuur, en de bijbehorende complexiteit, in welke de ontologie is gerepresenteerd, is heel geschikt voor redeneren en dergelijke, echter deze structuur is moeilijk te vertalen naar een structuur die geschikt is voor visualisatie. Voor het snel begrijpen van de inhoud van een ontologie kan visualisatie een heel goed hulpmiddel zijn. Zeker nu ontologieën in toenemende mate worden gebruikt voor het annoteren en uitwisselen van gegevens in databases; met name in de levenswetenschappen en in het bijzonder in de bio-informatica kan een visualisatie haar toepassing vinden. Boom- en graafstructuren kunnen heel geschikt zijn voor visualisatie. Het onderzoek dat in dit proefschrift wordt beschreven heeft zich erop gericht hoe een hiërarchische structuur, het zogenaamde skelet van de ontologie, gecombineerd met de in de ontologie aanwezige relaties, kan worden omgebouwd tot een graaf-structuur. Deze afgeleide structuur kan dan worden gebruikt voor de vi-

sualisatie. Het onderzoek in dit proefschrift behandeld twee visualisatie technieken voor de graaf, te weten, de containment methode en de node-link methode.

De containment methode wordt gebruikt voor het visualiseren van hiërarchische boomstructuren. De hiërarchie in de ontologie wordt gevormd op basis van de "is_a" relatie tussen de concepten in de ontologie. Deze relatie geeft een verfijning aan binnen de ontologie die we aangeven als ouder-kind relaties. In een ontologie kunnen concepten zodanig worden gemodelleerd dat een concept meer dan een ouder kan hebben. Deze structuur is geen boom meer maar typisch een graaf. Nu, uitgaande van die graaf een boom te kunnen verkrijgen zijn alle concepten die meerdere ouders hebben gekloond zodat een nieuw kind gemaakt wordt per elke ouder-kind relatie. Deze boom structuur is nu geschikt voor visualisatie.

In dit proefschrift wordt een nieuwe methode voor containment visualisatie geïntroduceerd. Deze alternatieve methode gebruikt een driedimensionale ruimte voor de visualisatie en hierin worden de concepten als bolsegmenten gerepresenteerd. De ouder-kind relaties worden weergeven door de kind-concepten als bolsegment te plaatsen over het bolsegment van het ouder concept en wel zo dat de kind-concepten altijd een deelverzameling zijn van de ouder concepten. Deze sferische representatie geeft veel inzicht met betrekking tot de inhoud van de ontologie. Om zodanig gerepresenteerde ontologie te kunnen bestuderen moet een zoom methode aanwezig zijn. Met de toepassing van standaard zoom wordt er niet meer detail verkregen met betrekking tot de inhoud van de ontologie. Daarom wordt hier het concept van semantische zoom uitgewerkt. Met dit semantisch inzoomen wordt bewerkstelligd dat het niveau van detail afhankelijk wordt van de afstand tussen de kijker (de gebruiker) en het visuele object. Deze belangrijke aanvulling maakt het mogelijk dat een ontologie op een eenvoudige en intuïtieve manier bestudeerd kan worden.

In het voorgaande is aangegeven hoe uit de logische structuur van de ontologie een graaf kan worden afgeleid die geschikt is voor visualisatie. In de node-link methode zijn de knopen van de graaf verbonden door links en deze links stellen de relaties voor die worden afgebeeld door lijnen. Voor de visualisatie van de graaf

is gekozen voor het zogenaamde spanning-tree skeleton. Dit spanning-tree skeleton geeft in feite de lay-out van de graaf; er zijn dan ook links die niet tot de spanning-tree horen en deze worden afgebeeld als stippellijnen. Voor het visualiseren van de graaf kunnen verschillende geometrische representaties worden gebruikt; het uitgangspunt voor het onderzoek is geweest dat er geen beperking moet zijn tot een bepaalde geometrische representatie en daarom is een implementatie gerealiseerd waarbinnen tussen verschillende geometrische representaties kan worden gewisseld tijdens de visualisatie. Op deze manier kan er optimaal gebruik worden gemaakt van de voordelen van elk van de geometrische representaties. Er zijn experimenten gedaan met verschillende geometrische representaties, te weten Euclidische, hyperbolische en sferische representaties. Deze geometrische representaties zijn in twee- en drie-dimensionale ruimtes gerealiseerd.

