Université Côte d'Azur · Università dell'Insubria

Master M2 Informatique Interactions - Laurea Magistrale in Informatica

Extraction of knowledge from RDF graphs

Student:
Valeria Bellusci

N°étudiant:
21914057

Supervisor (France):
Prof. Andrea G.B. Tettamanzi

Supervisor (Italy):
Prof. Alberto Trombetta

Academic Year: 2019/2020

Acknowledgment

First of all, I would like to thank my supervisor, Prof. Andrea G.B. Tettamanzi, for his assistance and involvement in every phase of the work and, above all, for his understanding.
I am grateful to have worked with him.

Thanks to Duc Minh Tran, Lucie Cadorel, Raphael Gazzotti, and Franck Michel for the suggestions they gave me during the internship.
I thank the I3S and INRIA laboratories for welcoming me and creating an ideal work environment, despite the problems caused by COVID-19. Thanks to Fabien Gandon and every member of the Team Wimmics for the moral support given during the months of lockdown.

A special thanks go to the Universiy of Insubria for allowing me the opportunity to do the Double Degree experience and for funding it.
I thank everyone I met at the Université Côte d'Azur for welcoming me into their world and for allowing me to learn new things every day.
I would also like to thank DS4H for funding my internship, which was an important experience for the start of my career.

Last but not least, I thank my Italian supervisor, Prof. Alberto Trombetta, for reviewing my final report for my degree in Italy.

Abstract

In the context of the Semantic Web, Linked Data (LD) is one of the main concepts and it is used to describe the practice of exposing, sharing, and linking data, information, and knowledge on the Semantic Web. A unique identifier (URI) is used for each resource and a standard format is defined for the data (RDF).
The Resource Description Framework (RDF) is an XML-based framework used to describe resources of the Semantic Web in the form of subject-predicate-object triples and link them together; RDF triples can be queried by SPARQL (SPARQL Protocol and RDF Query Language), which is a standard language for querying graph data. Queries are performed through a set of SPARQL endpoints.
In this work, we aim to analyze and refactor the works previously carried out in the knowledge extraction context, in particular the research work of Duc Minh Tran which aims to discover new knowledge in the form of multi-relational association rules from ontology loaded as a file *.owl.
Subsequently, the objective is to modify the algorithm, inserting a new approach that allows to apply it directly to RDF graphs. Using a SPARQL endpoint, the modified algorithm queries the RDF graph through SPARQL queries to obtain useful information for generating the association rules.
Finally, the consistency of the rules with the knowledge base is checked querying the RDF graph.
The refactored algorithms of Dun Minh Tran and the new algorithm were then published on an Open Source repository to make them available to the scientific community.

Keywords: Semantic web, SWRL, RDF, SPARQL query, Evolutionary algorithms

Contents

List of Figures

Introduction

In these months I did my work in collaboration with the I3s Laboratory, in particular, I was included in the Team SPARKS which is directed by Prof. Andrea G. B. Tettamanzi, who is also my supervisor.
During these months I analyzed, modified, and adapted the work of a previous Ph.D. student, Duc Minh Tran, in the context of the Semantic Web. His work is well presented in his doctoral thesis "Discovering multi-relational association rules from ontological knowledge bases to enrich ontologies" [16].
The main objective of my internship was to refactor his code to publish it as an Open Source and to adapt the modified code to the dataset **Covid-On-The-Web** [1], in order to try to discover new knowledge from it and to make it possible to use this algorithm on generic RDF graphs.

This work is structured as follows:
In this Chapter Introduction (1) I will introduce the environment in which I worked, the main concepts of my internship, with their definitions, and the dataset I worked with. Since it was the first time I approached this topic, I dedicated the first part of the internship to understand the whole context described in this Chapter.
For the description of the environment, the information are taken from the site of I3S [8]. For the definitions and the explanations of the context I mainly used two sources: a site about the introduction to the Semantic Web [13] and the doctoral thesis of Duc Minh Tran [16].
In Chapter 2 (State Of The Art) we will see the state of the art, with particular attention to the work of Duc Minh Tran.
In Chapter 3 (Refactoring and personal contributes) we will see my contribution to his

work and the experiments made.
In Chapter 4 (Generated rules) there will be a presentation of the results obtained.
In Chapter 5 (Conclusion and future perspectives) there will be a conclusion about my internship and the future possible works.

1.1 Work enviroment

The I3S Laboratory [8] is one of the largest information and communication science laboratories in the French Riviera.
It was one of the first ones to settle down on Sophia Antipolis Science and Technology Park. It consists of a little less than 300 people.
It is specialized in the research field of information and communication sciences.
In partnership with CNRS (National Center for Scientific Research, one of the most important research institutions in the world) and INRIA (National research institute in digital sciences and technologies), thanks to so many industrial collaborations, I3S works on innovating research area at the forefront of science and technology: ubiquitous networks and systems, biology and e-health, modeling for the environment, interactions and usages.
The I3S laboratory is composed of four teams:

- Team COMRED (Communications, Networks, Embedded Systems, Distributed Systems),
- Team MDSC (Discrete Models for Complex System),
- Team SIS (Signal, Images et Systems)
- Team SPARKS (Scalable and Pervasive software and Knowledge Systems).

1.1.1 Team SPARKS

The team, composed of 90 members, investigates the organization, the representation, and the distributed processing of knowledge, as well as its extraction from data and its semantic formalization.
A particular attention is dedicated, on the one hand, to large-scale architectures and massive data and, on the other hand, to the design of human- and knowledge-centered,

evolutionary and adaptive software systems.
One of the main fields is the Semantic Web, on which my project is based.
Specifically, I worked in a division of the Team SPARKS, which is the Team WIMMICS (Web-Instrumented Man-Machine Interactions, Communities, and Semantics).

Team WIMMICS

The main challenge of the Team is to bridge formal semantics and social semantics on the web. It is composed by a joint research team between INRIA Sophia Antipolis - Méditerranée and I3S.
The research areas are graph-oriented knowledge representation, reasoning to model and support actors, actions, and interactions in web-based epistemic communities.

