

Mining the semantic Web for OWL axioms

Thu Huong Nguyen

▶ **To cite this version:**

Thu Huong Nguyen. Mining the semantic Web for OWL axioms. Web. Université Côte d'Azur, 2021. English. NNT : 2021COAZ4052 . tel-03406784

HAL Id: tel-03406784
https://theses.hal.science/tel-03406784

Submitted on 28 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

FOUILLE DU WEB SÉMANTIQUE À LA RECHERCHE D'AXIOMES OWL

Thu Huong NGUYEN

Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis I3S
Centre Inria Sophia Antipolis – Méditerranée

Présentée en vue de l'obtention du grade de docteur en Informatique de l'Université Côte d'Azur
Dirigée par:
Andrea G.B. TETTAMANZI, Professeur, Université Côte d'Azur, I3S
Soutenue le: 2 Juillet 2021

Devant le jury, composé de :
Fabien GANDON, Directeur de Recherche, INRIA, *Président*
Leonardo VANNESCHI, Professeur, Universidade Nova de Lisboa Information Management School, *Rapporteur*
Karim TABIA, MCF, HDR, Université d'Artois, Centre de Recherche en Informatique de Lens (CRIL), *Rapporteur*
Nathalie HERNANDEZ, MCF, HDR, Université Toulouse 2, Institut de Recherche en Informatique de Toulouse (IRIT), *Examinatrice*
Catherine FARON ZUCKER, MCF, HDR, Université Côte d'Azur, I3S, *Examinatrice*
Andrea G.B. TETTAMANZI, Professeur, Université Côte d'Azur, I3S, D*irecteur de thèse*

CAMPUS FRANCE campusfrance.org

i3S sophia antipolis

Inria inventeurs du monde numérique

WIMMICS

DOCTORAL SCHOOL STIC
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION







MINING THE SEMANTIC WEB FOR OWL AXIOMS

Thu Huong NGUYEN

Research Unit: Laboratoire I3S - UMR7271 – CNRS - INRIA

To obtain the title of PhD of Science
Specialty : COMPUTER SCIENCE of UNIVERSITÉ CÔTE D'AZUR
Thesis supervisor :
Andrea G.B. TETTAMANZI, Professor, Université Côte d'Azur, I3S
defended on: 2nd July 2021

Jury members:
Fabien GANDON, Research Director, INRIA, *President*
Leonardo VANNESCHI, Professor, Universidade Nova de Lisboa Information Management School, *Reviewer*
Karim TABIA, MCF, HDR, Université d'Artois, Centre de Recherche en Informatique de Lens (CRIL), *Reviewer*
Nathalie HERNANDEZ, MCF, HDR, Université Toulouse 2, Institut de Recherche en Informatique de Toulouse (IRIT), *Examinator*
Catherine FARON ZUCKER, MCF, HDR, Université Côte d'Azur, I3S, *Examinator*
Andrea G.B. TETTAMANZI, Professor, Université Côte d'Azur, I3S, *Thesis Director*

Acknowledgements

There is a famous saying “No one who achieves success does so without acknowledging the help of others”. My PhD journey is not an exception. I’m writing these lines to express my sincere gratitude to the persons helping and encouraging me during the last years.

I would first like to express my deepest appreciation to my supervisor, Prof. Andrea G.B. TETTAMANZI, for not only his enthusiastic guidance and insightful feedback to sharpen the right direction in the research ideas but also his sincere encouragement when I felt pressure due to the obstacles. I believe that I am one of the luckiest PhD students when I have a supervisor like him.

I would also like to extend my deepest gratitude to the two members of my Personal Monitoring Committee, Prof. Catherine FARON ZUCKER and Prof. Michael O’NEIL, who have followed the annual progress of my work and given me their expert advice. I hope they would be pleased with my current results.

I would particularly like to thank the jury, especially Prof. Leonardo VANNESCHI and Prof. Karim TABIA for spending their valuable time to review my thesis. Their assessments will be invaluable contributions to push my work to get better in the future. I also want to thank Prof. Fabien GANDON, Prof. Nathalie HERNANDEZ, Prof. Catherine FARON ZUCKER for being part of the examination committee.

Many thanks go to WIMMICIANS for their support throughout the duration of my PhD program. I would like to offer special thanks to the WIMMICS team leader, Fabien GANDON, for his warm welcome and precious assistance. I would also like to thank Olivier CORBY for his helpful explanation of SPARQL. I would like to express my thanks to Alain GIBOIN for his sincere encouragement and also mini-lessons in French in the short lunch time. I am particularly grateful for the assistance given by Christine FOGGIA in the critical paperwork. I would like to thank Franck MICHEL for his instructions in the use of the resources in the laboratory. I wish to thank Raphaël, Molka, Amine, Vorkit and Antonia for sharing the experience in the research and also the paperwork for the PhD students.

I would also like to thank the SPARKS team, I3S laboratory and Inria for giving me an ideal working environment. I would especially like to thank my friends in the SPARKS team: Stéphanie, Roque, Sébastien, Edson, John for sharing with me unforgettable memories of the PhD student’s life.

I would like to thank Campus France, Vietnamese MOET, Erasmus plus program for financial support to follow the PhD program in France and the mobility program in DNIIT. I would like to thank Prof. Nhan LE-Thanh for his support during the time when I was working in DNIIT.

I want to thank my old colleagues, Hue NGUYEN and Tuan PHAN, for encouraging me to pursue a PhD program in France.

I would like to thank my friends in Antibes: Hoang-Ngoc, Phu, Huong NGO, Tuan, Nhut-Duc Anh, Cuong-Giang, Nhan, Khoa-Ngan who shared all moments of joy to rest my mind outside the research time.

Warming words of gratitude go to my best friend in UK Thanh Thao NGUYEN for a special friendship and her intelligent advice when things would get a bit messy.

Finally, I would like to express my wholehearted gratitude ♡ to my family: my parents, my sisters, my brothers and my little angels: Bong, Alpha, Sol, Do who never make me feel lonely, unloved or unsupported.

Sophia-Antipolis, April 2021
Thu Huong NGUYEN

Abstract

In the Semantic Web era, Linked Open Data (LOD) is its most successful implementation, which currently contains billions of RDF (Resource Data Framework) triples derived from multiple, distributed, heterogeneous sources. The role of a general semantic schema, represented as an ontology, is essential to ensure the correctness and consistency in LOD and make it possible to infer implicit knowledge by reasoning. The growth of LOD creates an opportunity for the discovery of ontological knowledge from its raw RDF data itself to enrich relevant knowledge bases. In this work, we aim at discovering schema-level knowledge in the form of axioms encoded in OWL (Ontology Web Language) from RDF data. The approaches to automated generation of the axioms from recorded RDF facts on the Web may be regarded as a case of inductive reasoning and ontology learning. The instances, represented by RDF triples, play the role of specific observations, from which axioms can be extracted by generalization.

Based on the insight that discovering new knowledge is essentially an evolutionary process, whereby hypotheses are generated by some heuristic mechanism and then tested against the available evidence, so that only the best hypotheses survive, we propose a model applying Grammatical Evolution, one type of evolutionary algorithm, to mine OWL axioms from an RDF data repository. In addition, we specialize the model for the specific problem of learning OWL class disjointness axioms, along with the experiments performed on DBpedia, one of the prominent examples of LOD.

Furthermore, we use different axiom scoring functions based on possibility theory, which are well-suited to the open world assumption scenario of LOD, to evaluate the quality of discovered axioms. Specifically, we proposed a set of measures to build objective functions based on single-objective and multi-objective models, respectively.

Finally, in order to validate it, the performance of our approach is evaluated against subjective and objective benchmarks, and is also compared to the main state-of-the-art systems.

Keywords: Semantic Web, OWL, RDF, Description Logics, Ontology, Ontology learning, Ontology Enrichment, Axioms, Class Disjointness Axioms, Data mining, Evolutionary Algorithms, Grammatical Evolution, Genetic Programming, Fuzzy Sets, Possibility Theory.

Résumé

À l'ère du Web Sémantique, les données liées ouvertes (LOD) en sort l'implémentation la plus réussie qui contient actuellement des milliards de triplets RDF (Resource Data Framework) dérivés de sources multiples, distribuées et hétérogènes. Le rôle d'un schéma sémantique général, représentanté comme une ontologie, est essentiel pour assurer l'exactitude et la cohérence du LOD et permettre d'inférer des connaissances implicites par le raisonnement. La croissance du LOD a créé une opportunité pour la découverte de connaissances ontologiques à partir de ses données RDF brutes elles-mêmes pour enrichir les bases de connaissances pertinentes. Dans ce travail, nous visons à découvrir des connaissances au niveau des schémas sous forme d'axiomes encodés en OWL (Ontology Web Language) à partir de données RDF. Les approches de génération automatisée d'axiomes à partir de faits RDF enregistrés sur le Web peuvent être considérées comme un cas de raisonnement inductif et d'apprentissage d'ontologie. Les instances, représentées par des triplets RDF, jouent le rôle d'observations spécifiques, dont les axiomes peuvent être extraits par généralisation.

Partant du principe que la découverte de nouvelles connaissances est essentiellement un processus évolutif, dans lequel les hypothèses sont générées par un mécanisme heuristique puis testées par rapport à l'évidence disponible, de sorte que seules les meilleures hypothèses survivent, nous proposons un modèle appliquant l'évolution grammaticale, un type d'algorithme évolutionnaire, pour extraire les axiomes OWL d'un référentiel de données RDF. En outre, nous spécialisons le modèle pour le problème spécifique d'apprentissage d'axiomes de disjonction de classes OWL, ainsi que pour les expériences effectuées sur DBpedia, l'un des exemples les plus proéminents du LOD.

De plus, nous utilisons différentes fonctions de notation des axiomes basées sur la théorie des possibilités, qui sont bien adaptées au scénario d'hypothèse du monde ouvert du LOD, pour évaluer la qualité des axiomes découverts. Plus précisément, nous avons proposé un ensemble de mesures pour construire des fonctions objectif basées respectivement sur des modèles à objectif unique et multi-objectifs.

Enfin, afin de la valider, la performance de notre approche est évaluée par rapport à des benchmarks, subjectif et objectif, et est également comparée aux principaux systèmes de l'état de l'art.

Mots-clés: Web Sémantique, OWL, RDF, logiques de description, ontologies, apprentissage d'ontologies, enrichissement d'ontologie, axiomes, axiomes de disjonction de classe, exploration des données, algorithmes évolutionaires, évolution grammaticale, programmation génétique, ensembles flous, théorie des possibilités.

Contents

List of Figures **xi**

List of Tables **xii**

List of Abbreviations **xiii**

1 Introduction **1**
1.1 Context 1
1.2 Motivations 5
1.3 Objectives 5
1.4 Contributions 6
1.5 Publications 7
1.6 Thesis Outline 8

2 Foundation **10**
2.1 Ontology based Knowledge Representation 11
2.1.1 Ontologies 11
2.1.2 Description Logics 12
2.2 The Semantic Web 17
2.2.1 Uniform Resource Identifiers (URIs) 17
2.2.2 Resource Description Framework 18
2.2.3 Ontology Modeling Languages: RDFS and OWL 20
2.2.4 The SPARQL Query Language 27

3 Literature Review **31**
3.1 Ontology learning 32
3.1.1 Ontology learning in the LOD context 32
3.1.2 Learning techniques 33
3.2 Schema-level axiom learning 36
3.2.1 Property Axiom Learning 36
3.2.2 Class Axiom Learning 37
3.3 RDF Mining 37
3.4 Summary 39

4 Learning OWL Axioms From RDF data **41**
4.1 Introduction . . . 41
4.2 Grammatical Evolution . . . 44
4.2.1 Grammar-mediated Representation . . . 44
4.2.2 Evolutionary Process . . . 47
4.2.3 Terminology . . . 48
4.2.4 Grammatical Evolution Implementations . . . 49
4.3 Grammatical Evolution to Search for OWL Axioms . . . 50
4.3.1 Grammar Construction . . . 50
4.3.2 An Evolutionary Model to Search for OWL Axioms . . . 53
4.4 Summary . . . 59

5 Axiom Evaluation **61**
5.1 Introduction . . . 61
5.2 OWL Axiom Testing . . . 63
5.2.1 General Principles . . . 63
5.2.2 The Computational Definitions of Evidences . . . 67
5.3 Axiom Scoring Frameworks . . . 76
5.3.1 Probabilistic Evaluation of Axioms . . . 77
5.3.2 Possibilistic Evaluation of Axioms . . . 79
5.4 Summary . . . 85

6 Grammatical Evolution Models toward Class Disjointness Axiom Discovery **86**
6.1 Introduction . . . 87
6.2 Related Works . . . 88
6.3 Learning Atomic and Complex Axioms involving Union and Intersection Operators . . . 90
6.3.1 GE Characteristics . . . 90
6.3.2 Gold Standard toward a Subjective Benchmark . . . 98
6.3.3 Experimental Protocol . . . 101
6.3.4 Results & Discussions . . . 102
6.3.5 Limitations . . . 106
6.4 Learning Complex Axioms containing Value and Existential Restriction 107
6.4.1 GE Characteristics . . . 108
6.4.2 From A Training- Testing Model toward An Objective Benchmark . . . 109
6.4.3 Experimental Protocol . . . 111
6.4.4 Results & Discussions . . . 112
6.5 Summary . . . 116

7 A Multi-Objective GE Approach to Class Disjointness Axioms Discovery 118
7.1 Introduction . . . 118
7.2 Evolutionary Multi-Objective Optimization . . . 119
7.2.1 Multi-Objective Optimization . . . 119
7.2.2 Multi-objective Evolutionary Algorithms . . . 120
7.3 Multi-Objective GE for Axiom Discovery . . . 125
7.3.1 Multi-Objective Evaluation Framework . . . 125
7.3.2 Experimental Protocol . . . 128
7.3.3 Results & Discussions . . . 129
7.4 Summary . . . 132

8 Conclusions & Perspectives 134
8.1 Conclusions . . . 134
8.1.1 A General GE Framework of OWL Axioms Discovery from RDF data . . . 135
8.1.2 Evaluation Frameworks for Discovered OWL Axioms . . . 136
8.1.3 Toward Learning OWL Class Disjointness Axioms . . . 137
8.1.4 Performance Evaluation Frameworks . . . 138
8.2 Future Work . . . 139

APPENDICES 142

A Appendix- Gold Standard 143

B Appendix- Training Dataset 145

List of Figures

1.1 The Linked Open Data Cloud 4

2.1 The Semantic Web Layer Architecture 18
2.2 An RDF graph 19
2.3 An example of the structure of a SPARQL query 29

4.1 Grammatical Evolution mechanism 45
4.2 An illustration of a mapping process 47
4.3 An illustration of the parent selection mechanism. 55
4.4 An illustration of a single-point crossover 56

5.1 A schematic illustration of the disjointness of two classes C_i and C_j . 73

6.1 An illustration of mapping process to an expression of class disjointness axiom 93
6.2 The process of *Gold Standard* creation 99
6.3 The diversity of axioms over generations 103
6.4 The growth of average fitness over generations 104
6.5 Possibility and generality distribution of the discovered axioms. . . 106
6.6 Workflow of class disjointness axioms discovery using GE in the training- testing model 110
6.7 An illustration of the Training Dataset sampling procedure 111
6.8 Number of axioms discovered over 20 runs. 113

7.1 An illustration of Pareto front for a *minimization* problem 121
7.2 NSGA-II mechanism 124
7.3 Statistical comparison about the number of axioms discovered over 20 runs between GE and MOGE method. 130
7.4 Possibility and generality distribution of the discovered axioms with $\Pi(\phi) > \frac{2}{3}$ 131
7.5 The distribution of the discovered axioms in terms of measures $(\Pi(\phi) > \frac{2}{3})$ 132

List of Tables

2.1 Naming convention in DLs 13
2.2 $\mathcal{SROIQ}$ constructors [KSH12] 14
2.3 $\mathcal{SROIQ}$ axioms [KSH12] 16
2.4 OWL syntaxes 22
2.5 OWL 2 constructors [TFG17] 26
2.6 OWL 2 axioms [TFG17] 28

4.1 The conventions of W3C grammar notation 50

5.1 Translation of OWL axioms into FOL [TFG17] 65
5.2 Formal relationships between class expressions and their SPARQL queries 69

6.1 The first set of GE parameter values with *Fitness Function 1 (6.1)* 102
6.2 The second set of GE parameter values with *Fitness Function 2 (6.11)* 102
6.3 Experimental results 103
6.4 Experimental results 106
6.5 Parameter values for GE. 112
6.6 Number of valid distinct axioms discovered over 20 runs. 112
6.7 Average precision per run ($\pm$***std***) 114
6.8 Possibility and generality distribution of the discovered axioms for different population sizes (columns) and total efforts $k = 100,000, \ldots, 400,000$ (rows). 117

7.1 Matrix for the comparison between nodes 127
7.2 Parameter values for MOGE. 129
7.3 Number of valid distinct axioms discovered over 20 runs 129
7.4 Average precision per run ($\pm$***std***) 130

A.1 List sub-classes of the class 'Work' 144

B.1 List of classes in the training dataset 145

List of Abbreviations

SW . . . . . Semantic Web

LD . . . . . Linked Data

LOD . . . . . Linked Open Data

RDF . . . . . Resource Data Framework

OWL . . . . . Web Ontology Language

DL . . . . . Description Logics

KB . . . . . Knowledge Base

EC . . . . . Evolutionary Computation

EA . . . . . Evolutionary Algorithms

GE . . . . . Grammatical Evolution

GP . . . . . Genetic Programming

ILP . . . . . Inductive Logic Programming

MOO . . . . . Multi-objective Optimization

MOEA . . . . Multi-objective Evolutionary Algorithm

MOGE . . . . Multi-objective Grammatical Evolution

OWA . . . . . Open World Assumption

CWA . . . . . Closed World Assumption

1 Introduction

Contents

1.1 Context 1
1.2 Motivations 5
1.3 Objectives 5
1.4 Contributions 6
1.5 Publications 7
1.6 Thesis Outline 8

1.1 Context

Today, we witness the explosion of information over the Web which can supply knowledge and information for users to learn about a variety of topics or questions. In reality, there are powerful search engines which support for finding specific information from the Web based on keyword criteria. However, the organization of information on the Web is maintained in human-readable form only, which reduces the effectiveness of these search tools. For instance, only a few results of thousands of matches typically returned by search engines is truly relevant content. Some contents are hidden within the identified pages as well as classification and generalization of identifiers are irrelevant to the searching context. Thus, extending the current Web with the information given well-defined meaning that enables

computers to be easier in processing data in order to turn it into highly relevant information and knowledge is an expected development direction.

In May 2001, Tim Berners-Lee introduced the idea [BHL01] of an extension of the World Wide Web (WWW), called the Semantic Web (SW) [W3Ce], which can create a better environment for computers and people to work in cooperation. The Semantic Web may be viewed as the movement from the Web of documents to the Web of data [SBH06] in which information of the current Web 2.0, expressed in the form of unstructured or semi-structured data, is converted into structured format that machines can understand. In fact, the SW provides a standardized framework (introduced in Section 2.2) for describing a domain of interest with machine-processable information, known as *machine-interpretable metadata*, embedded within Web content. Among these metadata, *URIs* (*Uniform Resource Identifiers*) (presented in Section 2.2.1) are used to uniquely identify abstract or physical resources. For easy sharing, exposing and connecting data, information and knowledge, the SW uses a common standard framework for representing resources which is RDF (Resource Description Framework) [W3C14a], based on the notion of a *triple* (*subject*, *predicate*, *object*), i.e. an *RDF triple*. Each element of a triple is bound with a URI that performs a referential function, enabling it to be both human readable and machine processable. Any object of an RDF triple can become the subject of another triple, making chains of relationships and representing knowledge in the form of a graph or network.

Furthermore, the information available on the Web is fragmented from different data sources, thus, the data should be connected to generate a huge web corpus of domain datasets which can contribute to the global knowledge commons. The common principles in data integration have relied on specific applications to consolidate data from disparate sources into a single dataset within common data models. However, in the Web-scale data integration from a too large variety sources, the traditional model is not effective. In order to address this challenge, in 2006, Tim Berners-Lee recommended a set of best practises for publishing and interlinking data on the Web using URI and HTTP called *Linked Data*. Linked Data (LD) [W3Ca] is

defined as a method to create a Web of data through linking datasets over the Web; in this case, we talk about linking RDF datasets, using the structured model of the SW. LD comprises a set of principles for sharing machine-readable interlinked data on the Web [Ber] which relies on a set of standards of the SW technologies as follows:

1. Use URIs (or IRIs) to name things.
2. Use HTTP URIs (or HTTP IRIs) so that those names can be looked up.
3. Use the standard format RDF to represent information and use SPARQL to query it.
4. Include links to other URIs to connect the data between data sources so that connected information can be discovered

For the purpose of being freely available for sharing and reuse, LD is associated with Open Data constituting Linked Open Data(LOD). In 2010, Tim Berners-Lee proposed the star scheme to rate the availability of LD as LOD. Each star (i.e., rating) stands for a property added to the properties of the previous rating:

- ⋆ Data is available on the web in any format but with an open licence.
- ⋆⋆ Data is available and structured in the machine-readble form.
- ⋆⋆⋆ Data format is non-proprietary
- ⋆⋆⋆⋆ Use open standards from W3C: URIs to identify things, RDF to represent data and SPARQL to query data
- ⋆⋆⋆⋆⋆ Link datasets to other people's datasets to provide context.

In reality, the LOD community project works [W3Cb] aim at publishing open RDF datasets on the Web and establishing RDF links between entities from different datasets. This project has also published a cloud diagram visualizing available datasets illustrated in Figure 1.1. At this time, the volume of LOD has reached the status of "big RDF data". Indeed, in May 2020, the number of datasets increased to

1,255 with billions RDF triples compared only 12 in 2007. Additionally, published datasets already cover diverse topics such as life science, linguistics, social networking, geography, publication, media, etc. Some prominent representatives of the LOD are DBpedia[1](a rather rich collection of facts extracted from the Wikipedia), Freebase (linked dataset used by Google) or YAGO (linked dataset extracted from sources such as Wikipedia and WordNet). LOD is considered as the massive deployment

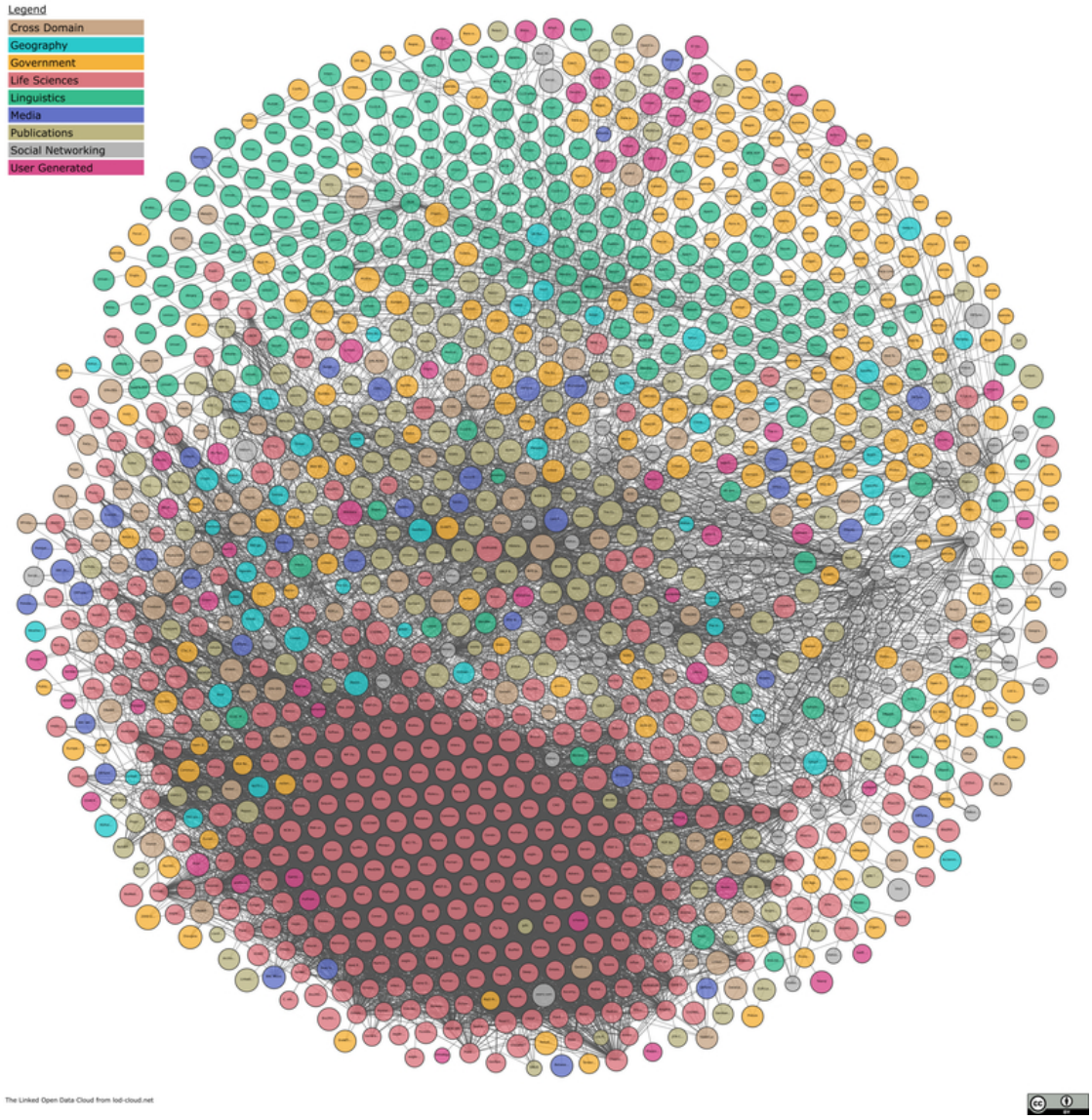


Figure 1.1: The Linked Open Data Cloud

of the Semantic Web according to the standards of technologies.

[1]https://wiki.dbpedia.org/

1.2 Motivations

The noticeable point is that the organization of LOD refers not only to linking "raw" RDF triples but also to embedding formal semantics for the triples through semantic schema captured in the concept of *ontology* (introduced in Section 2.1.1). In this sense, LOD contains a collection of RDF knowledge bases (RDF KBs) which integrates both *schema-level* knowledge represented in ontologies and *assertional* knowledge (assertions) given by RDF triples. In reality, RDF data published in LOD are mostly extracted and generated from different unstructured or semi-structured data sources, e.g. DBpedia extracted from Wikipedia, where there can exist incompleteness or that can be ridden with inconsistencies and errors in the information generated arbitrarily by the users. As a result, extracted data which can be erroneous, noisy or insufficient are added into KBs of LOD. Hence, the existence of ontologies, in particular, *axioms* expressing constrains, is critical to detecting data errors in LOD KBs. In addition, LOD KBs are only rich in factual information, i.e. raw RDF data, which is relatively abundant and easy to capture, but poor in knowledge models, i.e. ontologies, that make it limited to infer implicit knowledge by reasoning. This raises the demand of ensuring a co-evolution of ontologies and RDF data in KBs of LOD.

The common approach in the organization is to construct or reuse ontologies before filling data in them. This process is similar to the case when a database schema must be designed before a database can be populated. Nevertheless, this approach has some limitations. It is dogmatic in the knowledge organization. More specifically, obtained knowledge models in ontologies can be incomplete when they often do not provide all aspects that are required for specific domains of knowledge. Also, it does not represent a collaborative effort with actual populated data later.

1.3 Objectives

In the context mentioned in Section 1.1 and based on the motivations explained in Section 1.2, a conclusion is derived that it is increasingly important to enhance

ontological knowledge, i.e., schema-level knowledge, for RDF KBs of LOD. Ideally, these new knowledge should respect the existing knowledge along with the data in order to be maximally informative and avoid contradictions. In this sense, another more effective way is to use the facts themselves in LOD to learn new ontological knowledge which is able to account for them. The overall objective of this thesis is to tackle the challenges of learning new ontological knowledge from RDF datasets of LOD. The main research objective raises multiple specific research questions that need to be answered:

- **Research Question 1** (RQ_1): *What kinds of ontological knowledge need to be learned?*
- **Research Question 2** (RQ_2): *Which method is optimal to learn this kind of knowledge from RDF data?*
- **Research Question 3** (RQ_3): *How is the quality of learned knowledge evaluated?*
- **Research Question 4** (RQ_4): *How to evaluate the effectiveness of learning methods?*

1.4 Contributions

The contributions of this thesis consist of a number of answers to the above research questions:

- Regarding the first question RQ_1, this thesis is to address the problem of learning *axioms* which specify constrains describing the relationships between conceptual elements in ontologies and also supporting for reasoning activities. Axioms are considered as the theory derived from axiomatic statements describing the truth in the particular domain. In the Semantic Web, these axioms are expressed in OWL (Web Ontology Language) so called OWL axioms (introduced in Section 2.2.3). Specially, in the experiments we mainly

focus, for reasons that will be explained in Chapter 6, on the problem of mining OWL axioms describing the disjointness between classes.

- In order to answer the RQ_2, after considering the limitations of existing learning methods in general, we propose a completely novel learning model exploiting ideas from a heuristic approach of Evolutionary Computation. Specifically, we use Grammatical Evolution to learn OWL axioms from LOD. In particular, we apply this model to mining class disjointness axioms.

- Regarding the third question RQ_3, we exploit a possibilistic evaluation framework to measure the quality of discovered axioms, that is suitable to comply with the Open World Assumption (OWA) scenario. More specifically, we offer axiom scoring functions based on possibility theory.

- Regarding the fourth question RQ_4, we built two benchmarks: the subjective and the objective in order to measure the effectiveness of the learning method. The subjective benchmark is called *Gold Standard* that is constructed by knowledge engineers. The objective one is developed based on the *training-testing* model in which the test dataset is considered as an objective benchmark.

1.5 Publications

Work on this thesis has led to the following publications:

1. Thu Huong Nguyen and Andrea G. B. Tettamanzi. "**Learning Class Disjointness Axioms Using Grammatical Evolution**". In: Genetic Programming - 22nd European Conference, EuroGP 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24-26, 2019, Proceedings. Vol. 11451. Lecture Notes in Computer Science. Springer, 2019, pp. 278–294.

2. Thu Huong Nguyen and Andrea G. B. Tettamanzi. "**An Evolutionary Approach to Class Disjointness Axiom Discovery**". IEEE/WIC/ACM International Conference on Web Intelligence 2019, WI 2019, Thessaloniki, Greece, October 14-17, 2019, pages 68–75, ACM, 2019.

3. Thu Huong Nguyen and Andrea G. B. Tettamanzi. "**Grammatical Evolution to Mine OWL Disjointness Axioms Involving Complex Concept Expressions**" In: IEEE Congress on Evolutionary Computation, CEC2020, Glasgow, United Kingdom, July 19-24, 2020. IEEE, 2020, pp. 1–8.

4. Thu Huong Nguyen and Andrea G. B. Tettamanzi. "**Using Grammar-Based Genetic Programming for Mining Disjointness Axioms Involving Complex Class Expressions**". In: Ontologies and Concepts in Mind and Machine - 25th International Conference on Conceptual Structures,ICCS 2020, Bolzano, Italy, September 18-20, 2020, Vol. 12277. Lecture Notes in Computer Science. Springer, 2020, pp. 18–32.

5. Thu Huong Nguyen and Andrea G. B. Tettamanzi. "**A Multi-Objective Evolutionary Approach to Class Disjointness Axiom Discovery**". WI-IAT 2020 - IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. Melbourne/ Virtual, Australia, December 14-17, 2020.

1.6 Thesis Outline

This document is essentially split into three parts. The first part consists of three chapters presenting the basic knowledge and literature review involving the thesis topic. The second part comprises Chapter 4 and Chapter 5 providing formal models used in the thesis; Chapter 6 and Chapter 7 describing a detailed discussion of the contributions of the thesis. The last part is Chapter 8 comprising the conclusions of the thesis and perspectives. The content of the next chapters of the thesis are summarized as follows:

- **Chapter 2** introduces the basic concepts relating to the thesis. This includes concepts and notations in terms of ontologies and the Semantic Web technologies.

- **Chapter 3** provides a literature review for this thesis. The content of this chapter is composed of three main parts: (i) ontology learning including recent learning techniques in the context of LOD (ii) the recent studies of axiom learning (ii) mining RDF data.

- **Chapter 4** provides a general model based on an instance of an Evolutionary Algorithm, namely *Grammatical Evolution*(GE), to learn OWL axioms.

- **Chapter 5** introduces various evaluation frameworks to axiom scoring, specifically, *probabilistic* and *possibilistic* methods.

- **Chapter 6** shows the implementations of Chapter 4 and Chapter 5. In particular, two specialized models to discover OWL class disjointness axioms are given. Also, the chapter introduces two benchmarks to evaluate the performance of learning models.

- **Chapter 7** introduces a multi-objective extension to the learning models in Chapter 6 called MOGE.

- **Chapter 8** summarizes the contributions of the thesis and provides perspectives and future works.

2
Foundation

Contents

2.1 Ontology based Knowledge Representation 11
2.1.1 Ontologies 11
2.1.2 Description Logics 12
2.2 The Semantic Web 17
2.2.1 Uniform Resource Identifiers (URIs) 17
2.2.2 Resource Description Framework 18
2.2.3 Ontology Modeling Languages: RDFS and OWL 20
2.2.4 The SPARQL Query Language 27

Originated in the research problems and the research questions presented in the previous chapter, we investigate essential background to possibly capture the whole content of the thesis. In this chapter, we initially study ontology-based knowledge representation in Section 2.1. Specifically, we investigate the specification of ontologies and a popular language for ontology formalisation, called *Description Logics*. Section 2.2 presents the concepts and notations concerning the Semantic Web which is the basis for research problem.

2.1 Ontology based Knowledge Representation

2.1.1 Ontologies

Ontologies are considered as conceptual models of things in several domains transformed into machine-interpretable form by means of knowledge representation(KR) techniques. The term "Ontology" was derived from philosophy where ontology is considered as a philosophical investigation of existence [Cra98]. In computer science, there are various definitions of an ontology listed and compared in [GOS09] but the most popular one being known is the definition of Gruber [Gru95]: "*An ontology is a formal, explicit specification of shared conceptualization*". In terms of this definition, there are several characteristics of an ontology as follows:

- *formality*: An ontology provides a formal semantics which is machine-processable and is being interpreted in a well-defined way.
- *explicitness*: An ontology defines knowledge explicitly to make it accessible for machines.
- *sharebility*: An ontology captures consensual knowledge, that is, it is accepted by a group

In terms of KR, the conceptualization in the definition refers to knowledge of a domain represented in a declarative formalism, whereas explicit specification reflects in the representational terms for the respective domain of interest [SPA07]. An ontology can be referred to as a formal representation of a shared domain knowledge [LV14b].

An ontology can be defined as a quadruple [Che+10] $\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle$, where $\mathcal{C}$ is the set of concepts; $\mathcal{R}$ is the set of relations; $\mathcal{I}$ is the set of instances; $\mathcal{A}$ is the set of axioms. *Concepts* represent types of the named and identifiable concrete objects in the domain of interest. *Relations* specify the way in which concepts and instances can be related to one another. *Axioms* are the statements that are fully axiomatized theories about the domains. An example of an axiom imposing a

restriction on relation between two concepts is that "only students of a particular school can use services provided by this organization".

2.1.2 Description Logics

Definition & Characteristics

Description Logics(DLs) [KSH12] are a family of formal languages for representing knowledge and reasoning about it that are widely used in ontological modelling. DLs provide means to model the relationship between entities in a domain of interest. DLs are essentially decidable fragments of FOL equipped with a *formal semantics* which allows humans and machines to exchange DL ontologies without ambiguity and support the capability of inferring additional knowledge. There are different types of DLs which have informal names roughly describing the operators allowed. Table 2.1 illustrates the labels for a logic expressivity in DLs. In this thesis, we investigate $\mathcal{SROIQ}$, which is one of the most expressive DLs and serves as the logical basis of ontology language OWL 2 DL (see in Section 2.2.3).

Constitution

DLs are based on three disjoint sets of primal elements [Rud11]:

- The set N_C of concept names contains names referring to categories, types or classes of entities in a domain of interest, e.g. Person, Country, City, ...
- The set N_R of role names describes binary relations between the individuals of a domain, e.g. isFatherOf, isPlaceOF, is ConnectedTo, ...
- The set N_I of individual names describes single individuals, singular entities, in a domain, e.g. the sun, Nice, Finland, ...

In addition, DLs can include concepts and roles containing a variety of different *constructors*, i.e., *concept expressions* (also called *complex concepts*) and *role expressions* (also called *complex roles*) for the description of more complex situations. The two first columns of Table 2.2 present $\mathcal{SROIQ}$ constructors, their syntax and semantics.

Table 2.1: Naming convention in DLs

$\mathcal{AL}$	Attribute language allowing atomic negation, concept intersection, concept intersection, universal restrictions, limited existential quatification
$\mathcal{EL}$	Existential language allowing concept intersection, existential restrictions
$\mathcal{FL}$	Frame based description language containing concept intersection, universal restrictions, limited existential quantification, role restriction
$\mathcal{F}$	Functional properties
$\mathcal{E}$	Full existential qualification
$\mathcal{U}$	Concept union
$\mathcal{C}$	Complex concept negation
$\mathcal{H}$	Role hierachy
$\mathcal{R}$	Complex role inclusion axioms, role disjointness, reflexitivity and irreflexitivity
$\mathcal{O}$	Nominals
$\mathcal{I}$	Inverse properties
$\mathcal{N}$	Cardinality restrictions
$\mathcal{Q}$	Qualified cardinality restrictions
$(\mathcal{D})$	Used of datatype properties, data values or data types
$\mathcal{S}$	An abbreviation for $\mathcal{ALC}$ with transitive roles
$\mathcal{EL}^{++}$	Alias for $\mathcal{ELRO}$

DLs Knowledge Base

A $\mathcal{SROIQ}$ knowledge base (KB) [Rud11] defined by tuples $\mathcal{K} = (\mathcal{A}, \mathcal{T}, \mathcal{R})$ consists of a set of axioms classifying into three groups:

- *Assertional axioms (ABox $\mathcal{A}$)*: describe a specific state of affairs of an application domain in terms of concepts and roles, i.e *assertions* about named individuals. They can be *concept assertions*, e.g. `Father(Jim)` states that Jim is a father or *role assertions*, e.g. `fatherOf(Jim,John)` states that Jim is John's father.

- *Terminological axioms (TBox $\mathcal{T}$)*: describe the relationships between concepts. In the most general cases, TBox axioms are divided into two kinds: *inclusions* and *equalities*. For example, `Mother` $\sqsubseteq$ `Parent` states the fact that all mothers are parents, while `Person` $\equiv$ `Human` states that the two concepts have the same instances.

Table 2.2: $\mathcal{SROIQ}$ constructors [KSH12]

	Syntax	Semantics
Individuals:		
individual name	a	a^I
Roles:		
atomic role	R	R^I
inverse role	R^-	$\{\langle x,y\rangle \mid \langle y,x\rangle \in R^I\}$
universal role	U	$\Delta^I \times \Delta^I$
Concepts:		
atomic concept	A	A^I
intersection	$C \sqcap D$	$C^I \cap D^I$
union	$C \sqcup D$	$C^I \cup D^I$
complement	$\neg C$	$\Delta^I \setminus C^I$
top concept	$\top$	Δ^I
bottom concept	$\bot$	$\emptyset$
existential restriction	$\exists R.C$	$\{x \mid \text{some } R^I\text{-successor of } x \text{ is in } C^I\}$
universal restriction	$\forall R.C$	$\{x \mid \text{all } R^I\text{-successors of } x \text{ are in } C^I\}$
at-least restriction	$\geqslant n\, R.C$	$\{x \mid \text{at least } n\ R^I\text{-successors of } x \text{ are in } C^I\}$
at-most restriction	$\leqslant n\, R.C$	$\{x \mid \text{at most } n\ R^I\text{-successors of } x \text{ are in } C^I\}$
local reflexivity	$\exists R.Self$	$\{x \mid \langle x,x\rangle \in R^I\}$
nominal	$\{a\}$	$\{a^I\}$

where $a, b \in \mathsf{N}_I$ are individual names, $A \in \mathsf{N}_C$ is a concept name, $C, D \in \mathbf{C}$ are concepts, $R \in \mathbf{R}$ is a role

- *Relational axioms (RBox $\mathcal{R}$)*: describe the relationships between relations. As for concepts, DLs support *role inclusion* and *role equivalence* axioms. For example, the inclusion `brotherOf` $\sqsubseteq$ `siblingOf` states that `brotherOf` is a *subrole* of `siblingOf`. In addition, RBox axioms include *role disjointness*, e.g. `Disjoint(parentOf,childOf)` states that nobody can be both a parent and a child of the same named individual. Role inclusion axioms can be *complex role inclusion axioms* containining *role composition*, e.g. `brotherOf` $\circ$ `parentOf` $\sqsubseteq$ `uncleOf`. More RBox axioms include role characteristeristics such as reflexivity, symmetry and transitivity of roles [Hoe09]

A pairs of TBox and RBox is the structural and intensional component of conceptual relationships (concepts and roles) as conceptual schemas. Meanwhile, the ABox specifies knowledge at extensional level containing the facts about specific individuals.

DLs Semantics

The semantics of DLs is defined in a *model-theoretic* way based on *interpretations*. Instead of using default assumptions to fully define one particular interpretation for an ontology[KSH12], i.e. *Closed World Assumption (CWA)*, where complete information about a given state affairs is provided [Kee13a], the DLs semantics refer to all possible situations where the axioms of an ontology would hold, i.e., *Open World Assumption (OWA)*, that relevant to incomplete information about a given state of affairs [Kee13b]. An interpretation [Kaz10] is a pair $\mathcal{I}$=($\Delta^{\mathcal{I}}$, $\cdot^{\mathcal{I}}$) where $\Delta^{\mathcal{I}}$ is a non-empty set called *the domain of interpretation* and $\cdot^{\mathcal{I}}$ is the interpretation function that maps individual names to elements in the domain. The semantics of complex concepts and roles formalizing the meaning of the $\mathcal{SROIQ}$ constructors are listed in the third column of Table 2.2.

Reasoning in DLs

- *Satisfaction of Axioms*: An interpretation $\mathcal{I}$ *satisfies* an axiom α, i.e., α holds in $\mathcal{I}$ (written: $\mathcal{I} \models \alpha$), if it makes the axiom α true (the corresponding condition in Table 2.3 is met) and is considered as a *model* of that axiom α. Conversely, an axiom is unsatisfiable if none of the interpretations makes it true.

- *Models*

  - An interpretation $\mathcal{I}$ is a model of TBox $\mathcal{T}$ (written: $\mathcal{I} \models \mathcal{T}$) if it satisfies every axiom in $\mathcal{T}$.

  - An interpretation $\mathcal{I}$ is a model of RBox $\mathcal{R}$ (written: $\mathcal{I} \models \mathcal{R}$) if it satisfies every axiom in $\mathcal{R}$.

  - An interpretation $\mathcal{I}$ is a model of ABox $\mathcal{A}$ (written: $\mathcal{I} \models \mathcal{A}$) if it satisfies every assertional axiom in $\mathcal{A}$.

  - An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K}$ (written: $\mathcal{I} \models \mathcal{K}$) if it satisfies every axiom in $\mathcal{K}$.