Integratie van ontologieën is het vinden van de overeenkomsten tussen de verschillende entiteiten uit verschillende ontologieën en het samenvoegen van deze ontologieën op basis van deze overeenkomsten. Het onderzoek in dit proefschrift is gericht geweest op twee aspecten van ontologie integratie die afzonderlijk onderzocht zijn.

Met "ContextVis" wordt een methode geïntroduceerd waarmee een concept in haar verschillende contexten kan worden afgebeeld. Het is een visualisatie methode die vooral gericht is op het bestuderen van integratie. De "ContextVis" toont de ontologische context zoals die wordt verkregen uit verschillende ontologieën en deze contexten worden gevisualiseerd. Een ontologische context is een vereniging van sub-grafen die elk zijn geëxtraheerd uit de originele ontologieën. Elk van deze subgrafen representeert een bepaald concept dat wordt omringd door een omgeving van ontologische context die wordt gevormd door concepten gerelateerd aan dit concept. Deze methode is heel nuttig voor het bestuderen van de representatie van een concept in verschillende ontologieën; daarmee is het een typisch ondersteunend gereedschap dat gebruikt kan worden in het beginstadium van een ontologie integratie proces.

Bij het proces van integratie worden hele ontologieën of delen daarvan samen-

gevoegd tot een consistente ontologie. Bij hele grote ontologieën kan deze integratie een probleem worden. De integratie kan tot een hele grote ontologie leiden met het gevolg dat het bevragen (het doen van query's) en ook het automatisch redeneren (reasoning) te veel tijd gaan vergen. Daarnaast kan het integratieproces ook conflicten introduceren; dit is het gevolg van het feit dat een entiteit op verschillende manieren gemodelleerd kan worden. Integratie van ontologien is een proces dat wordt uitgevoerd door de ontwikkelaar van een ontologie. Deze is meestal juist alleen geïnteresseerd in een specifiek deel van een ontologie, namelijk dat deel dat is toegesneden op het onderwerp dat wordt bestudeerd. Het onderzoek dat is uitgevoerd in het kader van dit proefschrift gaat over een nieuwe methode waarmee langs semiautomatische weg een nieuwe ontologie gecreëerd kan worden. Hierbij wordt gebruik gemaakt van het feit dat er modules uit de ontologie kunnen worden geëxtraheerd. Deze modules komen uit verschillende ontologieën en worden samengevoegd tot een nieuwe ontologie over een bepaald onderwerp. Het samenvoegen wordt zodanig uitgevoerd dat de nieuwe ontologie ook consistent is in termen van de concepten die in deze ontologie zijn verankerd.

Curriculum Vitae

Julia Dmitrieva is geboren in Moskou op 4 Februari 1971. In 1988 heeft ze haar VWO diploma behaald. Van 1988 tot 1991 studeerde ze scheikunde aan de Rijksakademie voor Fijne Chemische Technologie te Moskou. In 1993 verhuisde ze naar Nederland. In 1996 begon ze haar Informatica studie aan de Universiteit Leiden. In 2003 heeft ze haar doctoraal diploma behaald in de afstudeerrichting Theoretishe Informatica. Van 2004 tot 2005 werkte ze as software ontwikkelaar in het bedrijfsleven. Vanaf september 2005 tot februari 2006 werkte ze als wettenschappelijk programmeur in de sectie Imaging & BioInformatics aan de Universiteit Leiden. In maart 2006 begon ze haar promotieonderzoek aan het Leiden Institute of Advanced Computer Science in dezelfde sectie onder begeleiding van Fons J. Verbeek. Sedert Februari 2011 is ze werkzaam in het Centre de Biophysique Moléculaire Numérique (CBMN) van de Universiteit Liége-Gembloux, België.

Acknowledgements

This thesis was not possible without help of a lot of people. Here I would like to thank you.

First of all, my husband and my son, who were supporting me during very difficult time, with a lot of downs and ups. Thank you, that you was always persuading me to proceed, when I was once again willing to give it up.

Thank you Walter and Jeannette for bringing me in the world of Computer Science.