1.2 Semantic Web and Linked Open Data

As we know, the web contains tons of information.
In this context, raw data is not available, but only HTML documents structured from data are displayed. Hence, a high level of human effort is necessary for finding, retrieving, and exploiting information.
For example, contemporary search engines are extremely fast, but tend to be very scarce in producing relevant results. Of the thousands of matches generally returned, only a few are relevant. Such problems drastically reduce the value of the information discovered and the ability to automate the consumption of such data.

The **semantic web** tries, in several ways, to remedy this problem: it defines an open standard format for data and encourages the use of data already available on the web.
In this way, a data collection and acquisition environment is created and thanks to this, the data on the web are treated and searched as a single database.
The goal is to share and reuse existing data.
It adopts unique identifiers for the concepts and relationships between them.
These identifiers, called Universal Resource Identifiers (URI), are similar to the URLs of Web pages, but do not necessarily identify documents from the Web.
Their only purpose is to identify unique objects or concepts and the relationships between them.

There exists also another type of identifier, called IRI. The Internationalized Resource Identifier (IRI) is an Internet protocol standard which builds on the Uniform Resource Identifier (URI) protocol by greatly expanding the set of permitted characters.
Thanks to the use of IRIs (or URIs), disambiguity has been almost completely removed. The semantic web also allows concepts to be associated with classification hierarchies, thus making it possible to infer new information based on an individual's classification and relationship with other concepts.
This is accomplished by making use of ontologies (hierarchical structures of concepts) to classify individual concepts.

In the context of the Semantic Web, **Linked Data (LD)** is one of the main concepts and it is used to describe the practice for exposing, sharing, and linking data, information, and knowledge on the semantic web, using URI and RDF. We will see the definition of RDF graphs in the Section 1.2.2.
Linked Open Data (LOD) is Linked Data which is released under an open license, which does not impede its reuse for free.
The LOD is seen as a giant knowledge base from which it is possible to extract "intelligent data", and so to learn knowledge from it. In Figure 1.1 we can see the representation of the Linked Open Data.

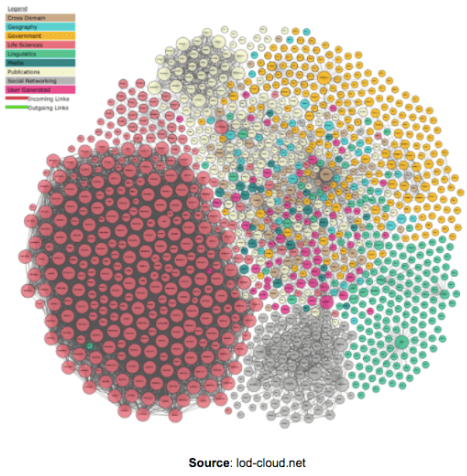

Figure 1.1: An image that represents the data in the Linked Open Data.

1.2.1 Ontology and Knowledge Base

In general, an **ontology** formally describes a domain of related concepts (classes of objects) and their relationships (properties between classes).
For example, in a school setting, staff members, faculty, exams, offices, and departments could be some important classes. Relationships typically include class hierarchies.
A hierarchy specifies that class C is a subclass of another class C' if every object in C is also included in C'.
For example, all teachers are staff members.
In addition to subclass relations, ontology can include information such as:

- property (X is subordinate to Y);
- value restrictions (only managers can direct departments);
- disjunction statements (managers and general employees are disjointed);
- specific of the logical relationships between objects (each office must have at least two teachers).

Ontologies are important because semantic repositories use ontology as semantic schemes. Since the most essential relationships between concepts are integrated into ontology, this makes automated reasoning about data possible (and easy to implement).
Ontologies are described using logical formalisms, such as OWL, which allow automatic inferences about these ontologies and datasets that use them, that is, as vocabulary.
An important role of ontologies is to act as "intelligent" schemes or views on information resources.
This is also the role of ontologies on the Semantic Web.

The **knowledge base (KB)** is a broader term than ontology. Similar to an ontology, a KB allows for automatic inference. It could include multiple axioms, definitions, rules, facts, statements, and any other primitives. Contrary to ontologies, however, KBs are not intended to represent a shared or consensual conceptualization. Hence, we can view an ontology as a specific type of KB. Many KBs can be divided into ontologies and instance data parts.

1.2.2 Resource Description Framework

The Resource Description Framework (RDF) is an XML-based framework used to describe and connect resources.
The data of the Semantic Web must be mapped to RDF.
The World Wide Web has grown rapidly and contains huge amounts of information that cannot be interpreted by machines that cannot understand its Web content. For this reason, most attempts to retrieve some useful information from the Web require manual retrieval of information from multiple sources, "digging" through multiple search engine results by comparing differently structured result sets (most incomplete), and so on.

To make an automatic interpretation of semantic content possible, there are two prerequisites:

- Each concept should be uniquely identified. For example, if a particular person owns a website, he is articles' author on other sites, he has profiles on a couple of social media sites like Facebook and LinkedIn, the occurrences of his name / identifier in all of these places should be linked to the same identifier. It is made thanks to the use of URI.
- There must be a unified system for transmitting and interpreting the meaning of the data.

RDF developed by the World Wide Web Consortium (W3C) makes possible the automatic semantic processing of information, structuring information. RDF is primarily a data model.
In particular, the information is represented as triples of the form *subject-predicate-object* [Fig 1.2].

- **Subject**: the resource to be described.
- **Predicate**: a property of the resource.
- **Object**: the value of the property.

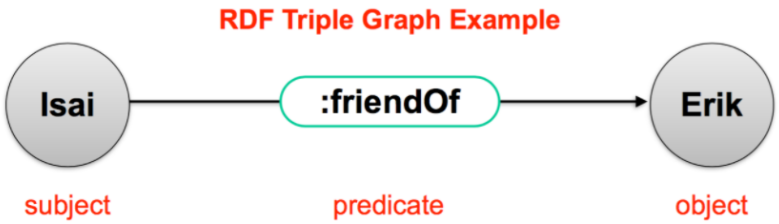

Figure 1.2: RDF triple graph example.