Table 2.3: $\mathcal{SROIQ}$ axioms [KSH12]

	Syntax	Semantics
ABox:		
concept assertion	$C(a)$	$a^I \in C^I$
role assertion	$R(a,b)$	$\langle a^I, b^I \rangle \in R^I$
individual equality	$a \approx b$	$a^I = b^I$
individual inequality	$a \not\approx b$	$a^I \neq b^I$
TBox:		
concept inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
concept equivalence	$C \equiv D$	$C^I = D^I$
RBox:		
role inclusion	$R \sqsubseteq S$	$R^I \subseteq S^I$
role equivalence	$R \equiv S$	$R^I = S^I$
complex role inclusion	$R_1 \circ R_2 \sqsubseteq S$	$R_1^I \circ R_2^I \subseteq S^I$
role disjointness	$Disjoint(R,S)$	$R^I \cap S^I = \emptyset$

- *Satisfiability or Consistency*:
  - A concept $\mathcal{C}$ is satisfiable or consistent if it has at least a model.
  - $\mathcal{K}$ is satisfiable or consistent if it has at least a model.
- *Entailment*:
  - An axiom α is called a *logical consequence* of a TBox $\mathcal{T}$, i.e., $\mathcal{T}$ entails α, if every model of $\mathcal{T}$ satisfies α (written: $\mathcal{T} \models \alpha$), i.e. α holds in all the interpretations that satisfy $\mathcal{T}$
  - An axiom α is called a *logical consequence* of a RBox $\mathcal{R}$, i.e., $\mathcal{R}$ entails α, if every model of $\mathcal{R}$ satisfies α (written: $\mathcal{R} \models \alpha$), i.e. α holds in all the interpretations that satisfy $\mathcal{R}$
  - An axiom α is called a *logical consequence* of a knowledge base $\mathcal{K}$, i.e., $\mathcal{K}$ entails α, if every model of $\mathcal{K}$ satisfies α (written: $\mathcal{K} \models \alpha$), i.e. α holds in all the interpretations that satisfy $\mathcal{K}$

2.2 The Semantic Web

A set of technologies, tools and standards of the Semantic Web is organised into the so called Semantic Web stack. Figure 2.1 illustrates different layers of the SW architecture. The tasks and features of each layer are describes as follow:

- The bottom layers focus on the syntactic interoperability by using *Unicode*, *URI* and *XML*.
- The middle layers concern technologies to enable building semantic web applications such as a standard model for data interchange on the Web *RDF*, ontology languages *RDFs* and *OWL*, a query languages *SPARQL* used to query any RDF-based data and rule languages *RIF/SWRL*. One notable point is the role of ontologies in this critical layer. Functionally, ontologies provide data schemas and a set of conceptual vocabularies with explicit semantics which fit the goal of SW in terms of comprehensive and transportable machine understanding.
- The top layers include Logic, Proof and Trust, are currently being researched and are being constructed. In this, the Logic layer enables intelligent reasoning by creating logical relations that cannot be defined in OWL. The Proof layer concerns the rules and evaluates cooperating with the Trust layer to define the credibility of the given proof.

Next, we consider in detail technologies for representing resources and knowledge in the Semantic Web: Uniform Resource Identifiers (URIs), Resource Description Framework (RDF), RDF Schema (RDFS) and Web Ontology Language (OWL). Afterwards, the query language SPARQL is also explained.

2.2.1 Uniform Resource Identifiers (URIs)

In the SW, each entity is defined by a specific name identified, i.e. *URI* (*Uniform Resource Identifier*). A URI [Mas05] consisting of a string of characters can be identified as a locator (*URL—Uniform Resource Locator*), a name (*URN—Uniform*

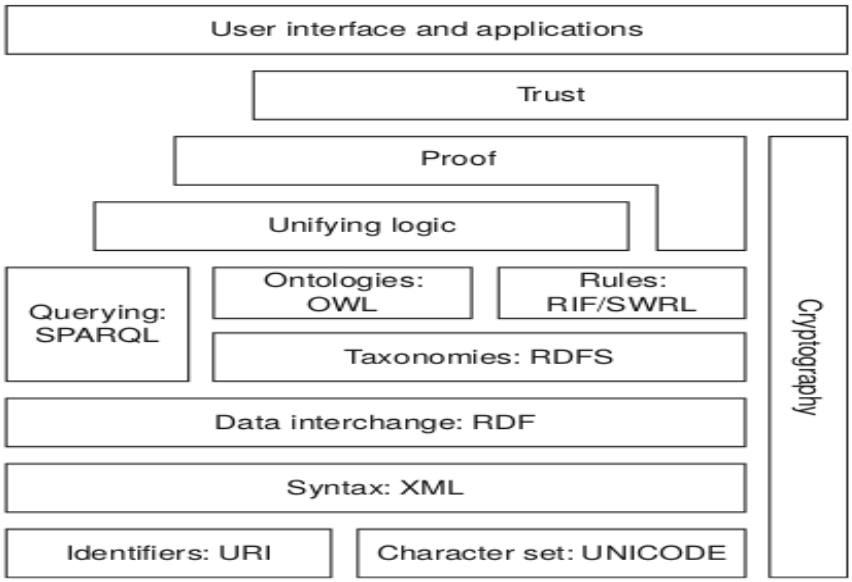


Figure 2.1: The Semantic Web Layer Architecture [Gri+11]

Resource Name) or both. A URI that provides a means of locating the resource by describing its primary access mechanism is referred to as a URL. Meanwhile, an URI used as an URN refers to providing a globally unique name for a resource. Also, according to *RFC3987* [Sui05], a complement of URIs (an upgraded version of URIs) are *IRIs (International Resource Identifiers)* which extend the *ASCII* characters of the URI version to a wide range of characters from the *Universal Character Set (Unicode)* including many special characters in different languages. Using URIs to identify entities and relations between them is essential for a global and unique namespace. The use of such a scheme greatly reduces the ambiguity, e.g. homonym problem, due to distributed data representation in the traditional databases like relational databases.

2.2.2 Resource Description Framework

Resource Description Framework (RDF) [W3C14a] [PF02] is mainly a data model of SW for semantically representing resources on the Web. RDF is a Web-oriented framework and identifies resources and relationships between them, i.e. properties, using URIs [Gan+11]. In terms of the structure, RDF uses as

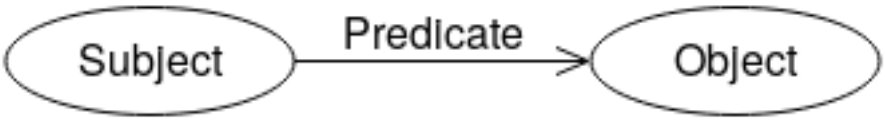


Figure 2.2: An RDF graph

statements of triples of the form

$$\langle \textbf{Subject}, \textbf{Predicate}, \textbf{Object} \rangle.$$

, where:

- the subject is a URI or a blank node.
- the predicate is a URI.
- the object is a URI, a literal or a blank node.

The RDF data model can be presented in the form of a directed-labeled graph, i.e. *RDF graph*[1]

Example 2.2.1 *: The content of the sentence "The 1997 film Titanic was produced by James Cameron" can be expressed in machine-accessible form as an RDF statement as follow:*

- *the subject is "Film_Titanic_1997"*
- *the predicate is "hasProducer"*
- *the object is "James_Cameron"*

Each part of the statement can be described in the form of URIs.

- *the subject is "http://dbpedia.org/resource/Titanic_(1997_film)"*
- *the predicate is "http://dbpedia.org/ontology/producer"*
- *the object is "http://dbpedia.org/resource/James_Cameron"*

[1]https://www.w3.org/TR/rdf11-concepts/#section-rdf-graph

or in the shorter representation associated with the prefix aliases,[2]

PREFIX dbr: http://dbpedia.org/resource/

PREFIX dbo: http://dbpedia.org/ontology/

- *the subject is "dbr:Titanic_(1997_film)"*
- *the predicate is "dbo:producer"*
- *the object is "dbr:James _Cameron"*

A notable point is that RDF triples are interpreted according to the open-world assumption (OWA). In this sense, the RDF semantics assumes that whatever is not explicitly stated could be true [FS11]. In Example 2.2.1, the fact in RDF triple indicates that "*Titanic was produced by James Cameroon*" does not mean that only *James Cameroon* is the producer; it only means that there is at least one named producer.

2.2.3 Ontology Modeling Languages: RDFS and OWL

RDF Schema (RDFS)

RDFS [W3C14b] is a set of of data-modelling vocabularies for RDF data which is a semantic extension of RDF. RDFS is used to declare and describe the resource types , i.e. classes and resource relationship, and attribute types, i.e. properties, and to organise them in hierarchies [Gan+11]. These schema are also published and exchanged in RDF. However, RDFS has expressive limitations compared with other ontology model languages like OWL in that it lacks some schema definitions in RDFS such as equality of individuals, equivalence or disjointness of properties and classes which restrict in reasoning. Hence, RDFS is used to define simple ontologies, i.e. lightweight ontologies [Vol+03].

[2] https://docs.microsoft.com/en-us/windows/desktop/winrm/uri-prefixes

Web Ontology Language (OWL)

OWL [W3C04b][W3C12a] is a family of knowledge representation languages for publishing and sharing ontologies. OWL is a vocabulary extension of RDF and RDFS but much more expressive regarding the description of classes and properties. OWL is based on DLs and its profile or sub-languages or species (see below) have been defined as syntatic variants of certain DLs.

OWL Syntaxes

OWL supports a variety of syntaxes which cover from the *high level* syntaxes aimed to the structural specification, e.g., *functional style syntax*, to *exchange* syntaxes for general use, e.g., *Manchester OWL syntax*, *OWL/XML*, *RDF Turtle* and *RDF/XML*. The description of syntaxes is illustrated in Table 2.4. In the table, attached examples [W3C12b] describe an equivalent class which indicates that the *Mother* class is equivalent to the set of objects which are instances of both class *Woman* and *Parent*.

OWL Versions

Following W3C recommendations, two versions of OWL, namely OWL 1 [W3C04a] and OWL 2 [W3C12c], have been proposed in which not only inherit a number of vocabularies from RDFS but also provide new sophisticated terms to automated reasoning support. The first version of OWL can be classified into three sub-languages called"profiles" corresponding to the degree of expresiveness: OWL 1 Full, OWL 1 DL and OWL 1 Lite. Each of these species is a syntactic extension of its simpler predecessor and also RDFS. The differences between three variants of OWL 1 are listed as follow:

- ***OWL 1 Lite*** is the syntactically simplest species of OWL 1 and corresponds to $\mathcal{SHIF}(\mathcal{D})$ in DLs. It is used in conceptually simple hierarchies and simple constraints. It has fewer constructs compared other species. For examples, it does not support explicit negation, union, nominal operators or only cardinality values of 0 or 1 are allowed.

Table 2.4: OWL syntaxes

Syntaxes	**Specification**	**Examples** [W3C12b]
Functional Style	is a concrete syntax for OWL ontologies which closely obeys the structural specification and is used in the definitions of the semantics of OWL 2 ontologies [W3Cf].	EquivalentClasses(:Mother ObjectIntersectionOf(:Woman :Parent))
Manchester OWL	is a user-friendly syntax used to describe OWL ontologies. It is based on a single construct known as *frame* containing all information about particular class, property or individual [W3Cc].	Class: Mother EquivalentTo: Woman and Parent
OWL/XML	specifies an XML serialization that closely models the structure of an OWL2 ontology [W3Cd]	<EquivalentClasses> <Class IRI="Mother"/> <ObjectIntersectionOf> <Class IRI="Woman"/> <Class IRI="Parent"/> </ObjectInteresectionOf> </EquivalentClasses>
RDF Turtle	is a syntax for expressing data in the RDF data model which its syntactic mapping possibly specified languages in the OWL family	:Mother owl:equivalentClass [rdf:type owl:Class ; owl:intersectionOf (:Woman :Parent)] .
RDF/XML	is the primary exchange syntax which must be supported by all OWL2 tools and is serialized in XML documents	<owl:Class rdf:about="Mother"> <owl:equivalentClass> <owl:Class> <owl:intersectionOf rdf:parseType="Collection"> <owl:Class rdf:about="Woman"/> <owl:Class rdf:about="Parent"/> </owl:intersectionOf> </owl:Class> </owl:equivalentClass>

- ***OWL 1 DL*** is much more expressive than OWL Lite and is equivalent to $\mathcal{SHOIN}(\mathcal{D})$ in DLs. It includes all OWL constructors used only under some restrictions, e.g. type separation in which a class cannot also individual or property and a property can not also be an individual or class. Also, it allows cardinality statements for arbitrary non-negative integers. OWL 1 DL supports automated reasoning, i.e. possible to automatically compute the classification hierarchy and check for inconsistencies in an ontology.

- ***OWL 1 Full*** is the most expressive OWL 1 sub-language. It is used in the situation referring to very high expressiveness but no computational guarantees. Thus, it is not possible to perform automated reasoning on OWL 1 Full ontologies.

Although OWL 1 has been successful, there have been several limitations in its design [Gra+08]:

- OWL 1 lacks expressivity, i.e., the absence of some constructors for modeling complex domains, e.g. constructors for qualified cardinality restrictions, or the absence of relational expressivity in properties, datatype or key constrains on data properties.

- OWL 1 has syntax issues in using two syntaxes: abstract syntax and OWL 1 RDF that their relationship is rather complex causing problems in transforming an ontology from one syntax into the other, e.g. RDF represents everything using triples to specify relationships between pairs of objects, whereas, many OWL 1 constructs cannot be represented using triples without the introduction of new objects.

- OWL 1 disallows the usage of annotation properties in OWL 1 DL axioms. Also, OWL 1 does not allow some important axioms to be annotated, for instance, to represent provenance information, e.g., who wrote a particular axiom, or for language extensions, e.g., to represent the confidence in the validity of axioms.

- OWL 1 DL and OWL 1 Lite were designed, on the one hand, as annotational variants of the expressive description logic $\mathcal{SHOIN}(\mathcal{D})$ and $\mathcal{SHIF}(\mathcal{D})$, respectively; on the other hand, it requires the compatibility of OWL 1 with existing Semantic Web languages such as RDF. Semantic differences between $\mathcal{SHOIN}(\mathcal{D})$ or $\mathcal{SHIF}(\mathcal{D})$ and RDF made it difficult to satisfy both requirements.

In order to address the limitations of OWL 1, the second version of OWL (OWL 2) has been proposed as an extension and revision of OWL 1 by adding new functionalities, new constructors and offering new expressivity presented in some examples as follows [CB15]:

- OWL 2 constructs qualified cardinality restrictions and annotation properties, e.g. `owl:minQualifiedCardinality`, `maxQualifiedCardinality` and `owl:qualifiedCardinality`.
- OWL 2 adds property chains and keys, e.g. `owl:propertyChainAxiom` and `owl:hasKey`.
- OWL 2 provides constructors expressing for new characteristics of properties, e.g. `owl:ReflexiveProperty, owl:IrreflexiveProperty, owl:AssymmetricProperty`, annotation properties, e.g. `owl:priorVersion` or incompatibility of properties, e.g. `owl:propertyDisjointWith`.
- OWL 2 provides new datatypes, e.g. `owl:real` and new construct to define data types `rdfs:DataType` and restriction definitions, combination of data ranges, e.g. `owl:IntersectionOf`, `owl:unionOf` and `owl:complementOf`.

Along with using the RDF/XML and Manchester exchange syntaxes, OWL 2 uses a functional-style syntax which replaces the abstract syntax of OWL 1. In addition to OWL 2 DL and OWL 2 Full, OWL 2 provides three new tractable profiles: OWL 2 RL, OWL 2 EL and OWL 2 QL. Like OWL 1, all sub-languages of OWL 2 can reuse some vocabularies in respect of RDFs. The characteristics of each profile is briefly presented in the following:

- OWL 2 DL is defined as the primitives for OWL 2 and is backward compatible with OWL 1 DL. In DLs, it corresponds to $\mathcal{SROIQ}(\mathcal{D})$, which encompasses $\mathcal{SHOIN}(\mathcal{D})$, i.e. OWL 2 DL contains OWL 1 DL. OWL 2 DL is considered as a very expressive language with high computational complexity of reasoning.

- OWL 2 Full, like the previous version OWL Full, is not decidable and rarely used in modelling ontologies.

- OWL 2 RL, also called OWL-based rule language, which provides restrictions on OWL 2 using rule based technologies such as DBMS, Jess and Prolog.

- OWL 2 EL, based on the $\mathcal{EL}++$ DL, defines restrictions on classes used in axioms. It provides polynomial-time reasoning for schema and data based on terminological expressivity. OWL 2 EL is particularly suitable for ontologies with a large TBox, i.e. large concept part.

- OWL 2 QL is relevant to a standard relational query language, i.e. SQL rewriting on RDBMS for query answering. It is useful for lightweight ontologies where there is a large ABox, i.e. large number of individuals.

In this thesis, we investigate OWL 2 DL, which is often used for OWL 2 ontologies, and the designation OWL 2 DL will be shortened to OWL 2.

The model-theortic semantics of the various constructors in OWL 2, i.e., OWL 2 DL, is shown in Table 2.5. In this table, the first column presents OWL 2 expressions in the functional style syntax, the second column contains their corresponding $\mathcal{SROIQ}(\mathcal{D})$ DLs syntax, and the last column shows their semantics.

OWL Axioms

In contrast to a DL knowledge base, where conceptual and instance level statements are usually split into, respectively, a set of TBox- RBox axioms and a set of ABox assertions, an OWL 2 ontology consists of a single set of axioms known as *OWL 2 axioms* that includes both conceptual (in terms of classes and properties) and instance (in terms of intances of classes and properties) level statements. There are

Table 2.5: OWL 2 constructors [TFG17]

OWL 2 (functional-style)	DL Syntax	Interpretation
ObjectInverseOf(R)	R^-	$(R^-)^\mathcal{I} = \{\langle y, x\rangle \mid \langle x, y\rangle \in R^\mathcal{I}\}$
DataIntersectionOf(D_1 ... D_n)	$D_1 \sqcap \ldots \sqcap D_n$	$D_1^\mathcal{I} \cap \ldots \cap D_n^\mathcal{I}$
DataUnionOf(D_1 ... D_n)	$D_1 \sqcup \ldots \sqcup D_n$	$D_1^\mathcal{I} \cup \ldots \cup D_n^\mathcal{I}$
DataComplementOf(D)	$\neg D$	$\mathsf{D}^{\text{arity}(D)} \setminus D^\mathcal{I}$
DataOneOf(d_1 ... d_n)	$\{d_1, \ldots, d_n\}$	$\{d_1^\mathcal{I}, \ldots, d_n^\mathcal{I}\}$
DatatypeRestriction(D F_1 d_1 ... F_n d_n)		$D^\mathcal{I} \cap \langle F_1, d_1\rangle^\mathcal{I} \cap \ldots \cap \langle F_n, d_n\rangle^\mathcal{I}$
ObjectIntersectionOf(C_1 ... C_n)	$C_1 \sqcap \ldots \sqcap C_n$	$C_1^\mathcal{I} \cap \ldots \cap C_n^\mathcal{I}$
ObjectUnionOf(C_1 ... C_n)	$C_1 \sqcup \ldots \sqcup C_n$	$C_1^\mathcal{I} \cup \ldots \cup C_n^\mathcal{I}$
ObjectComplementOf(C)	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
ObjectOneOf(a_1 ... a_n)	$\{a_1, \ldots, a_n\}$	$\{a_1^\mathcal{I}, \ldots, a_n^\mathcal{I}\}$
ObjectSomeValuesFrom(R C)	$\exists R.C$	$\{x \mid \exists y.\langle x, y\rangle \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
ObjectAllValuesFrom(R C)	$\forall R.C$	$\{x \mid \forall y.\langle x, y\rangle \in R^\mathcal{I} \Rightarrow y \in C^\mathcal{I}\}$
ObjectHasValue(R a)	$\exists R.\{a\}$	$\{x \mid \langle x, a^\mathcal{I}\rangle \in R^\mathcal{I}\}$
ObjectHasSelf(R)	$\exists R.\mathsf{Self}$	$\{x \mid \langle x, x\rangle \in R^\mathcal{I}\}$
ObjectMinCardinality(n R)	$\geq nR.\top$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I}\}\| \geq n\}$
ObjectMaxCardinality(n R)	$\leq nR.\top$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I}\}\| \leq n\}$
ObjectExactCardinality(n R)	$= nR.\top$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I}\}\| = n\}$
ObjectMinCardinality(n R C)	$\geq nR.C$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}\| \geq n\}$
ObjectMaxCardinality(n R C)	$\leq nR.C$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}\| \leq n\}$
ObjectExactCardinality(n R C)	$= nR.C$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}\| = n\}$
DataSomeValuesFrom(R D)	$\exists R.D$	$\{x \mid \exists y.\langle x, y\rangle \in R^\mathcal{I} \wedge y \in D^\mathcal{I}\}$
DataAllValuesFrom(R D)	$\forall R.D$	$\{x \mid \forall y.\langle x, y\rangle \in R^\mathcal{I} \Rightarrow y \in D^\mathcal{I}\}$
DataHasValue(R d)	$\exists R.d$	$\{x \mid \langle x, d^\mathcal{I}\rangle \in R^\mathcal{I}\}$
DataMinCardinality(n R)	$\geq nR.\top$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I}\}\| \geq n\}$
DataMaxCardinality(n R)	$\leq nR.\top$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I}\}\| \leq n\}$
DataExactCardinality(n R)	$= nR.\top$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I}\}\| = n\}$
DataMinCardinality(n R D)	$\geq nR.D$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I} \wedge y \in D^\mathcal{I}\}\| \geq n\}$
DataMaxCardinality(n R D)	$\leq nR.D$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I} \wedge y \in D^\mathcal{I}\}\| \leq n\}$
DataExactCardinality(n R D)	$= nR.D$	$\{x \mid \|\{y \mid \langle x, y\rangle \in R^\mathcal{I} \wedge y \in D^\mathcal{I}\}\| = n\}$

32 types of OWL 2 axioms, listed in Table 2.6. The structure of the table is similar to the previous table of OWL 2 constructors except for the first columns showing OWL 2 axioms in the functional style syntax. All axioms whose semantics share the same characteristics in terms of theoretic set will be grouped together as follows:

- *Assertions* involve named individuals or literals: `ClassAssertion`, `ObjectPropertyAssertion`, `NegativeObjectPropertyAssertion`, `DataPropertyAssertion` and `NegativeDataPropertyAssertion`.

- *Subsumption axioms* include axioms whose semantics are expressed in terms of set inclusion: `SubClassOf`, `SubObjectPropertyOf`, `SubDataProperty`, `ObjectPropertyDomain`, `ObjectPropertyRange`, `SymmetricObjectProperty`, `AsymmetricObjectProperty`, `TransitiveObjectProperty`, `DataProperty-`

`Domain` and `DataPropertyRange`.

- *Equivalence axioms* include axioms whose semantics are expressed in terms of set equality: `EquivalentClasses`, `EquivalentObjecProperties`, `Inverse-ObjectProperties`, `ReflexiveObjectProperties`, `EquivalentDataProperty` and `DataDefinition`

- *Disjointness axioms* include axioms whose semantics are expressed in terms of set disjunction: `DisjointClasses`, `DisjointObjecProperties`, `Inreflexive-ObjectProperties` and `DisjointDataProperties`.

- *Identity axioms* contain the three axiom types `HasKey`, `SameIndividual` and `DifferentIndividuals`. Also, we may add `FunctionalObjectProperty`, `InverseFunctionalObjectProperty` and `FunctionalDataProperty` in which identity plays an improtant role.

In this thesis, we only work on OWL 2, thus, OWL 2 axiom can be called shortly OWL axioms without risk of ambiguity.

2.2.4 The SPARQL Query Language

SPARQL [W3C08] [W3C13], an acronym for **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage is a structured and semantic query language for RDF knowledge bases. SPARQL can be considered as an interface to access knowledge on the Web of Data. A SPARQL query contains a set of triple patterns called *basic graph patterns (BGPs)* used to match a subset or a sub-graph from the queried RDF data or RDF graph. The results of SPARQL queries can be sets or RDF sub-graphs. Each triple pattern in BGP are RDF triples where subject, predicate and object can be unknown or replaced by a variable with a question mark, e.g. `?p` in which `p` is a variable. The conjunctions and the disjunctions of triple patterns are performed in the graph pattern match to provide the results.

The general SPARQL queries comprise three major parts [HKR10]:

Table 2.6: OWL 2 axioms [TFG17]

OWL 2 (functional-style)	DL Syntax	Semantics
SubClassOf$(C\ D)$	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
EquivalentClasses$(C_1\ \ldots\ C_n)$	$C_i \equiv C_j, i,j \in \{1,\ldots n\}$	$C_i^{\mathcal{I}} = C_j^{\mathcal{I}}, i,j \in \{1,\ldots n\}$
DisjointClasses$(C_1\ \ldots\ C_n)$	$\mathsf{Dis}(C_1,\ldots,C_n)$	$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \emptyset, i,j \in \{1,\ldots n\}, i \neq j$
DisjointUnion$(C\ C_1\ \ldots\ C_n)$	$C = C_1 \sqcup \ldots \sqcup C_n$, and	$C^{\mathcal{I}} = C_1^{\mathcal{I}} \cup \ldots \cup C_n^{\mathcal{I}}$, and
	$\mathsf{Dis}(C_1,\ldots,C_n)$	$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \emptyset, i,j \in \{1,\ldots n\}, i \neq j$
SubObjectPropertyOf(S,R)	$S \sqsubseteq R$	$S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
SubObjectPropertyOf(w,R), with	$S_1 \ldots S_n \sqsubseteq R$	$S_1^{\mathcal{I}} \circ \ldots \circ S_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$, i.e., $\forall y_0,\ldots,y_n,$
$w =$ ObjectPropertyChain$(S_1\ \ldots\ S_n)$		$\langle y_0,y_1\rangle \in S_1^{\mathcal{I}} \wedge \ldots \wedge \langle y_{n-1},y_n\rangle \in S_n^{\mathcal{I}}$
		$\Rightarrow \langle y_0,y_n\rangle \in R^{\mathcal{I}}$
EquivalentObjectProperties$(R_1\ \ldots\ R_n)$	$R_i \equiv R_j, i,j \in \{1,\ldots n\}$	$R_i^{\mathcal{I}} = R_j^{\mathcal{I}}, i,j \in \{1,\ldots n\}$
DisjointObjectProperties$(R_1\ \ldots\ R_n)$	$\mathsf{Dis}(R_1,\ldots,R_n)$	$R_i^{\mathcal{I}} \cap R_j^{\mathcal{I}} = \emptyset, i,j \in \{1,\ldots n\}, i \neq j$
ObjectPropertyDomain$(R\ C)$	$\geq 1R \sqsubseteq C$	$\langle x,y\rangle \in R^{\mathcal{I}} \Rightarrow x \in C^{\mathcal{I}}$
ObjectPropertyRange$(R\ C)$	$\top \sqsubseteq \forall R.C$	$\langle x,y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}$
InverseObjectProperties$(S\ R)$	$S \equiv R^-$	$S^{\mathcal{I}} = \{\langle y,x\rangle \mid \langle x,y\rangle \in R^{\mathcal{I}}\}$
FunctionalObjectProperty(R)	$\mathsf{Fun}(R)$	$\langle x,y\rangle \in R^{\mathcal{I}} \wedge \langle x,z\rangle \in R^{\mathcal{I}} \Rightarrow y = z$
InverseFunctionalObjectProperty(R)	$\mathsf{Fun}(R^-)$	$\langle x,y\rangle \in R^{\mathcal{I}} \wedge \langle z,y\rangle \in R^{\mathcal{I}} \Rightarrow x = z$
ReflexiveObjectProperty(R)	$\mathsf{Ref}(R)$	$\langle x,x\rangle \in R^{\mathcal{I}}$
IrreflexiveObjectProperty(R)	$\mathsf{Irr}(R)$	$\langle x,x\rangle \notin R^{\mathcal{I}}$
SymmetricObjectProperty(R)	$\mathsf{Sym}(R)$	$\langle x,y\rangle \in R^{\mathcal{I}} \Rightarrow \langle y,x\rangle \in R^{\mathcal{I}}$
AsymmetricObjectProperty(R)	$\mathsf{Asy}(R)$	$\langle x,y\rangle \in R^{\mathcal{I}} \Rightarrow \langle y,x\rangle \notin R^{\mathcal{I}}$
TransitiveObjectProperty(R)	$\mathsf{Tra}(R)$	$\langle x,y\rangle \in R^{\mathcal{I}} \wedge \langle y,z\rangle \in R^{\mathcal{I}} \Rightarrow \langle x,z\rangle \in R^{\mathcal{I}}$
SubDataPropertyOf(S,R)	$S \sqsubseteq R$	$S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
EquivalentDataProperties$(R_1\ \ldots\ R_n)$	$R_i \equiv R_j, i,j \in \{1,\ldots n\}$	$R_i^{\mathcal{I}} = R_j^{\mathcal{I}}, i,j \in \{1,\ldots n\}$
DisjointDataProperties$(R_1\ \ldots\ R_n)$	$\mathsf{Dis}(R_1,\ldots,R_n)$	$R_i^{\mathcal{I}} \cap R_j^{\mathcal{I}} = \emptyset, i,j \in \{1,\ldots n\}, i \neq j$
DataPropertyDomain$(R\ C)$	$\geq 1R \sqsubseteq C$	$\langle x,y\rangle \in R^{\mathcal{I}} \Rightarrow x \in C^{\mathcal{I}}$
DataPropertyRange$(R\ D)$	$\top \sqsubseteq \forall R.D$	$\langle x,y\rangle \in R^{\mathcal{I}} \Rightarrow y \in D^{\mathcal{I}}$
FunctionalDataProperty(R)	$\mathsf{Fun}(R)$	$\langle x,y\rangle \in R^{\mathcal{I}} \wedge \langle x,z\rangle \in R^{\mathcal{I}} \Rightarrow y = z$
DatatypeDefinition$(T\ D)$	$T \equiv D$	$T^{\mathcal{I}} = D^{\mathcal{I}}$
HasKey$(C\ (R_1\ \ldots\ R_n)\ (S_1\ \ldots\ S_m))$	n/a	$a,b \in C^{\mathcal{I}}$ a,a_i,b,b_i named individuals
with R_i object properties		$\wedge\langle a,a_i\rangle \in R_i^{\mathcal{I}} \wedge \langle b,b_i\rangle \in R_i^{\mathcal{I}}$
and S_i data properties		$\wedge\langle a,d_i\rangle \in S_i^{\mathcal{I}} \wedge \langle b,e_i\rangle \in S_i^{\mathcal{I}} \Rightarrow a = b$
SameIndividual$(a_1\ \ldots\ a_n)$	$a_i \doteq a_j, i,j \in \{1,\ldots,n\}$	$a_i^{\mathcal{I}} = a_j^{\mathcal{I}}, i,j \in \{1,\ldots n\}$
DifferentIndividuals$(a_1\ \ldots\ a_n)$	$a_i \neq a_j, i,j \in \{1,\ldots,n\}, i \neq j$	$a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}, i,j \in \{1,\ldots n\}, i \neq j$
ClassAssertion$(C\ a)$	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
ObjectPropertyAssertion$(R\ a\ b)$	$R(a,b)$	$\langle a^{\mathcal{I}},b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$
NegativeObjectPropertyAssertion$(R\ a\ b)$	$\neg R(a,b)$	$\langle a^{\mathcal{I}},b^{\mathcal{I}}\rangle \notin R^{\mathcal{I}}$
DataPropertyAssertion$(R\ a\ d)$	$R(a,d)$	$\langle a^{\mathcal{I}},d^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$
NegativeDataPropertyAssertion$(R\ a\ d)$	$\neg R(a,d)$	$\langle a^{\mathcal{I}},d^{\mathcal{I}}\rangle \notin R^{\mathcal{I}}$

- The first one specifies a set of namespace prefix, defined by the keyword `PREFIX`.
- The second one defines the query form with four different types:
  - `SELECT` query form is used to return variables and their bindings in a query pattern match.
  - `CONSTRUCT` query form builds an RDF graph constructed by replacing variables in a set of triple templates.
  - `DESCRIBE` query form returns the description of query pattern resource found.
- The third one is the `WHERE` clause which initiates the actual query, i.e. *query graph pattern*, containing a BGP and enclosed in curly braces.

We consider an example of the structure of SPARQL query illustrated in Figure. 2.3:

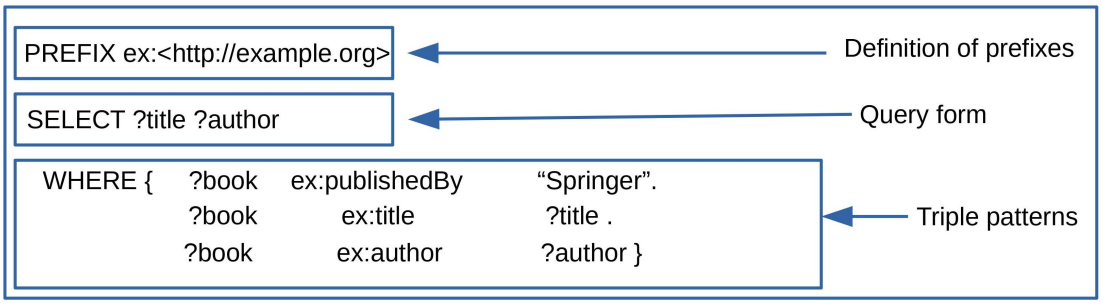


Figure 2.3: An example of the structure of a SPARQL query

This query comprises a BGP of 3 triple patterns defining different query patterns and implicit conjunction operations. The results of applying this query will be the *title*, the *author* for each book published by "*Springer*" in the RDF dataset.

Along with BGPs, SPARQL allows us to build more *complex graph patterns*(CGPs) combining multiple BGPs to construct various query graph pattern of `WHERE` clause:

- The query graph pattern can be a union of BGPs: `P1 UNION P2`, where `P1, P2` are BGPs.

- The query graph pattern have optional BGPs which are not required but they can allow the results of mandatory patterns, i.e. patterns without `OPTIONAL` operators, to be extended with additional information: `P1 OPTIONAL P2`, where `P1, P2` are BGPs.
- The graph pattern contains several restrictions on the patterns results by using operator `FILTER`: `P FILTER (E)`, where `P` is a BGP and `E` are restriction expressions.

Also, SPARQL provides several operators to modify the query results:

- `DISTINCT` operator is used to eliminate duplicate solutions from the solution set.
- `REDUCED` operator is used to permit duplicate results to be eliminated.
- `ORDER BY` operator is used to order the solutions following to a set of expression and an optional order modifier, i.e. either ascending by `ASC()` or descending by `DESC()`.
- `OFFSET` operator is used to split all solutions into a number of solution subsets with a specific size.
- `LIMIT` is used to give the largest number of solutions to be allowed to return.
- `REGEX` operator is used to match a text against a regular expression.

In addition, SPARQL can use a protocol, e.g. HTTP protocal, for transferring queries towards remote servers, i.e. *SPARQL endpoints*, and returning a set of solutions.

3
Literature Review

Contents

3.1 Ontology learning 32
3.1.1 Ontology learning in the LOD context 32
3.1.2 Learning techniques 33
3.2 Schema-level axiom learning 36
3.2.1 Property Axiom Learning 36
3.2.2 Class Axiom Learning 37
3.3 RDF Mining 37
3.4 Summary 39

As the main research topic of this thesis dissertation is to learn new ontological knowledge, i.e., schema-level knowledge, for KBs of LOD from their RDF datasets themselves, in this chapter we provide an overview of the research directions concerning learning ontological knowledge from LOD, in particular, from RDF data. In the first part (Section 3.1), we offer a view of the learning of ontologies, in particular, in the context of LOD and then adopt recent popular learning techniques. In Section 3.2, we review recent various studies of learning separately schema-level axioms as a little step for enriching the entire ontology. In Section 3.3, we survey the studies concerning mining RDF data for new knowledge.

3.1 Ontology learning

In reality, the development (construction or enhancement) of ontologies is an attracting research problem which concerns the activities of knowledge acquisition from various sources (human knowledge, diverse data sources or different existing knowledge sources, etc.). However, this process is limited by the obstacles arising from the requirement of involving domain experts and knowledge engineers, which is highly expensive and time-consuming. These obstacles known as *knowledge acquisition bottleneck* [LV14a] may be tackled by the set of methods and techniques that go under the name of *ontology learning*.

Ontology learning [MS01; MS04] is the field of research aiming to automatically extract formal schema information from scratch or from existing ontologies. Methods and techniques in ontology learning, by adopting learning algorithms from several existing knowledge and information sources, can help alleviate the overall cost of ontology construction by reducing or eliminating altogether the efforts of domain experts. Ontology learning may be viewed as a special case of knowledge discovery from data (KDD) or data mining in which ontological elements (conceptual knowledge) are extracted from data and an ontology is constructed from them.

3.1.1 Ontology learning in the LOD context

An interesting research direction considered as a little step in ontology learning involves the tasks of *ontology alignment and matching*. These tasks concern the process of determining correspondences between concepts in independent ontologies of LOD. Specifically, they include finding schema alignments [Jai+11; ZI11; SAS11; SE11; Hec+15; SK17] or finding alignments between concepts defined as disjunctions of conjunctions of (RDF) types and value restrictions [PKA12] when links between the datasets of LOD are almost exclusively on the level of instances and schema-level information is being ignored [Jai+10].

In addition, due to the lack of sophisticated schemata, the focus of ontology learning spins around enriching schemata for existing KBs known as knowledge base enrichment (or ontology enrichment). Having such schemata allows more powerful

querying, consistency checking and debugging as well as improved inference [BL13b]. Ontology enrichment is a sub-discipline of ontology learning which typically uses already existing data, i.e., RDF datasets, as input to detect hidden schemata, i.e., schema-level axioms, to refine an existing ontology. This process of enrichment begins with the *learning step* concerning the discovery of schema-level axioms before performing the *placement step* in terms of finding the appropriate place to use them in the original ontology [Ido+16]. This thesis only focuses on the former step involving learning schema-level axioms.

3.1.2 Learning techniques

Inductive Logic Programming (ILP)

ILP [MR94] techniques characterize the combination of machine learning and logical programming in which logic programs are derived from examples (i.e., assertions) and background knowledge. In the ILP setting, background knowledge consists of logical formulations and examples classified into positive and negative examples. The relationship between a hypothesis and an example, whereby the example provides evidence supporting the hypothesis is encompassed in the definition of *coverage*. Specifically, a hypothesis $\mathcal{H}$ covers an example $\mathcal{E}$ with respect to the background $\mathcal{B}$ if $\mathcal{B} \bigwedge \mathcal{H} \models \mathcal{E}$ where '$\models$' is the symbol of logical entailment. The aim of ILP is to find hypotheses ($\mathcal{H}$) covering all positive examples($\mathcal{E}^+$), i.e., $\mathcal{B} \bigwedge \mathcal{H} \models \mathcal{E}^+$, and not covering any negative example ($\mathcal{E}^-$), i.e., $\mathcal{B} \bigwedge \mathcal{H} \not\models \mathcal{E}^-$, with respect to a given background knowledge (B). In general, ILP based approaches obey the model of inductive reasoning that build general conclusions from specific instances, assuming that the latter exemplify a general truth.

Along with the adoption of OWL and $\mathcal{DL}$ in knowledge representation, ILP based methods with DLs settings have successfully been applied to learning concepts and their descriptions. For this purpose, a comprehensive collection of algorithms developed by Jens Lehmann et al. is included in the DL-Learner system [Leh09], which provides a framework for learning concepts and their descriptions in description logics and OWL, whereby learned concepts are used to construct and maintain OWL

ontologies or to solve problems similar to those in ILP. The component of learning algorithm in DL-Learner is based on a combination of using top-down refinement operators for the most fundamental DLs $\mathcal{ALC}$ [LH08] and a search algorithm, i.e., genetic programming. [LH10] extended DL-Learner with a concept learning algorithm based on refinement operators for the DLs $\mathcal{ALCQ}$ including support for concrete roles. The aim of DL-Learner is to find concepts and their descriptions covering as many positive examples while only applying to as few as possible negative examples. For this purpose, refinement operators are used to explore the search space of possible concept descriptions following the structure of a tree where child nodes are representing concept descriptions that are more specific than the class expression of their parent nodes. In order to reduce the number of steps required to find the final results, a heuristic was used to guide the search. A big obstacle of ILP-based approaches in DL-Learner is its dependency on reasoning techniques, which is hardly applicable to the very large KBs like LOD. One temporary solution was proposed in [HLA09] to increase the scalability of OWL learning algorithm on very large KBs through intelligent pre-processing. Instead of considering the complete knowledge bases, a knowledge fragment selection procedure was applied to select a piece of relevant knowledge that holds enough information to induce good results and allow efficient reasoning. Another prominent system, DL-FOIL [FdE08], developed a method derived from the FOIL algorithm [Qui90] (used to learn Horn clauses from data expressed as relations) to learn concept descriptions expressed in expressive DLs supporting OWL-DL. Its main components are represented by a set of refinement operators derived from other systems like DL-Learner and by a different gain function involving the OWA. A revised algorithm implemented in a new release of DL-FOIL [Fan+18] adapted the original approach by exploiting a refinement operators and a heuristic to select among candidate specializations the one that is able to better approximate a target concept. In addition, ILP-based methods are successfully investigated on learning *onto-relational rules* that complement and extend ontologies on the Semantic Web [Lis12] or learning *multi-relation association rules* in SWRL that may suggest new axioms at schema level [dAm+16].

In general, although ILP-based approaches perform very well in the generation of highly axiomatized ontologies, they are less scalable and robust when they need to handle the huge data of LOD. One of the reasons stems from their dependency on reasoning techniques. In addition, most ILP-based approaches are supervised, which requires determining positive and negative examples. This faces several obstacles when working on semantic KBs in OWA where the absence of instances can not be used as negative examples either.

Statistic-based Methods

Statistic-based methods are merely based on instance data in the repository. The difference of most statistic-based techniques with respect to ILP-based techniques is that they do not rely on reasoning tasks to be performed on the instances of a knowledge base, but involve data mining approaches. These research directions have been mentioned in the collection of works developed by Volker et al., which focus on mining the Semantic Web to enrich the schema of ontologies [Ret+12]. A classic statistic-based approach was proposed on *statistical correlation analysis* to learn disjointness axioms [FV11]. Later approaches employed *statistical schema induction* (SSI) via Association Rule Mining (ARM) to learn concept-centric [VN11; VFS15] and property-centric [FVS12] axioms from association rules. Specifically, instance data contained in LOD are translated into transaction tables. Association rules are discovered from these databases via ARM methods which point out certain conditions to hold in the data. These approaches rely on the assumption that the data contained in the RDF repository obeys the rules of a schema whose semantics are found in the patterns of its usage in the repository, i.e. the Closed World Assumption (CWA). This approach focuses on weakly expressive languages. The restrictions of this approach are its CWA and mutual interactions between discovered axioms, since they are induced independently.

3.2 Schema-level axiom learning

In ontology learning, enriching schema-level knowledge for RDF knowledge bases (KBs) published in LOD concerns axiomatizing the concepts and relationships to induce different types of *schema-level axioms*. Learning these axioms is also one of the critical tasks in the entire ontology learning which is called *schema-level axiom learning*. In the SW, schema-level axioms (also called *conceptual axioms*) are classified into class axioms and properties axioms possibly represented in OWL (explained in Section 2.2.3, corresponding to TBox and RBox axioms in DLs, respectively (explained in Section 2.1.1). Subsumption or equivalence axioms can be derived from the Horn rules mined on large RDF knowledge bases by AMIE [Gal+13] and its extension AMIE+ [Gal+15]. Similarly, role transitivity, symmetry, role/concept subsumption axioms can be suggested from multi-relation rules discovered from assertional knowledge given by RDF data [dAm+16; dTT16; Tra+17]. [SPS17] detected the evolution of axioms concerning four types of OWL axioms, namely, *DataPropertyAxiom*, *ObjectPropertyAxiom*, *ClassAxiom* and *HasKey* based on the needed part of the RDF data changed (updated or modified) which is relevant to not only extracting additional axioms for the ontology but also defining axioms needed to be deleted from the existing ontology. A recent work [LLS16] proposed a parallel mining of OWL2 RL axioms from LOD.

3.2.1 Property Axiom Learning

An increasing amount of research concentrates on discovering different types of property axioms. [FVS12] applied statistical schema induction via association rule mining to discover property axioms (subsumption, disjointness, transitivity, domain, range, symmetry, asymmetry, inverse, functionality, inverse functionality, reflexivity and irreflexivity) from RDF data. Also, some of works concentrate on discovering simple subsumption axioms used for the hierarchy organization in the ontology and equivalent axioms aiding for matching and aligning across ontologies mentioned above. PARIS [SAS11] proposed a probabilistic model for discovering equivalent predicates (i.e., equivalence properties axioms) across two datasets based

on estimating the overlap between instances of two properties in the datasets. [KPV17] proposed a supervised machine learning approach for relation alignment, called SORAL, based on the overlap of instances across disparate linked datasets and different schemas to discover subsumption and equivalent property axioms.