Amalia, for you a very special *thank you*, for your listening, understanding and of course for your art-radiating personality. Your presence filled my life with literature, films, Greek language, and Revenge of Frogger.

Thank you Joris for helping me when my computer was behaving strange. Thank you my colleagues and group mates, Jeroen L., Floris, Laura, Yanju, Yun for providing such a warm environment that made it possible to survive Dutch weather, especially these long, rainy, dark and cold winters. It was always a lot of fun to talk with you.

Jeannette and Jetty, thank you for you wise advises, motivations and moral support. Thank you Andre for you scientific talks, motivations, and for you: "break a leg". Thank you Michael for all that fun at the beginning of my PhD and for joining us in our crazy hike in Schwarzwald.

Jouri, Lena and Wadim your friendship and your warm house, where we were

always welcome, were very important parts of my life in Leiden.

And also very important, I would like to thank my group mates from Dutch climbing Society *NKBV regio Rijnland*. You give me the possibility to discover that it was also possible to an alpinist to survive in a country such as the Netherlands where the highest point is 322.7 m [1].

Thank you strange citizens of our LIACS garden: red cat, red bird (according to Michael), great tit and blue tit, strange night birds, may be owls, ferret, who suddenly appeared once in November 2009, for keeping me a company. And you biologists who were always doing something interesting and inexplicable in our garden, you were always a source of inspiration and envy.

[1] The Vaalserberg, http://en.wikipedia.org/wiki/Vaalserberg.

List of Publications

1. Méndez-Vilas, A. (Ed.), Proceedings MICTE 2006, Mubarak Q., Dmitrieva J., Vermeulen T., Zhou Y., Groenewegen L., Basmagi, S., van Beek R., Verbeek F.J.: BioTune, a component based repository of BioInformatics teaching modules.

2. Proceedings of SPIE Vol. 6506, MultiMedia Content Access: Algorithms & Systems, (2007) (Eds Hanjalic, A., Schettini, R., Sebe, N.), Bei, Y., Dmitrieva, J., Belmamoune, M., Verbeek, F.J.: Ontology Driven Image Search Engine.

3. Proceedings of OWLED 2007 Third International Workshop Innsbruck, Austria 67 June 2007, Julia Dmitrieva, Yun Bei and Fons J. Verbeek: Ontological Context Visualization, `http://www.webont.org/owled/2007/Proceedings.html`

4. Poster Proceedings of FOIS 2008 5th international conference on formal ontology in information systems Saarbrcken, Germany, Oct 31st, Nov 3rd, 2008, Julia Dmitrieva, Fons J. Verbeek: Information Visualization from Ontology, `http://fois08.dfki.de/joomla/index.php/papers`

5. 11th International Protégé Conference June 23 – 26, 2009, Amsterdam, Julia Dmitrieva, Fons J. Verbeek: Multiview Ontology Visualization, `http://protege.stanford.edu/conference/2009/index.html`

6. Proceedings of the 5th International Workshop on OWL: Experiences and

Directions (OWLED 2009), Chantilly, VA, United States, October 23-24, 2009 Julia Dmitrieva, Fons J. Verbeek: NodeLink and Containment Methods in Ontology Visualization, http://www.webont.org/owled/2009/index.html

7. Proceedings of Workshop on Semantic Web Applications and Tools for Life Sciences (2009): Vol. 559, Amsterdam, Dmitrieva, J. and Verbeek, F.J.: Integration of Information from different ontologies.

8. Visualization and Data Analysis 2010 conference, Proceedings of SPIE, Volume 7530, January 2010 San Jose, California, USA, Julia Dmitrieva, Fons J. Verbeek: Different Geometries in Ontology Visualization, http://spie.org/electronicimaging.xml

9. Proceedings of Workshop on Semantic Web Applications and Tools for Life Sciences (2010), Dmitrieva, J. and Verbeek, F.J.: Creating a new Ontology: a Modular Approach. http://arxiv.org/abs/1012.1658

10. Proceedings of 5th International Workshop on Modular Ontologies (WoMO 2011)- ESSLLI 2011, Julia Dmitrieva and Fons Verbeek: Modular Approach for a new Ontology.