Every describable thing is a subject. Each subject has a predicate (property) which has an object. Each of these three elements is identified by an IRI.
What we have now is the logical formula *P(x,y)*, where the binary predicate P relates the object x to the object y. We can write this formula as x, **P**, y.
In this case, the subject and the object are individuals (instances) of some classes. The class of the subject is the domain of the property and the class of the object is the range of the property.
RDF data can be considered as an oriented and labeled multigraph, composed of nodes attached by edges, where both the nodes and edges have labels.
There are three kinds of nodes in RDF Graphs [Fig. 1.3] :

- Resource Nodes (IRI's) which represent everything that can be said about it through an IRI (Internationalized Resource Identifier) within an RDF graph.
- Literal Nodes which are used for values such as Strings, Numbers, or Dates. The term literal is a fancy word for value.
- Blank Nodes which is a resource without a URI.

A **Named Graph (NG)** is a set of triples called by a URI. This URI can be used to refer to the graph itself.
The ability to name a graph allows you to identify different graphs and also allows you to create statements related to graphs.
The Named Graphs represent an extension of the RDF data model, in which the quadruples <*s, p, o, ng*> are used to define instructions in a multi-graph RDF.
This mechanism allows, for example, provenance management when multiple RDF graphs are integrated into a single repository.

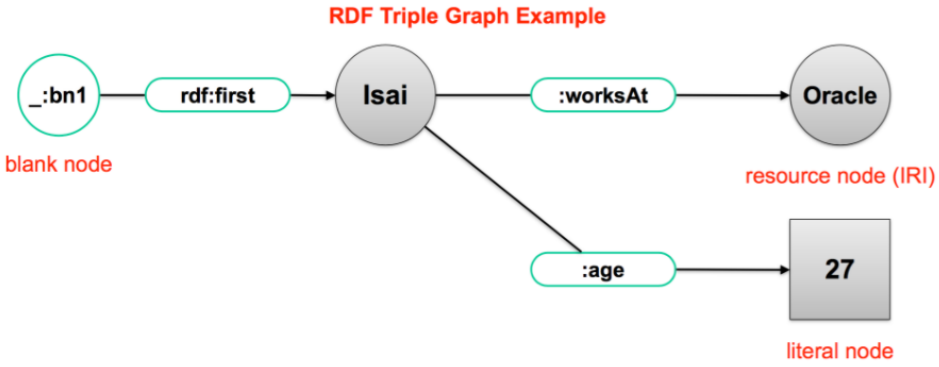

Figure 1.3: RDF different types of nodes.

There exist sets of terms used to describe things, that are the **RDF Vocabularies**.
A **term** is either a *class* or a *property*:

- Class: it is a construct that represents things in the real and/or information world, (a person, an organisation, a concepts). There are also relationship between two classes (in RDF they are called Object Type Properties).
- Property: it is a characteristic of a class in a particular dimension.

1.2.3 Accessing the RDF-graph

The RDF-triples can be queried by the **SPARQL (SPARQL Protocol and RDF Query Language)**, which is a standard language to query graph data in the form of RDF-triples; it is very similar to SQL.
The queries are made through a number of SPARQL endpoints.

Definition *(SPARQL Protocol client).*
It is an HTTP client that sends HTTP requests for SPARQL Protocol operations.

Definition *(SPARQL Protocol service).*
It is an HTTP server that services HTTP requests and sends back HTTP responses for SPARQL Protocol operations. The URI at which a SPARQL Protocol service listens for requests is generally known as a SPARQL endpoint.

Definition *(SPARQL endpoint).*
It is the URI at which a SPARQL Protocol service listens for requests from SPARQL Protocol clients.

1.2.4 SELECT queries

A select-query is usually written with this structure:

- **PREFIX**: it allows you to write prefixed names in queries instead of having to use full URIs everywhere.
  So it is a syntax convenience mechanism for shorter, easier to read (and write) queries.
  (ex: *PREFIX a: <http://www.w3.org/2005/Atom>).*
- **FROM**: the URI of the graph to be interrogated. If several graphs are mentioned, they are merged. If the clause is not specified, all the graphs in the same repository will be queried.
- **WHERE**: RDF triple patterns, so the conditions that have to be met.
- **Query Modifiers**: slicing, ordering, and otherwise rearranging query results.

So, the SPARQL language is used to perform semantic queries which enable the retrieval of both explicitly and implicitly derived information based on the syntax of SPARQL.

One of the goals is to analyze the wealth of data that are available on the LOD to extract information from it, concerning a specific domain of discourse, i.e., learn knowledge from it.

1.3 Covid-on-the-Web Dataset

The dataset [1] is an initiative of the $WimmicsTeam$, $i3sLaboratory$, *University Cote d'Azur*, *Inria*, $CNRS$. In this difficult period caused by the COVID-19 pandemic they wanted to make a contribution to research in this area.
They analyzed the scholarly articles of the **COVID-19 Open Research Dataset (CORD-19)**, which contains articles about the *COVID-19* and the *corona virus family virus* and they obtained two knowledge graphs:

- the CORD-19 Named Entities Knowledge Graph: it describes named entities identified by NCBO BioPortal annotator, Entity-fishing, and DBpedia Spotlight.
- CORD-19 Argumentative Knowledge Graph: it describes argumentative components and PICO elements extracted from the articles by the Argumentative Clinical Trial Analysis platform (ACTA).

These two main graphs are contained in an RDF dataset, the *Covid-on-the-Web Dataset*. The dataset is downloadable as a set of RDF dumps (in Turtle syntax) from Zenod or it can be queried through the **Virtuoso OS SPARQL endpoint** from this link: `http://covidontheweb.inria.fr/sparql`.
For the internship, the Virtuoso SPARQL endpoint has been used, to take constantly updated data, without having to download them every time.
To do this, SPARQL Query Language is used to make the queries and the framework *Jena* is used to manage the queries results in Java.

1.4 Goals of the internship

The main goals of my internship are three:

1. Since it was the first time I approached this topic, the first objective was to understand all the context about the Semantic Web and to understand some previous works described in the next Chapter 2.
   Specifically, the main goal was to understand the work "Discovering multi-relational association rules from ontological knowledge bases to enrich ontologies" [16]. It is the thesis of a previous Ph.D. student, Duc Minh Tran, and the purpose was to modify it suitably in order to publish it as Open Source code and to adapt it for the Covid-on-The-Web dataset [1] in the later phases.
2. Refactor the codes, with particular attention to the work of Duc Minh Tran.
3. Adapt the code of Minh to the dataset Covid-On-The-Web [1] to make it possible to use the code directly on RDF graphs, rather than on *.owl files.