Concerning the problem of defining the domain and range of properties used in multi-context, [Ton+15] induce domain and range restrictions from RDF data used to improve the correctness of domain and range in LOD. The occurrences of an original property can be replaced by using the new sub-properties corresponding to particular contexts. Similarly, Topper et al. [TKS12] proposed a method based on the class types of the instances in the subject and object of a property to suggest the domain and range of properties in LOD.

3.2.2 Class Axiom Learning

Similarly to property axioms, a large number of approaches concerns discovering class subsumption axioms [SAS11; GPS13] and specialized form with respect to restriction classes [VN11]. Considered as a pillar type of axioms characterizing the rich expressiveness and ensuring the quality of ontologies, discovering disjointness axioms between two classes is increasingly getting attention. A set of tools developed by Volker et al. [VHC07; FV11; VFS15], specifically, are *LeDA* [FV11], acquiring disjointness by supervised machine learning, and *GoldMiner*, [VFS15] using unsupervised machine learning to automatically extract class disjointness axioms. Other works include Redescription Mining (RM) [RTN19] based on mining alternate descriptions from two datasets related to the same set of individuals, in order to discover definitions of classes (i.e. equivalent axioms) and incompatibility (i.e., disjointness axioms) between classes or [Riz+17] which provided an unsupervised approach to disjointness learning based on terminological cluster trees.

3.3 RDF Mining

Under the different point of view of the field of KDD and data mining, ontology learning from RDF data of LOD can be regarded as "*RDF mining*" with the

data of RDF triples. RDF mining is a close relative to the fields of Linked Data mining [RBP15] and Semantic Web mining [Ret+12; QS13; TRa16], where Linked Data and Semantic Web data, including also themselves RDF triples, are used as the input for mining patterns and knowledge. RDF mining generally involves the discovery of meaningful patterns and correlations within RDF data, which is achieved via various data mining techniques.

A considerable number of research involving *Association Rule Mining* (ARM) [AS94] aims at finding out frequent patterns or interesting relations between variables from given datasets using some measures of interestingness. Accordingly, association rules (ARs) will satisfy the prior minimum support and confidence from a given dataset. An ARM problem can be divided into two tasks: (i) finding itemsets whose occurrences exceed a prior threshold in the datasets; those itemsets are called *frequent* itemsets; (ii) extracting ARs from those frequent itemsets under some constraints of minimal confidence. A group of ARM works [NB10; BBL17; Gal+13; Gal+15; TEE20] was proposed in the Semantic Web which concerns mining Semantic Web data (including RDF data). [NB10] proposed a method to efficiently extract items and transactions suited for traditional association rules mining algorithms. SWARM [BBL17] extracted common behavioural patterns associated with knowledge at both the instance-level and schema-level, i.e., semantic ARs. A similar work [TEE20] concerning the use of schema-level knowledge in the mining process is the extraction of ARs in the medical field based on ontology-based Apriori algorithm. In addition, AMIE [Gal+13] proposed a formal model for rule mining under the OWA with a novel measure to simulate counterexamples thanks to the *partial completeness assumption* (PCA) and a scalable algorithm for the faster mining. An extension of AMIE, known as AMIE+ [Gal+15], improves its performance by adding a series of pruning and query rewriting techniques that are used to discover Horn rules on large RDF knowledge bases. Claudia d'Amato et al. [dAm+16; dTT16; Tra+17] also proposed two algorithms, namely a level-wise generated-and-test algorithm and an evolutionary algorithm, to discover multi-relational association rules encoded in the Semantic Web rule language

(SWRL), by exploiting the evidence coming from the assertional data in KBs (i.e., RDF data). Some association rules mined from published RDF data can be exploited for the later creation of schema-level knowledge for ontologies like rules possibly translated into OWL 2 EL subsumption and equivalence axioms [Gal+13; Gal+15] or transitivity and symmetry of a role, and/or concept/role inclusion axioms [dAm+16; dTT16; Tra+17].

One notable point is that RDF data can also be regarded as an oriented, labeled multi-graphs. As a consequence, there have been efforts in mining RDF graphs. This research direction focuses on using data mining methods for extracting complex graph patterns known as *graph pattern mining* from RDF graphs. Mined graphs [Zha+12; Ram+05; Bas+10] characterize *instance-level data* but do not support any inferential mechanism on data. For example, [Zha+12] allows to detect interesting associations between elements in RDF graphs or [Ram+05; Bas+10] detect informative structures within RDF graphs. In addition, there are a few studies that refer to mining different types of restrictions axioms based on knowledge graphs like [Pot20], involving the extraction of cardinality restrictions in order to extend an ontology describing the graph or [EHC16], inducing independent domain and range restrictions as well as coupled domain/range restrictions from an RDF graph. However, the focus of this thesis is not on the graph structure of the knowledge, i.e., *knowledge graph* using the RDF formalism but on its *semantics* and on the *logical schema-level knowledge*.

3.4 Summary

In this chapter, we provided a broader view of ontology learning in the context of LOD. Specifically, we studied recent learning and mining techniques from RDF data. Also, we reviewed recent works concerning learning various types of schema-level axioms. In reality, recent approaches focus on the use of *inductive learning* methods for discovering various ontological knowledge from the linked data itself, thus avoiding a manual creation. In addition, each ontology learning approach outperforms others in different aspects. There is no approach that covers all

of the following capabilities: handling the massive data of LOD, using highly expressive languages like semantic OWL, not requiring the supervision of domain experts, handling uncertainty of data under the OWA, etc... In this thesis we refer to the model of inductive learning of rich representations, along with the ability of handling massive datasets.

4

Learning OWL Axioms From RDF data

Contents

4.1 Introduction 41
4.2 Grammatical Evolution 44
4.2.1 Grammar-mediated Representation 44
4.2.2 Evolutionary Process 47
4.2.3 Terminology 48
4.2.4 Grammatical Evolution Implementations 49
4.3 Grammatical Evolution to Search for OWL Axioms . . 50
4.3.1 Grammar Construction 50
4.3.2 An Evolutionary Model to Search for OWL Axioms . . 53
4.4 Summary 59

4.1 Introduction

With respect to the target of our research, we adopt the discovery of general OWL axioms, which can be considered as a generalization of all the learning targets from RDF data. The efforts towards discovering knowledge from RDF data may be regarded as a form of *inductive reasoning*, in that it proceeds from specific instances of concept and relations (RDF triples) to broader generalizations (OWL 2 axioms).

In Machine Learning (ML), a computer system can be viewed as an inductive machine which is a device used to perform inductive inferences, i.e., *inductive*

learning. At this time, ML provides the theoretical and practical framework within which the task of inductive learning from different datasets is addressed algorithmically. The problem we study may be stated in the view of ML as the learning task of axioms from RDF data sources in which axioms are expressed in the form of logical programs in OWL (introduced in Section 2.2.3). This task is viewed as an inductive synthesis of logic programs, which can be found in recent problems of ILP. In terms of ILP learning settings, the *learning from entailments* is the most appropriate for our case. In the context of the imperfection of RDF data repository containing noisy and incomplete data, axioms will be regarded as a tentative explanation of how knowledge may be expanded through the observation of facts. Precisely, the entire RDF repository is considered as a set of interpretations that agree with the facts contained in it.

However, induction in ML does not only consider the inference from observations to induce hypotheses and strive to justify them from the test but also includes the search for hypotheses in a large set of possibilities [Ber91]. In this sense, axiom induction is also regarded as a search problem, whereby a hypothesis space is traversed to find plausible axioms. Thus, the settings of learning algorithms are essentially to select an effective searching algorithm. In reality, the simplest approach is based on *deterministic* generate-and-test methods, which essentially perform an exhaustive search. Nevertheless, such methods are computationally too expensive to deal with massive datasets [RR01]. Several proposals coupled with heuristic pruning or syntactic biases have been used to handle the complexity of the search in structuring and traversing the hypothesis space, but they limit the search exploration and may give rise to the problem of local optima. Although the expressive power of the representation is taken into account in ILP, more powerful search methods are required for handling the large search space and for inducing more complex axioms. Among the techniques proposed to ensure scalability in the context of an ever expanding volume of RDF triples, Evolutionary approach (EA), i.e., Evolutionary Computation (EC) inspired by natural selection, is a potential search solution for this purpose. Along with global optimization capabilities, EA is less sensitive to

local optima and can adapt with both symbolic and numerical data. However, in the traditional EA like in Genetic Algorithms (GA), the representation of knowledge is syntactically restricted. Although Genetic Programming (GP) [Koz93; Lan+08] allows the exploration of computer programs, it encounters the *closure problem* that requires the validation of generated programs. The trade-off between the expressiveness of representation and the performance of the search has been the subject of numerous works [Div06; DM05; Div10; TM00; TM02]. These approaches aiming to combine EC with ILP have been investigated to alleviate expensive computation arising from inductive learning of rich representations.

Another line of research is the grammar-based GP methods that are developed on the ideas of declarative description for the search space which are represented in traditional GP or grammatical biased ILP [Coh94]. Specifically, grammars are used to guide the formation of a hypothesis in the form of programs or to direct the search for programs in the hypothesis space, i.e. the search bias. In particular, context-free grammars were used in [Whi95; Whi96] to control the search algorithm of GP, known as context-free grammar genetic programming (CFG-GP). Another combination of GP and ILP based on a formalism of a logic grammar in LOGENPRO [MK95b; MK95a] was proposed to induce logic programs from imperfect data. Instead of using any specific algorithms of ILP, it imitates the mechanism of logic programming to describe the grammar, but does not possess any characteristics completely concerning logic programming environment.

A recent idea of using grammar-based GP also known as Grammatical Evolution (GE) was invented by Michael O'Neill et al. [RCO98; OR01]. Instead of changing the paradigm of traditional tree-based GP as in LOGENPRO, GE uses the mapping process from genotype to phenotype. The grammars are designed to encode domain knowledge in any language, e.g. programs, whereas the search is itself driven by traditional evolution.

In this chapter, we first explore the fundamental characteristics of GE introduced in Section 4.2. Then, the deployment of GE to develop a general model in terms of automatically discovering OWL axioms from RDF datasets of LOD is introduced in

Section 4.3. In it, we investigate how to formulate axioms in OWL language based on the grammar in addition to an evolutionary search of OWL axioms.

4.2 Grammatical Evolution

Grammatical Evolution (GE for short) [OR01; DOB09; RCO98] is known as a relatively new Evolutionary Computation (EC for short) technique pioneered by Michael O'Neill and his collaborators. It is a grammar-based form of *Genetic Programming* (GP for short) [Koz92; VP12] that allows the exploration of the space of computer programs and differs from traditional GP in that it distinguishes the *search space* from the *solution space*, through the use of *a grammar-mediated representation*. At the starting point of the GE process, known as an *initialization*, a population consisting of individuals maintained within the search space are randomly initialized, which are then translated into corresponding programs based on a given grammar. The generated programs are then "bred" using iterative improvement of a population of programs, known as an *evolutionary process*. This process will stop when it meets the *termination criterion*. An illustration of the GE mechanism is shown in Figure 4.1

4.2.1 Grammar-mediated Representation

In GE, the search space contains variable-length binary strings, i.e., *genotypic individuals* or *genotypes*, which are decoded to generate programs (hypotheses), viewed as *phenotypic solutions* or *phenotypes* in the solution space through a transformation called *mapping process*. According to it, the variable-length binary string genomes, or *chromosomes*, are split into consecutive groups of bits, called *codons*, representing an integer value, used to select, at each step, one of a set of production rules, from a formal grammar, typically in *Backus-Naur form (BNF)*, which specifies the syntax of the desired programs.

• A *BNF grammar* is a context-free grammar consisting of terminals and non-terminals. A grammar can be represented in the form of a four-tuple $\{N, T, P, S\}$, where

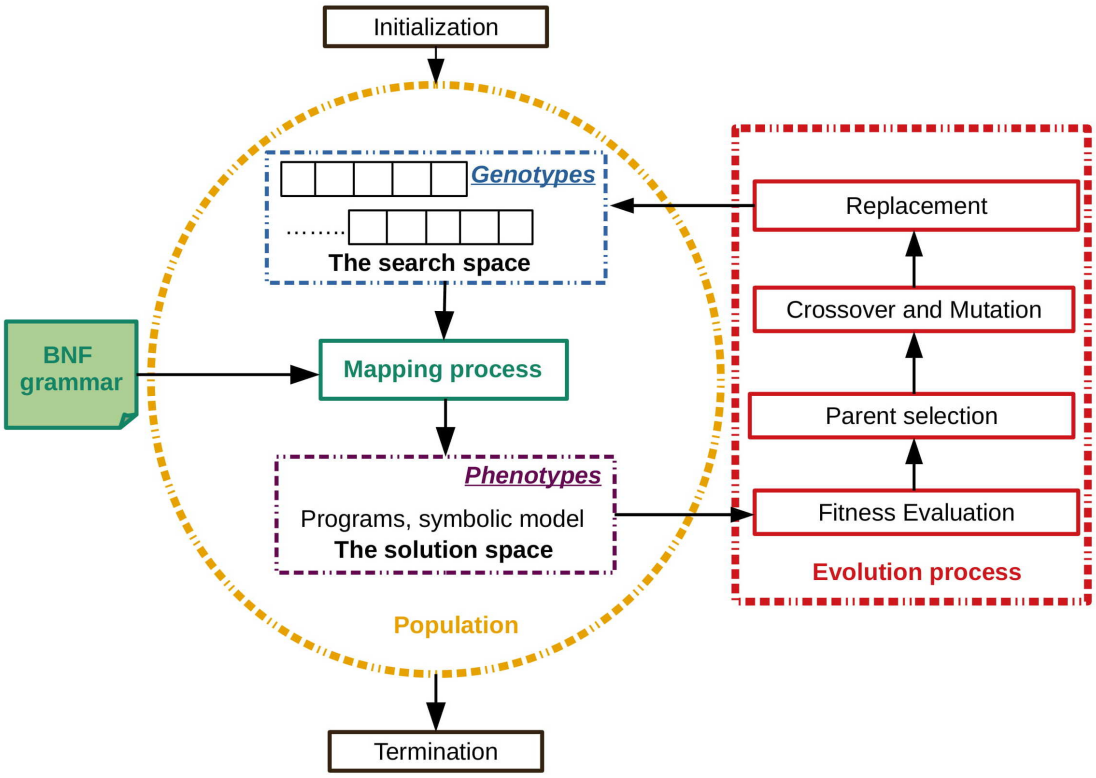


Figure 4.1: Grammatical Evolution mechanism

- N is the sets of non-terminals, which can be extended into one or more terminals;
- T is the set of terminals which are items in the language;
- P is the set of the production rules that map N to T;
- S is the start symbol and a member of N.

When there are a number of productions that can be used to rewrite one specific non-terminal, they are separated by the '|' symbol.

Example 4.2.1 (A sample of BNF grammar) *The tuple {N,T,P,S} of the grammar:*

$N = \{\langle \texttt{sentence} \rangle, \langle \texttt{subject} \rangle, \langle \texttt{predicate} \rangle, \langle \texttt{direct-object} \rangle, \langle \texttt{article} \rangle, \langle \texttt{noun} \rangle, \langle \texttt{verb} \rangle\}$
$T = \{\texttt{THE}, \texttt{A}, \texttt{TEACHER}, \texttt{STUDENT}, \texttt{INFORMATICS}, \texttt{STUDIES}, \texttt{TEACHES}, \texttt{WORKS}\},$
$S = \langle \texttt{sentence} \rangle$

The productions in the grammar:

```
(A) <sentence> ::= <subject>    (0)
                 | <predicate>  (1)
(B) <subject> ::= <article>     (0)
                | <noun>        (1)
(C) <predicate> ::= <verb>           (0)
                  | <direct-object>  (1)
(D) <direct-object> ::= <article>   (0)
                  | <noun>      (1)
(E) <article> ::= THE   (0)
                | A     (1)
(F) <noun> ::= TEACHER      (0)
             | STUDENT      (1)
             | UNIVERSITY   (2)
             | INFORMATICS  (3)
(G) <verb> ::= STUDIES        (0)
             | TEACHES        (1)
             | WORKS          (2)
```

• During the *mapping process*, codons are used consecutively to choose production rules from P in the BNF grammar according to the function:

$$production = codon \ \textbf{modulo} \begin{bmatrix}\text{Number of productions for}\\ \text{the current non-terminal}\end{bmatrix} \tag{4.1}$$

Example 4.2.2 (Mapping process) *Given the BNF grammar in the above example 4.2.1, let the chromosome be (16, 20, 355, 420, 506, 25, 81, 106, 682, 48, 125, 67).*

An illustration of a complete mapping process to a symbolic program is shown in Fig 4.2. The mapping process is begun by considering the first rule `(A)` *for non-terminal* $\langle sentence \rangle$ *and decoding the first codon value being read produces the integer* 16. *Following Equation (4.1), the result, i.e* $16 \bmod 2 = 0$, *is used to determine*

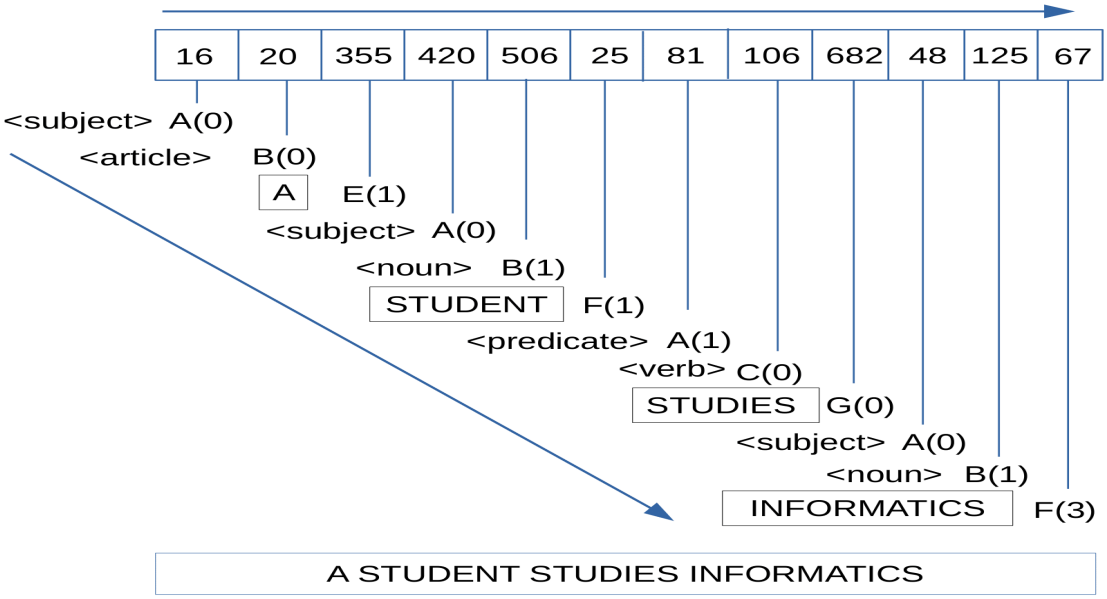


Figure 4.2: An illustration of a mapping process

which production is chosen to replace the leftmost non-terminal ⟨`subject`⟩*. When a production rule is chosen to map from a non-terminal, another codon is read. The genome is traversed consecutively by this procedure until eventually there is no non-terminal left in the expression.*

During the mapping process, it is possible for individuals to run out of codons. In this case, codons can be reused two or more times, a technique called *wrapping* [OR01; DOB09]. A wrapping operator is applied and the reading of codons will repeat from the beginning of the chromosome, until the maximum allowed number of wrapping events is reached. An incomplete mapping will happen when the threshold on the number of wrapping events is reached but the individual is still not completely mapped. Such individual is considered *invalid* and is assigned the lowest possible fitness.

4.2.2 Evolutionary Process

After the initialization of a population of individuals, an evolutionary process is iteratively refined with a series of operations in order to identify a set of highest level of individuals. At each iteration, known as a *generation*, improvements are made

possible by *stochastic variation*, i.e. by a set of *genetic operators* or *variation operators*, usually *crossover* and *mutation* [CJ10; ONe+03] and probabilistic selection according to pre-specified criteria for judging the quality of an individual (solution or phenotypic programs). The *fitness-based selection* of individuals is performed to create a list of better qualified individuals as input for generating a new set of candidate solutions (i.e., offspring) in the next generation. This process of selecting individuals is viewed as *parent selection*. The offspring of each generation is bred by applying genetic operators (crossover and mutations) on the selected parents. *Replacement* or *survival selection* is the last step and decides which individuals stay in a population and which are replaced on a par, with selection influencing convergence.

The evolutionary process repeats until the termination criterion can be met as follows:

- reaching a pre-defined maximum number of generations, i.e. iterations. The best solution in the final generation is considered as the optimal solution.
- meeting the optimal solutions.

In practice, there are many different variants of the algorithmic elements used for GE. However, we are not ambitious in traversing all their variants but only follow some particular variants to solve the problem of axiom discovery, described in Section 4.3.

4.2.3 Terminology

- ***Population*** is a subset of all individuals that are then decoded into programs, i.e., solutions (hypotheses), to the given problem.
- ***Generation*** is a complete cycle corresponding to an iteration step of the evolutionary process which consists of the reproduction and evaluation of individuals.
- ***Codon*** is a consecutive group of 8 bits representing an integer value

- ***Chromosome*** is a variable length binary string that is used to represent individuals. A chromosome is made up of a sequence of codons.

- ***Genotype*** is a solution representation in the search space in which the solution is represented in a way that can be easy for a machine to process and compute.

- ***Phenotype*** is a solution representation in the solution space in which the solution is represented in the form of programs mapped from the individual's genotype through a mapping operation.

- ***Crossover***, known as one type of variation operator, is a form of recombination where two parents (individuals) exchange genes to produce two offspring (new individuals) according to a given probability distribution. Crossover in GE can be applied at the genotypic level as the standard crossover of EA, or at the phenotypic level, like the sub-tree crossover of GP.

- ***Mutation*** is a variation operator which changes the information contained in the genome of a parent according to a given probability distribution.

- ***Fitness*** is an evaluation of the quality of individuals based on a set of objective values representing a function also called *fitness function*. This operation applies directly to the phenotypic solutions.

4.2.4 Grammatical Evolution Implementations

Currently, there exist some publicly available implementations of GE, namely AGE[1], GEVA[2] [ONe+08], PonyGE[3] [Fen+17], gramEvol [NSL16],... In this thesis, we only focus on using GEVA, which is a GE framework developed by the Natural Computing Research & Applications group at University College Dublin. The latest version is 2.0. The framework provides both command line tools and a Java source library in

[1] http //nohejl.nameage/
[2] http://ncra.ucd.ie/SiteGEVA.html
[3] https://github.com/PonyGE/PonyGE2

terms of GE. The advantages of GEVA are the organization of the algorithms in the form of modules which can be combined with one another into pipelines.

4.3 Grammatical Evolution to Search for OWL Axioms

Once presented the GE framework above, we formulate OWL axiom discovery from a given RDF dataset as a GE problem where the definition of "programs" or "phenotypic solutions" in GE are OWL axioms whose syntax is defined by a BNF grammar. The first essential task before performing the GE process is to construct a BNF grammar for structuring well-designed axioms in OWL, explained in Section 4.3.1. A description of the GE framework involving discovering OWL axioms is then presented.

4.3.1 Grammar Construction

The syntax of the logical language from which the axioms are to be generated in OWL is given by a functional style grammar expressed in the extended BNF notation used by W3C [W3Cf]. The grammar specifications of OWL and its fragments are published by W3C in the standard notation obeying the conventions indicated in Table 4.1. However, only a part of productions of the W3C grammar is considered in

Table 4.1: The conventions of W3C grammar notation

Construct	**Syntax**	**Example**
production	:=	Class:=IRI
non-terminal symbol	boldface	**ClassExpression**
terminal symbol	single quoted	'DisjointClasses'
zero or more	curly braces	{ClassExpression}
zero or one	square brackets	[ClassExpression]
alternative	vertical bar	SubClassOf | DisjointClasses
grouping	parentheses	(ClassExpression ' ' ClassExpression)

order to generate OWL axiom. The aim is to generate well-defined axioms describing the facts contained in a given RDF triple store, thus, only resources that actually occur in the RDF dataset should be generated. In order to ensure that the changes

in the contents of RDF repositories will not require to rewrite the grammar, the BNF grammar is organized into two main parts, as *static* and *dynamic* productions.

Static Productions

Static productions are high-level production rules used to define the structure of the axioms and do not depend on the content of RDF repositories. Different static productions will generate different kinds of axioms. Static productions are loaded from a hand-crafted text file. In the case of OWL axioms, the static productions are designed based on the production rules extracted from normative grammar of OWL 2 given in the appendix of W3C as an extended BNF grammar[4]. Following W3C, OWL axioms are divided into eight groups. However, axiom annotations and declarations are not the targets to generate, thus, all symbols and productions related to annotations and declarations have been ignored. Also, since we only consider built-in datatypes and datatypes used in RDF repository, there are no productions relevant to the definition of new datatypes, i.e., datatype definition in the grammar. The target production of axioms is thus alleviated to five categories corresponding to the following static production:

```
Axiom := ClassAxiom | ObjectPropertyAxiom | DataPropertyAxiom | HasKey | Assertion
```

Each category of axioms consists of different types of axioms expressed in the axioms-level productions:

```
ClassAxiom := SubClassOf | EquivalentClasses | DisjointClasses | DisjointUnion

ObjectPropertyAxiom :=  SubObjectPropertyOf | EquivalentObjectPropertyOf
                      | DisjointObjectPropertyOf | ObjectPropertyDomain
                      | ObjectPropertyRange | InverseObjectProperties
                      | FunctionalObjectProperty | InverseFunctionalObjectProperty
                      | ReflexiveObjectProperty | IrreflexiveObjectProperty
                      | SymmetricObjectProperty | AsymmetricObjectProperty
                      | TransitiveObjectProperty
DataPropertyAxiom := SubDataPropertyOf | EquivalentDataPropertyOf
                      | DisjointDataPropertyOf | DataPropertyDomain
                      | DataPropertyRange | FunctionalDataProperty
HasKey := 'Haskey''('ClassExpression'('ObjectPropertyExpression')' '('DataPropertyExpression')'')'
Assertion := SameIndividual | DifferentIndividual | ClassAssertion
           | ObjectPropertyAssertion | NegativeObjectPropertyAssertion
```

[4]https //www.w3.org/TR/owl2syntax/#Axioms

The remaining axiom-level productions are written according to the definition of each type of axioms. With the exception of two assertions of individuals as *SameIndividual* and *DifferentIndividuals*, each type of axioms consists of expressions formulating classes, object properties or data properties. Expressions can contain logical operators such as conjunction, disjunction, etc.

Example 4.3.1 *An instance of the axiom-level productions concerning* ***DisjointClasses****:*

```
DisjointClassess := 'DisjointClasses''('ClassExpression''ClassExpression {''ClassExpression } ')'
```

All productions are related to formulating expressions, data ranges, individuals are categorized to the expression-level productions.

Example 4.3.2 *An instance of the expression-level productions concerning* ***ClassExpression****:*

```
ClassExpression := Class | ObjectIntersectionOf | ObjectUnionOf | ObjectComplementOf | ObjectOneOf
                  | ObjectSomeValuesFrom | ObjectAllValuesFrom | ObjectHasValue | ObjectHasSelf
                  | ObjectMinCardinality | ObjectMaxCardinality | ObjectExactCardinality
                  | DataSomeValuesFrom | DataAllValuesFrom | DataHasValue | DataMinCardinality
                  | DataMaxCardinality | DataExactCardinality
```

Dynamic Productions

Dynamic productions are production rules for the low level non-terminals, which we called *primitives*. These primitives can be:

- Class - the IRI of a class mentioned in the RDF store, including owl:Thing.
- ObjectProperty - the IRI of an object property used in the RDF store.
- DataTypeProperty - the IRI of a data type property used in the RDF store.
- NamedIndividual - the IRI of an individual appearing in the RDF store.
- Literal - any literal apprearing in the RDF store

These production rules are automatically built at run-time by querying the SPARQL endpoint of the RDF repository at hand as follow:

- For the `Class` primitive:

```
SELECT DISTINCT ?class WHERE { ?subj a ?class }
```
(4.2)

- For the `ObjectProperty` primitive, we select the properties whose fillers are individual, i.e. RDF resources represented by IRIs:

```
SELECT DISTINCT ?prop WHERE { ?subj ?prop ?obj.
FILTER ( isIRI(?obj)) }
```
(4.3)

  To include the properties whose fillers are blank nodes, the filter has to be changed into `isIRI(?obj)) || isBlank(?obj))`

- For the `DataProperty` primitive, we select the properties whose values are literals:

```
SELECT DISTINCT ?prop WHERE { ?subj ?prop ?obj.
FILTER ( isLiteral(?obj)) }
```
(4.4)

- For the `NameIndividual` primitive, we select all RDF resources as the individuals of classes which appear as the subject x in a triple of the form x `rdf:type` *class*

```
SELECT DISTINCT ?ind WHERE { ?ind a ?class.
FILTER ( isIRI(?ind)) }
```
(4.5)

- For the `Literal` primitive, we select the objects whose values are literals:

```
SELECT DISTINCT ?obj WHERE { ?subj ?prop ?obj.
FILTER ( isLiteral(?obj)) }
```
(4.6)

The results s_1, s_2,...,s_n returned by the above queries for each primitive P are added into the grammar as new productions: P := s_1| s_2 | ... |s_n

4.3.2 An Evolutionary Model to Search for OWL Axioms

This section introduces an evolutionary model on Grammatical Evolution (GE) to mine an RDF repository for axioms. A population of candidate axioms is maintained by a GE algorithm and iteratively refined to find axioms that are both *general* and *credible* (two key quality measures for discovered knowledge). The quality of

the generated axioms can be improved gradually during the evolutionary process by applying standard genetic operators (crossover and mutation) on genotypic axioms. The overall flow of such GE algorithm is shown in Algorithm 1. Our model to axiom learning relies on a quite standard implementation of GE. In particular, we have adopted the reference implementation found in the GEVA framework. In this section, we only focus on different specific adaptations of the standard model to the problem at hand.

Initialization

In order to initialize a population of OWL axioms with the size *popSize*, a set of *popSize* chromosomes, i.e., genotypic individuals, are randomly initialized once and for all (line 2 of Algorithm 1). Each chromosome *Chr* is a set of integers with the initialized length *initlenChrom*. Its length can be extended to the scope of the maximum wrapping times *maxWrapping*. The transformation of genotypes into OWL axioms by means of the mapping process is based on the input BNF grammar *Gr*. The population of axioms is created by iterating *popSize* times the *CreateNewAxiom()* operator described in Algorithm 2.

Algorithm 2: Create_New_Axiom()

Input: *Chr*: Chromosome - a set of integers; *Gr*: BNF grammar
Output: *A*: a new axiom individual

1 $maxlenChrom \leftarrow initlenChrom * maxWrap$
2 $ValCodon \leftarrow random(maxValCodon)$
3 Set up *Chr* as input genotype *gp* used in mapping proccess to axiom *A*
while ($Chr.length \leq maxlenChrom$) && (*incomplete mapping*) **do**
4 | mapping from input genotype *gp* to output phenotype of individual axiom according to grammar *Gr*
5 **return** *A*

Parent Selection

Before executing the parent selection, the axioms in the population are evaluated and ranked in descending order of their fitness. The parent selection mechanism often amounts to choosing the fittest individuals from the population for reproduction.

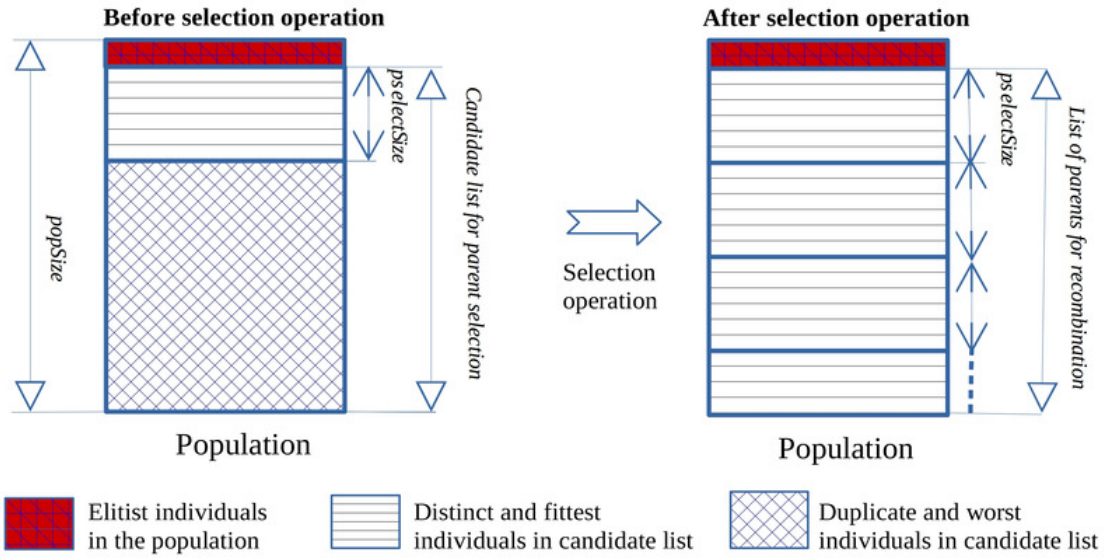


Figure 4.3: An illustration of the parent selection mechanism.

In addition, in order to combat the loss of fittest axioms as a result of the application of the variation operators, an elitism selection can be also applied to copy a small proportion *pElite* of the best axioms into the next generation (line 7-8 of Algorithm 1). In the remaining part of the population, the elimination of duplicates is carried out to ensure only distinct individuals will be included in the *candidate list* for parent selection. Figure 4.3. illustrates the process of selecting potential candidate solutions for recombination following elitism selection, i.e., generating a *list of parents*. The top proportion *pselectSize* of distinct individuals in the candidate list is selected and it is replicated to maintain the size *popSize* of population. The list of parents is shuffled (line 12 of Algorithm 1) and the individuals are paired in order from the beginning to the end of the list.

Variation Operators

- **Crossover**: We consider two standard variants employed for the function $Crossover(parent1, parent2)$ (line 16 of Algorithm 1) in terms of GE, namely single-point crossover [ONe+01] and sub-tree crossover as follows:

  – *Single- point crossover* can be employed in the search space of genotypes, whereby one crossover point on the chromosomes of both parents is chosen

randomly. The sets of codons beyond those points are exchanged between the two parents with probability *pCross*. The result of this exchange is two offspring genotypes. An illustration of a single point crossover mechanism is depicted in Figure 4.4.

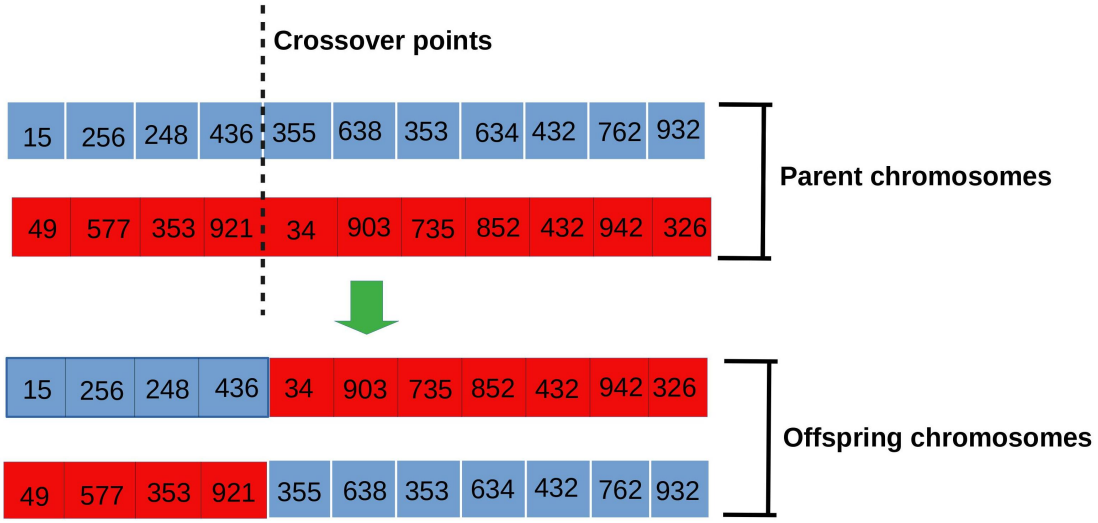


Figure 4.4: An illustration of a single-point crossover

– *Sub-tree crossover*: We also apply another standard crossover based on the trees, whereby parent-genotypes turn into derivation trees. The tree nodes will state which codons each branch uses to determine its child production. Random crossover points are picked up from the chromosomes. The tree-nodes are relevant to the chosen crossover points in the chromosomes. The random point chosen from the second chromosome is only a starting point from which a matching type with the random point of the first one will be searched. The search runs right and left of the point on the second chromosome evenly on both sides to find the nearest type that matches. As the crossover-point is chosen from the list of used codons, this will always find the node in the tree, otherwise, this will find the tree-node nearest to the chosen crossover point that has the same type. The crossover swaps selected sub-trees of the same type in parent-trees with probability *pCross* to create two offspring-trees. The offspring trees are then serialized back to the original representation of genotypes.

- **Mutation**: a standard single- point mutation in $Mutation(offspring)$ operator (line 17 of Algorithm 1) is applied to the offspring genotypes of crossover with probability *pMut*. In the selected individual for the mutation, a codon is selected at random, then is replaced with a new randomly generated codon.

After carrying out variant operators (crossover and mutation), well-formed individuals will then be generated syntactically from the new genotypes in the genotype-to-phenotype mapping process. Specifically, the transformations from offspring genotypes into phenotypic axioms in OWL are performed by executing the *Create_New_Axiom()* operator (Algorithm 2) again multiple times with the offspring chromosomes as input.

Replacement

In order to preserve population diversity and prevent premature convergence, a variant using the *Crowding method* is embedded in the survival selection. In terms of the properties of the Crowding method, there are two main steps, namely *repairing* and *replacing*. In the repairing phase, new individuals, i.e. offsprings, are paired with individuals in the current population, i.e. parents, according to a similarity metric. Specifically, the distances between parents and children in a family are computed. In the replacement phase, each offspring competes with its most similar peers to be selected for inclusion in the population of the next generation. Specifically, we follow the representative Crowding method called *Deterministic Crowding (DC) method* [Mah92] developed by Mahfoud. In DC method, the replacement rule is *deterministic*, and always picks the individuals with the higher fitness scores. Algorithm 3 describes this approach in detail.

Algorithm 3: Crowding (parent1, parent2, offspring1, offspring2)

Input: *parent1, parent2, child 1, child 2*: a crowd of individual axioms
Output: *A*: *A*: *ListWinners*- a list containing two winners of individuals

1 $d1 \leftarrow Distance(parent1, child1) + Distance(parent2, child2)$
2 $d2 \leftarrow Distance(parent1, child2) + Distance(parent2, child1)$
3 **if** $d1 > d2$ **then**
4 $\quad ListWinners[0] \leftarrow Compare(parent1, child1)$
5 $\quad ListWinners[1] \leftarrow Compare(parent2, child2)$
6 **else**
7 $\quad ListWinners[0] \leftarrow Compare(parent1, child2)$
8 $\quad ListWinners[1] \leftarrow Compare(parent2, child1)$
10 **return** $ListWinners$

The $Distance(parent, child)$ operator (line 1-2 of Algorithm 3) define the distinct between individuals in the pair $(parent, child)$ which can be defined for *genotypic* or *phenotypic distance*. According to this, the genotypic distance between individuals can be quantified as the *Hamming distance* [Ham50], which is much faster and easier to compute. On the other hand, computing distance at the phenotypic level can be based on the computation of *Levenshtein distance (Edit distance)* [Lev66] with the expectation of obtaining more accurate results. The $Compare(parent, child)$ operator (line 3-4 and 6-7 of Algorithm 3) defines which individual in the pair of $(parent, child)$ has the higher fitness value.

Fitness Evaluation

Fitness evaluation is determined in different quality criteria. The relations between different quality measurements are represented through a *fitness function* used to quantify the fitness of individuals. In general, a high fitness indicates that an axiom is meaningful (general and accurate) , thus, a fitness function is used for scoring axioms through at least two metrics of generality and accuracy, which are based on the *evidence* available in the form of a set of facts contained in a chosen RDF dataset, known as *axiom testing*. A detailed description of various evaluation frameworks for candidate OWL axioms will be introduced in next Chapter 5.

4.4 Summary

This chapter provides a formal GE framework for discovering OWL axioms. It formally defines a learning model from the perspective of the GE approach. Chapter 6 and Chapter 7 later will specialize this framework and incorporate it into a multi-objective optimization framework, respectively, to apply to learning OWL class disjointness axioms.

Algorithm 1: GE for discovering axioms from an RDF dataset

Input: T: RDF triples data; Gr: BNF grammar; $popSize$: the size of the population; $initlenChrom$: the initialized length of chromosome; $maxWrap$: the maximum number of wrapping; $pElite$: elitism propotion; $pselectSize$: parent selection propotion; $pCross$: the probability of crossover; $pMut$: the probability of mutation.

Output: Pop: a set of axioms discovered based on Gr

1 Initialize a list of chromosomes L of length $initlenChrom$. Each codon value in chromosome is an integer.
2 Create a population P of size $popSize$ mapped from list of chromosomes L on grammar Gr by iterating *Create_New_Axiom()* described in Algorithm 2
3 Compute the fitness values for all axioms in Pop.
4 Initialize current generation number: $currentGeneration = 0$
5 **while** $currentGeneration < maxGenerations$ **do**
6 Sort Pop by descending fitness values
7 Create a list of elite axioms $listElites$ with the propotion $pElite$ of fittest axioms in Pop
8 Add all axioms of $listElites$ to a new population $newPop$
9 Select the remaining part of population after elitism selection: $Lr \leftarrow Pop \backslash listElites$
10 Eliminate the duplicates in Lr
11 Create a list of axioms $listCrossover$ used for crossover operation with the propotion $pselectSize$ of the number of the fittest individuals in Lr
12 Shuffle $listCrossover$
13 **for** $(i = 0, 1....listCrossover.length - 2)$ **do**
14 $parent1 \leftarrow listCrossover[i]$
15 $parent2 \leftarrow listCrossover[i + 1]$
16 $child1, child2 \leftarrow Crossover(parent1, parent2)$ with the probability $pCross$
17 **for each** $offspring$ $\{child1, child2\}$ **do** $Mutation(offspring)$
18 Compute fitness values for $child1, child2$
19 Select $w1, w2$ - winners of competition between parents and offspring
20 $w1, w2 \leftarrow Crowding(parent1, parent2, child1, child2)$
`/*` $Crowding(parent1, parent2, child1, child2)$ `is described in Algorithm 3 */`
21 Add $w1, w2$ to new population $newPop$
22 $Pop = newPop$
23 Increase the number of current generation by 1: $curGeneration++$
24 **return** Pop

5
Axiom Evaluation

Contents

5.1 Introduction 61
5.2 OWL Axiom Testing 63
5.2.1 General Principles 63
5.2.2 The Computational Definitions of Evidences 67
5.3 Axiom Scoring Frameworks 76
5.3.1 Probabilistic Evaluation of Axioms 77
5.3.2 Possibilistic Evaluation of Axioms 79
5.4 Summary 85

5.1 Introduction

Along with the discovery of OWL axioms, there exists a need for evaluating the quality of discovered axioms. On the one hand, the evaluation of induced axioms can be verified by looking at their interactions with the background knowledge (TBox) and also mutual interactions between them. Some logical quality measures proposed in [SSB15; SS17] involve this evaluation model as follow:

- *consistency*: verifying whether axioms are consistent with the background knowledge.