State of the art

The union of the two scientific research areas of Semantic Web and Web Mining is known as Semantic Web Mining. A first survey [14], dating back to 2006, analyzes the work done in those two research areas up to that point, then outlines ways of how closer integration could be profitable. Another state-of-the-art survey on Semantic Web Mining in 2013 [12] gives a detailed account of the advances in this new research area.

In recent years, many research works have been done in this area. I will mention a few, from research groups of several universities.

- Institute for Web Science and Technologies (WeST) at the Universität Koblenz-Landau, founded by Prof. Dr. Steffen Staab: it is a group of researchers investigating many topics such as the semantic web, information retrieval, and web engineering and aim to a better understanding of the impact of the web on our society.
  One of their work is "Ontology learning for the Semantic Web" [10]: they present a framework for the ontology learning that extends typical ontology engineering environments by using semiautomatic ontology construction tools. The framework includes ontology import, extraction, pruning, refinement, and evaluation.

- The works of Claudia d'Amato et al. at the University of Bari concern the integration of inductive and deductive reasoning to discover useful patterns in datasets with a rich basic knowledge (ontologies).
  Among theme, we mention the work "DL-FOIL Concept Learning in Description Logic" [5]: a FOIL-like algorithm that can be applied to general DL languages. They present an experimental evaluation of the implementation of the algorithm

performed on some real ontologies to empirically assess its performance.

- Smart Data Analytics Research Group at the University of Bonn, led by Prof. Dr. Jens Lehmann: the group investigates machine learning techniques ("analytics") using structured knowledge ("smart data").
  A very interesting system proposed by Jens Lehmann is described in "DL-Learner: Learning Concepts in Description Logics" [9]: it introduces DL-Learner, a framework for learning in description logics and OWL. It includes several learning algorithms, support for different OWL formats, reasoner interfaces, and learning problems. It is a cross-platform framework implemented in Java. The framework allows easy programmatic access and provides a command-line interface, a graphical interface.

- The Karlsruhe Institute of Technology (KIT), led by York Sure-Vetter, is the only German university of excellence with a large-scale national research sector. They are committed to creating and transmitting knowledge and they provide pioneering contributions to research, especially in the areas of the energy transition, future mobility, and technologies for the information society. We mention in particular two works of Johanna Völker et al:

  – "Ontology Learning and Reasoning — Dealing with Uncertainty and Inconsistency" [7]: from a logical perspective, learned ontologies are potentially incoherent knowledge bases, which as such do not allow meaningful reasoning. In this article, an approach to generate coherent OWL ontologies from automatically generated or enriched ontology models is presented, which takes into account the uncertainty of the acquired knowledge;
  – "Statistical Schema Induction" [17]: a statistical approach for the induction of association rules from large RDF datasets, then translated into OWL 2 EL axioms.

- The team LACODAM, located at INRIA/IRISA in Rennes, with the purpose of facilitating the process of making sense from large quantities of data, either for deriving new knowledge or for taking better actions.
  Luis Galárraga et al. proposed a new system, AMIE, described in the article "AMIE: association rule mining under incomplete evidence in ontological knowledge bases" [6]: they developed a rule mining model (the AMIE system) that is

explicitly tailored to support the OWA scenario, meaning that absent data cannot be used as counterexamples. It is inspired by association rule mining and introduces a novel measure for confidence.

- Team Wimmics, the team in which I am involved, composed by members from I3s and Inria, led by Fabien Gandon. Their challenge is to bridge formal semantics and social semantics on the web. We will see some of their works in the next section.

2.1 RDF Mining

RDF Mining could be viewed as a large project about the semantic Web, which includes research works bringing together the semantic Web, possibility theory, and evolutionary algorithms.
It is described in the working paper by Andrea G.B. Tettamanzi [15].
The question behind the RDF Mining project is: what can be gained from Linked Open Data?
A first answer is that knowledge can be extracted from them.
Some preliminary works have already been done in this direction and together they form the RDF Mining project.
For my internship we considered mostly these works:

- Ontology Enrichment by Discovering Multi-Relational Association Rules from On-Ontological Knowledge Bases [2]: they present a method for discovering multi-relational association rules, coded in SWRL, from ontological knowledge bases. The discovered rules can be integrated within the ontology, to enrich its expressive power and to augment the assertional knowledge that can be derived. Discovered rules may also suggest new axioms to be added to the ontology;

- Evolutionary Discovery of Multi-Relational Association Rules from Ontological Knowledge Bases [3]: the method is similar to the one explained above, but in this case, they use genetic operators. The discovered rules are represented in SWRL;

- Comparing Rule Evaluation Metrics for the Evolutionary Discovery of Multi- Relational Association Rules in the Semantic Web [4]: carry out a comparison of popular asymmetric metrics, used to score candidate multi-relational association

rules in an evolutionary approach to the enrichment of populated knowledge bases in the context of the Semantic Web;

- Learning Class Disjointness Axioms Using Grammatical Evolution [11]: the method proposes the use of Grammatical Evolution, one type of evolutionary algorithm, for mining disjointness OWL 2 axioms from an RDF data repository such as DBpedia.

The approach proposed by these researchers works as follows:
the hypotheses are generated by evolutionary algorithms or by the level-wise and generated-and-test algorithm; they are tested using a deductive process and then they are added to the population of axioms if the test results are positive [Fig. 2.1].

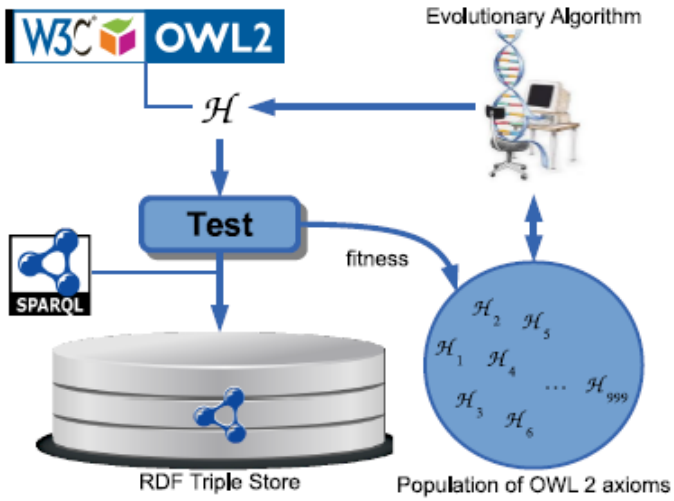

Figure 2.1: An illustration of the methodology used, with evolutionary algorithms.

In all the phases of the internship the work of Duc Minh Tran "Evolutionary Discovery of Multi-Relational Association Rules from Ontological Knowledge Bases" [3] was fundamental.
The idea behind his method is to discover frequent hidden patterns in the form of multi-relational association rules from knowledge bases.
Thanks to these rules it is possible to enrich the assertional knowledge, in particular, to make predictions of new assertions in the ontology.
Now, we will see in details this work.

2.2 Discovery of Multi-Relational Association Rules

The aim is to build an algorithm that is used for inductive learning from assertional data in the ontology, to discover hidden knowledge patterns from ontological knowledge bases.
The kind of learned knowledge can be reintegrated into the ontology to enrich the data, and from them, it is possible to induce new knowledge.

The goal, in particular, is to discover frequent hidden patterns in the form of multi-relational association rules (Horn-like clause of the form body $\rightarrow$ head), coded in SWRL.

Definition *(Relational Association Rules).*
Given a populated ontological knowledge base K, a relational association rule r for K is a $Horn-like$ clause of the form body $\rightarrow$ head, where:

- Body is a generalization of a set of assertions in K co-occurring together.
- Head is a consequent that is induced from K and body.

Definition *(Semantic Web Rule Language (SWRL)).*
Rules in SWRL are implication rules and they can be used to infer new knowledge from the existing OWL knowledge bases, respecting the language bias. The head and body consist of a conjunction of one or more atoms.

The **language bias** is a set of constraints that give a specification of the pattern worth considering.
In this work they considered only connected and non-redundant rules satisfying the safety condition:

- The constraint to connected rules avoids mining rules with completely unrelated atoms;
- The constraint to non-redundant rules avoids obvious mining rules;
- ***Definition*** *(Safety Condition).*
  Given a knowledge base K and a rule $r = B_1 \wedge B_2 \cdots \wedge B_n \rightarrow H$, r satisfies the

safety condition if all the variables appearing in the rule's head also appear in the rule's body.

When a rule is created, to assess the quality of it and add it in the ontology, it is necessary to use metrics:

- Set of distinct bindings of the variable in the head of the considered rule,
- Set of distinct bindings of the variable in the head of the considered rule provided that the body and the head are satisfied,
- Set of distinct bindings of the variable in the head of the considered rule also appearing as binding for the variables in the body.

There are also some methods for ensuring the monotony:

- Rule Support: given by the number of distinct bindings of the variable in the head.
- Head Coverage: given by the proportion of the distinct variable-bindings from the head that are covered by the prediction of the rule.
- Rule Confidence: the ratio of the binding of the predicting variables in the rule head and their binding in the rule's body.
- Rule Precision: the ratio of the number of correct predictions made by the rule and the total number of correct and incorrect predictions.

So, it is possible to evaluate the rule to put it in the set of new rules discovered.

2.2.1 Genetic algorithm

As Minh explains in his doctoral thesis [16], genetic algorithms are algorithms based on the idea of natural selection and genetic inheritance.
They are used to find the optimal solutions (or near-optimal), although this depends on the number of generations used.
There is a population of individuals (patterns) in the search space; each individual describes a possible solution for a given problem and it has an associated fitness value that is its competitiveness with respect to the other individuals.
New generations of population are created by applying genetic operators as selection,

recombination (or crossover), and mutation on the individuals of the previous generations. So, the new individuals are added in the population and the individual with the lowest fitness value is removed.
The aim is to create offspring which are better than their parents and from the last population generated to discover rules capable of making a large number of predictions.

The weakness of the genetic algorithm is that it can miss some rules (we call these rules the undiscovered rules). In the thesis of Minh is described also a deterministic approach to find rules, the Level-Wise and Generating-And-Test [2]: thanks to this approach all the possible rules are discovered in a given space; specifically, space is determined by the maximum length of the rules to be discovered. However, this method has a fundamental disadvantage: since the size of the search space increases exponentially with the maximum rule length, it is difficult to obtain long rules because the execution time explodes. For this reason the use of the genetic algorithm is an improvement: although the genetic algorithm can hardly find all possible rules in the space determined by the given maximum length, it can discover rules of long length along with selecting and keeping the best rules that it traverses in the search space.
To overcome the weakness of the genetic algorithms, they chose the appropriate number of generations to minimize the number of undiscovered rules.
In Figure 2.2.1 we can see the difference between the two approaches.

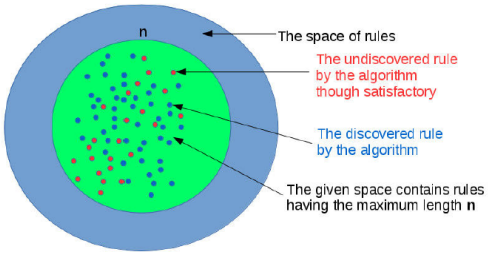

Figure 2.2: The search space of rules used in genetic algorithm.

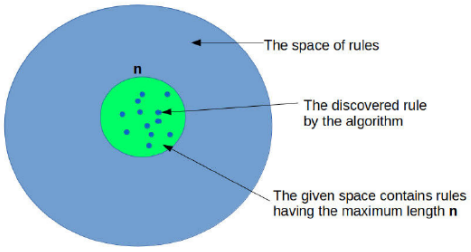

Figure 2.3: The search space of rules used in level-wise and generate-and-test algorithm.

2.2.2 An algorithm for discovering SWRL Rules

The algorithm is a genetic algorithm that repeats the following phases until a stopping criterion is met [Fig. 2.4]:

1. At first, it starts by randomly initializing the population, associating a fitness' value for all the individuals in the population, and selecting parents from this population for mating.

2. Then, it applies crossover and mutation operators to generate new offspring.

3. Finally, these offspring are added into the population and so they compete with all the individuals of it. In this way, only the best individuals survive until the final generation.