- *redundancy*: verifying whether axioms are redundant with respect to the existing knowledge.
- *logical strength*: verifying whether a set of axioms is weaker (more general) than another set of axioms.
- *dissimilarity*: measuring how "dissimilar" axioms are with respect to the TBox. The more dissimilar they are, the more interesting the axioms are for the TBox.
- *complexity*: measuring the complexity of TBox added by the new axiom, compared to the old TBox, by quantifying how many entailments the new TBox has.

In addition, the quality of generated axioms also depends on the data, which can be incorrect, noisy and incomplete. In reality, in [SSB15; SS17] statistical quality criteria consisting of the *support*, the *coverage*, the *contradiction* were also proposed to measure how well the data in the ABox supporting axioms take the background knowledge in the TBox into account. However, the evaluation models based on the mentioned quality measures critically rely on the reasoning mechanism which would be expensive. Also, the background knowledge (terminology) can be incomplete, thus, using it to evaluate new induced axioms respecting the OWA can lead to misleading results. Hence, ignoring the use of reasoning and the existence of prior knowledge, the validation process of candidate axioms should consist of automatically checking whether they fit or explain the available RDF repository, known as *axiom testing against RDF data* [TFG17], which would provide a cheap and scalable assistance. This approach corresponds to the *hypothetico-deductivism*[1] method of hypothesis testing which is used to assess hypotheses in the light of empirical data, i.e., RDF data. According to this approach, evidence e confirms a hypothesis, i.e., an axiom, h if the latter entails it, i.e $h \models e$, and dis-confirms it if the former entails the negation of the latter, i.e. $e \models \neg h$.

[1] https://www.britannica.com/science/hypothetico-deductive-method

In addition, *confirmation* and *falsification* are strategies for testing hypotheses and describing the results of those tests. More specifically, confirmation is the act of using evidence to verify that a hypothesis is true or approximately true, whereas falsification is the act of defining that a hypothesis is false in the light of observations. In terms of confirmation versus falsification, one of the most influential and controversial views was given by Karl Popper (1902-1994) [Pop35], which seems particularly well-suited to the context of axiom induction from incomplete RDF repositories. In his view, all theories are hypotheses and may be overthrown. More important, he proposed the principle of *falsifiability*, in which all scientific knowledge, i.e., theories, is provisional, conjectural, hypothetical, and we can merely (provisionally) confirm or (conclusively) refute them. Back to our problem of axiom induction, we can consider axioms as conjecturing hypotheses that can potentially be refuted by facts contained in the RDF repository known as *contradicting evidence*, i.e., *counterexamples*. The facts recorded in the RDF triples satisfying axioms are known as *supporting evidence*, i.e. *confirmations*. In epistemology, the term "*confirmation*"is used whenever observational data and evidence are "in favor of" or support hypotheses. Testing a single axiom involves checking whether the formulas entailed by it are confirmed or falsified by the facts contained in RDF datasets. Furthermore, the validation of an axiom also requires an axiom scoring framework computed based on the available evidence and a set of measures.

In this section, we first investigate the principles of testing a single OWL axiom agianst a given RDF dataset introduced in Section 5.2. Then, we consider two axiom scoring frameworks, based on probability and possibility theory in Section 5.3, which are the bases to develop the functions for evaluating the fitness of discovered axiom, as it will become clear in the following chapters.

5.2 OWL Axiom Testing

5.2.1 General Principles

In order to test an axiom against a given RDF dataset, Tettamanzi et al. [TFG17] proposed the definition of the *development of an OWL axiom* with respect to

a RDF dataset developed on Hempel's proposal in terms of the *development of hypothesis*, whereby an OWL axiom can be translated into a first order logic (FOL) formula used for querying the RDF dataset later. The development of OWL axiom ϕ with respect to RDF dataset $\mathcal{K}$ is the formula $D_{\mathcal{K}}(\phi)$, such that $\phi \models D_{\mathcal{K}}(\phi)$ is defined recursively, whose notion relies on a transformation $t(., x, y)$ into a FOL formula based on the set-theoretic formulas of OWL direct semantics (introduced in Tables 2.5, 2.6). The authors gave the definition of a transformation which is relevant to the development of an axiom. In particular, $t(\phi, x, y)$ is the translation from axiom ϕ into a FOL formula which is recursively defined in such a way that the resulting formulae involves two variables x, y. The translation of different types of axioms into FOL is depicted in Table 5.1.

Example 5.2.1 *Let an OWL axiom* $\phi =$`SubClassOf(dbo:Fish dbo:Animals)`. *The transformation of* `SubClassOf` *axiom can be defined following the principle of [TFG17] as follows:*

- $t(C_1 \sqsubseteq C_2, x, y) = \forall x(\neg t(C_1, x, y) \vee t(C_2, x, y))$, *where the first argument is axiom* $C_1 \sqsubseteq C_2$ *with* C_1, C_2 *as classes(concepts).*
- $t(C, x, y) = C(x)$, *where the first argument is entity* C *as an atomic class (concept).*

As a result, the transformation of axiom ϕ *into FOL formula is defined as follows:*

$t(\phi; x, y) =$

$t($`SubClassOf(dbo:Fish dbo:Animals)`$, x, y) =$

$\forall x(\neg t($`dbo:Fish`$, x, y) \vee t($`dbo:Animals`$, x, y)) =$

$\forall x(\neg($`dbo:Fish`$(x) \vee ($`dbo:Animals`$(x))$

The development of axiom ϕ with respect to RDF dataset $\mathcal{K}$, noted by $D_{\mathcal{K}}(\phi)$, can be transformed either into *conjunctive* normal form or into *disjunctive* normal form of the ground formulas ψ_i, i.e. $D_{\mathcal{K}}(\phi) = \bigwedge_i \psi_i$ or $D_{\mathcal{K}}(\phi) = \bigvee_i \psi_i$, respectively. These ground formulas ψ_i are called *basic statements*, which are tested against the available facts in RDF data.

Table 5.1: Translation of OWL axioms into FOL [TFG17]

OWL axioms (Functional-style)	**DL syntax**	**Translation into FOL**
$SubClassOf(C\,D)$	$C \sqsubseteq D$	$t(C \sqsubseteq D; x, y) = \forall(x(\neg t(C; x, y))$
$EquivalentClasses(C\,D)$	$C \equiv D$	$t(C \equiv D; x, y) = \forall(t(; x, y) \wedge (D; x, y) \vee (\neg t(C; x, y) \wedge \neg t(D; x, y)))$
$DisjointClasses(C_1...C_n)$	$Dis(C_1, ..., C_n)$	$t(Dis(C_1, ..., C_n); x, y) = \bigwedge_{i=1}^{n} \bigwedge_{j=i+1}^{n} (\neg t(C_i; x, y) \vee \neg t(C_j; x, y))$
$DisjointUnion(C\ C_1...C_n)$	$C \equiv C_1 \sqcup ... \sqcup C_n$ and $Dis(C_1, ..., Cn)$	$t(C \equiv C_1 \sqcup ... \sqcup C_n,\ Dis(C_1, ..., Cn); x, y) = t(C \equiv C_1 \sqcup ... \sqcup C_n; x, y) \wedge t(Dis(C_1, ..., Cn); x, y))$
$SubObjectPropertyOf(S\,R)$	$S \sqsubseteq R$	$t(S \sqsubseteq R; x, y) = \forall x \forall y(\neg t(S; x, y) \vee t(R; x, y))$
$SubObjectPropertyOf(nR)$, with $n = ObjectPropertyChain(S_1...S_n)$	$S_1...Sn \sqsubseteq R$	$t(S_1...Sn \sqsubseteq R; x, y) = \forall x \forall z_1 ....\forall z_{n-1} \forall y(\neg t(S_1; x, z_1) \vee \neg t(S_2; z_1, z_2) \vee ... \vee \neg t(S_n; z_{n-1}, y) \vee t(R; x, y)$
$EquivalentObjectProperties(R_1...R_n)$	$R_1 \equiv R_2$	$t(R_1 \equiv R_2; x, y) = \forall x \forall y((t(R_1; x, y) \wedge t(R_2; x, y)) \vee (\neg t(R_1; x, y) \wedge \neg t(R_2; x, y)))$
$DisjointObjectProperties(R_1...R_n)$	$Dis(R_1...R_n)$	$t(Dis(R_1...R_n; x, y)) = \bigwedge_{i=1}^{n} \bigwedge_{j=i+1}^{n} (\neg t(R_i; x, y) \vee \neg t(R_j; x, y))$
$ObjectPropertyDomain(R\,C)$	$\geq 1R \sqsubseteq C$	$t(\geq 1R \sqsubseteq C; x, y) = \forall x \forall y(\neg t(R; x, y) \vee t(C; x, y))$
$ObjectPropertyRange(R\,C)$	$\top \sqsubseteq \forall R.C$	$t(\top \sqsubseteq \forall R.C; x, y) = \forall x \forall y(\neg t(R; x, y) \vee t(C; y, z))$
$InverseObjectProperties(S\,R)$	$S \equiv R^-$	$t(S \equiv R^-; x, y) = \forall x \forall y((t(S; x, y) \wedge t(R; y, x)) \vee (\neg t(S; x, y) \wedge \neg t(R; y, x)))$
$FunctionalObjectProperty(R)$	$Fun(R)$	$t(Fun(R); x, y) = \forall x \forall y \forall z(\neg t(R; x, y) \vee \neg t(R; x, z) \vee y = z)$
$InverseFunctionalObjectProperty(R)$	$Fun(R^-)$	$t(Fun(R); x, y) = \forall x \forall y \forall z(\neg t(R; x, y) \vee \neg t(R; z, y) \vee x = z)$
$ReflexiveObjectProperty(R)$	$Ref(R)$	$t(Ref(R); x, y) = \forall x(t(R; x, x))$
$IrreflexiveObjectProperty(R)$	$Irr(R)$	$t(Irr(R); x, y) = \forall x(\neg t(R; x, x))$
$SymmetricObjectProperty(R)$	$Sym(R)$	$t(Sym(R); x, y)) = \forall x \forall y(\neg t(R; x, y) \vee t(R; y, x))$
$AsymmetricObjectProperty(R)$	$Asy(R)$	$t(Asy(R; x, y)) = \forall x \forall y(\neg t(R, x, y) \vee \neg t(R; y, x))$
$TransitiveObjectProperty(R)$	$Tra(R)$	$t(Tra(R); x, y) = \forall x \forall y \forall z(\neg t(R; x, y) \vee t(R; x, z))$
$DataPropertyRange(R\,D)$	$\top \sqsubseteq \forall R.D$	$t(\top \sqsubseteq \forall R.D; x, y) = \forall x \forall y(\neg t(R; x, y) \vee t(D; y, z))$
$DataTypeDefinition(R)$	$T \equiv D$	$t(T \equiv D; x, y) = \forall x((t(T; x, y) \wedge (t(D; x, y)) \vee (\neg t(T; x, y) \wedge \neg t(D; x, y)))$
$HasKey(C(R_1, ..., R_n)$ with R_i object properties	$Key(C) = \{R_1, ..., R_n\}$	$t(Key(C) = \{R_1, ..., R_n\}; x, y) = \forall x \forall z \forall z_1 ...\forall az_n(\neg t(C; x, y) \vee t(C; z, y) \vee$ $\bigvee_{i=1}^{n} (\neg R_i(x, z_i \vee \neg R_i(z, z_i)) \vee x = z)$
$SameIndividual(a\,b)$	$a \doteq b$	$t(a \doteq b; x, y) = (a = b)$
$DifferentIndividuals(a\,b)$	$a \neq b$	$t(a \neq b; x, y) = \neg(a = b)$
$ClassAssertion(C\,a)$	$C(a)$	$t(C(a); x, y) = C(a)$
$ObjectPropertyAssertion(R\,a\,b)$	$R(a, b)$	$t(R(a, b); x, y) = R(a)$
$NegativeObjectPropertyAssertion(R\,a\,b)$	$\neg R(a, b)$	$t(\neg R(a, b); x, y) = \neg R(a)$
$DataPropertyAssertion(R\,a\,b)$	$R(a, d)$	$t(R(a, d); x, y) = R(a)$
$NegativeDataPropertyAssertion(R\,a\,b)$	$\neg R(a, d)$	$t(\neg R(a, d); x, y) = \neg R(a)$

As we can observe in example 5.2.1, the axiom translates into the FOL formula $\forall x(\neg(\texttt{dbo:Fish}(x) \vee (\texttt{dbo:Animals}(x))$ and is developed according to the RDF dataset $\mathcal{K}$ into: $D_{\mathcal{K}}(\phi) = \bigwedge_{r \in I(\mathcal{K})}(\neg(\texttt{dbo:Fish}(r) \vee (\texttt{dbo:Animals}(r))$ where $I(\mathcal{K})$ is the set of resources (individuals) occurring in $\mathcal{K}$.

The set of all basic statements ψ_i of $D_{\mathcal{K}}(\phi)$ is defined as the *content* of axiom ϕ, which serves as the foundation of axiom testing.

Definition 5.2.1: Content of an Axiom [TFG17]

Let ϕ be an OWL axiom that we wish to test (i.e., theory). Let $\mathcal{K}$ be a RDF dataset. The content of an axiom ϕ is defined as a set of logical consequences $D_{\mathcal{K}}(\phi)$ obtained through the instatiation of ϕ to the vocabulary of $\mathcal{K}$:

$$content(\phi) = \{\psi : \phi \models \psi\}$$

The content of axiom ϕ in example 5.2.1 can be expressed as follows:

$content_{\mathcal{K}}(\phi) = \{\neg\texttt{dbo:Fish}(r) \vee \texttt{dbo:Animals}(r)\} : r$ is a resource occurring in $\mathcal{K}$.

Definition 5.2.2: Confirmation and Counterexample of an Axiom

Let ψ be a formula in the content of axiom ϕ with respect to a given RDF dataset $\mathcal{K}$, i.e. $\psi \in content_{\mathcal{K}}(\phi)$.

- ψ is a confirmation of axiom ϕ if $\mathcal{K} \models \psi$.
- ψ is a counterexample of axiom ϕ if $\mathcal{K} \models \neg\psi$.
- ψ is neither a confirmation nor a counterexample of axiom ϕ if $\mathcal{K} \not\models \psi$ and $\mathcal{K} \not\models \neg\psi$.

In order to refine ψ for the content of axiom ϕ, the concept of *selective confirmation*, proposed by Scheffer and Goodman [SG72], is applied. Selective confirmation of a hypothesis involves an evidence which not merely confirms the hypothesis but also dis-confirms its contrary. The definition of $content(\phi)$ restricts to ψ which can be counterexamples of ϕ and leaves out simpliciter confirmation. In addition, depending on the form of the axiom, the definition of $content(\phi)$ refined by the principle of selective confirmation is different. For example, in the case of $DisjointClasses(C\ D)$ axiom, all ψ involving the occurrence of a resource r will be confirmations when there is at least $\mathcal{K} \not\models C(r)$ or $\mathcal{K} \not\models D(r)$ or both,

i.e. $\mathcal{K} \not\models \psi$ and $\mathcal{K} \not\models \neg\psi$. Such ψ is only *simpliciter* confirmation and there is no *selective* confirmation in the case of *DisjointClasses* axiom. Therefore, the presence of confirmations of *DisjoinClasses* axiom is not very interesting and necessary in the content of $DisjointClasses(C\,D)$. As an example in the case of $SubClassOf(C\,D)$, all ψ involving $\mathcal{K} \not\models C(r)$ (if $\mathcal{K} \models D(r)$) or confirmation $\mathcal{K} \not\models \psi$ and $\mathcal{K} \not\models \neg\psi$ will be trivial confirmations, i.e. not selective ones, and should be left out of the content $SubClassOf(C\,D)$. As a result, $content(\phi)$ and the number of ψ that need to be checked are greatly lessened.

In order to quantify the notions in terms of the content, confirmations and counterexamples of an axiom, some concepts were also introduced in [TFG17] as follows:

- The *support* of an axiom ϕ is defined as the cardinality of the content of ϕ:

$$u_\phi = \| \, content(\phi) \, \|$$

- The *number of confirmations* u_ϕ^+ of an axiom ϕ: is defined as the number of basic statements ψ that are satisfied by the RDF dataset (confirmations).

- The *number of counterexamples* u_ϕ^- of an axiom ϕ: is defined as the number of basic statements ψ that are falsified by the RDF dataset (counterexamples).

5.2.2 The Computational Definitions of Evidences

In order to measure the quantities relevant to evidence for an axiom, like the support, the number of confirmations and the number of counterexamples, the translation of the axiom into the corresponding SPARQL queries to count them is performed. Axioms are composed of expressions which will be translated into SPARQL graph patterns before formalizing SPARQL queries for each axiom. We consider the studies proposed by Lorenz Buhmann and Jens Lehmann [BL13a] and Tettamanzi et al. [TFG17] to construct graph patterns that are a direct mapping of the expressions of the OWL axiom considered.

Let `E` be an expression in an axiom ϕ and `x`, `y` be formal parameters which can be the name of a SPARQL variable, a resource identifier, or a literal as value. A mapping

Q(E, x, y) involves transforming OWL expression E into a graph pattern which is used to take the relevant set of all resources occurring in a given RDF repository. Q(E, x, y) is defined depending on the type of expression E depicted in Table 5.2. The query

```
SELECT DISTINCT ?x ?y
WHERE {
 Q(E, ?x, ?y)
}
```
(5.1)

returns the extension of expression E which is equivalent to the semantics of E, i.e. $E^{\mathcal{I}}$.

However, the table does not contain the case of concept of negation Q(¬C, ?x, ?y) which is slightly more complicated due to the absence of negation expressed in RDF. Hence, we will consider three definitions with respect to the concept negation:

1. The graph pattern proposed by Buhmann and Lehmann, which is used for the CWA, in which the negation is considered as a failure:

   Q(¬C, ?x, ?y) = ?x ?p ?o. FILTER NOT EXISTS Q(C, ?x, ?y)} (5.2)

2. Another graph pattern is proposed for the case of open-world semantics, in which the concept negation Q(¬C ?x ?y) consists of all individuals x of the concepts that are disjoint from C:

   ```
   Q(¬C, ?x, ?y) = { ?x a ?dc)                          (5.3)
                     FILTER NOT EXISTS {
                     ?z a ?dc.
                     Q(C, ?z, ?y1)}}.
   ```

   In this case, testing the disjointness of two concepts is required but it is based on negation as failure as in the first option.

3. One way to define whether two concepts are disjoint is to find a disjointness axiom involving classes in the ontology, i.e. owl:disjointWith. In this sense, the concept negation Q(¬C, ?x, ?y) will be defined as:

Table 5.2: Formal relationships between class expressions and their SPARQL queries

Expression		**SPARQL Graph Partern**
Atomic Expression	Atomic Concept	$\mathtt{Q(A, ?x, ?y)} = \mathtt{?x}\ \ a\ \ \mathtt{A}$. where A is a valid IRI.
	Simple Relation	$\mathtt{Q(S, ?x, ?y)} = \mathtt{?x\ \ S\ \ ?y}$. where S is a valid IRI.
	Datatype	$\mathtt{Q(D, ?x, ?y)} = \mathtt{?x\ \ ?p\ \ ?y.\ \ FILTER\ \ (datatype(?x) = D)}$. where D is a valid datatype IRI
Complex Expression	Inverse Relation (ObjectInverseOf)	$\mathtt{Q(R^-, ?x, ?y)} = \mathtt{Q(R, ?x, ?y)}$ where R is a simple relation
	Extensional Concept (ObjectOneOf)	$\mathtt{Q(\{a_1, .., a_n\}, ?x, ?y)} = \mathtt{?x\ ?p\ ?o.\ \ FILTER\ ?x\ IN\ (a_1, .., a_n)}$ where $\mathtt{a_i}$ is a valid IRI with $\mathtt{i = 1...n}$.
	Extensional Concept (DataOneOf)	$\mathtt{Q(\{d_1, .., d_n\}, ?x, ?y)} = \mathtt{?x\ ?p\ ?o.\ \ FILTER\ ?x\ IN(d_1, .., d_n)}$ where $\mathtt{d_i}$ is a literal with $\mathtt{i = 1...n}$.
	Intersection (ObjectIntersectionOf)	$\mathtt{Q(C_1 \sqcap ... \sqcap C_n, ?x)} = \mathtt{Q(C_1, ?x)...Q(C_n, ?x)}$ where $\mathtt{C_i}$ is a class expression with $\mathtt{i = 1...n}$
	Union (ObjectUnionOf)	$\mathtt{Q(C_1 \sqcup \ldots \sqcup C_n, ?x, ?y)} = \mathtt{\{Q(C_1, ?x)\}\ UNION\ \cdots\ UNION\ \{Q(C_n, ?x)\}}$. where $\mathtt{C_i}$ is a class expression with $\mathtt{i = 1...n}$
	Existential Restriction	$\mathtt{Q(\exists R.C, ?x, ?y)} = \mathtt{Q(R, ?x, ?z1)}\ \ Q(C, \mathtt{?z1, ?z2})$ where C is a class expression and R is a simple relation.
	Value Restriction	$\mathtt{Q(\forall R.C, ?x, ?y)} = \{\ \mathtt{Q(R, ?x, ?z0)}$ $\mathtt{FILTER\ NOT\ EXISTS\ \{Q(R, ?x, ?z1)}$ $\mathtt{FILTER\ NOT\ EXISTS\ \{Q(C, ?z1, ?z2)\}\}\}}$. where C is a class expression and R is a simple relation.
	Number Restriction	$\mathtt{Q(\Theta nR.C, a, ?y)} = \{\ \mathtt{SELECT\ \ ?z0\ WHERE\ \{}$ $\mathtt{BIND\ \ (a\ AS\ ?z0)}$ $Q(R, \mathtt{?z0, ?z1})$ $Q(C, \mathtt{?z1, ?z2})\}$ $\mathtt{GROUP\ BY\ ?z0}$ $\mathtt{HAVING\ (count(DISTINCT\ \ ?z1)\ n)\}}$. $\mathtt{Q(\Theta nR.\top, a, ?y)} = \{\ \mathtt{SELECT\ \ ?z0\ WHERE\ \{}$ $\mathtt{BIND\ \ (a\ AS\ ?z0)}$ $Q(\mathtt{R, ?z0, ?z1})\}$ $\mathtt{GROUP\ BY\ ?z0}$ $\mathtt{HAVING\ (count(DISTINCT\ \ ?z1)\ n)\}}$. where C is a class expression, R is a simple relation and $\Theta \in \{\leq, =, \geq\}$

$$\texttt{Q}(\neg\texttt{C}, \texttt{?x}, \texttt{?y}) = \{\ \texttt{?x a ?dc}) \\ \texttt{?dc owl:disjointWith C }\}. \tag{5.4}$$

In term of the real extension of ¬C, i.e., $\neg\texttt{C}^{\mathcal{I}}$, the first case presented in Equation (5.2) will regard all individuals a for which "a a C" is not found in RDF repository as the instances of $\neg\texttt{C}^{\mathcal{I}}$. In this sense, the extension of $\neg C^{\mathcal{I}}$ is overestimated. For the second case, the extension of $\neg C^{\mathcal{I}}$ only consists of all individuals a such that "a a C" is not known, but there is some classes D for which "b a D" is known and no instance of D is known to be also an instance of C. The extension of C is still overestimated but is reduced significantly. The third option is restricted to atomic concepts which directly appear in the graph pattern, i.e., C, instead of in the form of Q(C ?x ?y), thus, it cannot be extended to complex concepts. Indeed, if there does not exist any declaration of disjointness axioms in the RDF repository, Q(¬C ?x ?y) in Equation (5.4) returns an empty set, which might underestimate the extension of ¬C, i.e., $\neg\texttt{C}^{\mathcal{I}}$. Also, an individual is an instance of C even though it does not belong to any disjoint class with C. Among the above three options, we refer to the second one with Equation (5.3) which is suitable in the context of an open-world dataset, e.g. DBpedia, not too optimistic as in Equation (5.2) and not too pessimistic as in Equation (5.4).

Regardless of the kind of axioms, the support u_ϕ of an axiom ϕ composed of a set of concept expressions or relation expressions $E_1, ..., E_n$ is defined in the following SPARQL query:

$$\texttt{SELECT(count (DISTINCT ?x) } AS \texttt{ ?u)} \\ \texttt{WHERE \{}Q(E_1, \texttt{?x}) \texttt{ UNION } \cdots \texttt{ UNION } Q(E_n, \texttt{?x})\texttt{\}.} \tag{5.5}$$

Conversely, there are various patterns of SPARQL queries for each type of axioms to measure the quantities supporting evidence (confirmations) u_ϕ^+ and refuting evidence (counterexamples) u_ϕ^- stored in the RDF triple repository. We will consider the semantics of different axioms introduced in Table 2.6 as the basis for defining evidence and developing specific computational definitions for testing them.

As for defining evidences for the group of subsumption axioms, the general principle for this axiom group is the following: let be E_{sub}, E_{super} the extension of the subsumed expression and the subsuming expression, respectively, which are retrieved by executing the relevant SPARQL queries. Let be $E^{\neg}_{sub}$, $E^{\neg}_{super}$ the extension of their negated expressions. As said in the previous section, in terms of confirmations, we are interested in resources as evidence in favor of a hypothesis including only selective confirmations. In this sense, resources are individuals x such that $x \notin E^{\neg}_{sub}$ and $x \notin E^{\neg}_{super}$ or $x \notin E_{sub}$ will not be treated as confirmations. Also, resources $x \in E_{sub}$ such that $x \notin E^{\neg}_{super}$ will be not counterexamples because there might be missing assertions such that $x \in E_{super}$ in the incomplete RDF repository. The confirmations and counterexamples in the case of subsumption axioms will be as follows:

- confirmations are individuals x such that $x \in E_{sub}$ and $x \in E_{super}$
- counterexamples are individuals x such that $x \in E_{sub}$ and $x \notin E^{\neg}_{super}$

Then, according to [TFG17] the transformation into the corresponding SPARQL queries is carried out to count the confirmations and counterexamples as follows:

```
SELECT (count (DISTINCT ?x) AS ?numberOfConfirmations)
WHERE {
 Q(C, ?x, ?y)
 Q(D, ?x, ?y)
}
```
(5.6)

and

```
SELECT (count (DISTINCT ?x) AS ?numberOfCounterexamples)
WHERE {
 Q(C, ?x, ?y)
 Q(¬D, ?x, ?y)
}
```
(5.7)

, respectively.

In this thesis, we are not ambitious to cover the computational definitions for all kinds of axioms. Instead, we only set out to develop a general testing model for the case of *disjointness axioms* which will be mentioned below and will be applied in practice in the next chapters. Intuitively, expressions in a disjointness axiom will be divided into left-hand side and right-hand side whose extensions are noted E_{rs} and E_{ls}, respectively. The confirmations and counterexamples are defined as follow:

- confirmations are those individuals x such that either $x \in E_l s$ and $x \in E_{rs}^{\neg}$ or $x \in E_{ls}^{\neg}$ and $x \in E_{rs}$.
- counterexamples are those individual x such that either $x \in Ers$ and $x \in E_{ls}$.

Counting the counterexamples of a disjoint axiom $DisjointClasses(C_1, C_2, ..., C_n)$ will be done simply with the following conjunctive SPARQL query:

```
SELECT (count (DISTINCT ?x) AS ?numberOfCounterexamples)
WHERE {
 Q(C_1, ?x, ?y)
 Q(C_2, ?x, ?y)
 .                                                        (5.8)
 .
 .
 Q(C_n, ?x, ?y)
}
```

However, counting the confirmations is more complicated in open-world RDF data. In general, defining the confirmation for $DisjointClasses(C_1, C_2, ..., C_n)$ is to find all the individuals $\mathtt{x}$ for which an assertion $\mathtt{C}_i(\mathtt{x})$ with $i \in 1...n$ is in the RDF repository but $x^{\mathcal{I}} \notin \mathtt{C}_j^{\mathcal{I}}$ with $j \in 1...n$, $j \neq i$. The definition of $x^{\mathcal{I}} \notin \mathtt{C}_j^{\mathcal{I}}$ corresponds to the definition of the negation $x^{\mathcal{I}} \in \neg\mathtt{C}_j^{\mathcal{I}}$ in the open-world dataset mentioned in Equation (5.3). In this sense, confirmations are only instances that do not belong to class $\mathtt{C}_j$ independently of the class $\mathtt{C}_i$ to which instances belong. The disjointness of $\mathtt{C}_j$ and $\mathtt{C}_i$ is illustrated in Figure 5.1, whereby an instance $\mathtt{x}$ belongs only to one

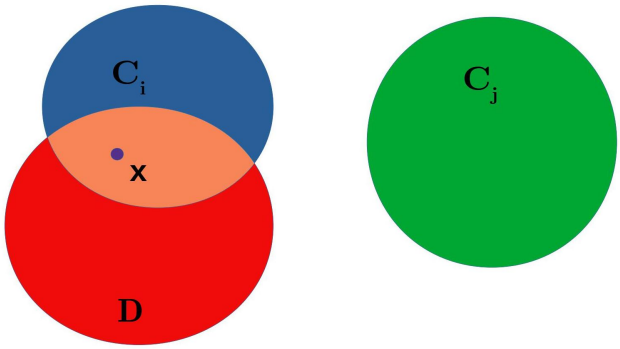


Figure 5.1: A schematic illustration of the disjointness of two classes $\mathtt{C}_i$ and $\mathtt{C}_j$

of the supposedly disjoint classes $\mathtt{C}_i$ and $\mathtt{x}$ can belong to a class $\mathtt{D}$ that is not a sub-class of $\mathtt{C}_i$ and does not share any common instance with $\mathtt{C}_j$, $j \neq i$.

Based on the above definition of confirmations, counting the confirmations for a disjointness axiom involving two concepts $\mathtt{C}_i$ and $\mathtt{C}_j$, i.e., $DisjointClasses(C_i\, C_j)$, is translated into the SPARQL query as follows:

```
SELECT (count (DISTINCT ?x) AS ?numberOfConfirmations) WHERE {
    {
      Q(C_i, ?x, ?y)
      ?x  a  ?dc1)
      ?z1  a  ?dc1)
      Q(¬C_i, ?z1, ?y1)
      FILTER NOT EXISTS {
      ?z2 a ?dc1.
      Q(C_j, ?z2, ?y2)
      }
    }
    UNION
    {
      Q(C_j, ?x, ?y)
      ?x  a  ?dc2)
      ?z3  a  ?dc2)
      Q(¬C_j, ?z3, ?y3)
      FILTER NOT EXISTS {
      ?z4 a ?dc2.
      Q(C_i, ?z4, ?y4)
      }
    }
  }
```

(5.9)

We can represent the above query in a simpler format using the following graph pattern:

```
Q_Dis(C_j | C_i, ?x, ?y) = {
                              ?x  a  ?dc1
                              ?z1  a  ?dc1
                              Q(¬C_i, ?z1, ?y1)
                              FILTER NOT EXISTS {
                              ?z2 a ?dc1.
                              Q(C_j, ?z2, ?y2)
                              }
                          }
```

(5.10)

The query is shortened as follows:

```
SELECT (count (DISTINCT ?x) AS ?numberOfConfirmations) WHERE {
    {
      Q(C_i, ?x, ?y)
      Q_Dis(C_j| C_i, ?x, ?y)
    }
    UNION
    {
      Q(C_j, ?x, ?y)
      Q_Dis(C_i|C_j, ?x, ?y)
    }
  }
```

(5.11)

The query for testing a disjointness axiom involving a number of classes (concepts), i.e., $DisjointClasses(C_1\, C_2\, ...C_n)$ with n the number of classes will be generalized as follows:

```
SELECT (count (DISTINCT ?x) AS ?numberOfConfirmations) WHERE {
    {
      Q(C1, ?x, ?y)
      QDis(C2| C1, ?x, ?y)
      .
      .
      .
      QDis(Cn| C1, ?x, ?y)
    }
    UNION
    {
      Q(C2, ?x, ?y)
      QDis(C1| C2, ?x, ?y)
      QDis(C3| C2, ?x, ?y)
      .
      .
      .
      QDis(Cn| C2, ?x, ?y)
    }
    UNION
      .
      .
      .
    UNION
    {
      Q(Cn, ?x, ?y)
      QDis(C1| Cn, ?x, ?y)
      .
      .
      .
      QDis(Cn−1| Cn, ?x, ?y)
    }
  }
```

(5.12)

5.3 Axiom Scoring Frameworks

Scoring an axiom ϕ is defined based on a corpus of evidence computed on the facts stored in RDF repositories of LOD. Yet, as a result of the incompleteness (due to the lack of information) and the noise (due to the heterogeneous and collaborative character) of the LOD, these facts contain *uncertain* and *imprecise*

information items known as *imperfect* information [Sme96]. Accordingly, *imprecision* involves the content of information which is not sufficient to answer a question of interest due to missing or erroneous information. Also, *uncertainty* concerns a property that represents the relation between the world and the statement about the world [Sme96] when it cannot induce a decision about the truth of the statement due to the lack of information, known as *incomplete knowledge*. In order to deal with the representation of imperfect information, there are two basic frameworks: *probability theory* and *possibility theory*. Probability theory is a traditional approach to formalizing the *ontic* uncertainty typical of random processes; as such, it is appropriate for the situations where all evidence is available. On the other hand, possibility theory determines a sort of *epistemic* uncertainty which is well-suited to incomplete knowledge. In this section, we restrict our attention in investigating these two frameworks in terms of scoring discovered axiom from the RDF repository.

5.3.1 Probabilistic Evaluation of Axioms

In axiom scoring method, probability is defined as a logical relation between a proposition (an axiom) and a corpus of evidence in which an axiom is expressed as being probable with respect to current evidence. A method based on the probabilistic estimation to measure the credibility of an axiom is mentioned in the work [BL12] of Buhmann and Lehmann in which experiments are performed to check whether its logical consequences confirmed by the facts stored in the RDF repository. Possible outcomes for each trial are *success* or *failure* which imply the *confirmations* and the *counterexample*, respectively. The probability of success or failure is the same for each observation which is statistically independent. In view of the CWA, the approach expresses that the probability of confirmation of an axiom ϕ can be simply estimated by $\hat{p}_\phi = u_\phi^+ / u_\phi$. According to this, the parameter estimation is performed by a statistical inference with a confidence interval for it.

In this approach, probabilistic measure is merely based on the supporting evidence, i.e., confirmations, and the absence of confirmations induces a failure in the calculation of the confidence interval. This is only proper under the CWA

where the total of the probability of finding confirmations and the probability of finding counterexamples is equal to 1. Yet, as mentioned in Section 5.2.2, there are cases in the incomplete RDF repository, in which some resources should be treated as neither confirmations nor counterexamples. As a result, there is always a non-zero probability for every potential assumption of an axiom. In order to suit for this open-world assumption, in [TFG17] Tettamanzi et al. proposed the correction of the probabilistic scoring method by using the following proportion: $\hat{p}^*_\phi = u^+_\phi/(u^+_\phi + u^-_\phi)$ instead of $\hat{p}$.

The likelihood of an axiom ϕ, i.e., ϕ is true, is computed on a posterior or conditional probability when given evidence e as follow:

$$Pr(\phi|e) = \frac{Pr(e|\phi)Pr(\phi)}{Pr(e)} \tag{5.13}$$

However, evidence may be logical consequence of the content of axiom ϕ with respect to the given RDF repository or it may be not. Hence, the probability $Pr(e)$ is calculated by adding up the condition that axiom ϕ holds which obeys the extended Bayes' theorem as follow [TFG17]:

$$Pr(\phi|e) = \frac{Pr(e|\phi)Pr(\phi)}{Pr(e|\phi)Pr(\phi) + Pr(e|\neg\phi)Pr(\neg\phi)} \tag{5.14}$$

Such conditional probability can be computed at least on the estimation of probabilities as follow:

- the probability that a fact confirming ϕ exists in the repository, i.e., given that axiom ϕ holds.
- the probability that a fact contradicting ϕ exists in the repository in error, i.e. given that axiom ϕ holds.
- the probability that a fact confirming ϕ exist in the repository in error, i.e., given that ϕ does not hold.
- the probability that a fact contradicting ϕ exist in the repository, i.e. given that ϕ does not hold.

Estimating the above probabilities involves *subjective* or, in other words, *qualitative* properties of uncertainty which are linked to the subjective opinion (the prior) about the true value of facts as derived from the available RDF repository. In reality, estimating these probabilities is hard and requires the consideration of all available evidence. Otherwise, a large number of experiments is performed whose results would be hard to generalize [TFG17]. In addition, with respect to incomplete RDF data, the facts contained in RDF repositories, e.g. DBpedia, cannot be representative of all possible facts that could be recorded, unless that RDF stores are the results of a planned and well-designed effort aimed at building a knowledge base providing uniform coverage of a specific domain. Hence, the results of favour cannot increase the probability of a hypothesis, i.e., an axiom. In order to use the number of facts supporting a hypothesis to estimate its probability, we have to make sure that the finite number of recorded facts can be randomly extracted according to a uniform distribution from the infinite number of all facts of the real world. However, adopting a probabilistic approach can fail due to the lack of fullfiled conditions.

5.3.2 Possibilistic Evaluation of Axioms

Derived from the obstacles mentioned in the previous probabilistic model, we turn to consider another heuristic evaluation framework which is expected to be a more promising solution for incomplete RDF datasets. We exploit preliminarily the concept of *degree of verisimilitude* proposed by Popper, which is used to determine the truth-likeness of a hypothesis which will be encountered in the similar idea of the *possibility theory* introduced later. Accordingly, the level of verisimilitude of hypotheses is always defined insufficiently and the finite amount of data cannot confirm any scientific theory, they only can falsify the theory. This means that a hypothesis is more verisimilar when it could be closer to the truth, i.e., it has passed the test even though it is false. In addition, there is a clear distinction between the verisimilitude and the probability of a hypothesis. In which, the results in favour cannot increase the probability of a hypothesis, i.e., an axiom, but only increase the "*degree of corroboration*" or shortly "*corroboration*". This

term "degree of corroboration" here corresponds to the possibilty of an axiom against the attempts to falsify it.

In this section, we first investigate the essential background for introducing the heuristic evaluation model, namely *fuzzy sets* and *possibility theory*, before deploying it in the specific problem of axiom scoring. We harness the ideas from a series of studies of Tettamanzi et al. in terms of possibilistic axiom scoring [TFG15; TFG17] which obtained some promising results in evaluating candidate OWL axioms.

Fuzzy Sets and Possibility theory

- **Fuzzy Sets**

Definition 5.3.1: A Fuzzy Set [DP80; Zim96]

Let U be a *classical set* called the *universe* defined as a collection of generic elements x:

$$U = \{u | u \in U\}$$

A *fuzzy set* F is a subset of U defined by a value set in the real interval [0,1]. A is characterized by ordered pairs:

$$F = \{(x, \mu_F(u)) | u \in U\}$$

in which:

- $\mu_F(u)$ is called a *membership function* of u in F that maps U to the membership space containing points in the interval of [0,1]:

  $$\mu_F(u) : U \to [0, 1]$$

  $\mu_F(u)$ is *the grade of membership* of u in F
- when the membership space contains only two point 0 or 1, F is a non-fuzzy set ,i.e., classic set, and $\mu_F(u)$ is a characteristic function of a classic set.

– *Set theoretic operations for fuzzy sets [Gog73]*:

Given A, B - fuzzy sets.

∗ The membership function $\mu_{A \cap B}(u)$ of the *intersection* $A \cap B$ is defined by

$$\mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u)), \ \forall u \in U \tag{5.15}$$

* The membership function $\mu_{A\cup B}(u)$ of the *union* $A \cup B$ is defined by

$$\mu_{A\cup B}(u) = \max(\mu_A(u), \mu_B(u)),\ \forall u \in U \tag{5.16}$$

* The membership function of the *complement* of a normalized fuzzy set A, $\mu_{\overline{A}}(u)$, is defined by

$$\mu_{\overline{A}}(u) = 1 - \mu_A(u),\ \forall u \in U \tag{5.17}$$

- **Possibility theory**
  - Possibility theory is also an uncertainty theory used for the handling of incomplete information which stands at the *cross-road* between fuzzy set and probability theory [DP93]. It is comparable to probability theory because it is bound on set functions. However, it uses two dual measures, namely *possibility* and *necessity* in order to model available information whereas probability theory uses only one, namely *probability measure* being *additive*.
  - *Possibility distribution*: There is a relation between possibility theory and fuzzy sets expressed in the concept of a *possibility distribution* [Zad99]. According to it, a possibility distribution is defined as a fuzzy restriction, i.e. an elastic constraint, on the values assigned to a variable. Let A be a fuzzy set of a universe of discourse $U = \{u\}$ characterizing membership function μ_A and x be a variable taking values in U. A possibility distribution π_x can be defined as an interpretation of the membership function μ_A of a fuzzy set A by the elastic restriction: $\forall u \in U,\ \pi_X(u) = \mu_A(u)$. In which, $\pi_x(u)$ represents the possibility of x taking value u in U and the membership function $\mu_A(u)$ represents the degree of compatibility of the value u with the fuzzy set A.

Definition 5.3.2: Possibility distribution

A *possibility distribution* on a universal U is a mapping $\pi : U \to [0, 1]$

In terms of the representation of an imperfect knowledge, U stands for a (mutually exclusive) set of states of affairs and a possibility distribution π represents the state of affairs $u \in U$. It is used for distinguishing the plausibility,

i.e. the possibility, of the different states. In addition, it represents a flexible restriction on what is the actual state with the following conventions [DP15]:

  * $\pi(u) = 0$ means that state u (i.e., $x = u$) is totally *impossible*
  * $\pi(u) = 1$ means that state u (i.e., $x = u$) is completely *possible*, i.e. *plausible*, which is said to be *normalized*

– *Possibility and Necessity measures*: denoted by Π and N, respectively, are built from a possibility distribution π. Both measures apply to a subset of states $X \subseteq U$ and are defined as follows:

$$\Pi(X) = \sup_{u \in X} \pi(u) \tag{5.18}$$

$$N(X) = 1 - \Pi(\overline{X}) = \inf_{u \notin X} \{1 - \pi(u)\} \tag{5.19}$$

While the possibility measure of X, $\Pi(X)$, is equivalent to the greatest degree of possibility associated to its elements, the necessity measure of X, $N(X)$ corresponds the impossibility of its complement $\overline{X}$. In others words, $\Pi(X)$ evaluates to what extent X is logically consistent with π, whereas $N(X)$ evaluates to what extent X is certainly implied by π. Some basic properties with respect to possibility and necessity measures are induced by a normalized possibility distribution on a finite universe of discourse U as follows [DP16]:

  * $\Pi(U) = N(U) = 1$ and $\Pi(\emptyset) = N(\emptyset) = 1$
  * $\Pi(X) = 1 - N(\overline{X})$ (duality)
  * $N(X) \leqslant \Pi(X)$
  * $N(X) > 0$ implies $\Pi(X) = 1$
  * $\Pi(X) < 1$ implies $N(X) = 0$
  * Possibility measures satisfy the "maxitivity" property:

$$\Pi(X \cup Y) = \max(\Pi(X), \Pi(Y))$$

  * Necessity measures satisfy the "minitivity" property:

$$N(X \cap Y) = \min(N(X), N(Y))$$

Possibility and Necessity Score of an Axiom

The conjunctive or disjunctive form in the development of an axiom ϕ, $D(\phi)$ (see Section 5.2.1), will influence the use of confirmations and counterexamples to verify and refute an axiom. More specifically, there are two cases as follows:

1. When the development of axiom ϕ, $D(\phi)$, is in *conjunctive* normal form: only one counterexample is enough to falsify an axiom, i.e., axiom ϕ becomes totally *impossible*, regardless of the number of confirmations.

2. When the development of axiom ϕ, $D(\phi)$, is in *disjunctive* normal form: a single confirmation is enough to verify an axiom, i.e., axiom ϕ is completely *possible* or *plausible*, regardless of the number of counterexamples.