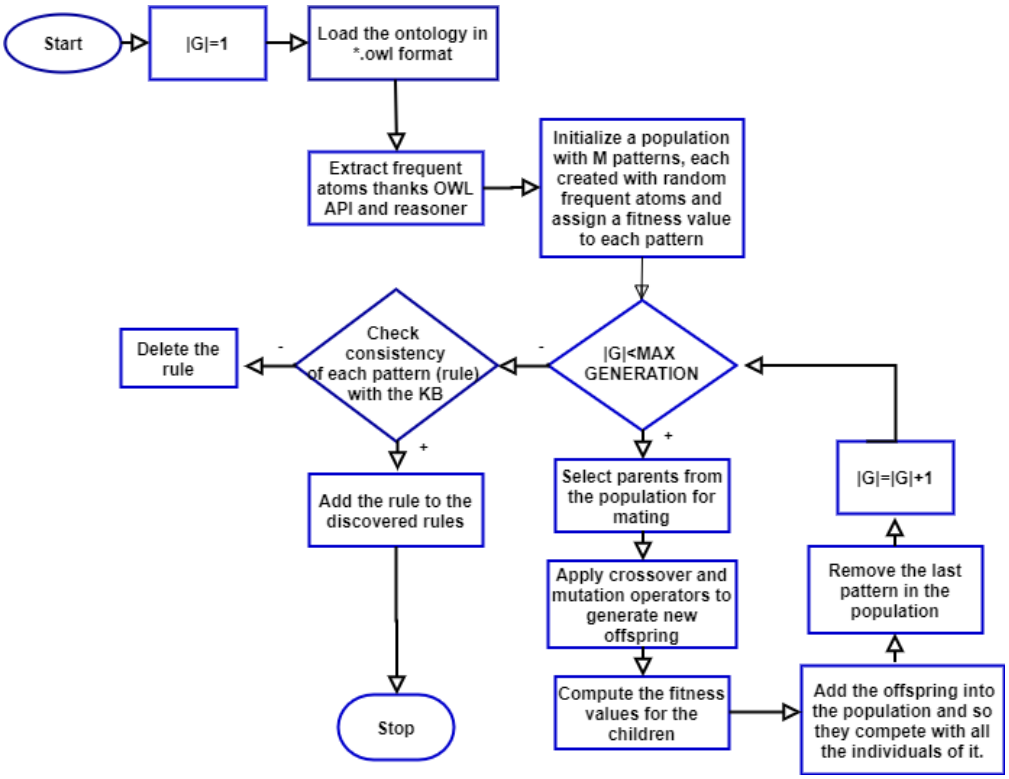

Figure 2.4: An overview of the algorithm.

Initialization of the population

The algorithm initializes once and for all a list of frequent atoms A_f. An atom could be taken from the Concept Atoms, which represent the Classes in the KB or it could be taken from the Role Atoms, which represent the Properties of the KB.

The function $CREATENEWPATTERN$ is called n times to create a population of n patterns. It initializes the maximum length of the pattern and then it chooses a frequent atom from the list A_f and it adds that atom at the end of the pattern (in respect to the language bias).
This process is repeated until the maximum length of the pattern is reached.

Then, the fitness value for each pattern is computed.
The patterns are sorted by decreasing fitness values, to have the best individuals to the top of the list and so to choose the best ones for the crossover's phase.

Crossover and mutation operators

Thanks to the crossover operator it is possible to produce two offspring patterns from two parents patterns chosen from the previous phase.
The atoms of the offspring are randomly selected from the atoms of the parents:
A set L is created, containing all the atoms in the parents' patterns and choosing a target length for the two offspring.
Each output pattern is created by choosing an atom at random from L and adding it to the end of the pattern. This is performed iteratively until the maximum length is reached.

The mutation operator perturbs a pattern from the offspring with a given probability and it is based on the idea of specialization and generalization.
If the fitness value of the corresponding rule of the pattern is above a given threshold, then it applies the specialization (it appends a new atom to a pattern, preserving the language bias).
If it is below the threshold then it applies the generalization (it removes the last atom of the pattern, preserving the language bias).

Fitness and selection

When the fittest individuals are selected for the reproduction, the remaining individuals are replaced by the offspring of the selected individuals.
A pattern from the final generation is evaluated by constructing a rule from it, coded in SWRL.
A first fitness function is the head coverage of that rule and a second fitness function is a combination between the head coverage and the confidence of the rule. With these two different fitness functions, they proposed two independent experiments and the best rules were obtained using the second fitness value.

Consistency of the rules

The consistency of the patterns is checked only in the final population, without checking during the evolution, for two reasons explained in the thesis:

1. Checking rules for consistency may be computationally very expensive.
2. It is not necessary to immediately omit the inconsistent rules during evolution because even if we apply a genetic operator to inconsistent rules, its offspring may still be consistent rules.

The consistency check is developed in a separate algorithm, which uses Threads to have multiple processes checking multiple rules at the same time. The consistency is verified by reasoners, using the Hermit API or Pellet API library (see Section 3.1.1 for more details).

Refactoring and personal contribute

After understanding the context, analyzing the previously created algorithms and in particular the thesis of Duc Minh Tran [16], the next step was to refactor the last version of these codes to publish them as OpenSource: the algorithm to generate the rules and the algorithm to check these rules.

3.1 Code Refactoring

The main changes that have been made are the following:

- Added the Java Doc to all the methods in the program since there was not.
- Added comments where the code is not easy to understand.
- Deleted unused code.
- Modified some methods, to improve readability.
- Delete and replace deprecated methods.

The main changes have been made to the algorithm to generate the rules.
It is composed by 11 packages and 23 classes.
The most important packages are:

- *MKnowledge*: it contains the class *KnowledgeBase*, which loads the ontology and check if it is consistent thanks to the use of reasoners. This package will be removed in the code re-adaptation for RDF graphs, since we don't want to upload the ontology as a file.

- *MDataProcessing*: it contains the classes with which the data structures containing all the elements necessary to create the patterns will be formed, taking them from the ontology loaded. This package will be modified to allow the extrapolation of the data directly from the SPARQL endpoint.

- *MGeneticAlgorithm*: it contains the classes with which the evolution of the population is managed. This package will stay the same.

- *MPatternComputation*: it contains the classes with which the fitness value and all the metrics of a pattern are calculated. This package will stay the same.

The code of this algorithm was written by Duc Minh Tran in Java, without any comments or JavaDoc, because the purpose of the work was to discover a new approach and so in the first moment it was not so important to write a commented code.

3.1.1 Libraries

At the beginning of my internship, the algorithm has been analyzed and understood. After this, we tried to run it and so all necessary libraries have been searched, also trying to understand which of these would have been used to adapt the code in the last phase of the internship.
The libraries that need to be installed are as follows:

- HermiT and Pellet API: these two are called reasoners and they are necessary to understand if the ontology is consistent and then when the new rules are discovered, the algorithm uses the two reasoners to check the consistency of the rules.
  It is possible to download the HermiT reasoner from the official site: download HermiT Reasoner.
  For the Pellet Reasoner we used the version used by Minh because this reasoner is no longer available so it is not possible to find a new version which can be downloaded.
  For the version on RDF graphs, described in the next section 3.2, this library will not be needed.

- Json: it is a format used to show data from external sources and paginate them according to our solutions. It is possible to download the library from here: download JSON.

- JUnit and Hamcrest: these are open-source frameworks that support the writing and the automatic execution of Test Cases and Test Suites. The download of the latest version of the former is available here: download JUnit.
  Hamcrest is a framework for writing matcher objects allowing 'match' rules to be defined explicitly. The latest download is available here: download Hamcrest-Core.

- OWL API: it is a Java API and reference implementation for creating, manipulating, and serializing OWL Ontologies. Concerning this algorithm, the library is used to extract the classes, properties, and all necessary elements. The latest version of the API is focused on OWL 2 and it is available from here: download owlapi-distribution.
  In this contest, we need also JPaul and JDom APIs.
  Thanks to Jpaul it is possible to have algorithms (like graph algorithms, data-flow engines, set constraint solvers, etc) implemented independently of any compiler infrastructure project. Download available here: download JPaul API.
  JDom is a complete, Java-based solution for accessing, manipulating, and outputting XML data. It is available here: download JDom API.
  For the version, described in the next section, on RDF graphs, these libraries will not be needed.

- Apache-Jena: it is a free and open source Java framework for building Semantic Web and Linked Data applications. The framework also includes various logical deduction engines. It is available from this link: download JENA.
  This library will allow in the new version of the code to execute queries to extract data from RDF graphs.

After the download of these libraries, the algorithm works and it produces SWRL rules.

3.1.2 Comments

The algorithm was written without any comments or Java DOC.
What has been done was understand all the methods and comment on the unclear parts of the code to an inexperienced user.
The basic information on automatically generated packages, classes, methods and fields can be enriched with further details by means of JavaDoc comments; these are enclosed between the sequences of characters / ** and * / (in fact they are a particular form of

multi-line comment), and are added to the documentation of the element that follows them:

- The first paragraph is a description of the method documented.
- Following the description are a varying number of descriptive tags, signifying:
  - The parameters of the method (@*param*)
  - What the method returns (@*return*)
  - Any exceptions the method may throw (@*throws*)
  - Other less-common tags such as @*see* (a "see also" tag)

We also removed all the commented code and all the warnings.

3.1.3 Methods

In the algorithm, some lines could be written differently.
For example, very often in the class *Pattern* three identical *ConceptAtoms* were created and each of them was inserted in three different data structures.
Instead of the three concept atom now we have only one (since they were identical) and this is inserted in the three different data structures.
Another example is that there were some $if - else$ clauses which can be written with a single if, since the $if - corp$ and the $else - corp$ were equals, except for one command.
These are some of the changes made to the algorithm to make it less repetitive but with the same result.

As for the algorithm for verifying that the rules are consistent, the main changes have been about JavaDoc, comments and deprecated methods.
In particular, the method Thread.stop() used by Minh to kill a Thread is now deprecated in Java and there is no method to replace it.
Thus, a method called "terminate()" was created and it could replace the method stop(), using a Boolean variable:

- If the variable is false, the process can continue to work,
- If the variable is true, then it must terminate.

3.2 Adapting the code

At the beginning of the code to generate rules, there are nine principal methods containing in the package $MDataProcessing$, used to populate the data structures containing the main elements of the datasets:

- $createFrequentConceptsStratified$: an $ArrayList$ used to collect all the couple $(class, individuals)$, where $class$ is a concept of the ontology and the second element is the set of individuals which belong to this class.
- $createConceptIsSubsumedByConcept$: a Map used to collect all the pairs (A, B_i), where B_i are the superclasses of the class A.
- $createConceptSubsumsConcepts$: a Map used to collect all the pairs (A, B_i), where A is a superclass of the classes B_i.
- $createFrequentRolesStratified$: an $ArrayList$ used to collect all the properties and their individuals (similar to the concepts data structure)
- $createRoleIsSuperPropertyRoles$: used to collect all the pairs (A, B_i), where B_i are the sub properties of A.
- $createRoleIsSubPropertyRoles$: a Map used to collect all the pairs (A, B_i), where A is the sub property of the proprieties B_i.
- $createConceptsDomainOfRole$: a Map used to collect all the pairs $(property, domains)$, where the first is a property and the second is the set of domain classes of this property.
- $createConceptsRangeOfRole$: a Map used to collect all the pairs $(property, range)$, where the first is a property and the second is the set of range classes of this property.
- $getAllOfIndividuals$: a Set used to collect all the individuals containing in the ontology.

In this algorithm the ontology is loaded by a file $*.owl$ and all these data structures are created using the OWL API, which allows to extrapolate concepts, roles, etc. directly from ontology, using the methods of the library itself.

Our goal, however, is not to upload the dataset **Covid-On-The-Web** [1] directly as a file, but rather to extrapolate the data directly from the created *SparqlEndpoint* in order to have recent results, according to the changes of the dataset. To do this, it is necessary to do some queries to the endpoint and populate the data structures thanks to their results.
The methods explained above have therefore been replaced with methods that collect the results of the queries.