These are the basic principles for establishing a set of measures for an axiom in terms of a possibilistic framework, namely *possibility* and *necessity* of an axiom. According to the presence of any evidence in the RDF repository ($u_\phi > 0$), the *possibility* of an axiom ϕ involves the absence of counterexamples to ϕ in the RDF dataset. Axiom ϕ is more *possible* as it is not contradicted by any fact. The degree of possibility of an axiom ϕ equals to 1, i.e., $\Pi(\phi) = 1$, means that it is *completely possible*, i.e. plausible, is not contradicted by facts in the knowledge base. When the number of counterexamples u_ϕ^- increases, $\Pi(\phi) \to 0$ strictly monotonically. Possibly formalize the description of the above intuitions in the variant of mathematical definitions for $\Pi(\phi)$ with $u_\phi > 0$ as follows:

- if $D(\phi)$ is in conjunctive normal form:

$$\Pi(\phi) \quad = \quad 1 - \sqrt{1 - \left(\frac{u_\phi - u_\phi^-}{u_\phi}\right)^2} \tag{5.20}$$

- if $D(\phi)$ is in disjunctive normal form:

$$\Pi(\phi) = \begin{cases} 1 - \sqrt{1 - \left(\frac{u_\phi - u_\phi^-}{u_\phi}\right)^2}, & \text{if } u_\phi^+ = 0, \\ 1, & \text{if } u_\phi^+ > 0. \end{cases} \tag{5.21}$$

Meanwhile, axiom ϕ is more *necessary* as it is explicitly supported by facts and not contradicted by any fact [TFG15]. In the specific case of $D(\phi)$ being conjunctive, if there is only one counterexample to ϕ in the RDF dataset, the degree of necessity of axiom ϕ will be zero, i.e. $N(\phi) = 0$. Otherwise, if the number of confirmations increases and no counterexamples are found, $N(\phi) \to 0$ strictly monotonically. The possible mathematical definitions for $N(\phi)$ are given correspondingly in the case of $u_\phi > 0$ as follows:

- if $D(\phi)$ is in conjunctive normal form:

$$N(\phi) = \begin{cases} \sqrt{1 - \left(\frac{u_\phi - u_\phi^+}{u_\phi}\right)^2}, & \text{if } u_\phi^- = 0, \\ 0, & \text{if } u_\phi^- > 0. \end{cases} \tag{5.22}$$

- if $D(\phi)$ is in disjunctive normal form:

$$N(\phi) = \sqrt{1 - \left(\frac{u_\phi - u_\phi^+}{u_\phi}\right)^2} \tag{5.23}$$

We can derive three extreme epistemic attitudes pertaining to an axiom ϕ also found in the certainty of an information item in [DDP20] as follows:

- the certainty that ϕ is true: $N(\phi) = 1$, hence $\Pi(\phi) = 1$;
- the certainty that ϕ is false: $\Pi(\phi) = 0$, hence $N(\phi) = 0$;
- ignorance pertaining to ϕ: $\Pi(\phi) = 1$ and $N(\phi) = 0$ when $u_\phi = 0$, given that no evidence is available in the RDF dataset to assess the credibility of ϕ..

In principle, the combined values of necessity $N(\phi)$ and possibility $\Pi(\phi)$ of axiom ϕ is considered as representative of its degree of *credibility* which the fitness functions later for axiom evaluation should be directly proportional to.

5.4 Summary

As an alternative to statistics-based heuristics applied in probabilistic evaluation framework, possibilistic approach is currently preferable in the condition of an incomplete RDF repository. We will investigate the implementation of this framework in terms of class disjointness axiom testing against the DBpedia database in the next chapters.

6

Grammatical Evolution Models toward Class Disjointness Axiom Discovery

Contents

6.1 Introduction 87
6.2 Related Works 88
6.3 Learning Atomic and Complex Axioms involving Union and Intersection Operators 90
6.3.1 GE Characteristics 90
6.3.2 Gold Standard toward a Subjective Benchmark 98
6.3.3 Experimental Protocol 101
6.3.4 Results & Discussions 102
6.3.5 Limitations 106
6.4 Learning Complex Axioms containing Value and Existential Restriction 107
6.4.1 GE Characteristics 108
6.4.2 From A Training- Testing Model toward An Objective Benchmark 109
6.4.3 Experimental Protocol 111
6.4.4 Results & Discussions 112
6.5 Summary 116

6.1 Introduction

The different types of negation are one of the requirements for expressive ontologies [Flo+06]. Unfortunately, ontology languages based on DLs are not expressive enough to express axiom negations. Among different types of axioms, class disjointness axioms, which, despite their importance, are little used in knowledge bases, express the incompatibility between pairs of concepts known as *concept disjointness* based on negation. Even though class disjointness axioms are supported by ontology languages, e.g. OWL with the keyword `owl:disjointWith`, disjointness information is often neglected when building logical modeling [Rud11], and one can find only a few axioms of this type currently in existing ontologies. For example, in the DBpedia ontology, the query `SELECT ?x ?y { ?x owl:disjointWith ?y }` executed on March 09, 2021 returned only 27 solutions, whereas the realistic number of class disjointness axioms that one would expect to hold among the hundreds of classes in DBpedia (738 classes in DBpedia version 2015-04, 760 classes in DBpedia version 2016-04)[1] must be much larger, in the order of thousands or tens of thousands.

This type of axiom is essentially useful for checking data quality, in particular, for testing the logical consistency and for detecting the undesired usage patterns or incorrect assertions. The definition of *concept disjointness* with respect to an interpretation $\mathcal{I}$ is given as follow:

Definition 6.1.1: Concept Disjointness [Rud11]

Given C, D are concepts. Two concepts C and D are *disjoint* with respect to an intepretation $\mathcal{I}$ if they do no possess any common individual according to their extensions, i.e. $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$.

A simple example can demonstrate the potential advantages obtained by the addition of this kind of axioms to an ontology. A knowledge base (KB) defining terms of classes like *Mother, Man* and asserting that individual `Tyler` is both a `Mother`, i.e., `Mother(Tyler)`, and a `Man`, i.e., `Man(Tyler)`, would be logically consistent, without any errors being recognized by a reasoner. Yet, if a constraint of

[1] These figures can be obtained by executing the querry " `SELECT (COUNT( DISTINCT ?subject) as ?numberClasses` " on `http://dbpedia.org/sparql`

disjointness between classes `Mother` and `Man`, i.e., as this statement can be expressed in DLs: `Mother`$\sqsubseteq \neg$`Man`, is added, the reasoner will be able to reveal an error in the modeling of such a knowledge base. As a consequence, logical inconsistencies of facts can be detected and excluded—thus enhancing the quality of ontologies.

In reality, learning implicit knowledge in terms of class disjointness axioms from a LOD repository in the context of the Semantic Web has been the object of research in several different approaches which are introduced in Section 6.2. Along the line of the general GE model to axiom discovery evaluation introduced in Chapter 4, we develop specific models for mining different types of class disjointness axioms from RDF datasets described in Section 6.3 and Section 6.4. In the functional-style syntax of OWL[2], class disjointness axioms have the form $\mathsf{DisjointClasses}(C_1\, C_2 ... C_n)$. In order to simplify our discussion and without loss of generality, we can only focus on binary axioms such as $\mathsf{DisjointClasses}(C_1\, C_2)$, where C_1 and C_2 can be atomic expressions or complex expressions. In addition, the idea of axiom evaluation arising in Chapter 5 is applied to develop different evaluation frameworks to assess the certainty level of induced axioms. Finally, we conclude this chapter by pointing out the major contributions of the developed models in Section 6.5.

6.2 Related Works

The most prominent related work relevant to learning disjointness axioms consists of the contributions by Johanna Völker and her collaborators [Völ+07; FV11; VFS15]. In early work, Völker developed supervised classifiers from LOD incorporated in the *LeDA* tool [FV11]. However, the learning algorithms need a set of labeled data for training that may demand expensive work by domain experts. In contrast to *LeDA*, statistical schema induction via association rule mining [VFS15] was given in the tool *GoldMiner*, where association rules are representations of implicit patterns extracted from large amount of data and no training data is required. Association rules are compiled based on statistical analysis of a transaction table, which is built from the results of SPARQL queries. That research only focused

[2]https://www.w3.org/TR/owl2-syntax/#Functional-Style_Syntax

on generating axioms involving *atomic* classes, i.e., classes that do not consist of logical expressions, but only of a single class identifier.

Another relevant research is the one by Lorenz Bühmann and Jens Lehmann, whose proposed methodology is implemented in the DL-Learner system [Leh09] for learning general class descriptions (including disjointness) from training data. Their work relies on the capabilities of a reasoning component, but suffers from scalability problems for the application to large datasets like LOD.

Also, a recent contribution of Reynaud *et al.* [RTN19] uses *Redescription Mining (RM)* to learn class equivalence and disjointness axioms with the *ReReMi* algorithm. *RM* is about extracting a category definition in terms of a description shared by all the instances of a given class, i.e. equivalence axioms, and finding incompatible categories which do not share any instance, i.e. class disjointness axioms. Their method, based on *Formal Concept Analysis (FCA)*, a mathematical framework mainly used for classification and knowledge discovery, aims at searching for data subsets with multiple descriptions, like different views of the same objects. While category redescriptions, i.e., equivalence axioms, refer to complex types, defined with the help of relational operators like $A \equiv \exists r.C$ or $A \equiv B \sqcap \exists r.C$, in the case of incompatible categories, the redescriptions are only based on the set of attributes with the predicates of dct:subject, i.e. axioms involving atomic classes only.

Another procedure for extracting disjointness axioms [Riz+17] requires a *Terminological Cluster Tree (TCT)* to search for a set of pairwise disjoint clusters. A decision tree is built and each node in it corresponds to a concept with a logical formula. The tree is traversed to create concept descriptions collecting the concepts installed in the leaf-nodes. Then, by exploring the paths from the root to the leaves, intensional definitions of disjoint concepts are derived. Two concept descriptions are disjoint if they lie on different leaf nodes. An important limitation of the method is the time-consuming and computationally expensive process of growing a *TCT*. A small change in the data can lead to a large change in the structure of the tree. Also, like other intensional methods, that work relies on the services of a reasoning

component, but suffers from scalability problems for the application to large datasets, like the ones found on the LOD, caused by the excessive growth of the decision tree.

6.3 Learning Atomic and Complex Axioms involving Union and Intersection Operators

Preliminarily, we only focus on mining class disjointness axioms containing atomic expressions, e.g. DisjointClasses(Film WrittenWork), or complex expressions in the cases of relational operators, i.e., intersection and union, e.g. DisjointClasses(Film ObjectIntersectionOf(Book ObjectUnionOf(Comics MusicalWork))). The learning method which we use in the following is based on the general GE model for axiom discovery introduced in Chapter 4, with specific settings.

6.3.1 GE Characteristics

BNF Grammar Pattern

According to the method for BNF grammar construction introduced in Section 4.3.1, we initially build a BNF grammar pattern for generating this kind of class disjointness axioms consisting of two parts, namely *static part* and *dynamic part* as follows:

BNF Grammar Pattern 6.3.1

% Static part

```
(r1) Axiom := ClassAxiom
(r2) ClassAxiom := DisjointClasses
(r3) DisjointClasses := 'DisjointClasses' '(' ClassExpression ' 'ClassExpression ')'
(r4) ClassExpression :=   Class                (0)
                        | ObjectUnionOf        (1)
                        | ObjectIntersectionOf (2)
(r5) ObjectUnionOf := 'ObjectUnionOf' '(' ClassExpression ' ' ClassExpression ')'
(r6) ObjectIntersectionOf := 'ObjectIntersectionOf' '(' ClassExpression' 'ClassExpression ')'
```

% Dynamic part - Primitives

```
(r7) Class := % production rules are constructed by using SPARQL queries
```

The production rules of the primitive `Class` in the dynamic part will be filled by using the SPARQL queries mentioned in Equation (4.2) to extract the IRI of a class mentioned in the RDF store.

Example 6.3.1 (RDF data) *An example representing a small excerpt of an RDF triple repository is the following:*

```
PREFIX dbr: http://dbpedia.org/resource/
PREFIX dbo: http://dbpedia.org/ontology/
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

dbr:Quiet_City_(film)     rdf:type      dbo:Film.
dbr:Cantata               rdf:type      dbo:MusicalWork.
dbr:The_Times             rdf:type      dbo:WrittenWork.
dbr:The_Hobbit            rdf:type      dbo:Book.
dbr:Fright_Night_(comics) rdf:type      dbo:Comic
```

and options for the `Class` *non-terminal are represented as follows:*

```
(r.7) Class := dbo:Film             (0)
            |  dbo:MusicalWork      (1)
            |  dbo:WrittenWork      (2)
            |  dbo:Book             (3)
            |  dbo:Comic            (4)
```

Example 6.3.2 (Mapping Process)

- *Let Grammar Pattern 6.3.1.*
- *Let RDF store be as in Example 6.3.1*
- *Let a chromosome be* $253, 213, 397, 387, 268, 342, 321, 408, 182, 132$.

We apply Equation (4.1) to choose production rules from the grammar. Figure 6.1 illustrates the steps of the mapping to a class disjointness axiom expression relevant to the considered example. There is only one production for non-terminals `Axiom`*,* `ClassAxiom`*,* `DisjointClasses`*,* `ObjectIntersectionOf`*, and* `ObjectUnionOf` *as it can be seen from Rules 1–3, 5, and 6. In these cases, we skip using any codons for mapping and concentrate on reading codons for non-terminals having more than one production, like in Rules 4 and 7. We begin by decoding the first codon, i.e.* `253`*, by Eq. 4.1. The result, i.e* `253 modulo 3 = 1`*, is used to determine which production is chosen to replace the leftmost non-terminal (*`ClassExpression`*) from its relevant rule (Rule 4). In this case, the leftmost* `ClassExpression` *will*

be replaced by the value of `ObjectUnionOf`*. The mapping goes on like this until eventually there is no non-terminal left in the expression. Not all codons were required and extra codons have been simply ignored in this case.*

In the mapping process based on Grammar 6.3.1 and Equation (4.1), the production rule for `ClassExpression` is recursive and may lead to a large fan-out. In order to alleviate this problem and promote "reasonable" axioms, one of the solutions is to increase the probability of obtaining a successful mapping to complex axiom expressions. In practice, we enforce doubling the appearance probability of non-terminal `ClassExpression`. Rule (r4) in the grammar is modified to

```
(r4) ClassExpression :=   Class                (0)
                        | Class                (1)
                        | ObjectUnionOf        (2)
                        | ObjectIntersectionOf (3)
```

Evolutionary Process

- **Initialization**: The initial population is seeded with *popSize* random chromosomes of *initlenChrom* codons uniformly distributed over $\{0, \ldots, maxValCodon-1\}$.

- **Genotype-to-Phenotype Mapping**: The standard genotype-to-phenotype mapping is used, with at most *maxWrap* wrapping events. In case of an unsuccessful mapping (because after the maximum allowed number of wrapping events the individual is not yet completely mapped), the individual is assigned a fitness of zero, i.e., the lowest possible fitness.

- **Parent selection**: We use the parent selection mechanism described in Section 4.3.

- **Variant operators**: The single-point crossover operator is applied to genotypes, with probability *pCross*. The standard mutation operator is also applied with probability *pMut*.

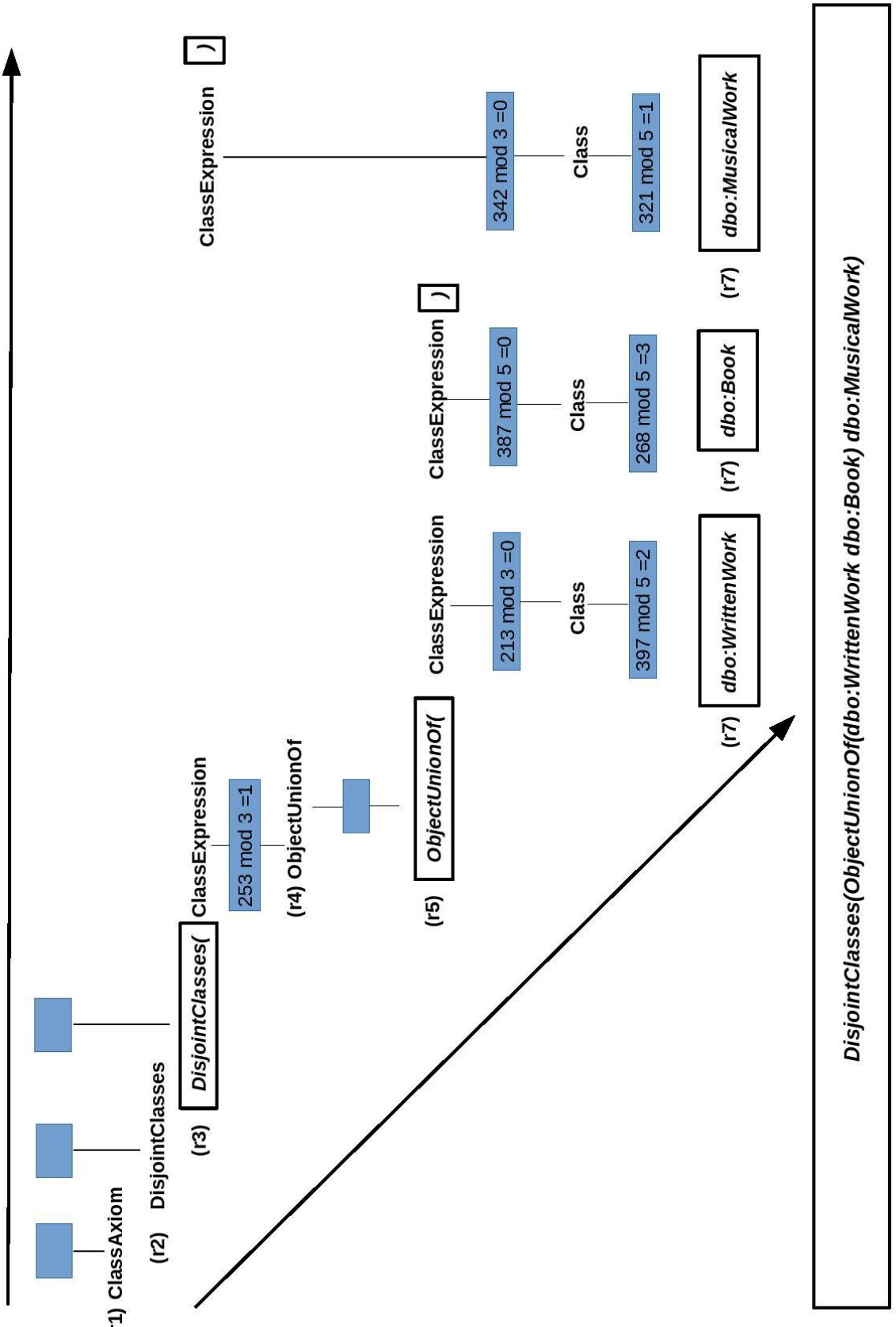


Figure 6.1: An illustration of mapping process to an expression of class disjointness axiom

- **Survival selection**: We use the *Crowding* function presented in *Algorithm* 2 implementing the *Deterministic Crowding* method to improve the diversity of the population. In it, the comparison is performed at the *genotypic level* to decide whether an individual is to be selected for inclusion in the population of the next generation. The genotypic distance between individuals is computed as their *Hamming distance*, with the expectation of obtaining more accurate results.

- **Fitness Functions**: In order to build the fitness functions, we follow the possibilistic approach to axiom scoring presented in Section 5.3.2. In practice, what we are looking for is not only credible axioms, but also general ones. Hence, the credibility of an axiom should be directly proportional to its necessity $N(\phi)$, its possibility $\Pi(\phi)$, whereas the generality g_ϕ of an axiom is based on the support of the axiom, u_ϕ.

  Deriving from those basic ideas, we propose the first version of the fitness function as follows:

  Fitness Function 1

  $$f(\phi) = u_\phi \cdot \frac{\Pi(\phi) + N(\phi)}{2}, \tag{6.1}$$

  As mentioned in Section 5.2.1, the transformation of a disjointness class axiom is based on the definition indicated in Table 5.1. We compact it to suit for the binary class disjointness axioms $Dis(C_1, C_2)$ as follows: $t(Dis(C_1, C_2), x, y) = \forall x(\neg t(C_1, x, y) \vee \neg t(C_2, x, y))$ in which C_1, C_2 are class expressions (concepts).

  In addition, the development of class disjointness axiom ϕ with respect to RDF dataset $\mathcal{K}$, noted $D_\mathcal{K}(\phi)$, is transformed into *conjunctive* normal form, i.e. $D_\mathcal{K}(\phi) = \bigwedge_i \psi_i$ in which ψ_i are basic statements which are tested against the available facts in RDF data. Thus, the possibility measure $\Pi(\phi)$ and the necessity measure $N(\phi)$ of axiom ϕ are defined by Equation (5.20) and Equation (5.22), respectively. Whereby, these quantities are measured by defining the number of confirmations and the number of counterexamples. According

to the computational definitions described in Section 5.2.2, these values are counted by executing the corresponding SPARQL queries based on *graph patterns*, via an accessible SPARQL endpoint. Practically, for a given class disjointness axiom $\mathsf{DisjointClasses}(C, D)$ (or $\mathsf{Dis}(C, D)$ in Description Logic notation) we use Equation (5.8) to figure out its number of counterexamples $u^{-}_{\mathsf{Dis}(C,D)}$ as follow:

```
SELECT( count (DISTINCT ?x)  AS  ?numberOfCounterexamples)
 WHERE {Q(C, ?x)  Q(D, ?x)}
```
(6.2)

We can also use Equation (5.10) and Equation (5.11) to define its number of confirmations as follows:

```
SELECT (count (DISTINCT ?x) AS ?numberOfConfirmations)
 WHERE {
    {
      Q(C, ?x, ?y)
      Q_Dis(D| C, ?x, ?y)
    }
    UNION
    {
      Q(D, ?x, ?y)
      Q_Dis(C|D, ?x, ?y)
    }
 }
```
(6.3)

In this query, $\mathtt{Q}_{Dis}(C \mid D, \mathtt{?x}, \mathtt{?y})$ and $\mathtt{Q}_{Dis}(D \mid C, \mathtt{?x}, \mathtt{?y})$ are defined as follows:

$$\texttt{Q}_{Dis}(C \mid D, \texttt{?x}, \texttt{?y}) = \{$$
```
?x  a  ?dc1
?z1  a  ?dc1
Q(¬D, ?z1, ?y1)
FILTER NOT EXISTS {
?z2 a ?dc1.
Q(C, ?z2, ?y2)
}
```
$$\}$$ (6.4)

$$\texttt{Q}_{Dis}(D \mid C, \texttt{?x}, \texttt{?y}) = \{$$
```
?x  a  ?dc1
?z1  a  ?dc1
Q(¬C, ?z1, ?y1)
FILTER NOT EXISTS {
?z2 a ?dc1.
Q(D, ?z2, ?y2)
}
```
$$\}$$ (6.5)

However, it should be noticed that negation is not supported by RDF. Negated assertions can of course be expressed using the vocabulary of OWL, but then the services of an OWL reasoner would have to be used to infer the negation of an assertion thus expressed; however, that would be way more expensive than using SPARQL to query the dataset and also of little use, since very few or no negated assertions at all do occur in real-world RDF datasets. As a result, an RDF dataset will naturally provide counterexamples for the disjointness axioms (e.g., an individual that is asserted to belong to two supposedly disjoint classes). On the other hand, confirmations, which should take the form of negated assertions, like "such individual, which belongs to either of the supposedly disjoint classes, *does not* belong to the other", will be completely missing. The simple solution of taking the absence of a counterexample as a confirmation $u^{+}_{\mathsf{Dis}(C,D)}$ can be as follows:

$$u^{+}_{\mathsf{Dis}(C,D)} = u_{\mathsf{Dis}(C,D)} - u^{-}_{\mathsf{Dis}(C,D)} \tag{6.6}$$

Also, based on Equation (5.5), the support $u_{Dis(C,D)}$ of $Dis(C, D)$ the cardinalities of the extension of the two classes C and D is computed with the following SPARQL query:

```
SELECT( count (DISTINCT ?x)  AS  ?u)
      WHERE {Q(C, ?x)  UNION  Q(D, ?x)}
```
(6.7)

In addition, a second refinement of the definition of fitness stems from the observation that, for a disjointness axiom of the form $\mathsf{Dis}(C, D)$, a better measure of its generality would be given by the minimum of the cardinalities of the extensions of the two classes involved, C and D, in the RDF dataset, whereas $u_{\mathsf{Dis}(C,D)}$ is the cardinality of the extension of $C \sqcup D$. Let us denote by $[C]$ the extension of class C in the RDF dataset at hand: this is the set of instances of C returned by a SPARQL query of the form

```
SELECT DISTINCT ?x WHERE { ?x a C}
```
(6.8)

We will define the generality of axiom $\mathsf{Dis}(C, D)$ as

$$g_{\mathsf{Dis}(C,D)} = \min\{\|[C]\|, \|[D]\|\} \tag{6.9}$$

and use it instead of u_ϕ in Equation (6.1), with the following SPARQL queries.

```
SELECT( count (DISTINCT ?x)  AS  ?u_C)  WHERE {Q(C, ?x)}
SELECT( count (DISTINCT ?x)  AS  ?u_D)  WHERE {Q(D, ?x)}
```
(6.10)

Also, the solution for counting confirmations as in Equation (6.6) would betray the open-world hypothesis that underlies the SW. Hence, this problem can be overcome by actually scoring axioms based on counterexamples only, which is, after all, much in agreement with the falsificationist approach that underlies the current practice in Science (to corroborate a hypothesis, one

should not look for easy confirmations, but should rather try hard to find counterexamples). Since the number of confirmations u_ϕ^+ only appears in the definition of $N(\phi)$, we can safely drop $N(\phi)$ from the fitness function. This yields the following improved second definition of the fitness function,

Fitness Function 2

$$f(\phi) = g_\phi \cdot \Pi(\phi), \tag{6.11}$$

We will run two independent experiments with the two above fitness functions.

6.3.2 Gold Standard toward a Subjective Benchmark

In order to evaluate the effectiveness of our method in discovering disjointness class axioms, we use a *subjective benchmark* called the *Gold Standard*, created by domain experts and knowledge engineers.

Gold Standard Construction

The process of creating the *Gold Standard* was carried out by both manual and automatic mechanisms depending on the evaluation results of prior pairs of classes. In general, the Gold Standard construction consists of two phases, illustrated in Figure 6.2.

In the first phase, the disjointness of the top-most classes to their siblings was assessed manually. As a result, two sibling classes being disjoint will automatically imply the disjointness of their corresponding pairs of subclasses. This process was repeated in the same way on the next level of concepts.

The second phase consisted in manually annotating the disjointness for the pairs of classes not yet assessed because they did not belong to the cases given in the previous phase. The result of the completion of the *Gold Standard* is a matrix representing the disjointness evaluation between pairs of distinct atomic class expressions. We first constructed the (62×62) matrix[3] of class disjointness axioms relevant to the *Work* topic of DBpedia 2015-04. This matrix contains (0

[3] https://bitbucket.org/RDFMiner/disjointnessclassaxiomge/src/master/GoldStandard.csv

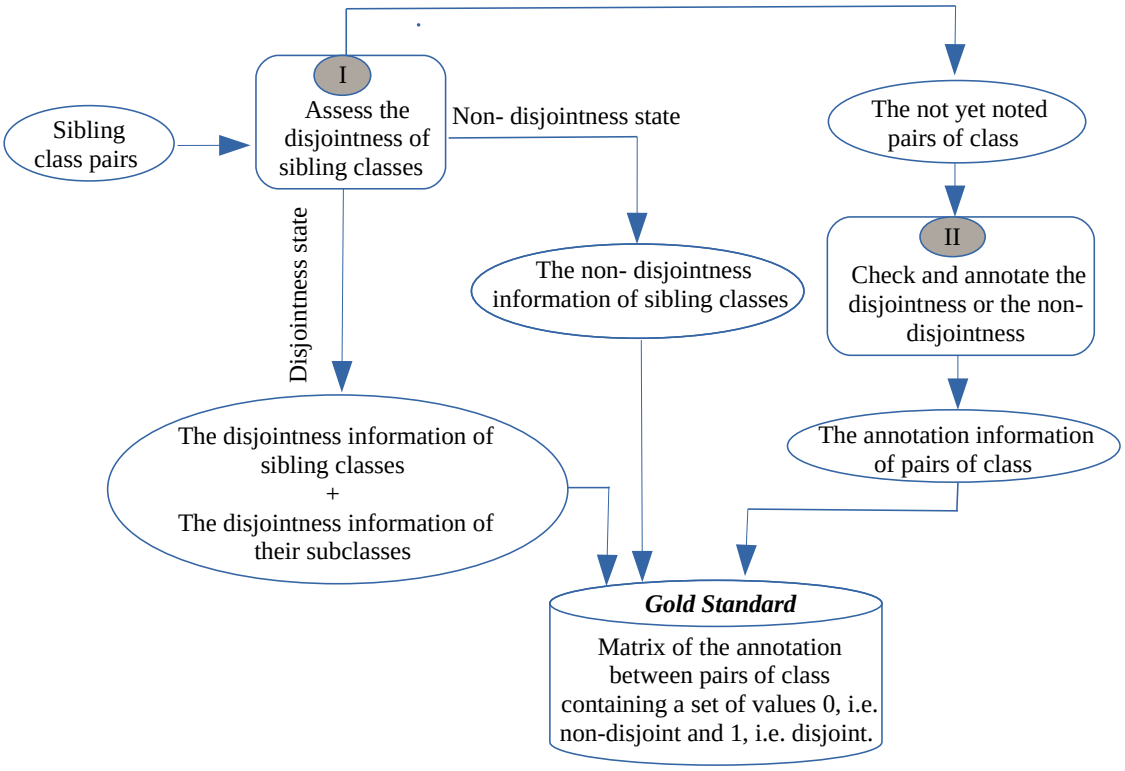


Figure 6.2: The process of *Gold Standard* creation

and 1) values representing the disjointness evaluation between 3,844 pairs of classes relevant to the topic, with 1,891 pairs of distinct asymmetric classes.

Gold Standard based Performance Assessments

In compliance with the *Gold Standard* thus constructed, we measure the quality of class disjointness axioms involving both atomic and more complex types, i.e. involving the intersection and union operators. Algorithm 4 describes in detail how a complex axiom is assessed using the *Gold Standard*. Specifically, this depends on considering whether its class expressions, i.e. axiom arguments, are mutually disjoint or not. A recursive method is applied to check disjointness between two class expressions, namely $expr_1$ and $expr_2$. The base steps (lines 1-4) involve the case when $expr_1$ and $expr_2$ are atomic classes in which the disjointeness between them is defined by looking up the Gold Standard, i.e. the function *CheckDisjointAtomicClasses*. The recursive steps (lines 6-13) occur if at least one expression is complex involving relational operators, i.e., union or intersection. In

this case, checking the complex expressions can be solved by converting them into the simpler ones, i.e. containing at least one atomic expression, until the base case is reached. The union operator refers to the sum of the recursive cases while the intersection operator corresponds to their multiplication.

Algorithm 4: CheckDisjoint(expr1(n),expr2(m))

Input: $expr_1(n), expr_2(m)$: class expressions being arguments in axiom;
n,m: the numbers of the classes contained in the class expressions;
n, m: the numbers of the classes contained in the class expressions;
G: matrix of Gold Standard

Output: R: results of disjointness - returns a non- negative integer value; if the return value is greater than 0, *expr1(n)* and *expr2(m)* are disjoint; if the return value equals 0, *expr1(n)* and *expr2(m)* are non-disjoint

1 **if** *both* $classexpr_1(n)$ *and* $classexpr(_2(m)$ *are atomic expressions* **then**
2 $expr_1(1) \leftarrow expr_1(n)$
3 $expr_2(1) \leftarrow expr_2(m)$
4 $R \leftarrow$ **CheckDisjointAtomicClasses**$(expr_1[1], expr_2[1])$
/* **CheckDisjointAtomicClasses**$(expr_1, expr_2)$ scans in the matrix G and returns 0, i.e. non-disjoint or 1, i.e disjoint. */
5 **else**
6 **if** $expr_1(n)$ *is a complex expression containing union operator* **then**
7 $R \leftarrow$ **CheckDisjoint**$(expr_1[1], expr_2(m)) +$ **CheckDisjoint**$(expr_1(n-1), expr_2(m))$
8 **if** $expr_1(n)$ *is a complex expression containing intersection operator* **then**
9 $R \leftarrow$ **CheckDisjoint**$(expr_1[1], expr_2(m)) *$ **CheckDisjoint**$(expr_1(n-1), expr_2(m))$
10 **if** $expr_2(m)$ *is a complex expression containing union operator* **then**
11 $R \leftarrow$ **CheckDisjoint**$(expr_1[n], expr_2(1)) +$ **CheckDisjoint**$(expr_1(n), expr_2(m-1))$
12 **if** $expr_2(m)$ *is a complex expression containing intersection operator* **then**
13 $R \leftarrow$ **CheckDisjoint**$(expr_1[n], expr_2(1)) *$ **CheckDisjoint**$(expr_1(n), expr_2(m-1))$
14 **return** R

Example 6.3.3 *Based on the complex axiom generated from mapping process in Example 6.3.2.*

`ClassDisjointness(ObjectUnionOf(dbo:WrittenWork dbo:Book)) dbo:MusicalWork)` *The steps of the Algorithm 4 to validate the axiom based on the Gold Standard are as follows:*

1. $R \leftarrow$ ***CheckDisjoint***`(ObjectUnionOf(dbo:WrittenWork dbo:Book)), dbo:MusicalWork)`

2. $R \leftarrow$ ***CheckDisjoint***`(dbo:WrittenWork, dbo:MusicalWork)` +
CheckDisjoint`(dbo:Book, dbo:MusicalWork)`

3. *Because dbo:Book, dbo:MusicalWork dbo:WrittenWork are atomic classes,* ***CheckDisjoint*** *will call the base function* ***CheckDisjointAtomicClasses****. The returning values of the base function are scanned through the matrix of Gold Standard.*

   1 ← ***CheckDisjointAtomicClasses****(*`dbo:Book`*,* `dbo:MusicalWork`*)*

   1 ← ***CheckDisjointAtomicClasses****(*`dbo:WrittenWork`*,* `dbo:MusicalWork`*)*

4. $R = 1 + 1 = 2$. $R > 0$ *means that the axiom is* ***disjoint*** *based on the validation of the Gold Standard.*

6.3.3 Experimental Protocol

We apply the GE approach with the settings introduced in Section 6.3.1, to mine class disjointness axioms. The axioms involving atomic or complex expressions of union and intersection relevant to topic *Work* are systematically generated and then evaluated on DBpedia version 2015-04 in English as the reference RDF fact repository. Of 62 classes about the *Work* topic in DBpedia 2015-04, 53 classes with 5,195,019 instances are relevant to our experiments. The data used in this experiment are represented by RDF triples, as in Example 6.3.1. We use the BNF grammar pattern 6.3.1 of disjointness axioms with the double appearance probability of non-terminal `ClassExpression` in Rule (r4). Although the desirable purpose of our research is to focus on exploring complex disjointness axioms (atomic axiom can be considered as a special case of complex ones), we also performed experiments to generate axioms involving atomic classes only, for comparison purpose. In that case, Rule (r4) is simplified to only one option `ClassExpression := Class` .

We set up two different sets of the algorithm parameters summarized in Table 6.1 and Table 6.2 involving the first fitness function (6.1) and the second one (6.11), respectively, which were empirically determined by performing a systematic exploration of a grid of possible settings.

A prototype system of the proposed method was developed in Java, using Apache Jena to interface with SPARQL endpoints and GEVA[4] v.2.0 , a Java implementation

[4] http://ncra.ucd.ie/Site/GEVA.html

Table 6.1: The first set of GE parameter values with *Fitness Function 1 (6.1)*

Parameter	Value
popSize	500
numGenerations	30
initlenChrom	20
maxWrap	2
pCross	80%
pMut	1%
pselectSize	70%
pElite	2%
$f(\phi)$	$u_\phi \cdot \frac{\Pi(\phi)+N(\phi)}{2}$

Table 6.2: The second set of GE parameter values with *Fitness Function 2 (6.11)*

Parameter	Atomic Axioms	Complex Axioms
popSize	2,000	2,000
numGenerations	25	5
initLenChrom	5	30
maxWrap	2	2
pCross	80%	80%
pMut	1%	1%
pselectSize	70%	70%
pElite	2%	2%
$f(\phi)$	$g_\phi \cdot \Pi(\phi)$	

of GE. In order to avoid overloading DBpedia's SPARQL endpoint, we set up a local mirror[5] using the Virtuoso Universal Server.

All the experiments have been performed on a HP ZBook 15 G3 Mobile Workstation equipped with an eight-core Intel i7 CPU 6820HQ processor at 2.7GHz clock speed, with 32 GB RAM, 1 TB of disk space under the Ubuntu 16.04 LTS 64-bit operating system.

6.3.4 Results & Discussions

For both indicated sets of GE parameters above, we ran the GE for axiom discovery by repeating the sample procedure of Algorithm 1 for each run with the same parameters of each case.

[5] https://joernhees.de/blog/2015/11/23/setting-up-a-linked-data-mirror-from-rdf-dumps-dbpedia-2015-04-freebase-wikidata-linkedgeodata-with-virtuoso-7-2-1-and-docker-optional/

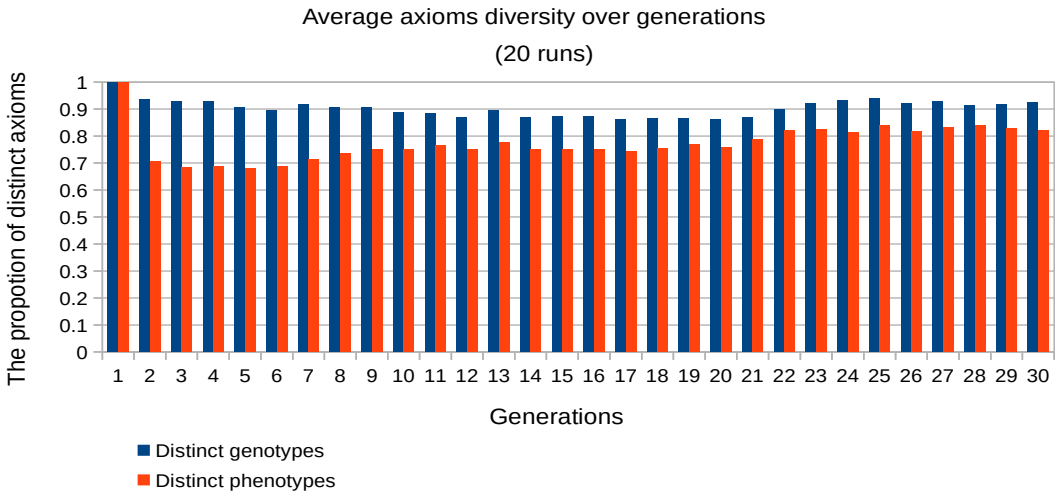


Figure 6.3: The diversity of axioms over generations

The first set of GE parameter values

The chart in Figure 2 illustrates the average diversity of the population of axioms over the generations of the evolutionary process. It shows how many different "species" of axioms are contained in the population, i.e., axioms that cover different aspects of the known facts. One of the remarkable points here is that there is a more rapid loss of diversity in the phenotype axioms compared with this decrease in the genotype ones. The use of the *Crowding method* on genotypes instead of phenotypes can be the reason of this difference. Likewise, a set of codons of two parent chromosomes which are used for the mapping to phenotypes can fail to be swapped in the single-point crossover operator. From the chart in Figure 3, we can observe a gradual increase in the quality of discovered axioms over generations.

Table 6.3: Experimental results

	Our approach		GoldMiner
	Complex axioms	Atomic axioms	Atomic axioms
Precision (per run)	0.867 ± 0.03	0.95 ± 0.02	0.95
Recall (per run)	N/A	0.15 ± 0.017	0.38
Recall (over 20 runs)	N/A	0.54	0.38

The precision and recall are computed by comparison to the *Gold Standard*. Regarding to atomic axioms, the results in Table 6.3 confirm the high accuracy

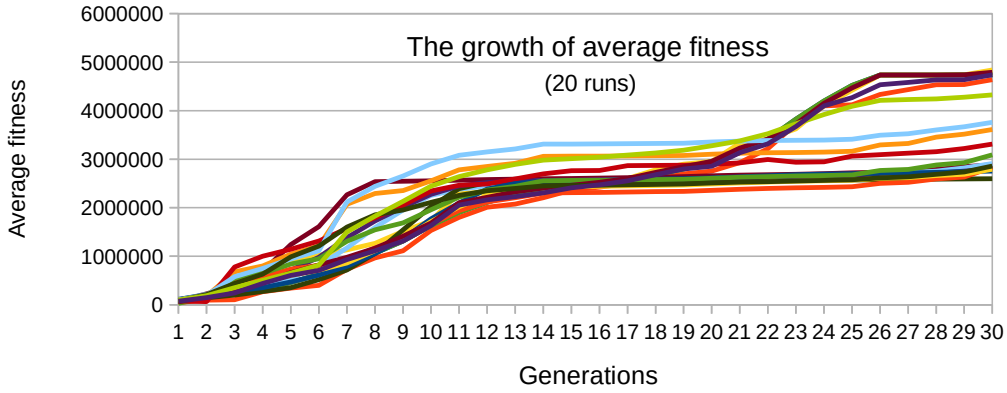


Figure 6.4: The growth of average fitness over generations

of our approach in discovering class disjointness axioms (Precision $= 0.95 \pm 0.02$). The recall value is higher than the value in *GoldMiner* [VFS15]. In addition, there are a number of discovered class disjointness axioms being absent in the result of *GoldMiner*. For instance, there are no axioms relevant to class Archive in the axioms generated by *GoldMiner*.

In the case of more complex axioms, there is a smaller degree of precision (Precision $= 0.867 \pm 0.03$). The reason may stem from the complexity in the content of generated axioms which is relevant to more different classes. We do not present the recall for the case of complex axioms, since the discovery process of this type of axioms cannot define how many of the complex axioms should have been generated. After 20 runs, from 10,000 candidate individual axioms, we got 5,728 qualified distinct complex axioms. We performed an analysis of the discovered axioms and found some noticeable points. Almost all generated axioms have high fitness values with millions of support instances from the DBpedia dataset, which witness the generality of the discovered axioms. However, we found some deficiencies in determining the disjointness of classes. As in the case of axiom DisjointClasses(dbo:MovingImage ObjectUnionOf(dbo:Article dbo:ObjectUnionOf(dbo:Image dbo:MusicalWork))), 4,839,992 triples in DBpedia confirm that this class disjointness axiom is valid. However, according to the *Gold Standard*, these two classes should not be disjoint *a priori*. Indeed, the class MovingImage can be assessed as a subclass of Image, which makes the disjointness between class MovingImage and any complex

class expression involving relational operator union of class Image altogether impossible. Another similar case is the axiom DisjointClasses(ObjectUnionOf(dbo:Article ObjectUnionOf(ObjectUnionOf(ObjectUnionOf (ObjectUnionOf(dbo:TelevisionShow, dbo:WrittenWork) dbo:MusicalWork) dbo:Image) dbo:Film)) dbo:UndergroundJournal), having 5,037,468 triples in its support. However, according to the *Gold Standard*, these two classes should not be disjoint. The main reason for such erroneous axioms may lie in the inconsistencies and errors in the DBpedia dataset. Another possible cause is the subjectiveness of Gold Standard to some extent, thus, the evaluation of axioms is quite sensitive to this benchmark. Specifically, the method we used to build it is too simplistic based on human experts and possibly fails to capture some disjointness axioms.

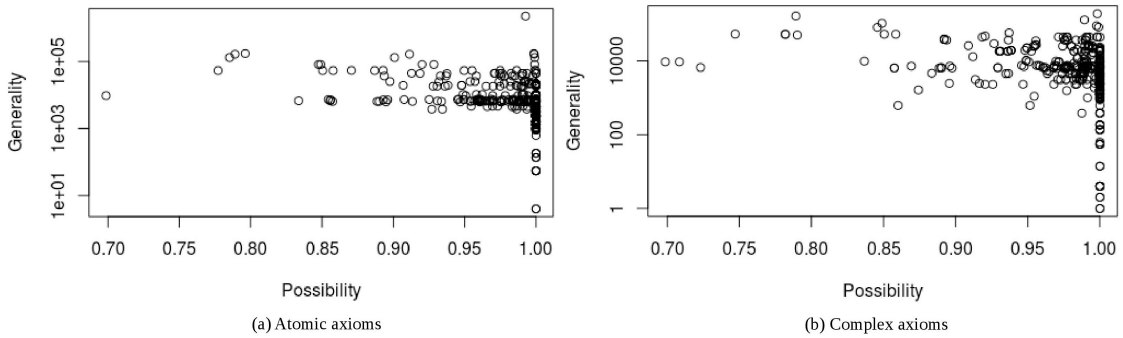
The second set of GE parameters

The results, shown in Table 6.4, confirm that the accuracy and the coverage of the second parameter settings in Table 6.2 in extracting atomic axioms are higher than the results of the first setting in Table 6.1 and *GoldMiner*. In terms of generating complex axioms, we witness a quite higher accuracy than in the first parameter settings involving the fitness function described in Equation (6.1) and a superiority of our method compared with *GoldMiner*. In the recall comparison for the case of atomic axioms, we can also observe that the coverage of the set of generated atomic axiom in each run is much higher than the result in 6.1. The recall value in *GoldMiner* is constant, namely 0.38, because that algorithm is deterministic. Meanwhile, the overall recall value of our method gets much higher, namely 0.323 over 3 runs but ours is stochastic, and would easily overtake the results of *GoldMiner* and the parametter setting in Table 6.1 simply by executing more runs. Therefore, the comparison in this case is unnecessary.

We do not present the recall for complex axioms, because it is not clear how the cardinality of the set of *all* true complex axioms should be computed; under the most general assumptions, this set is infinite, although enumerable.

Table 6.4: Experimental results

	Results involving Table 6.2		Results involving Table 6.1		GoldMiner
	Atomic axioms	Complex axioms	Atomic axioms	Complex axioms	Atomic axioms
Precision	0.958 ± 0.011	0.876 ± 0.01	0.95 ± 0.02	0.867 ± 0.03	0.95
Recall (per run)	0.247 ± 0.01	N/A	0.15 ± 0.017	N/A	N/A
Recall (overall)	0.323 (over 3 runs)	N/A	0.54 (over 20 runs)	N/A	0.38



(a) Atomic axioms (b) Complex axioms

Figure 6.5: Possibility and generality distribution of the discovered axioms.

Figure 6.5 plots the generality of discovered axioms against their possibility degree. Most discovered axioms are highly possible ($\Pi(\phi)$ close to 1) and most are supported by a large number of facts (instances) both in the atomic and complex case. In terms of generality, some discovered axioms have a particularly high generality, i.e. true axioms, such as DisjointClasses(dbo:Article dbo:Image) ($g_\phi = 2,220,106$) or DisjointClasses(dbo:Image ObjectUnionOf(ObjectUnionOf(dbo:Album dbo:TelevisionShow) dbo:Website)) ($g_\phi = 190,783$). This can be explained by the existence of classes supported by a huge number of instances (like dbo:Article or dbo:Image) in the content of the generated axioms.

6.3.5 Limitations

The proposed model is capable of discovering highly accurate and general axioms, however, the dependence on SPARQL endpoints (i.e., query engines) for testing mined axioms against facts, i.e. instances, in large RDF repositories limits the performance of the method. In addition, evaluating the effectiveness of the method requires the participation of experts in specific domains, in particular, the use of a

Gold Standard, which is directly proportional to the number of concepts. Hence, the extracted axioms are limited to the classes relevant to a small scope of topics, namely the Work topic of DBpedia. Also, complex axioms are defined with the help of relational operators of intersection and union, which can also be mechanically derived from the known atomic axioms.

6.4 Learning Complex Axioms containing Value and Existential Restriction

In order to overcome the above limitations as well as to enhance the diversity of discovered types of axioms indicated in previous experiments, we propose a new approach. In detail, we develop an *objective benchmark* introduced in Section 6.4.2 for evaluating the effectiveness of the system which is bound by applying a *training-testing model*. Additionally, the type of class disjointness axioms is extended to include existence restriction $\exists r.C$ and value restriction operators $\forall r.C$, where r is a property and C a class, which cannot be mechanically derived from a given set of atomic axioms. In particular, we only consider the case of binary axioms such as DisjointClasses(C_1 C_2) where C_1 and C_2 can be atomic or complex classes like DisjointClasses(Building ObjectSomeValuesFrom(hasWings Animals)). It is important to mention that, to the best of our knowledge, no other method has been proposed so-far in the literature to mine the Web of data for class disjointness axioms involving complex class expressions with existential and value restrictions in addition to conjunctions.

The grammar is updated to suit these changes. A set of candidate axioms is also improved in the evolutionary process through the use of evolutionary operators of crossover and mutation. Finally, the final population of generated axioms is evaluated on the full RDF dataset, specifically the whole DBpedia, which can be considered as the objective benchmark eliminating the need of domain experts to evaluate the ability of generating axioms on a wider variety of topics. The evaluation of generated axioms in each generation of the evolutionary process is thus performed

on a reasonably sized data sample, which alleviates the computational cost of query execution and enhances the performance of the method.

6.4.1 GE Characteristics

BNF Grammar Pattern

As we did for the construction of the BNF grammar pattern 6.3.1, we follow the approach proposed in Section 4.3.1 to structure another BNF grammar pattern which still ensures that changes in the contents of RDF repositories do not require the grammar to be rewritten. However, we specify it to mine only disjointness axioms involving at least one complex axiom, containing a relational operator of existence restriction $\exists$ or value restriction $\forall$, i.e., of the form $\exists r.C$ or $\forall r.C$, where r is a property and C is an atomic class. The remaining class expression can be an atomic class or a complex class expression involving an operator out of $\sqcap$, $\exists$ or $\forall$. The BNF grammar pattern is thus structured as follows:

BNF Grammar Pattern 6.4.1

% Static part

```
(r1) Axiom := ClassAxiom
(r2) ClassAxiom := DisjointClasses
(r3) DisjointClasses := 'DisjointClasses' '(' ClassExpression1 ' 'ClassExpression2 ')'
(r4) ClassExpression1 :=   Class                  (0)
                         | ObjectSomeValuesFrom   (1)
                         | ObjectAllValuesFrom    (2)
                         | ObjectIntersection     (3)
(r5) ClassExpression2 :=   ObjectSomeValuesFrom   (0)
                         | ObjectAllValuesFrom    (1)
(r6) ObjectIntersectionOf := 'ObjectIntersectionOf' '(' Class ' ' Class ')'
(r7) ObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '(' ObjectPropertyOf ' ' Class ')'
(r8) ObjectAllValuesFrom  := 'ObjectAllValuesFrom' '(' ObjectPropertyOf ' ' Class ')'
```

% Dynamic part - Primitives

```
(r9) Class := % production rules are constructed by using SPARQL queries
(r10) ObjectPropertyOf := % production rules are constructed by using SPARQL queries
```

The production rules for the two primitives in dynamic part, namely Class and ObjectPropertyOf, are filled by the SPARQL queries mentioned in Equations (4.2) and 4.3, respectively, to extract atomic classes and properties (represented by their IRI) from the RDF dataset.

Example 6.4.1 (RDF data and Primitives) *An example representing a small excerpt of an RDF triple repository is the following:*

```
PREFIX dbr: http://dbpedia.org/resource/
PREFIX dbo: http://dbpedia.org/ontology/
PREFIX dbprop: http://dbpedia.org/property/
PREFIX rdf: http://www.w3.org/1999/02/22\-rdf-syntax-ns\#

dbr:Amblycera       rdf:type            dbo:Animal.
dbr:Salweenia       rdf:type            dbo:Plant.
Dbr:Himalayas       rdf:type            dbo:NaturalPlace.
dbr:Amadeus         rdf:type            dbo:Work.
dbr:Cat_Napping     dbprop:director     dbr:William_Hanna.
dbr:With_Abandon    dbprop:artist       dbr:Chasing_Furies.
dbr:Idris_Muhammad  dbprop:occupation   dbr:Drummer.
dbr:Genes_Reunited  dbo:industry        dbr:Genealogy.
```

The productions for Class *and* ObjectPropertyOf *would thus be:*

```
(r9) Class :=  dbo:Animal       (0)
             |  dbo:Plant        (1)
             |  dbo:NaturalPlace (2)
             |  dbo:Work         (3)
(r10) ObjectPropertyOf :=  dbprop:director      (0)
                         |  dbprop:artist        (1)
                         |  dbprop:occuptation   (2)
                         |  dbo:industry         (3)
```

6.4.2 From A Training- Testing Model toward An Objective Benchmark

In order to evaluate the effectiveness of the system, we organize the dataset following the "*training-testing*" model shown in Figure 6.6. Specifically, the learning process is performed with the input data source derived from a *training dataset* consisting of RDF triples, in particular, a random sample of DBpedia version 2015-04 in English (the extraction for it will be presented in the later of this section). On the other hand, the evaluation of discovered axioms is based on a *testing dataset*, in particular, the entire DBpedia version 2015-04 in English(which contains 665,532,306 RDF triples). This testing dataset is considered as an *objective benchmark* which refers

to eliminating the subjective intervention of human experts and to enhancing for the scalability of our implementation.

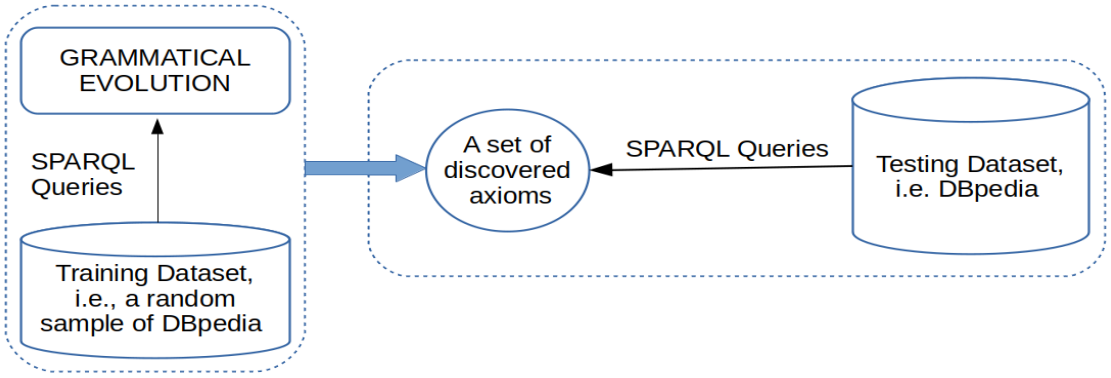


Figure 6.6: Workflow of class disjointness axioms discovery using GE in the training-testing model

Training Dataset Preparation

We randomly collect 1% of the RDF triples from DBpedia with the same version 2015-04 in English to create the *training dataset* [6] *(TD)*. A small linked dataset is generated, where RDF triples are interlinked with each other and the number of RDF triples accounts for 1% of the triples of DBpedia corresponding to each type of resource, i.e. subjects and objects. A demonstration of this mechanism to extract the sample training dataset is illustrated in Figure 6.7.

Let r be an initial resource for the extraction process, e.g., `http://dbpedia.org/ontology/Plant`; 1% of the RDF triples having r as subject, of the form $\langle r\ p\ r' \rangle$, and 1% of the triples having r as object, of the form $\langle r''\ p'\ r \rangle$, will be randomly extracted from DBpedia. Then, the same will be done for every resource r' and r'' mentioned in the extracted triples, until the size of the training dataset reaches 1% of the size of DBpedia. If the number of triples to be extracted for a resource is less than 1 (following the 1% proportion), we round it to 1 triple. In practice, we applied the proposed mechanism to extract a training dataset containing 6,739,240 connected RDF triples with a variety of topics from DBpedia.

[6] Available for download at `http://bit.ly/2Kl36wB`

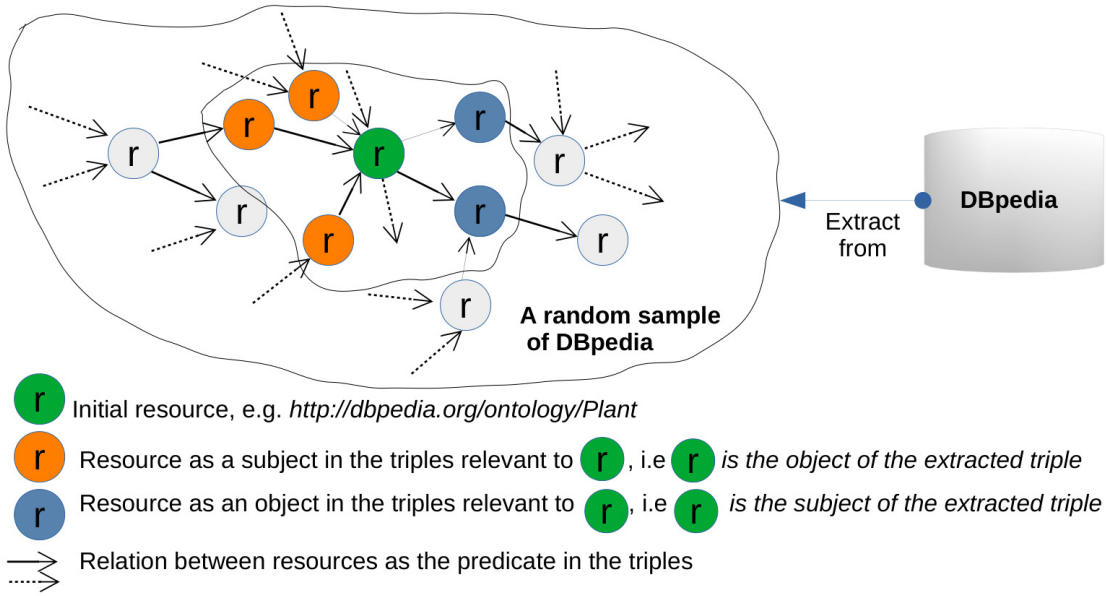


Figure 6.7: An illustration of the Training Dataset sampling procedure

6.4.3 Experimental Protocol

We use the BNF grammar pattern 6.4.1 introduced in Section 6.4.1. Given how the grammar was constructed, the mapping of any chromosome of length ≥ 6 will always be successful. Hence, we can safely set $maxWrap = 0$. Furthermore, in order to investigate the ability of the method to discover class disjointness axioms for different parameter settings, we ran our algorithm in 20 different runs for each of 4 distinct population sizes, namely 1,000; 2,000; 5,000 and 10,000 individuals, respectively. In addition, to make fair comparisons possible, a set of milestones of total effort k (defined as the total number of fitness evaluations) corresponding to each population size are also recorded for each run, namely 100,000; 200,000; 300,000 and 400,000, respectively. The maximum numbers of generations $maxGenerations$ (used as the stopping criterion of the algorithm) are automatically determined based on the values of the total effort k so that $popSize \cdot maxGenerations = k$. Also, we reuse the improved fitness function indicated in Equation (6.11). The parameters are summarized in Table 6.5.

Table 6.5: Parameter values for GE.

Parameter	Value
Total effort k	100,000; 200,000; 300,000; 400,000
initLenChrom	6
pCross	80%
pMut	1%
popSize	1000; 2000; 5000; 10000
Fitness function f	$f(\phi) = g_\phi \cdot \Pi(\phi)$

6.4.4 Results & Discussions

We ran the GE method 20 times with the parameters shown in Table 6.5 on the BNF grammar 6.4.1. Full results are available online.[7]

The number of valid distinct axioms, i.e., axioms ϕ such that $\Pi(\phi) > 0$ and $g_\phi > 0$, discovered is listed in Table 6.6 and demonstrated in Figure 6.8.

Table 6.6: Number of valid distinct axioms discovered over 20 runs.

k \ popSize	1000	2000	5000	10000
100000	8806	11389	4684	4788
200000	6204	13670	10632	9335
300000	5436	10541	53021	14590
400000	5085	9080	35102	21670

Statistical Analyses

We have statistically compared the number of distinct valid axioms using different settings of *popSize* and total effort k. Overall, we can see a trend whereby the number of discovered axioms increases steadily during the early stage of evolution, i.e. for low values of k, before gradually decaying at the end of the process. This trend is most clearly visible when $popSize = 2,000$ and $popSize = 5,000$. This phenomenon may be due to the prevalence of *exploration* in the early phases of the evolutionary process, as opposed to *exploitation*, when the population, despite our efforts to preserve diversity, begins to converge towards few axioms with particularly high fitness. Depending on the population size, this may happen before reaching

[7] http://bit.ly/32YEQH1

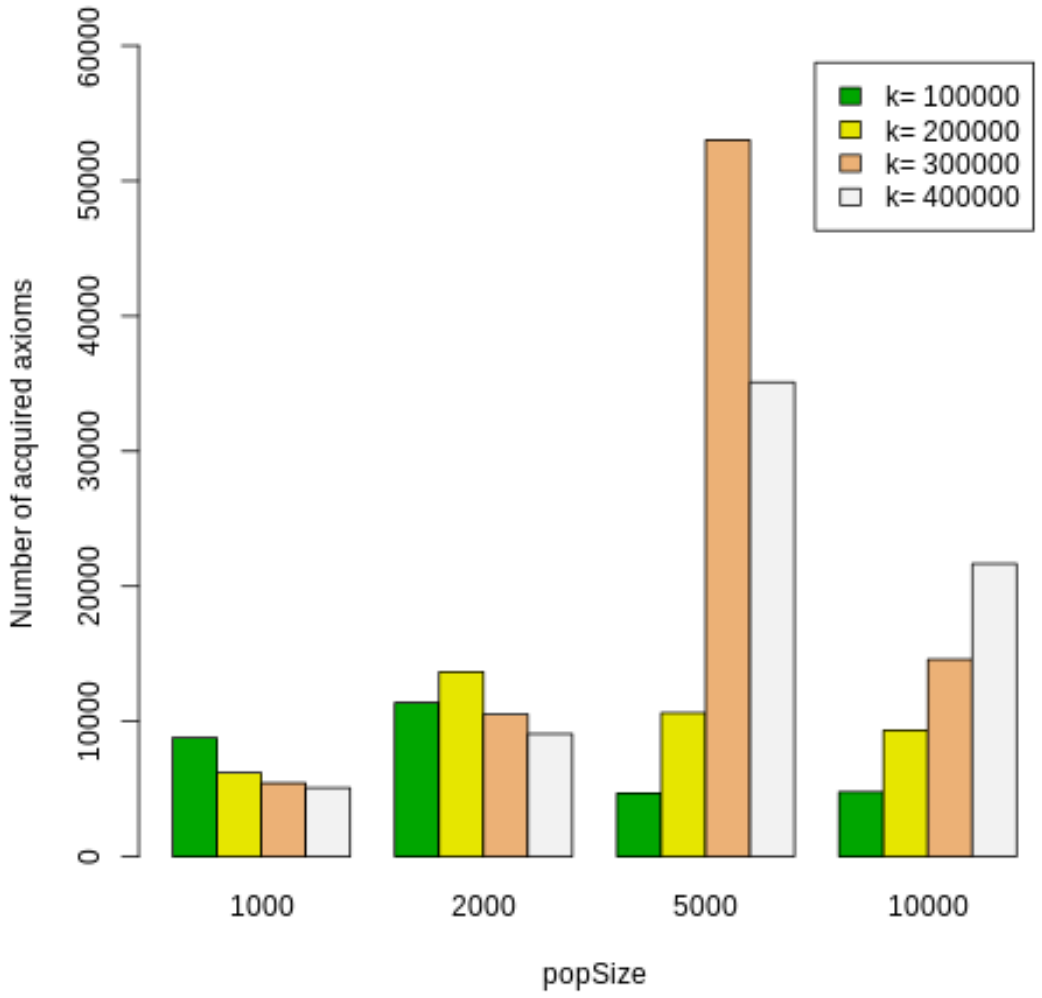


Figure 6.8: Number of axioms discovered over 20 runs.

the first milestone of total effort $k = 100,000$ (as it is the case for $popSize = 1000$) or in the generations following the last milestone, as one could expect to observe for $popsize = 10,000$, if the evolutionary process were allowed to continue.

In terms of the accuracy measurement of the results, given that the discovered axioms come with an estimated degree of possibility (introduced Section 5.3.2), which is essentially a fuzzy degree of membership, we propose to use a *fuzzy extension* of the usual definition of *precision*. According to which, the precision values are computed based on the following definition of a *fuzzy set cardinality* introduced by De Luca et al. in [DT72] as follows:

Definition 6.4.1: Fuzzy Set Cardinality [DT72]

Given a countable universe set Δ, the cardinality of a fuzzy set F is defined as follows:

$$\|F\| = \sum_{x \in \Delta} F(x), \tag{6.12}$$

In our case, we may view $\Pi(\phi)$ as the degree of membership of axiom ϕ in the (fuzzy) set of the "positive" axioms. The value of precision can thus be computed against the test dataset, i.e. DBpedia 2015-04 according to the formula

$$\text{precision} = \frac{\|\text{true positives}\|}{\|\text{discovered axioms}\|} = \frac{\sum_\phi \Pi_{DBpedia}(\phi)}{\sum_\phi \Pi_{TD}(\phi)}, \tag{6.13}$$

where Π_{TD} and $\Pi_{DBpedia}$ are the possibility measures computed on the training dataset and DBpedia, respectively. The results in Table 6.7 confirm the high accuracy of our axiom discovery method with a precision ranging from 0.969 to 0.998 for all the different considered sizes of population and different numbers of generations (reflected through the values of total effort).

Table 6.7: Average precision per run ($\pm$***std***)

k \ *popSize*	1,000	2,000	5,000	10,000
100,000	0.981 ± 0.019	0.999 ± 0.002	0.998 ± 0.002	0.998 ± 0.003
200,000	0.973 ± 0.024	0.979 ± 0.011	0.998 ± 0.001	0.998 ± 0.002
300,000	0.972 ± 0.024	0.973 ± 0.014	0.993 ± 0.007	0.998 ± 0.001
400,000	0.972 ± 0.024	0.969 ± 0.018	0.980 ± 0.008	0.998 ± 0.001

We also see that the accuracy remains stable across different values of total effort k in the case of large populations like $popSize = 10,000$, whilst there is an opposite trend in the case of smaller populations, where the values decrease slightly as the total effort k increases. This surprising behavior suggests that the method tends to *overfit* individuals in the population after a high number of generations (reflected by the values of total effort). This overfitting may be the only way to achieve higher fitness values (as computed against the training set), whereas the evaluated axioms

actually turn out to be incorrect when evaluated against the test dataset, i.e the full DBpedia. We can witness this phenomenon more clearly from the plots illustrating the distribution of axioms in terms of possibility and generality shown in Table 6.8. Even though most discovered axioms are highly possible ($\Pi(\phi)$ close to 1), the number of highly general axioms possessing a lower possibility increases slightly as total effort k increases. This suggests that the evolutionary process should be stopped early before axioms begin overfitting the training dataset. Indeed, with the same value of total effort, the larger populations, which correspond to a lower number of generations, as it is the case for $popSize = 10,000$, allow the method to discover axioms that correctly generalize to the full DBpedia and the evidence of the precision values in Table 6.7 seems to confirm this hypothesis.

Analyses of Axiom Contents

In order to obtain a more objective evaluation, we analyze in detail the axioms discovered by the algorithm with this best setting. First, we witness that together with the mandatory class expression containing the $\forall$ or $\exists$ operator, most extracted disjointness axioms contain an atomic class expression. This may be due to the fact that the support of atomic classes is usually larger than the support of a complex class expression. We also analyse axioms containing complex expressions in both their members. These axioms are less general, even though they are completely possible. An example is the case with DisjointClasses(ObjectAllValuesFrom(dbprop:author dbo:Place) ObjectAllValuesFrom(dbprop:placeofBurial dbo:Place))($\Pi(\phi) = 1.0$; $g_\phi = 4$), which states that "*what can only be buried in a place cannot be what can only have a place as its author*".

We also observe that some discovered axioms have a particularly high generality, as it is the case with DisjointClasses(dbo:Writer ObjectAllValuesFrom(dbo:writer dbo:Agent)) ($\Pi(\phi) = 0.982$; $g_\phi = 79,464$). This can be explained by the existence of classes supported by a huge number of instances (like dbo:Agent or dbo:Writer). From it, we might say that it is quite possible that "*writers are never written by agents*". Another similar case is axiom DisjointClasses(dbo:Journalist

ObjectAllValuesFrom(dbo:distributor dbo:Agent)) ($\Pi(\phi) = 0.992$; $g_\phi = 32,533$) whereby in general "*journalists are not distributed by agents*", although it would appear that some journalists are!

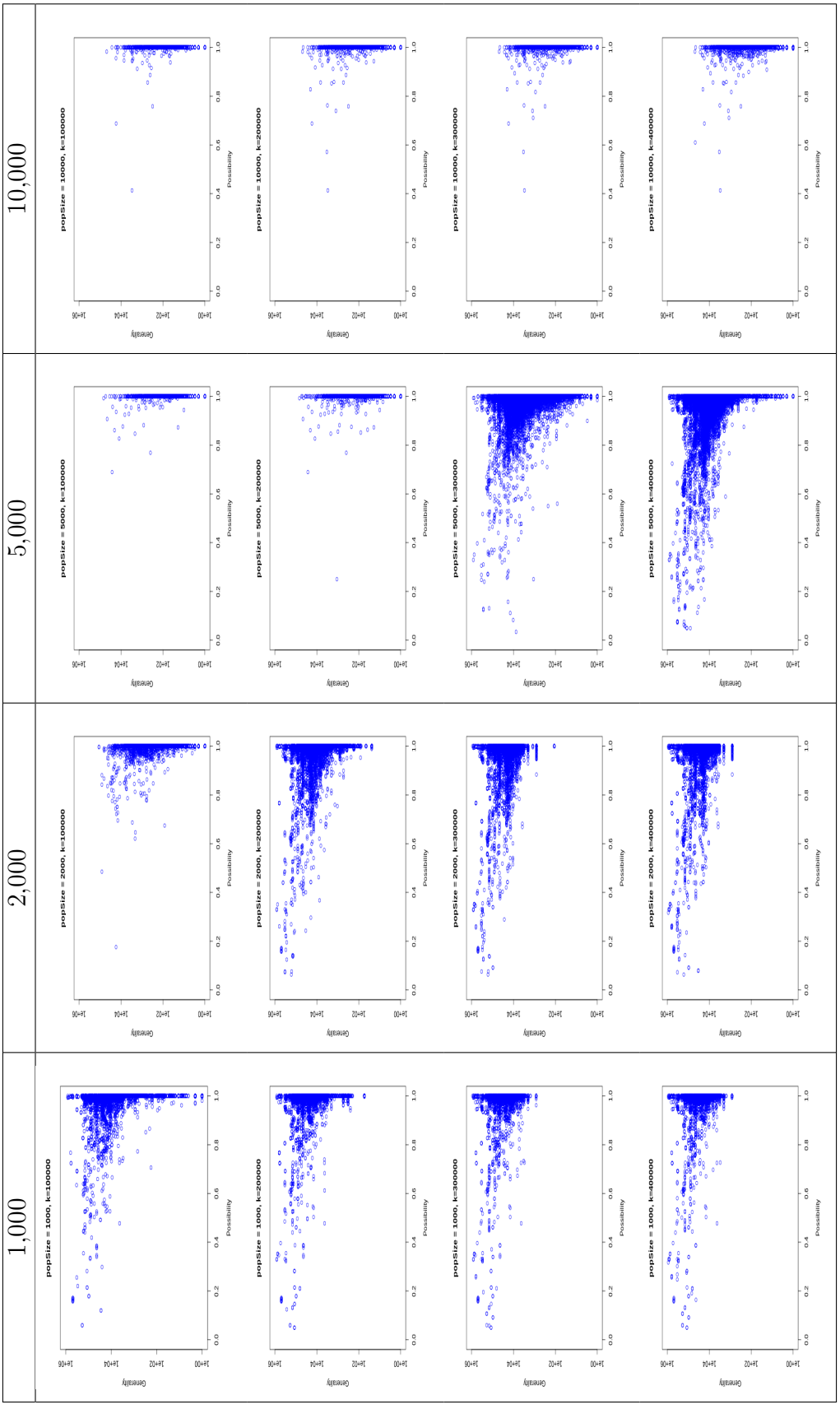
Finally, we analyze an example of a completely possible and highly general axiom, DisjointClasses(dbo:Stadium ObjectAllValuesFrom(dbo:birthPlace dbo:Place)) ($\Pi(\phi) = 1.0$; $g_\phi = 10,245$), which we can paraphrase as "*stadiums cannot have a place as their birthplace*". Knowing that Stadium and Place are *not* disjoint, this axiom states that Stadium and ∀birthPlace.Place are in fact disjoint; in addition, ∀.birthPlace.Place, i.e., "(people) whose birthplace is a place" is a class with many instances, hence the high generality of the axiom.

Generally, the experimental results confirm that the proposed method is capable of discovering highly accurate and general axioms containing the relational operators of existential quantification ∃ and value restriction ∀ on a wide variety of topics from DBpedia. A *training-testing* approach is also implemented to solve current limitations of performance and obtain a fair and objective assessment of its accuracy.

6.5 Summary

The works in this section aim at mining axioms in term of both atomic and complex axiom containing different types of relational operators. Two variants of the fitness function are proposed which ensure the obtained axioms being highly credible and general. In addition, two evaluation benchmarks, namely *subjective* and *objective*, are deployed to evaluate the performance of the models.

Table 6.8: Possibility and generality distribution of the discovered axioms for different population sizes (columns) and total efforts $k = 100,000, \ldots, 400,000$ (rows).



7

A Multi-Objective GE Approach to Class Disjointness Axioms Discovery

Contents

7.1 Introduction 118
7.2 Evolutionary Multi-Objective Optimization 119
7.2.1 Multi-Objective Optimization 119
7.2.2 Multi-objective Evolutionary Algorithms 120
7.3 Multi-Objective GE for Axiom Discovery 125
7.3.1 Multi-Objective Evaluation Framework 125
7.3.2 Experimental Protocol 128
7.3.3 Results & Discussions 129
7.4 Summary 132

7.1 Introduction

Within the evolutionary process, the evaluation framework quantifies the quality of axioms, which is the base for selecting individuals (solutions) for the recombination, mutation, and replacement phases. In the previous chapter, the evaluation framework based on possibility theory is introduced to determine the fitness values of generated axioms in the evolutionary cycle, i.e. the credibility and generality of axioms. However, the selection pressure in each phase of the evolutionary process

tends naturally to drive the diversity of the population down. In addition, there possibly exist candidate axioms in the population which have the high fitness values, but are invalid following the benchmarks (i.e., Gold Standard or testing dataset). This can be derived from unsuited fitness function in evaluation framework which based on a single criterion. In reality, the experimental analyses are specific evidence for these phenomena. A possible solution which aims at enhancing the capability to explore the diverse regions of the solution (axiom) space is to use multiple objectives optimized at one time known as *Multi-objective optimization* (for shortly, MOO). The goal of MOO is to find multiple solutions representing the possible non-dominated trade-offs among the objective functions known as a set of solutions lying on the optimal front. In addition, a set of obtained solutions is sought for that is also diverse enough to represent the entire range of the front known as the *Pareto-optimal front*. In this chapter, we first investigate the concepts and characteristics in terms of MOO and its variant in a heuristic approach of evolutionary algorithms known as *Evolutionary multi-objective optimization* (for shortly, *EMO*) [Deb11; ZLB04]. Along the lines of the studies using GE to mine class disjointness axioms, we extend the single-objective GE approach introduced in the previous chapter as a multi-objective problem called *multi-objective GE* (for shortly, *MOGE*) described in Section 7.3. Specifically, we use MOGE to refine the evaluation of candidate axioms satisfying a trade-off between a set of objectives that improves the adaptive fit of a population of candidate axioms constrained by two independent criteria, i.e. the credibility and generality. The experiments and results are also performed based on the new MOGE model. Conclusions are provided in the last section of the chapter.

7.2 Evolutionary Multi-Objective Optimization

7.2.1 Multi-Objective Optimization

A *Multi-objective Optimization* (MOO) [Deb11] problem involves a number of objective functions constituting a multi-dimensional objective space, in addition to the decision variable space. Specifically, a solution to a MOO problem is a vector of decision variables $x = (x_1, x_2, ..., x_n)^T$ in the decision space X. For each x, there

exists an objective vector $y = (y_1, y_2, ..., y_n)^T$ in the objective space Y mapped by a function $f : X \to Y$ with $y = f(x) = (f_1(x), f_2(x), ..., f_n(x))$. The term *domination* is used for the situation of comparing two solutions $x^{(1)}$ and $x^{(2)}$ defined as follow:

Definition 7.2.1: Domination

A solution $x^{(1)}$ *dominates* the other solution $x^{(2)}$ ($x^{(1)} \succ x^{(2)}$) if and only if $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives (e.g., for a *minimization* problem $\forall i \in [1..n]$ $y_i^{(1)} \leq y_i^{(2)}$ where $y^{(1)} = f_i(x^{(1)})$ and $y^{(2)} = f_i(x^{(2)})$) and $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective (i.e., $\exists i \in [1..n] y_i^{(1)} < y_i^{(2)}$).

The set of optimal solutions in the decision space X is called as *Pareto-optimal solutions* or *Pareto set*. In addition, there are corresponding optimal objective vectors, i.e. points, in the objective space Y, called as *Pareto-optimal front* or *non-domination front*. In MOO, all objectives are equally important, i.e., finding the optimum solution cannot be based on one objective alone while skipping other objectives. The goal of MOO is to find multiple solutions representing the possible non-dominated trade-offs among the objective functions, i.e., a set of solutions lying on the Pareto-optimal front. In addition, a set of obtained solutions is sought for that is also diverse enough to represent the entire range of the Pareto-optimal front.

An illustration of Pareto front for a *minimization* problem containing a set of solutions set that are not dominated by any other feasible solutions, i.e., Pareto optimal solutions, is indicated in Figure 7.1. In it, the blue points represent feasible choices in which the smaller values are preferred to the larger ones. Points $x^{(1)}$ and $x^{(2)}$ are in vector x of decision variables in the decision space X. A point as point $x^{(m)}$ is not on the Pareto frontier because it is dominated by both other points $x^{(1)}$ and point $x^{(2)}$. Points $x^{(1)}$ and $x^{(2)}$ are not strictly dominated by any other in all objectives represented in a objective vector $y = (y_1, y_2)$ in the objective space Y, where $y_1 = f_1(x)$, $y_2 = f_2(x)$, thus, do lie on the Pareto front.

7.2.2 Multi-objective Evolutionary Algorithms

Evolutionary multi-objective optimization (for short, *EMO*) involving *Multi-objective evolutionary algorithms* (for short, *MOEA*) [Deb11; ZLB04] is one of engineering

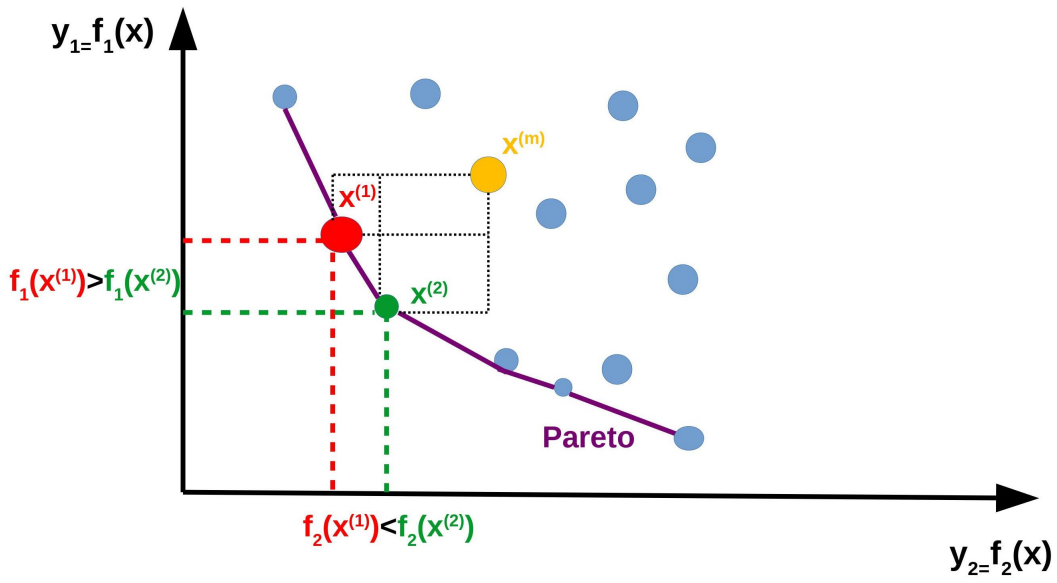


Figure 7.1: An illustration of Pareto front for a *minimization* problem

optimization techniques based on stochastic search strategies of evolutionary algorithms which follows the goal of MOO but refers to finding multiple non-dominated points as close to the Pareto-optimal front as possible, i.e., a *Pareto-optimal front approximation*, with respect to the trade-off among objectives. Also, it provides operators, i.e., recombination and mutation operators, to constantly improve the evolving non-dominated points. MOEA can lessen the computational complexity resulting in the expensive cost in generating Pareto set. They may not reach the optimal trade-offs, but try to find a set of good approximating solutions whose vectors are not too far away from the optimal objective vectors.

The fundamental goals of MOEA are not only to guide the search toward the Pareto set but also to maintain the diversity of the set of non-dominated solutions. While the former concerns *mating selection* (i.e., *parent selection* mentioned in Section 4.3.2) in which the assignment of fitness values refer to satisfying multiple optimization criteria, the latter is relevant to selection in general with respect to avoiding the convergence in the population with respect to both the objective space and decision space, i.e., limiting the identical solutions in the population. In order to deal with the first task, the studies in terms of fitness assignment and selection

in multi-objective optimization problems are investigated, in particular, both must satisfy multiple objectives with multi-criteria optimization problem. In general, there are different strategies [ZLB04] in terms of fitness assignment including *aggregation-based* [HL92; IM98], *criterion-based* [Sch85; Kur91] and *Pareto-based* [Deb+02; FF93; GH88; ZLT01; ZT99]. Meanwhile, the tasks involving the second goal focus on enhancing the diversity issue of the current Pareto set approximation of non-dominated solutions. In fact, there are various techniques in order to solve this task, e.g., Kernel methods [Sil86], fitness sharing [FF93; HNG94; SD94a], and nearest neighbor techniques [KC99; ZT99]. In addition, the studies addressing both tasks concern *elitism* that in this particular case is applied to preventing non-dominated solution from being lost. Elitist MOEA mostly obey the combination of the domination criterion and additional information to select the individuals for the elitist group at each generation. In terms of the domination criterion, the elitist group only consists of the current approximation of the Pareto set, i.e., dominated solutions are eliminated, which ensures non-dominated solution are preferable to dominated ones. Also, additional information concerns the density or the time when the individual put into the elitist group.

In reality, MOEA have been developing with the presence of various algorithms. The development of phases of MOEA is divided into different phases depicted in [HZL19]. Also, a survey of MOEA based on their own characteristics is introduced in [VDP15]. We will not discuss all algorithms here in detail which is not necessary for the purpose of application to solve our specific problem later. Instead of that, we only investigate one of the well-known MOEA, *NSGA-II* which concentrates on finding non-dominated solutions in addition to elitist and diversity preserving mechanisms.

NSGA-II

The *non-dominated sorting genetic algorithm*, for shortly, *NSGA-II* [Deb+02] being an improved version of NSGA [SD94b], is one of the most efficient MOEA proposed by K.Deb, which is suitable for the application in complex and real-world MOO problems. NSGA-II alleviates the obstacles in the previous version of NSGA in

terms of high computational complexity of non-dominated sorting, lack of elitism strategies and the need of sharing parameters [Deb+02]. In particular, NSGA-II provides a better sorting algorithm, incorporates an *elitist* principle and no sharing parameter needs to be chosen a *priori*. In order to obtain a uniformly spread of the Pareto-optimal front, NSGA-II also employs a *fast density estimation* in terms of computing the *crowding distance* of solutions in their own front and *crowded comparison operators* to guide the selection in each phase of the algorithm. The general principle of NSGA-II can be presented as follows:

1. *Population Initialization*: A population P_0 of size n is initialized based on the problem range and constraints.

2. *Non-dominated Sorting*: Evaluating the objective functions for the initial population. Each solution is assigned a fitness value according to its non-domination level.

3. *Genetic Operation*: Binary tournament selection, crossover and mutation are applied on P_0. Offspring population Q_0 of size n is created.

4. *Population Combination*: The combination of the offspring and parent population, i.e., $R_i = P_i \cup Q_i$ is performed to maintain the best solutions for the new population, i.e., *elitism*.

5. *Non-dominated Sorting*: All individuals in the combined population R_i are sorted by using the fast non-dominant sorting algorithm which returns a list of the non-dominated fronts according to ranking levels $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, ...\mathcal{F}_n)$ of the population R_i.

6. *Crowding Distance Calculation*: The crowding distance in fronts $\mathcal{F}$ of the sorted population R_i is calculated based on the *Euclidean distance* between the specific individuals and two adjacent individuals in each front based on the objectives.

7. *Selection and Offspring Population Generation*: The new population P_{i+1} is generated by adding the individuals from the *first* front until the population size exceeds n. Then, the individuals in the last accepted front are sorted according to front ranking and the first n individuals are picked. Tournament selection is used to choose the parents, whose selection criterion is based on *crowded comparison operators*. Crossover and mutation are performed on the selected parents in order to create a new offspring population Q_{i+1} of size n.

It is important to note that, the first three steps (1-3) appear only once in the random initial phase, while the remaining steps (4-7) are performed iteratively from the first generation on. An illustration of the above NSGA-II procedure is given in Figure 7.2.

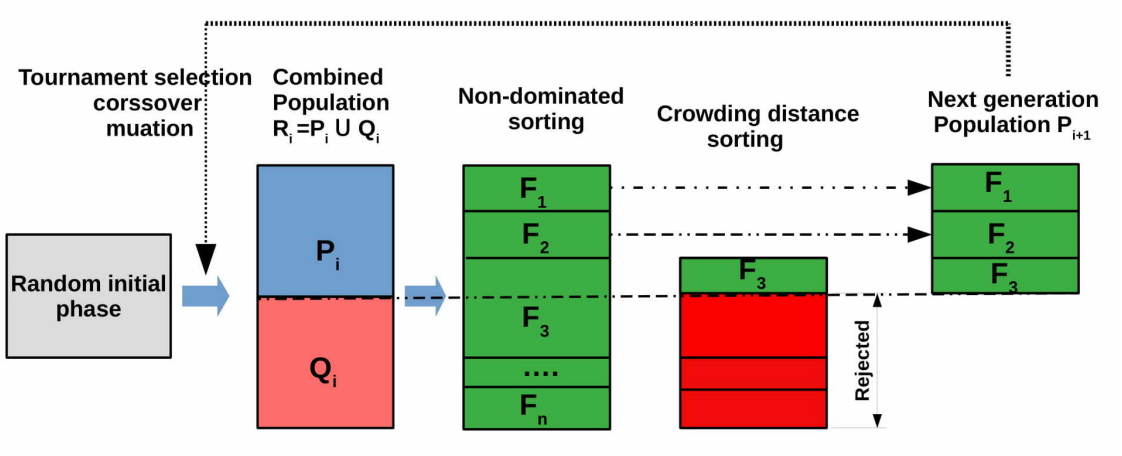


Figure 7.2: NSGA-II mechanism

Practical MOEA Framework

In reality, there are various practical frameworks developed for MOEA. One of them is a free and open source Java library for developing and experimenting with MOEA published on the site `http://moeaframework.org/`, which is an extensible framework for rapidly designing, developing, executing, and statistically testing MOEA. We exploit the advantages of this library, incorporated with GE, to develop a Multi-objective GE for discovering axioms.

7.3 Multi-Objective GE for Axiom Discovery

As a particular case of MOEA, the approach we propose comprises the integration of GE in MOEA, in particular, using NSGA-II for axiom discovery, which we call Multi-Objective GE (MOGE). Basically, the mechanism of MOGE is quite similar to the one of MOEA, except that we define multi-objective problems using integer arrays called *codons* as decision variables. The codons do not define axioms, i.e., the programs, themselves, but provide instructions for deriving axioms using the BNF grammar through the mapping process, as explained in Section 4.3. In practice, we focus on discovering axioms containing relational operators defined by the grammar 6.4.1. In addition, we will not pay attention to the description of the changes of codes in terms of GE embedding into NSGA-II of a given practical MOEA framework. Instead, we aim at enhancing several points in terms of the evaluation framework, whose results directly influence the non-dominated sorting.

7.3.1 Multi-Objective Evaluation Framework

The goodness of an axiom is determined by its dominance, whereby it obtains a score on each objective that is not dominated by the corresponding score of another axiom. To derive such axioms, we extend the classic GE approach presented in the previous chapters to MOGE. More importantly, we develop separate objective functions to evaluate the fitness of each axiom. In order to ensure the diversity of the obtained axioms, a scoring of the similarity of each axiom to the other axioms in the population (essentially, a local phenotypic crowding measure) is also considered in the evaluation framework. In addition to the use of axiom scoring according to possibility and generality, applied in Chapter 6, we add a new scoring, called *similarity*. Then, we propose two objective functions for this MOGE model.

Similarity measure

This measure characterizes the similarity of an axiom ϕ, $s(\phi)$, to the population of n axioms which is quantified by the average of similarity metrics $s(\phi, a_i)$ between

axiom ϕ and each axiom a_i in the population:

$$s(\phi) = \frac{1}{n-1} \sum_{i=1;a_i \neq \phi}^{n} s(\phi, a_i) \tag{7.1}$$

In order to measure the similarity coefficient $s(\phi)$ as in the above formula, the similarities $s(\phi, a_i)$ need to be computed. As mentioned in Section 6.4.1, class disjointness axioms are structured in the form of binary axioms of the form $\phi \equiv DisjointClasses(A, B)$ and $a_i \equiv DisjointClasses(C, D)$ where A, B, C, D can be atomic expressions or complex expressions containing relational operators. We define the similarity between two axioms based on the similarities between pairs of expressions as

$$s(\phi, a_i) = \max\{s(A, C), s(A, D), s(B, C), s(B, D)\} \tag{7.2}$$

Expressions in each axiom are represented in the form of binary trees where each node can be an atomic class or a relational operator, namely existential quantification ($\exists$), value restriction ($\forall$), or intersection ($\sqcap$) operators. Determining each similarity between expressions, e.g., $s(A, C)$, is performed on the corresponding binary trees t_1 and t_2. Binary trees are traversed simultaneously and each pair of corresponding nodes (p_j, q_j) in both trees, i.e., p_j in t_1 and q_j in t_2, is compared to each other and the value returned is the similarity between two nodes, i.e., $s(p_j, q_j)$, according to Table 7.1. One notable point is that if both nodes represent atomic classes, the value returned is 1 if the two nodes represent the same class; otherwise the value returned is 0. Each similarity between expressions, e.g. s(A,C), is defined as

$$s(A, C) = \frac{1}{k} \sum_{j=1}^{k} s(p_j, q_j) \tag{7.3}$$

where k is the number of pairs defined by the number of nodes in the smallest tree.

Example 7.3.1 (Similarity Degree Between Two Axioms) *Given two axioms ϕ_1 and ϕ_2 where*

- $\emptyset_1$ = DisjointClasses(ObjectSomeValuesFrom(dbo:industry dbo:Work) ObjectSomeValuesFrom(dbo:director dbo:Plant)

- $\emptyset_2$ = DisjointClasses(ObjectAllValuesFrom(dbo:artist dbo:Work) ObjectIntersectionOf(dbo Animal dbo:Plant))

Table 7.1: Matrix for the comparison between nodes

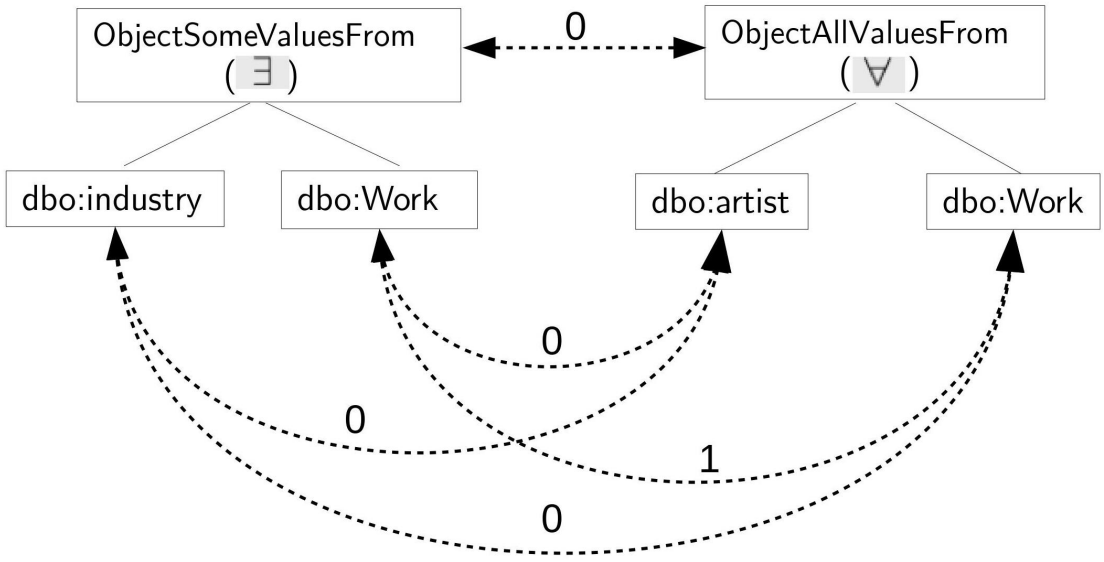
Node	Atomic class	⊓	∃	∀
Atomic class	0 or 1	0	0	0
⊓	0	1	0	0.5
∃	0	0	1	0
∀	0	0.5	0	1

In order to be more convenient for the explanation later, we use symbols A, B, C, D to stand for the expressions of the axioms as follow:

- *A =ObjectiveSomeValuesFrom(dbo:industry dbo:Work)*
- *B =ObjectiveSomeValuesFrom(dbo:director dbo:Plant)*
- *C =ObjectiveAllValuesFrom(dbo:artist dbo:Work)*
- *D =ObjectiveIntersectionOf(dbo:Animal dbo:Plant)*

In order to compute the similarity degree between ϕ_1 and ϕ_2, we need to define the similarity scores of the expression pairs (A, C), (A, D), (B, C), (B, D), respectively.

As in the case for the pair (A, C), the binary trees representing the expressions A and C are built, whose each node can be a relational operator, class or relation. Then, the comparisons between pairs of nodes are carried out based on the matrix in Table 7.1 and illustrated below:



Then, the similarity degree of the pair is induced by applying Equation (7.3) as follows:

$$s(A, C) = \frac{1}{5}(0 + 0 + 1 + 0) = 0.2$$

Similarly, we calculate the similarity degrees of the remaining pairs (A, D), (B, C) *and* (B, D) *as follows:*

$$s(A, D) = \frac{1}{5}(0.5 + 0 + 0 + 0) = 0.1$$
$$s(B, C) = \frac{1}{5}(0 + 0 + 0 + 0) = 0$$
$$s(B, D) = \frac{1}{5}(0.5 + 0 + 0 + 1) = 0.3$$

Finally, we apply Equation (7.2) to compute the similarity score between ϕ_1 *and* ϕ_2 *as follows:*

$$s(\phi_1, \phi_2) = \max\{0.2, 0.1, 0, 0.3\} = 0.3$$

Objective Functions

We propose two objective functions, f_1 and f_2, used in our approach, which aim at obtaining axioms ϕ that maximize the value of possibility $\Pi(\phi)$ and generality $g(\phi)$ while not being too similar among themselves, i.e., minimize the similarity score $s(\phi)$, as follows:

$$\begin{cases} \text{Maximize } f_1 = \Pi(\phi) \cdot \sqrt{1 - s(\phi)^2} \\ \text{Maximize } f_2 = g_\phi \cdot \sqrt{1 - s(\phi)^2} \\ \text{Where } 0 \leq \Pi(\phi) \leq 1;\ g_\phi \geq 0\ ;\ 0 < s(\phi) < 1 \end{cases} \tag{7.4}$$

7.3.2 Experimental Protocol

As mentioned above, we follow the BNF grammar pattern 6.4.1 for generating class disjointness axioms involving complex restriction expressions. In order to make fair comparisons possible with the previous study introduced in Section 6.4.1, which only applies a single-objective approach, i.e. the *GE* method, a set of milestones of total effort k (defined as the total number of fitness evaluations) corresponding to each population size are also recorded for each run, namely 100,000; 200,000; 300,000 and 400,000, respectively. The maximum numbers of generations, *maxGenerations*

(used as the stopping criterion for the algorithm) are automatically determined based on the values of the total effort k, thus $popSize \cdot maxGenerations = k$. The parameters listed in Table 7.2 are like the GE parameters in Table 6.2.

Table 7.2: Parameter values for MOGE.

Parameter	Value
Total effort k	100,000; 200,000; 300,000; 400,000
initLenChrom	6
pCross	80%
popSize	1000; 2000; 5000; 10000

7.3.3 Results & Discussions

We ran the *MOGE* method for 20 distinct runs for each of the different parameter settings summarized in Table 7.2, using the BNF grammar defined in Section 6.4.1. The full set of valid distinct axioms, i.e., axioms ϕ such that $\Pi(\phi) > 0$ and $g_\phi > 0$ discovered are available online.[1] Statistics for automatically generated axioms are presented in Table 7.3. In addition, we can see in Figure 7.3 that the number of valid distinct axioms for most parameter settings, i.e., population size *popSize* and total effort k, mined by *MOGE* is significantly greater than those mined by the single-objective *GE* method. This means that the diversity of an extracted set of axioms is considerably enhanced when we use the *MOGE* method.

Table 7.3: Number of valid distinct axioms discovered over 20 runs

k \ popSize	1000	2000	5000	10000
100000	8084	16085	41320	50535
200000	8713	17400	41813	76804
300000	7970	17680	40303	70562
400000	8457	16258	40656	67722

Furthermore, we follow the use of the fuzzy extension of the usual definition of *precision* in Section 6.4.1 to measure the accuracy of our results.

[1] `https://bit.ly/38crj4M`

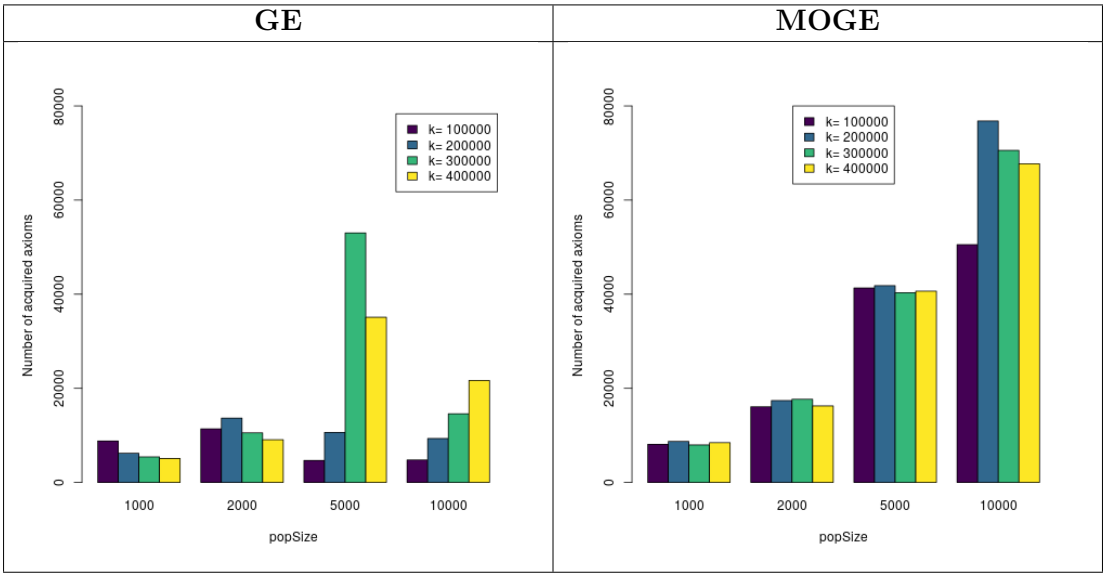


Figure 7.3: Statistical comparison about the number of axioms discovered over 20 runs between GE and MOGE method.

Table 7.4: Average precision per run ($\pm$***std***)

	GE				MOGE			
k \ *popSize*	1,000	2,000	5,000	10,000	1,000	2,000	5,000	10,000
100,000	0.981 0.019	0.999 ±0.002	0.998 ±0.002	0.998 ±0.003	0.988 ±0.007	0.990 ±0.005	0.989 ±0.003	0.996 ±0.001
200,000	0.973 ±0.024	0.979 ±0.011	0.998 ±0.001	0.998 ±0.002	0.989 ±0.007	0.990 ±0.004	0.987 ±0.004	0.988 ±0.004
300,000	0.972 ±0.024	0.973 ±0.014	0.993 ±0.007	0.998 ±0.001	0.989 ±0.007	0.989 ±0.003	0.986 ±0.004	0.986 ±0.003
400,000	0.972 ±0.024	0.969 ±0.018	0.980 ±0.008	0.998 ±0.001	0.989 ±0.008	0.990 ±0.003	0.985 ±0.004	0.984 ±0.004

The results shown in Table 7.4 confirm the high accuracy of the proposed *MOGE* method. The precision values are quite equivalent to the figures of *GE* method with the range from 0.984 to 0.996 for all the different considered population sizes and different numbers of generations (reflected through the values of total effort).

Figure 7.4 illustrates the distribution of axioms having $\Pi(\phi) > \frac{2}{3}$ in terms of the two objectives, i.e. possibility and generality, compared with the single-objective *GE* methods. We perform the comparison based on the results of the best setting, i.e., those yielding the largest number of obtained distinct axioms and the highest accuracy, for either method, i.e., $\{popSize = 10,000; k = 200,000\}$ and $\{popSize = 5,000; k = 300,000\}$, respectively. We can observe that the number of highly qualified axioms ($\Pi(\Phi) > \frac{2}{3}$ and $g_\Phi > 100$)) is maintained in the *MOGE*

system. More clearly, based on the specific resulting statistics, the number of obtained axioms from MOGE in the best setting is 38, 134, which is much greater than those extracted by the GE, i.e., 23, 767 axioms. In addition, with the smaller value of total effort k reflecting the cost of evaluations, i.e., $k = 200,0000$ compared with $k = 300,000$ in the GE method, *MOGE* is clearly more effective in inducing highly qualified axioms. We also show the distribution of the discovered axioms in this best setting in terms of similarity coefficient in Figure 7.5.

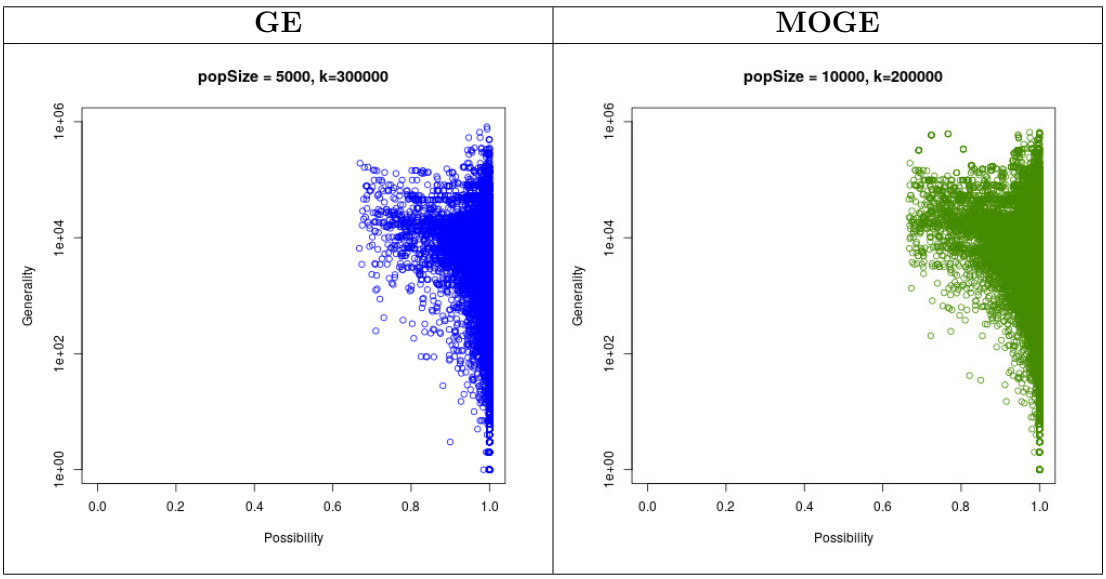


Figure 7.4: Possibility and generality distribution of the discovered axioms with $\Pi(\phi) > \frac{2}{3}$

The range of similarity scores recorded for these axioms lies below 0.35, which indicates a good diversity of the classes and properties in the components of axioms. Based on the given grammar, one part of the axioms is forced to contain a relational operator, i.e. $\exists$, $\forall$, or $\sqcap$, hence the overlap of the operators in the axioms does not allow the similarity score to be zero.

According to the results, we consider in detail the axioms discovered by the algorithm with this best setting. First, we witness that the number of obtained axioms containing the $\exists$ operator is slightly larger than the one of those with the $\forall$ operator, namely 40,122 and 36,682 axioms, respectively. However, together with the mandatory class expression containing the $\forall$ or $\exists$ operator, most extracted class disjointness axioms contain an atomic class expression. This may be due to the fact that the

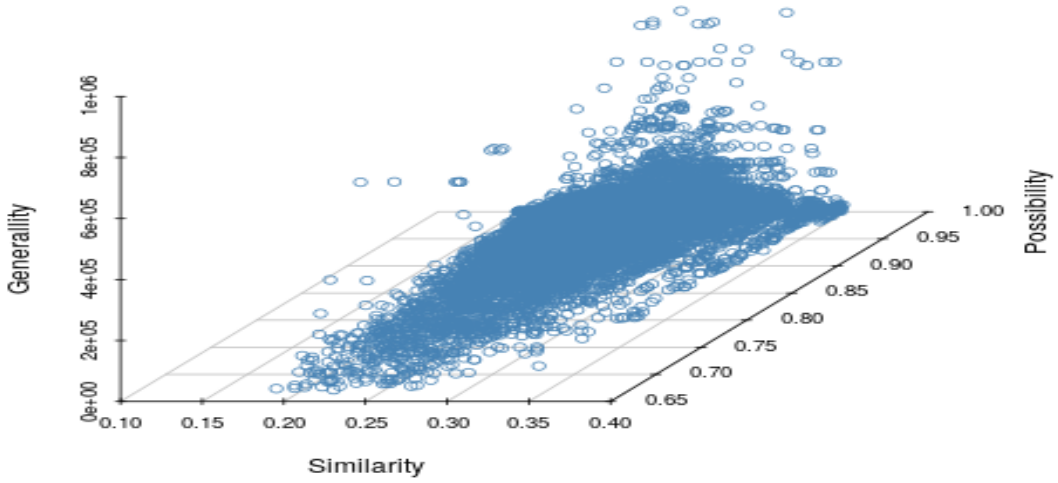


Figure 7.5: The distribution of the discovered axioms in terms of measures ($\Pi(\phi) > \frac{2}{3}$)

support of atomic classes is usually larger than the support of a complex class expression. Specifically, we obtain 7 axioms containing complex expressions in both their members. These axioms are less general, even though they are completely possible. An example is the case with DisjointClasses(ObjectAllValuesFrom(dbprop:operation dbo:MilitaryConflict) ObjectAllValuesFrom(dbprop:order dbo:MIlitaryUnit))($\Pi(\phi) = 1.0$; $g_\phi = 1$). Also, we analyze an example of a completely possible and highly general axiom, DisjointClasses(dbo:District ObjectSomeValuesFrom(dbo:birthPlace dbo:Place)) ($\Pi(\phi) = 1.0$; $g_\phi = 8,483$), which we can paraphrase as "*districts cannot have a place as their birthplace*". Knowing that District and Place are *not* disjoint, this axiom states that District and ∃birthPlace.Place are in fact disjoint; in addition, ∃birthPlace.Place, i.e., "(people) whose birthplace is a place" is a class with many instances, whence, the high generality of the axiom.

7.4 Summary

In this chapter, we presented a multi-objective extension to a grammar-based genetic programming approach to axiom discovery which consists of using two objectives plus a "similarity" score, which is in fact a sort of local phenotypic crowding factor. The experimental results confirm that the proposed method is capable of discovering

highly accurate and general axioms and is more effective when compared to the single-objective methods of the previous chapter.

8

Conclusions & Perspectives

Contents

8.1 Conclusions 134
8.1.1 A General GE Framework of OWL Axioms Discovery from RDF data 135
8.1.2 Evaluation Frameworks for Discovered OWL Axioms . . 136
8.1.3 Toward Learning OWL Class Disjointness Axioms . . . 137
8.1.4 Performance Evaluation Frameworks 138
8.2 Future Work 139

This chapter presents the outcomes of this thesis and develops some future directions. In the conclusions presented in Section 8.1, we highlight the main results relevant to OWL axioms discovery from RDF data. The chapter closes with several directions for future work in Section 8.2.

8.1 Conclusions

Along with the rapid extension of LOD consisting of an increasing number of new RDF data instances, the existing ontologies used as its schema-level knowledge model also need to be co-evolved. This involves the enrichment of ontologies with new knowledge defined in terms of axioms to enhance data quality and data cleaning in LOD. Exploiting ontological axioms in the form of logical assertions to be added

to an existing ontology can be useful for the automatic discovery of errors and inconsistencies in the structure of the ontology as well to infer new facts, thus increasing the deductive power of populated ontologies (or knowledge graphs).

In this thesis, we merely focus on the automated learning of axioms from recorded RDF facts which can be viewed as the first step of the enrichment process, i.e., *learning* steps whose outputs is the input for *replacement* steps applied to the existing ontologies. The process of learning from RDF data is viewed as a case of inductive reasoning and ontology learning. Based on the insight of Karl Popper [Pop35] that discovering new knowledge is essentially an evolutionary process, whereby hypotheses are generated by some heuristic mechanism and then tested against the available evidence, so that only the best hypotheses survive, we employed Grammatical Evolution, one type of evolutionary algorithm, to build the models for mining OWL axioms from an RDF data repository. While other methods are incapable of scaling up when the space of hypotheses, i.e. the axioms, becomes too large, and, as a consequence, their applicability is restricted to the discovery of relatively simple axioms, the application of an evolutionary heuristic method in our research overcome the limitations of other methods; specifically, it can handle the search for more complex axioms, whose search space is incomparably larger.

The main results of this thesis can be summarized into four aspects, presented in the next sections (from Section 8.1.1 to Section 8.1.4).

8.1.1 A General GE Framework of OWL Axioms Discovery from RDF data

We formalized a general framework for learning OWL axioms using GE, presented in Chapter 4. First, the construction of a BNF grammar for structuring OWL axioms is explained in Section 4.3.1. In order to avoid rewriting the whole grammar when there is any change in the contents of RDF repositories, the structure of the grammar is split into two parts: static and dynamic, respectively, with distinguished specifications. An evolutionary model for searching OWL axioms is then proposed (see Section 4.3.2) in which a population of candidate axioms is maintained by

Algorithm 1 and iteratively refined to find axioms with the highest level of credibility and generality. In addition to the standard implementation of GE found in the GEVA framework, we carried out different specific adaptations to the model by a series of algorithms. Initially, we built Algorithm 2 to generate a population containing a defined number of OWL axioms from integer strings with the initialized length. In order to avoid the loss of the fittest axioms, elitism selection was applied in the parent selection mechanism to keep the best axioms in the next generation. In order to maintain the diversity of the population and prevent the premature convergence, we used the deterministic crowding model (Algorithm 3) in survival selection.

8.1.2 Evaluation Frameworks for Discovered OWL Axioms

We used the model known as *axiom testing against RDF data* (see Section 5.2) [TFG17] to check discovered axioms whether they fit or explain the available RDF repository. This approach eliminated the reasoning tasks and the requirements of background knowledge which can be the obstacles in dealing with the big data of LOD. In addition, we adopted an axiom scoring heuristic based on possibility theory which is well-suited to the OWA where there is uncertain and insufficient information. We employed two measures in terms of the possibilistic framework (see Section 5.3.2), namely, *possibility* and *necessity*, to assess the credibility of axioms in addition to the generality measure defined through their extensions. We developed various functions to assess the fitness of OWL axioms based on the above measures. The computation of these measures is defined through performing the corresponding SPARQL queries.

- *Single-objective framework*: we developed two fitness functions based on the single-objective models which refer to finding the best axioms for a specific single criterion. Unlike the first function (see Equation (6.1)) involving both necessity and possibility, the second fitness function (see Equation (6.11)) dropped the necessity measure. Also, in order to remove the case of the components of an axiom being not supported by any facts, we modified the computational definition of the generality in the second fitness function as the minimum of the cardinality of the extensions instead of their total one.

Based on the experimental results (see Section 6.3.4), it is clear that the second fitness function outperforms the former. However, the disadvantages of both single-objective functions include the *overfitting* problem, whereby some axioms would be discovered possessing a high fitness but invalid.

- *Multi-objective framework*: We built a multi-objective framework (see Section 7.3.1) consisting of using two objectives plus a "similarity" score which refers to finding axioms satisfying the possible non-dominated trade-offs among the objective functions and enhancing the diversity of obtained axioms in the population. We built two objective functions (see Equation (7.4)) aiming at discovering axioms that maximize the values of possibility and generality while not being too similar among themselves. This framework reduced the problem of overfitting of the previous single-objective GE models.

8.1.3 Toward Learning OWL Class Disjointness Axioms

- In order to apply the general OWL axiom discovery framework using GE (presented in Chapter 4) to the specific problem of discovering OWL class disjointness, we developed two learning models (see Chapter 6). We developed the first model involving learning atomic and complex axioms containing *union* and *intersection* operators and involving topic 'Work' in DBpedia (see Section 6.3). We used the two versions of the fitness function obeying the mentioned *single-objective* model for evaluating the discovered OWL axioms. In terms of atomic axioms, we compared our system to GoldMiner [VHC07], which is a related system outperforming other state of-the-art systems in terms of discovering class disjointness axioms. The advantages of this model are the high accuracy and the wider coverage than the GoldMiner when there are a number of discovered atomic axioms that cannot be mined by GoldMiner. For the case of complex axioms, the results are slightly less accurate compared with the case of discovering atomic ones, but still considerably precise. Although the first model is effective in discovering axioms containing both conjunctions and disjunctions, its limitations are that this kind of axioms can also be

mechanically derived from the known atomic axioms and the extracted axioms are limited to the classes relevant to a small scope of topics, namely the Work topic of DBpedia.

- In the second model of the GE approach described in Section 6.4, we turned to discovering OWL class disjointness axioms involving *value* and *existential restrictions* in addition to *conjunctions* on a wider variety of topics, which are hard or impossible to be manually induced from atomic axioms. The experimental results confirm that the proposed method is capable of discovering highly accurate and general axioms with the wider range of topics from DBpedia.

- We extended the above GE approach as a multi-objective problem, i.e., MOGE, by combining GE with the NSGA-II algorithm and using the *multi-objective* evaluation framework. The experimental results (see Section 7.3.3) confirmed the increased effectiveness of the framework, when compared to the above single-objective GE models.

Based on all the experimental results from both GE and MOGE, we found that the GE and MOGE approach are currently the best methods for discovering OWL class disjointness axioms. Furthermore, the MOGE outperformed GE in discovering larger number of axioms. In addition, there are no other methods proposed so-far in the literature to mine complex axioms of the same kind.

8.1.4 Performance Evaluation Frameworks

We developed two frameworks categorized into *subjective* (i.e., *Gold Standard*) and *objective* (i.e., *training-testing* model) for evaluating the performance of the learning models.

- *Gold Standard*: we created a matrix called Gold Standard created by humans, i.e., three domain experts. This matrix contains binary values representing the disjointness evaluation between 3,844 pairs of classes relevant to the topic

Work of DBpedia. The disadvantages of this benchmark is the dependence on the human assessment, which can be subjective and incorrect in addition to the limitation of the scabability, i.e. only relevant to the small scope of the topic Work.

- *Training- Testing model*: we begun developing the training-testing model by generating the training dataset containing RDF data extracted from DBpedia. The advantages of this benchmark is to overcome the limitation of performance and to provide a more objective assessment of the accuracy.

8.2 Future Work

From the point of view of evolutionary computation, the way in which we have used grammar-based genetic programming in this thesis is somehow atypical and demonstrates how evolutionary algorithms can profitably serve as tools to explore a huge search space to discover multiple interesting solutions (the more, the better!), rather than finding a single "best" solution. In other words, while the common practice in the field of evolutionary computation is to view evolutionary algorithms as very powerful, albeit somehow slow, global optimization methods, and devise clever ideas to make them converge faster and consistently to the global optimum, our work provides an alternative type of problems where exploration and "divergence", as it were, is the real name of the game and suggests that it is exactly for problems of this kind that evolutionary algorithms might give the best of themselves. Investigating in depth such a perspective on evolutionary algorithms and all of its consequences may thus be viewed as one possible extension of this work.

In addition, there are many ways in which the research we have initiated with this thesis might be extended. The following list of directions for further work is thus far from exhaustive and focuses on the most immediate issues or opportunities that our results bring about:

- *Enhancing computing speed*: The computation of the complex SPARQL queries in our axiom evaluation framework in general is still rather slow, which

required hours or days of CPU time, depending on the type of axioms and the relevant parameter settings. It would thus be interesting to study promising performance improvements that lend themselves to massive parallelization, of the kind offered by general-purpose GPUs. In addition, we could study the use of the parallelized *MapReduce* and *Hadoop* framework that can reduce the processing time of the big datasets like LOD.

- *Exploring various possible combinations of the promising measures*: A possible direction to follow in the future is to extend the evaluation of candidate axioms with the inclusion of some measures of relevant to the structure of axioms. For example, we could study the complexity within the axiom contents to define their interestingness. The more the axiom is complex, the higher the computational cost. In the case when axioms are too complex and the computation is over the allowed threshold, they should be less expected to be mined.

- *Exploiting our method to clean datasets and improve ontology-based data access (OBDA)*: The application of our results for providing issuing warnings in the datasets when inconsistencies are discovered would be the base for the further direction in handling inconsistencies and fixing errors in datasets involving the problem of data quality and data cleaning in OBDA [Xia+18].

- *Extending the types of axioms that need to be mined*: Another way to extend our results would be to concentrate on mining different types of axioms relevant to broader topics or discovering axioms at instance-level. For instance, one can learn *entity* axioms consisting of three following axioms: $HasKey$, $SameIndividual$, and $DifferentIndividuals$ which concern the problem of discovering "same" entities in different data sets, i.e., *instance-level equivalences*. Whereby, if two distinct resource names linking to the same real-world object is verified, i.e., if they are synonyms, is one of the hardest problems, which is closely linked to defining what is a legal representative (i.e., *key* in the database sense) for an entity. This task is related to the problem

of "*ontology alignment*" which is getting a lot of attention in knowledge engineering.

Appendices

A

Appendix- Gold Standard

- The Gold Standard[1] includes the sub-classes of the class *Work* in DBpedia 2015-04. Querying all these sub-classes is based on the following SPARQL query:

```
SELECT DISTINCT ?class WHERE {?class a owl:Class.}
?class rdfs:subClassOf <http://dbpedia.org/ontology/Work>
```
(A.1)

Sub-classes of class *Work* are listed in Table A.1

- Checking each pairs of class whether siblings or not is presented in Algorithm 5. If two classes share a similar set of super-classes, they are siblings; otherwise, they are not (lines 5-7). In this, the form of SPARQL query used to filter super-classes of the class named `cl`, i.e., $Query(cl)$ is designed as follows:

```
SELECT DISTINCT ?superclass
WHERE {cl rdfs:subClassOf ?superClass.}
FILTER REGEX(?superClass,<http://dbpedia.org/ontology*>
```
(A.2)

[1] https://bitbucket.org/RDFMiner/disjointnessclassaxiomge/src/master/GoldStandard.csv

Table A.1: List sub-classes of the class 'Work'

http://dbpedia.org/ontology/Software	http://dbpedia.org/ontology/Manhwa
http://dbpedia.org/ontology/AcademicJournal	http://dbpedia.org/ontology/MovingImage
http://dbpedia.org/ontology/Album	http://dbpedia.org/ontology/MultiVolumePublication
http://dbpedia.org/ontology/Anime	http://dbpedia.org/ontology/Musical
http://dbpedia.org/ontology/Annotation	http://dbpedia.org/ontology/MusicalWork
http://dbpedia.org/ontology/Archive	http://dbpedia.org/ontology/NationalAnthem
http://dbpedia.org/ontology/Article	http://dbpedia.org/ontology/Newspaper
http://dbpedia.org/ontology/ArtistDiscography	http://dbpedia.org/ontology/Novel
http://dbpedia.org/ontology/Artwork	http://dbpedia.org/ontology/Opera
http://dbpedia.org/ontology/BiologicalDatabase	http://dbpedia.org/ontology/Painting
http://dbpedia.org/ontology/Book	http://dbpedia.org/ontology/PeriodicalLiterature
http://dbpedia.org/ontology/Cartoon	http://dbpedia.org/ontology/Play
http://dbpedia.org/ontology/ClassicalMusicComposition	http://dbpedia.org/ontology/Poem
http://dbpedia.org/ontology/CollectionOfValuables	http://dbpedia.org/ontology/Quote
http://dbpedia.org/ontology/Comic	http://dbpedia.org/ontology/RadioProgram
http://dbpedia.org/ontology/ComicStrip	http://dbpedia.org/ontology/Reference
http://dbpedia.org/ontology/Database	http://dbpedia.org/ontology/Resume
http://dbpedia.org/ontology/Document	http://dbpedia.org/ontology/Sculpture
http://dbpedia.org/ontology/Drama	http://dbpedia.org/ontology/Single
http://dbpedia.org/ontology/EurovisionSongContestEntry	http://dbpedia.org/ontology/Song
http://dbpedia.org/ontology/File	http://dbpedia.org/ontology/Sound
http://dbpedia.org/ontology/Film	http://dbpedia.org/ontology/StatedResolution
http://dbpedia.org/ontology/HollywoodCartoon	http://dbpedia.org/ontology/StillImage
http://dbpedia.org/ontology/Image	http://dbpedia.org/ontology/TelevisionEpisode
http://dbpedia.org/ontology/Law	http://dbpedia.org/ontology/TelevisionSeason
http://dbpedia.org/ontology/Letter	http://dbpedia.org/ontology/TelevisionShow
http://dbpedia.org/ontology/LightNovel	http://dbpedia.org/ontology/Treaty
http://dbpedia.org/ontology/LineOfFashion	http://dbpedia.org/ontology/UndergroundJournal
http://dbpedia.org/ontology/Magazine	http://dbpedia.org/ontology/VideoGame
http://dbpedia.org/ontology/Manga	http://dbpedia.org/ontology/Website
http://dbpedia.org/ontology/Manhua	http://dbpedia.org/ontology/WrittenWork

The execution of the designed queries returns the list of their super-classes (lines 3-4)

Algorithm 5: Check_Siblings(cl1, cl2)

Input: $cl1, cl2$: classes need to be checked

Output: *results*: the result of checking siblings -returns to boolean value; if the return value is *true*: $cl1$ and $cl2$ are siblings; otherwise: $cl1$ and $cl2$ are not siblings.

1 $query1 \leftarrow Query(cl1)$

2 $query2 \leftarrow Query(cl1)$ /* $Query(class)$, in which $class = \{cl1, cl2\}$ is formed in Equation (A.2) */

3 $listsuperclass1 \leftarrow ResultQuery(query1)$

4 $listsuperclass2 \leftarrow ResultQuery(query2)$ /* $ResultQuery(query)$, in which $query = \{query1, query2\}$, returns the results of $query$ containing super-classes of given class, i.e., $class1$ or $class2$ */

5 $result \leftarrow false$

6 **if** *$listsuperclass_1$ and $listsuperclass_2$ are the same and not empty* **then**

7 $\quad result \leftarrow true$

8 **return** *result*

B

Appendix- Training Dataset

The training dataset mentioned in Section 6.4.2 contains 449 classes (owl:class) listed in the Table B.1 with respect to the diverse topics in DBpedia version 2015-04. We can use the following SPARQL query in order to view all of them through SPARQL endpoint.

```
SELECT DISTINCT ?cl WHERE ?cl a Owl:Class        (B.1)
```

Also, it comprises 13,238 different object properties. In order to view them, we can execute the following SPARQL query with the prefix: `http://dbpedia.org`:

```
SELECT DISTINCT ?p WHERE ?s ?p ?o                (B.2)
```

Table B.1: List of classes in the training dataset

http://dbpedia.org/ontology/Installment	http://dbpedia.org/ontology/BaseballLeague
http://dbpedia.org/ontology/Abbey	http://dbpedia.org/ontology/BaseballPlayer
http://dbpedia.org/ontology/AcademicJournal	http://dbpedia.org/ontology/BaseballSeason
http://dbpedia.org/ontology/Actor	http://dbpedia.org/ontology/BaseballTeam
http://dbpedia.org/ontology/AdministrativeRegion	http://dbpedia.org/ontology/BasketballLeague
http://dbpedia.org/ontology/AdultActor	http://dbpedia.org/ontology/BasketballPlayer
http://dbpedia.org/ontology/Agent	http://dbpedia.org/ontology/BasketballTeam
http://dbpedia.org/ontology/Agglomeration	http://dbpedia.org/ontology/Beach
http://dbpedia.org/ontology/Airline	http://dbpedia.org/ontology/BeautyQueen
http://dbpedia.org/ontology/Airport	http://dbpedia.org/ontology/Beer
http://dbpedia.org/ontology/Ambassador	http://dbpedia.org/ontology/Beverage
http://dbpedia.org/ontology/AmericanFootballCoach	http://dbpedia.org/ontology/BiologicalDatabase
http://dbpedia.org/ontology/AmericanFootballPlayer	http://dbpedia.org/ontology/Biologist
http://dbpedia.org/ontology/Amphibian	http://dbpedia.org/ontology/Biomolecule
http://dbpedia.org/ontology/AmusementParkAttraction	http://dbpedia.org/ontology/Bird
http://dbpedia.org/ontology/AnatomicalStructure	http://dbpedia.org/ontology/Birth

http://dbpedia.org/ontology/Animal	http://dbpedia.org/ontology/Bodybuilder
http://dbpedia.org/ontology/Anime	http://dbpedia.org/ontology/Bone
http://dbpedia.org/ontology/Arachnid	http://dbpedia.org/ontology/Book
http://dbpedia.org/ontology/Archaea	http://dbpedia.org/ontology/Boxer
http://dbpedia.org/ontology/Archeologist	http://dbpedia.org/ontology/Brain
http://dbpedia.org/ontology/Archipelago	http://dbpedia.org/ontology/Brewery
http://dbpedia.org/ontology/Architect	http://dbpedia.org/ontology/Bridge
http://dbpedia.org/ontology/ArchitecturalStructure	http://dbpedia.org/ontology/BritishRoyalty
http://dbpedia.org/ontology/Area	http://dbpedia.org/ontology/BroadcastNetwork
http://dbpedia.org/ontology/Aristocrat	http://dbpedia.org/ontology/Broadcaster
http://dbpedia.org/ontology/Arrondissement	http://dbpedia.org/ontology/BrownDwarf
http://dbpedia.org/ontology/Artery	http://dbpedia.org/ontology/Building
http://dbpedia.org/ontology/ArtificialSatellite	http://dbpedia.org/ontology/BusinessPerson
http://dbpedia.org/ontology/Artist	http://dbpedia.org/ontology/Camera
http://dbpedia.org/ontology/ArtistDiscography	http://dbpedia.org/ontology/Canoeist
http://dbpedia.org/ontology/Artwork	http://dbpedia.org/ontology/Cardinal
http://dbpedia.org/ontology/Asteroid	http://dbpedia.org/ontology/Cartoon
http://dbpedia.org/ontology/Astronaut	http://dbpedia.org/ontology/Case
http://dbpedia.org/ontology/Athlete	http://dbpedia.org/ontology/Casino
http://dbpedia.org/ontology/Athletics	http://dbpedia.org/ontology/Castle
http://dbpedia.org/ontology/Attack	http://dbpedia.org/ontology/Cave
http://dbpedia.org/ontology/SoccerLeague	http://dbpedia.org/ontology/CelestialBody
http://dbpedia.org/ontology/AutomobileEngine	http://dbpedia.org/ontology/Cemetery
http://dbpedia.org/ontology/Award	http://dbpedia.org/ontology/Chancellor
http://dbpedia.org/ontology/Bacteria	http://dbpedia.org/ontology/Cheese
http://dbpedia.org/ontology/BadmintonPlayer	http://dbpedia.org/ontology/Chef
http://dbpedia.org/ontology/Baronet	http://dbpedia.org/ontology/ChemicalCompound
http://dbpedia.org/ontology/ChristianBishop	http://dbpedia.org/ontology/Department
http://dbpedia.org/ontology/Church	http://dbpedia.org/ontology/Deputy
http://dbpedia.org/ontology/ClassicalMusicArtist	http://dbpedia.org/ontology/Desert
http://dbpedia.org/ontology/ClassicalMusicComposition	http://dbpedia.org/ontology/Device
http://dbpedia.org/ontology/Cleric	http://dbpedia.org/ontology/Diocese
http://dbpedia.org/ontology/Wine	http://dbpedia.org/ontology/District
http://dbpedia.org/ontology/ClubMoss	http://dbpedia.org/ontology/Document
http://dbpedia.org/ontology/Coach	http://dbpedia.org/ontology/Dog
http://dbpedia.org/ontology/CollegeCoach	http://dbpedia.org/ontology/Drama
http://dbpedia.org/ontology/Colour	http://dbpedia.org/ontology/Drug
http://dbpedia.org/ontology/Comedian	http://dbpedia.org/ontology/Economist
http://dbpedia.org/ontology/Comic	http://dbpedia.org/ontology/Website
http://dbpedia.org/ontology/ComicStrip	http://dbpedia.org/ontology/Egyptologist
http://dbpedia.org/ontology/ComicsCharacter	http://dbpedia.org/ontology/Election
http://dbpedia.org/ontology/ComicsCreator	http://dbpedia.org/ontology/Engine
http://dbpedia.org/ontology/Community	http://dbpedia.org/ontology/Engineer
http://dbpedia.org/ontology/Company	http://dbpedia.org/ontology/Entomologist
http://dbpedia.org/ontology/Competition	http://dbpedia.org/ontology/Enzyme
http://dbpedia.org/ontology/ConcentrationCamp	http://dbpedia.org/ontology/EthnicGroup
http://dbpedia.org/ontology/Congressman	http://dbpedia.org/ontology/WineRegion
http://dbpedia.org/ontology/Conifer	http://dbpedia.org/ontology/Factory
http://dbpedia.org/ontology/Constellation	http://dbpedia.org/ontology/Farmer
http://dbpedia.org/ontology/Contest	http://dbpedia.org/ontology/Fashion
http://dbpedia.org/ontology/Continent	http://dbpedia.org/ontology/FashionDesigner
http://dbpedia.org/ontology/Convention	http://dbpedia.org/ontology/Fencer
http://dbpedia.org/ontology/ConveyorSystem	http://dbpedia.org/ontology/Fern
http://dbpedia.org/ontology/Country	http://dbpedia.org/ontology/FictionalCharacter
http://dbpedia.org/ontology/Crater	http://dbpedia.org/ontology/FigureSkater
http://dbpedia.org/ontology/CricketTeam	http://dbpedia.org/ontology/File
http://dbpedia.org/ontology/Cricketer	http://dbpedia.org/ontology/Fish
http://dbpedia.org/ontology/Criminal	http://dbpedia.org/ontology/Flag
http://dbpedia.org/ontology/Crustacean	http://dbpedia.org/ontology/FloweringPlant
http://dbpedia.org/ontology/CultivatedVariety	http://dbpedia.org/ontology/SoccerPlayer
http://dbpedia.org/ontology/Curler	http://dbpedia.org/ontology/FootballMatch
http://dbpedia.org/ontology/Currency	http://dbpedia.org/ontology/FormulaOneRacer
http://dbpedia.org/ontology/Cycad	http://dbpedia.org/ontology/Fungus
http://dbpedia.org/ontology/CyclingRace	http://dbpedia.org/ontology/GaelicGamesPlayer
http://dbpedia.org/ontology/CyclingTeam	http://dbpedia.org/ontology/Galaxy
http://dbpedia.org/ontology/Cyclist	http://dbpedia.org/ontology/Game

http://dbpedia.org/ontology/Dam	http://dbpedia.org/ontology/Garden
http://dbpedia.org/ontology/Dancer	http://dbpedia.org/ontology/Gate
http://dbpedia.org/ontology/DartsPlayer	http://dbpedia.org/ontology/Gene
http://dbpedia.org/ontology/Deity	http://dbpedia.org/ontology/Genre
http://dbpedia.org/ontology/GivenName	http://dbpedia.org/ontology/Model
http://dbpedia.org/ontology/Glacier	http://dbpedia.org/ontology/Mollusca
http://dbpedia.org/ontology/GolfCourse	http://dbpedia.org/ontology/Monarch
http://dbpedia.org/ontology/GolfPlayer	http://dbpedia.org/ontology/Monastery
http://dbpedia.org/ontology/GolfTournament	http://dbpedia.org/ontology/Monument
http://dbpedia.org/ontology/GovernmentAgency	http://dbpedia.org/ontology/Mosque
http://dbpedia.org/ontology/Governor	http://dbpedia.org/ontology/Moss
http://dbpedia.org/ontology/GrandPrix	http://dbpedia.org/ontology/Motorcycle
http://dbpedia.org/ontology/Grape	http://dbpedia.org/ontology/MotorcycleRider
http://dbpedia.org/ontology/GreenAlga	http://dbpedia.org/ontology/MotorsportRacer
http://dbpedia.org/ontology/GridironFootballPlayer	http://dbpedia.org/ontology/MotorsportSeason
http://dbpedia.org/ontology/Guitar	http://dbpedia.org/ontology/Mountain
http://dbpedia.org/ontology/Guitarist	http://dbpedia.org/ontology/MountainPass
http://dbpedia.org/ontology/Gymnast	http://dbpedia.org/ontology/MountainRange
http://dbpedia.org/ontology/HandballTeam	http://dbpedia.org/ontology/Municipality
http://dbpedia.org/ontology/Historian	http://dbpedia.org/ontology/Murderer
http://dbpedia.org/ontology/HistoricBuilding	http://dbpedia.org/ontology/Muscle
http://dbpedia.org/ontology/HockeyTeam	http://dbpedia.org/ontology/Museum
http://dbpedia.org/ontology/Holiday	http://dbpedia.org/ontology/Musical
http://dbpedia.org/ontology/HollywoodCartoon	http://dbpedia.org/ontology/MusicalArtist
http://dbpedia.org/ontology/HorseRace	http://dbpedia.org/ontology/MusicalWork
http://dbpedia.org/ontology/HorseRider	http://dbpedia.org/ontology/MythologicalFigure
http://dbpedia.org/ontology/HorseTrainer	http://dbpedia.org/ontology/NCAATeamSeason
http://dbpedia.org/ontology/Hospital	http://dbpedia.org/ontology/Name
http://dbpedia.org/ontology/Host	http://dbpedia.org/ontology/NascarDriver
http://dbpedia.org/ontology/HotSpring	http://dbpedia.org/ontology/SoccerManager
http://dbpedia.org/ontology/Hotel	http://dbpedia.org/ontology/NaturalPlace
http://dbpedia.org/ontology/Humorist	http://dbpedia.org/ontology/Nerve
http://dbpedia.org/ontology/IceHockeyLeague	http://dbpedia.org/ontology/Newspaper
http://dbpedia.org/ontology/IceHockeyPlayer	http://dbpedia.org/ontology/Noble
http://dbpedia.org/ontology/Image	http://dbpedia.org/ontology/Non-ProfitOrganisation
http://dbpedia.org/ontology/InformationAppliance	http://dbpedia.org/ontology/Novel
http://dbpedia.org/ontology/Infrastructure	http://dbpedia.org/ontology/Ocean
http://dbpedia.org/ontology/Insect	http://dbpedia.org/ontology/OfficeHolder
http://dbpedia.org/ontology/Instrument	http://dbpedia.org/ontology/OlympicEvent
http://dbpedia.org/ontology/Instrumentalist	http://dbpedia.org/ontology/OlympicResult
http://dbpedia.org/ontology/Island	http://dbpedia.org/ontology/Olympics
http://dbpedia.org/ontology/Jockey	http://dbpedia.org/ontology/Opera
http://dbpedia.org/ontology/Journalist	http://dbpedia.org/ontology/Organ
http://dbpedia.org/ontology/Judge	http://dbpedia.org/ontology/Organisation
http://dbpedia.org/ontology/LacrossePlayer	http://dbpedia.org/ontology/OrganisationMember
http://dbpedia.org/ontology/Lake	http://dbpedia.org/ontology/Orphan
http://dbpedia.org/ontology/LaunchPad	http://dbpedia.org/ontology/Painter
http://dbpedia.org/ontology/LawFirm	http://dbpedia.org/ontology/Painting
http://dbpedia.org/ontology/Lawyer	http://dbpedia.org/ontology/Parish
http://dbpedia.org/ontology/Legislature	http://dbpedia.org/ontology/Park
http://dbpedia.org/ontology/Letter	http://dbpedia.org/ontology/Parliament
http://dbpedia.org/ontology/Lieutenant	http://dbpedia.org/ontology/PeriodicalLiterature
http://dbpedia.org/ontology/Lighthouse	http://dbpedia.org/ontology/Philosopher
http://dbpedia.org/ontology/Linguist	http://dbpedia.org/ontology/Photographer
http://dbpedia.org/ontology/Lipid	http://dbpedia.org/ontology/Place
http://dbpedia.org/ontology/Locality	http://dbpedia.org/ontology/Planet
http://dbpedia.org/ontology/Magazine	http://dbpedia.org/ontology/Play
http://dbpedia.org/ontology/Mammal	http://dbpedia.org/ontology/PlayWright
http://dbpedia.org/ontology/Manga	http://dbpedia.org/ontology/PlayboyPlaymate
http://dbpedia.org/ontology/MartialArtist	http://dbpedia.org/ontology/Poem
http://dbpedia.org/ontology/Mayor	http://dbpedia.org/ontology/Poet
http://dbpedia.org/ontology/Media	http://dbpedia.org/ontology/PokerPlayer
http://dbpedia.org/ontology/Medician	http://dbpedia.org/ontology/PoliticalParty
http://dbpedia.org/ontology/Medicine	http://dbpedia.org/ontology/Politician
http://dbpedia.org/ontology/Meeting	http://dbpedia.org/ontology/Polysaccharide
http://dbpedia.org/ontology/MemberOfParliament	http://dbpedia.org/ontology/Pope
http://dbpedia.org/ontology/Memorial	http://dbpedia.org/ontology/PopulatedPlace

http://dbpedia.org/ontology/MilitaryConflict	http://dbpedia.org/ontology/Population
http://dbpedia.org/ontology/MilitaryPerson	http://dbpedia.org/ontology/Port
http://dbpedia.org/ontology/MilitaryStructure	http://dbpedia.org/ontology/PowerStation
http://dbpedia.org/ontology/MilitaryUnit	http://dbpedia.org/ontology/Prefecture
http://dbpedia.org/ontology/Mill	http://dbpedia.org/ontology/Presenter
http://dbpedia.org/ontology/Mine	http://dbpedia.org/ontology/President
http://dbpedia.org/ontology/Mineral	http://dbpedia.org/ontology/Priest
http://dbpedia.org/ontology/MixedMartialArtsEvent	http://dbpedia.org/ontology/PrimeMinister
http://dbpedia.org/ontology/Prison	http://dbpedia.org/ontology/SongWriter
http://dbpedia.org/ontology/Producer	http://dbpedia.org/ontology/Sound
http://dbpedia.org/ontology/Profession	http://dbpedia.org/ontology/SpaceStation
http://dbpedia.org/ontology/Professor	http://dbpedia.org/ontology/Spacecraft
http://dbpedia.org/ontology/ProgrammingLanguage	http://dbpedia.org/ontology/Species
http://dbpedia.org/ontology/Project	http://dbpedia.org/ontology/Sport
http://dbpedia.org/ontology/ProtectedArea	http://dbpedia.org/ontology/Winery
http://dbpedia.org/ontology/Protein	http://dbpedia.org/ontology/SportFacility
http://dbpedia.org/ontology/Psychologist	http://dbpedia.org/ontology/SportsLeague
http://dbpedia.org/ontology/PublicTransitSystem	http://dbpedia.org/ontology/SportsManager
http://dbpedia.org/ontology/Publisher	http://dbpedia.org/ontology/SportsSeason
http://dbpedia.org/ontology/Race	http://dbpedia.org/ontology/SportsTeam
http://dbpedia.org/ontology/RaceHorse	http://dbpedia.org/ontology/SportsTeamMember
http://dbpedia.org/ontology/Racecourse	http://dbpedia.org/ontology/SportsTeamSeason
http://dbpedia.org/ontology/RacingDriver	http://dbpedia.org/ontology/Square
http://dbpedia.org/ontology/RadioProgram	http://dbpedia.org/ontology/SquashPlayer
http://dbpedia.org/ontology/RadioStation	http://dbpedia.org/ontology/Stadium
http://dbpedia.org/ontology/RailwayLine	http://dbpedia.org/ontology/State
http://dbpedia.org/ontology/RailwayStation	http://dbpedia.org/ontology/Station
http://dbpedia.org/ontology/Rebellion	http://dbpedia.org/ontology/Statistic
http://dbpedia.org/ontology/RecordLabel	http://dbpedia.org/ontology/Stream
http://dbpedia.org/ontology/Referee	http://dbpedia.org/ontology/Street
http://dbpedia.org/ontology/Regency	http://dbpedia.org/ontology/SubMunicipality
http://dbpedia.org/ontology/Region	http://dbpedia.org/ontology/SumoWrestler
http://dbpedia.org/ontology/Religious	http://dbpedia.org/ontology/Writer
http://dbpedia.org/ontology/ReligiousBuilding	http://dbpedia.org/ontology/Surfer
http://dbpedia.org/ontology/Reptile	http://dbpedia.org/ontology/Surname
http://dbpedia.org/ontology/RoadJunction	http://dbpedia.org/ontology/Swimmer
http://dbpedia.org/ontology/RoadTunnel	http://dbpedia.org/ontology/Synagogue
http://dbpedia.org/ontology/Rocket	http://dbpedia.org/ontology/TableTennisPlayer
http://dbpedia.org/ontology/RollerCoaster	http://dbpedia.org/ontology/Tax
http://dbpedia.org/ontology/RouteOfTransportation	http://dbpedia.org/ontology/Taxon
http://dbpedia.org/ontology/Rower	http://dbpedia.org/ontology/TelevisionEpisode
http://dbpedia.org/ontology/Royalty	http://dbpedia.org/ontology/TelevisionSeason
http://dbpedia.org/ontology/RugbyClub	http://dbpedia.org/ontology/TelevisionShow
http://dbpedia.org/ontology/RugbyPlayer	http://dbpedia.org/ontology/TelevisionStation
http://dbpedia.org/ontology/Saint	http://dbpedia.org/ontology/Temple
http://dbpedia.org/ontology/SambaSchool	http://dbpedia.org/ontology/TennisPlayer
http://dbpedia.org/ontology/Satellite	http://dbpedia.org/ontology/Territory
http://dbpedia.org/ontology/School	http://dbpedia.org/ontology/Theatre
http://dbpedia.org/ontology/Scientist	http://dbpedia.org/ontology/TimePeriod
http://dbpedia.org/ontology/ScreenWriter	http://dbpedia.org/ontology/Tournament
http://dbpedia.org/ontology/Sculptor	http://dbpedia.org/ontology/Tower
http://dbpedia.org/ontology/Sculpture	http://dbpedia.org/ontology/Town
http://dbpedia.org/ontology/Senator	http://dbpedia.org/ontology/TradeUnion
http://dbpedia.org/ontology/Settlement	http://dbpedia.org/ontology/Train
http://dbpedia.org/ontology/ShoppingMall	http://dbpedia.org/ontology/Tunnel
http://dbpedia.org/ontology/Shrine	http://dbpedia.org/ontology/Type
http://dbpedia.org/ontology/Singer	http://dbpedia.org/ontology/UnitOfWork
http://dbpedia.org/ontology/Single	http://dbpedia.org/ontology/Valley
http://dbpedia.org/ontology/SoccerClubSeason	http://dbpedia.org/ontology/Venue
http://dbpedia.org/ontology/Skater	http://dbpedia.org/ontology/VideoGame
http://dbpedia.org/ontology/SkiArea	http://dbpedia.org/ontology/Village
http://dbpedia.org/ontology/Skier	http://dbpedia.org/ontology/Vodka
http://dbpedia.org/ontology/Skyscraper	http://dbpedia.org/ontology/VoiceActor
http://dbpedia.org/ontology/SnookerPlayer	http://dbpedia.org/ontology/Volcano
http://dbpedia.org/ontology/SoapCharacter	http://dbpedia.org/ontology/Watermill
http://dbpedia.org/ontology/SoccerClub	http://dbpedia.org/ontology/Weapon

http://dbpedia.org/ontology/SoccerTournament	http://dbpedia.org/ontology/WinterSportPlayer
http://dbpedia.org/ontology/SocietalEvent	http://dbpedia.org/ontology/Work
http://dbpedia.org/ontology/Software	http://dbpedia.org/ontology/WorldHeritageSite
http://dbpedia.org/ontology/Wrestler	http://dbpedia.org/ontology/WrittenWork
http://dbpedia.org/ontology/WrestlingEvent	http://dbpedia.org/ontology/Year
http://dbpedia.org/ontology/SupremeCourtOfTheUnitedStatesCase	
http://dbpedia.org/ontology/SiteOfSpecialScientificInterest	
http://dbpedia.org/ontology/EducationalInstitution	
http://dbpedia.org/ontology/AustralianRulesFootballPlayer	
http://dbpedia.org/ontology/ClericalAdministrativeRegion	
http://dbpedia.org/ontology/NationalFootballLeagueSeason	
http://dbpedia.org/ontology/EurovisionSongContestEntry	
http://dbpedia.org/ontology/FootballLeagueSeason	
http://dbpedia.org/ontology/SportCompetitionResult	

Bibliography

[Pop35] Karl Popper. “Logik der Forschung”. In: (1935).

[Ham50] R. W. Hamming. “Error detecting and error correcting codes”. In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160.

[Lev66] Vladimir I Levenshtein. “Binary Codes Capable of Correcting Deletions, Insertions and Reversals”. In: *Soviet Physics Doklady* 10 (1966), p. 707.

[DT72] A. De Luca and S. Termini. “A Definition of a Nonprobabilistic Entropy in the Setting of Fuzzy Sets Theory”. In: *Information and Control* 20 (1972), pp. 301–312.

[SG72] Israel Scheffler and Nelson Goodman. “Selective Confirmation and the Ravens: A Reply to Foster”. In: *The Journal of Philosophy* 69.3 (1972), pp. 78–83.

[Gog73] J. A. Goguen. “L. A. Zadeh. Fuzzy sets. Information and control, vol. 8 (1965), pp. 338–353. - L. A. Zadeh. Similarity relations and fuzzy orderings. Information sciences, vol. 3 (1971), pp. 177–200.” In: *Journal of Symbolic Logic* 38.4 (1973), pp. 656–657.

[DP80] D. Dubois and H. Prade. “Fuzzy sets and systems: theory and applications”. In: *Mathematics in science and engineering*. 1980.

[Sch85] J. David Schaffer. “Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”. In: *Proceedings of the 1st International Conference on Genetic Algorithms*. USA: L. Erlbaum Associates Inc., 1985, pp. 93–100.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, 1986.

[GH88] David E. Goldberg and John H. Holland. “Genetic Algorithms and Machine Learning”. In: *Machine Learning* 3.2 (1988), pp. 95–99.

[Qui90] J. Ross Quinlan. “Learning Logical Definitions from Relations”. In: *Mach. Learn.* 5 (1990), pp. 239–266.

[Ber91] Francesco Bergadano. “The Problem of Induction and Machine Learning”. In: *IJCAI*. Morgan Kaufmann, 1991, pp. 1073–1080.

[Kur91] Frank Kursawe. “A variant of evolution strategies for vector optimization”. In: *Parallel Problem Solving from Nature*. Ed. by Hans-Paul Schwefel and Reinhard Männer. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 193–197.

[HL92] P. Hajela and C.Y. Lin. “Genetic search strategies in multicriterion optimal design”. In: *Structural optimization* 4.2 (1992), pp. 99–107.

[Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[Mah92] Samir W. Mahfoud. “Crowding and Preselection Revisited”. In: *PPSN*. Elsevier, 1992, pp. 27–36.

[DP93] D. Dubois and H. Prade. “Fuzzy sets and probability: misunderstandings, bridges and gaps”. In: *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*. 1993, 1059–1068 vol.2.

[FF93] C. M. Fonseca and P. Fleming. “Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization”. In: *ICGA*. 1993.

[Koz93] John R. Koza. *Genetic programming - on the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1993.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. “Fast Algorithms for Mining Association Rules in Large Databases”. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB ’94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499.

[Coh94] William W. Cohen. “Grammatically Biased Learning: Learning Logic Programs Using an Explicit Antecedent Description Language”. In: *Artif. Intell.* 68.2 (1994), pp. 303–366.

[HNG94] J. Horn, N. Nafpliotis, and D. E. Goldberg. “A niched Pareto genetic algorithm for multiobjective optimization”. In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. 1994, 82–87 vol.1.

[MR94] Stephen Muggleton and Luc De Raedt. “Inductive Logic Programming: Theory and Methods”. In: *J. Log. Program.* 19/20 (1994), pp. 629–679.

[SD94a] N. Srinivas and K. Deb. “Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”. In: *Evolutionary Computation* 2.3 (1994), pp. 221–248.

[SD94b] N. Srinivas and Kalyanmoy Deb. “Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”. In: *Evol. Comput.* 2.3 (Sept. 1994), pp. 221–248. ISSN: 1063-6560.

[Gru95] Thomas R. Gruber. “Toward principles for the design of ontologies used for knowledge sharing?” In: *Int. J. Hum.-Comput. Stud.* 43.5-6 (1995), pp. 907–928.

[MK95a] Man Leung Wong and Kwong Sak Leung. "Applying logic grammars to induce sub-functions in genetic programming". In: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. Vol. 2. 1995, pp. 737–740.

[MK95b] Man Leung Wong and Kwong Sak Leung. "Combining genetic programming and inductive logic programming using logic grammars". In: *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. Vol. 2. 1995, pp. 733–736.

[Whi95] P. A. Whigham. "Inductive bias and genetic programming". In: *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*. 1995, pp. 461–466.

[Sme96] Philippe Smets. "Imperfect Information: Imprecision and Uncertainty". In: *Uncertainty Management in Information Systems*. Kluwer Academic Publishers, Boston, 1996, pp. 225–254.

[Whi96] P. A. Whigham. "Search Bias, Language Bias and Genetic Programming". In: *Proceedings of the 1st Annual Conference on Genetic Programming*. Stanford, California: MIT Press, 1996, pp. 230–237.

[Zim96] H.-J. Zimmermann. "Fuzzy Sets—Basic Definitions". In: *Fuzzy Set Theory—and Its Applications*. Dordrecht: Springer Netherlands, 1996, pp. 11–21.

[Cra98] Edward Craig. "Ontology". In: (1998). URL: `https://www.rep.routledge.com/articles/thematic/ontology/v-1`.

[IM98] H. Ishibuchi and T. Murata. "A multi-objective genetic local search algorithm and its application to flowshop scheduling". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28.3 (1998), pp. 392–403.

[RCO98] Conor Ryan, J. J. Collins, and Michael O'Neill. "Grammatical Evolution: Evolving Programs for an Arbitrary Language". In: *EuroGP*. Vol. 1391. Lecture Notes in Computer Science. Springer, 1998, pp. 83–96.

[KC99] J. Knowles and D. Corne. "The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation". In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 1. 1999, 98–105 Vol. 1.

[Zad99] L.A. Zadeh. "Fuzzy sets as a basis for a theory of possibility". In: *Fuzzy Sets and Systems* 100 (1999), pp. 9–34.

[ZT99] E. Zitzler and L. Thiele. "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach". In: *IEEE Transactions on Evolutionary Computation* 3.4 (1999), pp. 257–271.

[TM00] Alireza Tamaddoni-Nezhad and Stephen Muggleton. "Searching the Subsumption Lattice by a Genetic Algorithm". In: *ILP*. Vol. 1866. Lecture Notes in Computer Science. Springer, 2000, pp. 243–252.

[BHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. "The Semantic Web". In: *Scientific American* 284 (2001), pp. 34–43.

[MS01] A. Maedche and S. Staab. "Ontology learning for the Semantic Web". In: *IEEE Intelligent Systems* 16.2 (2001), pp. 72–79.

[OR01] Michael O'Neill and Conor Ryan. "Grammatical evolution". In: *IEEE Trans. Evolutionary Computation* 5.4 (2001), pp. 349–358.

[ONe+01] Michael O'Neill et al. "Crossover in Grammatical Evolution: The Search Continues". In: *EuroGP*. Vol. 2038. Lecture Notes in Computer Science. Springer, 2001, pp. 337–347.

[RR01] Philip G. K. Reiser and Patricia J. Riddle. "Scaling Up Inductive Logic Programming: An Evolutionary Wrapper Approach". In: *Appl. Intell.* 15.3 (2001), pp. 181–197.

[ZLT01] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Tech. rep. 2001.

[Deb+02] Kalyanmoy Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Trans. Evol. Comput.* 6.2 (2002), pp. 182–197.

[PF02] Peter F. Patel-Schneider and Dieter Fensel. "Layering the Semantic Web: Problems and Directions". In: *International Semantic Web Conference*. Vol. 2342. Lecture Notes in Computer Science. Springer, 2002, pp. 16–29.

[TM02] Alireza Tamaddoni-Nezhad and Stephen Muggleton. "A Genetic Algorithms Approach to ILP". In: *ILP*. Vol. 2583. Lecture Notes in Computer Science. Springer, 2002, pp. 285–300.

[ONe+03] Michael O'Neill et al. "Crossover in Grammatical Evolution". In: *Genetic Programming and Evolvable Machines* 4.1 (2003), pp. 67–93.

[Vol+03] Raphael Volz et al. "Views for Light-weight Web Ontologies". In: *SAC*. ACM, 2003, pp. 1168–1173.

[MS04] Alexander Maedche and Steffen Staab. "Ontology Learning". In: *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004, pp. 173–190.

[W3C04a] W3C. *OWL Web Ontology Language Overview*. 2004. URL: `https://www.w3.org/TR/2004/REC-owl-features-20040210/`.

[W3C04b] W3C. *OWL Web Ontology Language Reference*. 2004. URL: `https://www.w3.org/TR/owl-ref/`.

[ZLB04] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. "A Tutorial on Evolutionary Multiobjective Optimization". In: *Metaheuristics for Multiobjective Optimisation*. Ed. by Xavier Gandibleux et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 3–37.

[DM05] Federico Divina and Elena Marchiori. “Handling continuous attributes in an evolutionary inductive learner”. In: *IEEE Trans. Evol. Comput.* 9.1 (2005), pp. 31–43.

[Mas05] T. Berners-Lee; R. Fielding; L. Masinter. *RFC3986 - Uniform Resource Identifiers(URI): Generic Syntax*. 2005. URL: `http://www.ietf.org/rfc/rfc3986.txt`.

[Ram+05] Cartic Ramakrishnan et al. “Discovering informative connection subgraphs in multi-relational graphs”. In: *SIGKDD Explor.* 7.2 (2005), pp. 56–63.

[Sui05] M. Duerst; M. Suignard. *RFC3987- Internationalized Resource Identifiers(URI): Generic Syntax*. 2005. URL: `http://www.ietf.org/rfc/rfc3987.txt`.

[Div06] Federico Divina. “Evolutionary concept learning in First Order Logic: An overview”. In: *AI Commun.* 19.1 (2006), pp. 13–33.

[Flo+06] Giorgos Flouris et al. “Inconsistencies, Negations and Changes in Ontologies”. In: *AAAI*. AAAI Press, 2006, pp. 1295–1300.

[SBH06] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. “The Semantic Web Revisited”. In: *IEEE Intell. Syst.* 21.3 (2006), pp. 96–101.

[SPA07] Grimm st Stephan, Hitzler st Pascal, and Abecker st Andreas. “Knowledge Representation and Ontologies”. In: *Semantic Web Services: Concepts, Technologies, and Applications*. Ed. by Rudi Studer, Stephan Grimm, and Andreas Abecker. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 51–105.

[VHC07] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. “Acquisition of OWL DL Axioms from Lexical Resources”. In: *ESWC*. Vol. 4519. Lecture Notes in Computer Science. Springer, 2007, pp. 670–685.

[Völ+07] Johanna Völker et al. “Learning Disjointness”. In: *ESWC*. Vol. 4519. Lecture Notes in Computer Science. Springer, 2007, pp. 175–189.

[FdE08] Nicola Fanizzi, Claudia d’Amato, and Floriana Esposito. “DL-FOIL Concept Learning in Description Logics”. In: *ILP*. Vol. 5194. Lecture Notes in Computer Science. Springer, 2008, pp. 107–121.

[Gra+08] Bernardo Cuenca Grau et al. “OWL 2: The next step for OWL”. In: *J. Web Semant.* 6.4 (2008), pp. 309–322.

[Lan+08] William B. Langdon et al. “Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications”. In: *Computational Intelligence: A Compendium*. Ed. by John Fulcher and Lakhmi C. Jain. Vol. 115. Studies in Computational Intelligence. Springer, 2008, pp. 927–1028.

[LH08] Jens Lehmann and Pascal Hitzler. “A Refinement Operator Based Learning Algorithm for the $*mathcal{*ALC*}$*DescriptionLogic*”. In: *Inductive Logic Programming*. Ed. by Hendrik Blockeel et al. Springer Berlin Heidelberg, 2008.

[ONe+08] Michael O'Neill et al. "GEVA: Grammatical Evolution in Java". In: 3.2 (July 2008), pp. 17–22.

[W3C08] W3C. *SPARQL Query Language for RDF*. 2008. URL: `https://www.w3.org/TR/rdf-sparql-query/`.

[DOB09] Ian Dempsey, Michael O'Neill, and Anthony Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments - Chapter 2 Grammatical Evolution*. Vol. 194. Studies in Computational Intelligence. Springer, 2009.

[GOS09] Nicola Guarino, Daniel Oberle, and Steffen Staab. *What Is an Ontology? Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2009, pp. 1–17.

[HLA09] Sebastian Hellmann, Jens Lehmann, and Sören Auer. "Learning of OWL Class Descriptions on Very Large Knowledge Bases". In: *Int. J. Semantic Web Inf. Syst.* 5.2 (2009), pp. 25–48.

[Hoe09] Rinke Hoekstra. *Ontology Representation - Design Patterns and Ontologies that Make Sense*. Vol. 197. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009.

[Leh09] Jens Lehmann. "DL-Learner: Learning Concepts in Description Logics". In: *Journal of Machine Learning Research* 10 (2009), pp. 2639–2642.

[Bas+10] Adrien Basse et al. "DFS-based frequent graph pattern extraction to characterize the content of RDF Triple Stores". In: *Web Science Conference 2010 (WebSci10)*. Raleigh, United States, 2010.

[CJ10] Tom Castle and Colin G. Johnson. "Positional Effect of Crossover and Mutation in Grammatical Evolution". In: *EuroGP*. Vol. 6021. Lecture Notes in Computer Science. Springer, 2010, pp. 26–37.

[Che+10] Wen-hao Chen et al. "Generating ontologies with basic level concepts from folksonomies". In: *ICCS*. Vol. 1. Procedia Computer Science 1. Elsevier, 2010, pp. 573–581.

[Div10] Federico Divina. *Genetic Relational Search for Inductive Concept Learning: A Memetic Algorithm for ILP*. LAP LAMBERT Academic Publishing, 2010.

[HKR10] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies - Chapter 7: Query Language*. Chapman and Hall/CRC Press, 2010.

[Jai+10] Prateek Jain et al. "Ontology Alignment for Linked Open Data". In: *The Semantic Web – ISWC 2010*. Ed. by Peter F. Patel-Schneider et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 402–417.

[Kaz10] Yevgeny Kazakov. "An Extension of Complex Role Inclusion Axioms in the Description Logic SROIQ". In: *IJCAR*. Vol. 6173. Lecture Notes in Computer Science. Springer, 2010, pp. 472–486.

[LH10] Jens Lehmann and Pascal Hitzler. “Concept learning in description logics using refinement operators”. In: *Mach. Learn.* 78.1-2 (2010), pp. 203–250.

[NB10] Victoria Nebot and Rafael Berlanga. “Mining Association Rules from Semantic Web Data”. In: *Trends in Applied Intelligent Systems*. Ed. by Nicolás García-Pedrajas et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 504–513.

[Deb11] Kalyanmoy Deb. “Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction”. In: *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*. Springer, 2011, pp. 3–34.

[FS11] Reto Krummenacher Fabien Gandon and Ioan Toma Sung Kook Han. “The Resource Description Framework and its Schema”. In: *Handbook of Semantic Web Technologies*. Ed. by John Domingue, Dieter Fensel, and James A. Hendler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[FV11] Daniel Fleischhacker and Johanna Völker. “Inductive Learning of Disjointness Axioms”. In: *OTM Conferences (2)*. Vol. 7045. Lecture Notes in Computer Science. Springer, 2011, pp. 680–697.

[Gan+11] Fabien L. Gandon et al. “Semantic Annotation and Retrieval: RDF”. In: *Handbook of Semantic Web Technologies*. Ed. by John Domingue, Dieter Fensel, and James A. Hendler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 117–155.

[Gri+11] Stephan Grimm et al. “Introduction to the Semantic Web Technologies”. In: *Handbook of Semantic Web Technologies*. Ed. by John Domingue, Dieter Fensel, and James A. Hendler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 18–20.

[Jai+11] Prateek Jain et al. “Contextual Ontology Alignment of LOD with an Upper Ontology: A Case Study with Proton”. In: *ESWC (1)*. Vol. 6643. Lecture Notes in Computer Science. Springer, 2011, pp. 80–92.

[Rud11] Sebastian Rudolph. “Foundations of Description Logics”. In: *Reasoning Web*. Vol. 6848. Lecture Notes in Computer Science. Springer, 2011, pp. 76–136.

[SE11] François Scharffe and Jérôme Euzenat. “Linked Data Meets Ontology Matching - Enhancing Data Linking through Ontology Alignments”. In: *KEOD*. SciTePress, 2011, pp. 279–284.

[SAS11] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. “PARIS: Probabilistic Alignment of Relations, Instances, and Schema”. In: *CoRR* abs/1111.7164 (2011).

[VN11] Johanna Völker and Mathias Niepert. “Statistical Schema Induction”. In: *ESWC (1)*. Vol. 6643. Lecture Notes in Computer Science. Springer, 2011, pp. 124–138.

[ZI11] Lihua Zhao and Ryutaro Ichise. "Mid-Ontology Learning from Linked Data". In: *JIST*. Vol. 7185. Lecture Notes in Computer Science. Springer, 2011, pp. 112–127.

[BL12] Lorenz Bühmann and Jens Lehmann. "Universal OWL Axiom Enrichment for Large Knowledge Bases". In: *EKAW*. Vol. 7603. Lecture Notes in Computer Science. Springer, 2012, pp. 57–71.

[FVS12] Daniel Fleischhacker, Johanna Völker, and Heiner Stuckenschmidt. "Mining RDF Data for Property Axioms". In: *OTM Conferences (2)*. Vol. 7566. Lecture Notes in Computer Science. Springer, 2012, pp. 718–735.

[KSH12] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. "A Description Logic Primer". In: *CoRR* abs/1201.4089 (2012).

[Lis12] Francesca A. Lisi. *Learning Onto-Relational Rules with Inductive Logic Programming*. 2012. arXiv: `1210.2984 [cs.AI]`.

[PKA12] Rahul Parundekar, Craig A. Knoblock, and José Luis Ambite. "Discovering Concept Coverings in Ontologies of Linked Data Sources". In: *The Semantic Web – ISWC 2012*. Ed. by Philippe Cudré-Mauroux et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 427–443.

[Ret+12] Achim Rettinger et al. "Mining the Semantic Web - Statistical learning for next generation knowledge bases". In: *Data Min. Knowl. Discov.* 24.3 (2012), pp. 613–662.

[TKS12] Gerald Töpper, Magnus Knuth, and Harald Sack. "DBpedia ontology enrichment for inconsistency detection". In: *I-SEMANTICS*. ACM, 2012, pp. 33–40.

[VP12] Leonardo Vanneschi and Riccardo Poli. *Genetic Programming — Introduction, Applications, Theory and Open Issues*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 709–739.

[W3C12a] W3C. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. 2012. URL: `https://www.w3.org/TR/owl2-overview/`.

[W3C12b] W3C. *OWL 2 Web Ontology Language Primer)*. 2012. URL: `https://www.w3.org/2007/OWL/draft/ED-owl2-primer-20120905/`.

[W3C12c] W3C. *OWL 2 Web Ontology Language Profiles (Second Edition)*. 2012. URL: `https://www.w3.org/TR/owl2-profiles/`.

[Zha+12] Xiang Zhang et al. "Mining Link Patterns in Linked Data". In: *Web-Age Information Management*. Ed. by Hong Gao et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 83–94.

[BL13a] Lorenz Bühmann and J. Lehmann. "OWL Class Expression to SPARQL Rewriting". In: 2013.

[BL13b] Lorenz Bühmann and Jens Lehmann. "Pattern Based Knowledge Base Enrichment". In: *International Semantic Web Conference (1)*. Vol. 8218. Lecture Notes in Computer Science. Springer, 2013, pp. 33–48.

[GPS13] Luis Antonio Galárraga, Nicoleta Preda, and Fabian M. Suchanek. "Mining rules to align knowledge bases". In: *AKBC@CIKM*. ACM, 2013, pp. 43–48.

[Gal+13] Luis Antonio Galárraga et al. "AMIE: association rule mining under incomplete evidence in ontological knowledge bases". In: *WWW*. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 413–422.

[Kee13a] C. Maria Keet. "Closed World Assumption". In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky et al. New York, NY: Springer New York, 2013, pp. 415–415.

[Kee13b] C. Maria Keet. "Open World Assumption". In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky et al. New York, NY: Springer New York, 2013, pp. 1567–1567.

[QS13] Q. Quboa and M. Saraee. "A State-of-the-Art Survey on Semantic Web Mining". In: *Intelligent Information Management* 5.1 (2013), pp. 10–17.

[W3C13] W3C. *SPARQL 1.1 Overview*. 2013. URL: https://www.w3.org/TR/sparql11-overview/.

[LV14a] Jens Lehmann and Johanna Voelker. In: *Perspectives on Ontology Learning*. Ed. by Jens Lehmann and Johanna Voelker. AKA / IOS Press, 2014, pp. ix–xvi.

[LV14b] Jens Lehmann and Johanna Völker. *Perspectives on Ontology Learning*. Vol. 18. Studies on the Semantic Web. IOS Press, 2014.

[W3C14a] W3C. *RDF 1.1 Concepts and Abstract Syntax*. 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/Overview.html.

[W3C14b] W3C. *RDF Schema 1.1*. 2014. URL: https://www.w3.org/TR/rdf-schema/.

[CB15] Olivier Curé and Guillaume Blin. *RDF Database Systems: Triples Storage and SPARQL Query Processing*. Morgan Kaufmann, 2015.

[DP15] Didier Dubois and Henry Prade. "Possibility Theory and Its Applications: Where Do We Stand?" In: *Springer Handbook of Computational Intelligence*. Ed. by Janusz Kacprzyk and Witold Pedrycz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 31–60.

[Gal+15] Luis Galárraga et al. "Fast rule mining in ontological knowledge bases with AMIE+". In: *VLDB J.* 24.6 (2015), pp. 707–730.

[Hec+15] Thomas Hecht et al. "Ontology Alignment Using Web Linked Ontologies as Background Knowledge". In: *EGC (best of volume)*. Vol. 665. Studies in Computational Intelligence. 2015, pp. 207–227.

[RBP15] Petar Ristoski, Christian Bizer, and Heiko Paulheim. "Mining the Web of Linked Data with RapidMiner". In: *J. Web Semant.* 35 (2015), pp. 142–151.

[SSB15] Viachaslau Sazonau, Uli Sattler, and Gavin Brown. "General Terminology Induction in OWL". In: *OWLED*. Vol. 9557. Lecture Notes in Computer Science. Springer, 2015, pp. 1–13.

[TFG15] Andrea G. B. Tettamanzi, Catherine Faron-Zucker, and Fabien L. Gandon. "Dynamically Time-Capped Possibilistic Testing of SubClassOf Axioms Against RDF Data to Enrich Schemas". In: *K-CAP*. ACM, 2015, 7:1–7:8.

[Ton+15] Alberto Tonon et al. "Fixing the Domain and Range of Properties in Linked Data by Context Disambiguation". In: *LDOW@WWW*. Vol. 1409. CEUR Workshop Proceedings. CEUR-WS.org, 2015.

[VDP15] V. L. Vachhani, V. K. Dabhi, and H. B. Prajapati. "Survey of multi objective evolutionary algorithms". In: *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*. 2015, pp. 1–9.

[VFS15] Johanna Völker, Daniel Fleischhacker, and Heiner Stuckenschmidt. "Automatic acquisition of class disjointness". In: *J. Web Sem.* 35 (2015), pp. 124–139.

[dTT16] Claudia d'Amato, Andrea G. B. Tettamanzi, and Duc Minh Tran. "Evolutionary Discovery of Multi-relational Association Rules from Ontological Knowledge Bases". In: *EKAW*. Vol. 10024. Lecture Notes in Computer Science. 2016, pp. 113–128.

[dAm+16] Claudia d'Amato et al. "Ontology enrichment by discovering multi-relational association rules from ontological knowledge bases". In: *SAC*. ACM, 2016, pp. 333–338.

[DP16] Didier Dubois and Henri Prade. "Practical Methods for Constructing Possibility Distributions". In: *International Journal of Intelligent Systems* 31.3 (2016), pp. 215–239.

[EHC16] Basil Ell, Sherzod Hakimov, and Philipp Cimiano. "Statistical Induction of Coupled Domain/Range Restrictions from RDF Knowledge Bases". In: *KEKI/NLP&DBpedia@ISWC*. Vol. 10579. Lecture Notes in Computer Science. Springer, 2016, pp. 27–40.

[Ido+16] Rihab Idoudi et al. "Association rules-based Ontology Enrichment". In: *Int. J. Web Appl.* 8.1 (2016), pp. 16–25.

[LLS16] Yuanyuan Li, Huiying Li, and Jing Shi. "Mining RDF Data for OWL2 RL Axioms". In: *CCKS*. Vol. 650. Communications in Computer and Information Science. Springer, 2016, pp. 117–123.

[NSL16] Farzad Noorian, A. Silva, and P. Leong. "gramEvol: Grammatical Evolution in R". In: *Journal of Statistical Software* 71 (2016), pp. 1–26.

[TRa16] B.L.Shivakumar T.Raji. "A survey on semantic web mining technologies and web mining tools". In: *International Journal of Advanced Computational Engineering and Networking* (2016).

[BBL17] Molood Barati, Quan Bai, and Qing Liu. "Mining Semantic Association Rules from RDF Data". In: *Know.-Based Syst.* 133.C (Oct. 2017), pp. 183–196.

[Fen+17] Michael Fenton et al. "PonyGE2: Grammatical Evolution in Python". In: *CoRR* abs/1703.08535 (2017).

[KPV17] Maria Koutraki, Nicoleta Preda, and Dan Vodislav. "Online Relation Alignment for Linked Datasets". In: *ESWC (1)*. Vol. 10249. Lecture Notes in Computer Science. 2017, pp. 152–168.

[Riz+17] Giuseppe Rizzo et al. "Terminological Cluster Trees for Disjointness Axiom Discovery". In: *ESWC (1)*. Vol. 10249. Lecture Notes in Computer Science. 2017, pp. 184–201.

[SPS17] Fatiha Saıs, Cédric Pruski, and Marcos Da Silveira. "Inferring the evolution of ontology axioms from RDF data dynamics". In: *K-CAP*. ACM, 2017, 43:1–43:4.

[SS17] Viachaslau Sazonau and Uli Sattler. "Mining Hypotheses from Data in OWL: Advanced Evaluation and Complete Construction". In: *International Semantic Web Conference (1)*. Vol. 10587. Lecture Notes in Computer Science. Springer, 2017, pp. 577–593.

[SK17] Kiril Ivanov Simov and Atanas Kiryakov. "Accessing Linked Open Data via A Common Ontology". In: *NLPLOD@RANLP*. INCOMA, 2017, pp. 33–41.

[TFG17] Andrea G. B. Tettamanzi, Catherine Faron-Zucker, and Fabien Gandon. "Possibilistic testing of OWL axioms against RDF data". In: *Int. J. Approx. Reason.* 91 (2017), pp. 114–130.

[Tra+17] Duc Minh Tran et al. "An evolutionary algorithm for discovering multi-relational association rules in the semantic web". In: *GECCO*. ACM, 2017, pp. 513–520.

[Fan+18] Nicola Fanizzi et al. "DLFoil: Class Expression Learning Revisited". In: *Knowledge Engineering and Knowledge Management*. Ed. by Catherine Faron Zucker et al. Cham: Springer International Publishing, 2018, pp. 98–113.

[Xia+18] Guohui Xiao et al. "Ontology-Based Data Access: A Survey". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. Ed. by Jérôme Lang. ijcai.org, 2018, pp. 5511–5519.

[HZL19] Wenlan Huang, Yu Zhang, and Lan Li. "Survey on Multi-Objective Evolutionary Algorithms". In: *Journal of Physics: Conference Series* 1288 (2019), p. 012057.

[RTN19] Justine Reynaud, Yannick Toussaint, and Amedeo Napoli. "Redescription Mining for Learning Definitions and Disjointness Axioms in Linked Open Data". In: *ICCS*. Vol. 11530. Lecture Notes in Computer Science. Springer, 2019, pp. 175–189.

[DDP20] Thierry Denœux, Didier Dubois, and Henri Prade. "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility". In: *A Guided Tour of Artificial Intelligence Research: Volume I: Knowledge Representation, Reasoning and Learning*. Ed. by Pierre Marquis, Odile Papini, and Henri Prade. Cham: Springer International Publishing, 2020, pp. 69–117.

[Pot20] Jedrzej Potoniec. "Mining Cardinality Restrictions in OWL". In: *Foundations of Computing and Decision Sciences* 45.3 (2020), pp. 195–216.

[TEE20] Mohammed R. Thamer, S. El-Sappagh, and T. El-Shishtawy. "A Semantic Approach for Extracting Medical Association Rules". In: *International Journal of Intelligent Engineering and Systems* 13 (2020), pp. 280–292.

[Ber] Tim Berners-Lee. *Linked Data*. URL: https://www.w3.org/DesignIssues/LinkedData.html.

[W3Ca] W3C. *LinkedData*. URL: https://www.w3.org/wiki/LinkedData.

[W3Cb] W3C. *Linking Open Data*. URL: https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.

[W3Cc] W3C. *OWL 2 Web Ontology Language Manchester Syntax (Second Edition)*. URL: https://www.w3.org/TR/owl2-manchester-syntax/.

[W3Cd] W3C. *OWL 2 Web Ontology Language XML Serialization (Second Edition)*. URL: https://www.w3.org/TR/owl2-xml-serialization/.

[W3Ce] W3C. *Semantic Web*. URL: https://www.w3.org/standards/semanticweb/.

[W3Cf] W3C. *Structural Specification and Functional-Style Syntax*. URL: https://www.w3.org/2007/OWL/draft/ED-owl2-syntax-20081128/.