3.2.1 Queries

With respect to the method thanks to which Minh filled the data structure presented in the section above (Section 3.2), the number of queries is four for the classes and six for the properties.
Classes:

1. *individualsQuery*: it is the query about the classes and their individuals. It is motivated by a simple heuristic: every object of an rdf:type statement is a class, whereas the subjects of all such statements provide us with the instances of these classes.
   For this reason, in the beginning, the query was as follow (the keyword "a" is equivalent to rdf:type in SPARQL):
   SELECT ?class ?ind
   WHERE {
   ?ind a ?class. FILTER(isIRI(?ind)
   }.
   This query returns single pairs (class, individual) and, for each pair, the class and the individual are taken as a single element in the array list and, whenever the result gave a class that had already been entered, the individual is added to the individuals of that class.
   Since the number of class with individuals is about 8000 and the server has a limit of rows returned, the results were drastically reduced by the server, because a single class could have many individuals, and so the rows of this result could be also 10000 or more.

So, the new query takes only the class, and for each class, there is another query which takes all the individuals of this class :

- **SELECT DISTINCT ?class**
  WHERE {
  ?ind a ?class.
  FILTER(isIRI(?ind))
  }
- For each class:
  SELECT DISTINCT ?ind
  WHERE {
  ?ind a $< IRIclass >$ **.**
  FILTER(isIRI(?ind))
  }

In this way, there are many more queries to be made, but the results are complete and the array about the classes is filled. Each element corresponds to the pair (class, individuals).

2. *queryEndpointSubsumed*: it is the query which takes for each class, the super-classes. In this case, we used the statement rdfs:subClassOf, which is a property, instance of rdf:Property, that is used to state that all the instances of one class are instances of another.
   Also for this query there are two versions.
   The first version returns two results in the *SELECT clause* but it reduced the number of results.
   In the latest version the ArrayList filled by the previous query is scrolled and for each IRI class in the array the query about the superclasses is made:
   SELECT DISTINCT ?superClass
   WHERE {
   $< IRIclass >$ **rdfs:subClassOf ?superClass.**
   FILTER(?superClass != owl:Thing)
   }
   The results are inserted in a pair (class, super-classes) in a HashMap, where the class is the key and the set of superclasses is the value.

3. *queryEndpointSubsumes*: this query is very similar to the query about the superclasses of a single class.
   The latest version of this query is:
   SELECT ?subClass
   WHERE {
   ?subClass rdfs:subClassOf $< IRIclass >$
   }
   This is made for each IRI class in the ArrayList and the results are inserted in a HashMap: the class is the key and his set of subclasses is the value.

Finally, we decided to put all the queries in a single method (Algorithm [1]):
the first query about the class is called and for each result, the other queries are made.
So, for example, one result(row) of the query
SELECT ?class
WHERE {
?ind a ?class.
FILTER(isIRI(?ind))
}
could be the IRI "http://rdfs.org/sioc/argument/Justification".
As soon as this result is found, the queries are immediately executed: one to find its individuals, one to find superclasses, and one to find its subclasses. In this case, there is no superclass and there is one subclass, which is "http://rdfs.org/sioc/argument/Argument".
After doing this, the same thing is done for the other classes founded with the first query.
We made this change because for endpoint characteristics it is better to do in nested queries, rather than many separate queries with numerous results, otherwise it is possible to have an error like *"500 error: Request Failed"*.
Also from an algorithmic point of view, this choice is better, since in this way the array list of the classes is not scrolled every time.

The same idea is behind the queries about the **properties**.
In this case we have six queries and also for these there are different versions.
I show only the last (Algorithm [2]):

- *queryProperty*: it is the query about the properties: we want their IRIs, their subjects and objects, their sub-property/super-property, and their domain/range

classes.
There is a query which takes only the property, and, also in this case, as soon as a result is returned, the other queries are immediately executed.
In the first query, from the triples subject-property-object, we removed the property rdf:type, since in this case the objects returned would be classes and the subjects would be instances of classes.
Then, the rdfs:subPropertyOf property is used to state that one property is a subproperty of another.
The rdfs:range is an instance of rdf:Property that is used to state that the values of a property are instances of one or more classes.
The rdfs:domain is an instance of rdf:Property that is used to state that any resource that has a given property is an instance of one or more classes.

- **SELECT ?prop**
  WHERE {
  ?sub ?prop ?obj.
  FILTER (?prop != rdf:type)
  }
- For each property:
  1. SELECT DISTINCT ?sub ?obj
     WHERE {
     ?sub < $IRIprop$ > ?obj
     }
  2. SELECT DISTINCT ?subProp
     WHERE {
     ?subProp rdfs:subPropertyOf < $IRIprop$ >.
     FILTER (?subProp != owl:BottomObjectProperty)
     }
  3. SELECT DISTINCT ?superProp
     WHERE {
     < $IRIprop$ > rdfs:subPropertyOf ?superProp.
     FILTER (?superProp != owl:TopObjectProperty)
     }

4. SELECT DISTINCT ?domain
   WHERE {
   $< IRIprop >$ rdfs:domain ?domain.
   FILTER (?domain != owl:Thing)
   }
5. SELECT DISTINCT ?range
   WHERE {
   $< IRIprop >$ rdfs:range ?range.
   FILTER (?range != owl:Thing)
   }

Thanks to these queries the data structures are filled and the program can use this information to run correctly.
The problem that has been encountered is the memory. Indeed, the algorithm works pretty well with a small ontology (with few classes and properties).
But as we can imagine, the dataset about the COVID articles is very big and it was a problem for the program and after less than 50 generation "OutOfMemory" error appears.
The solution found was to limit the number of individuals of each class and each property to 500, because the algorithm when computes the metrics for a pattern (and so the fitness value) must control each individual and it takes a very long time.
Then, another thing to do is to query not all the graphs of the dataset but only the main ones, to decrease the amount of extrapolated data, leaving the most significant.
To do this it is necessary to modify the queries, inserting the *FROM clause* after the *SELECT clause* and before the *WHERE clause*.

Graph of the dataset Covid-On-The-Web

- CORD-19 Named Entities Knowlegde Graph:
  - http://ns.inria.fr/covid19/graph/articles: it contains the articles metadata (title, authors, DOIs, journal etc.).
  - http://ns.inria.fr/covid19/graph/metadata.: it contains a description of the dataset and the definition of a few properties.

  - http://ns.inria.fr/covid19/graph/dbpedia-spotlight: named entities identified by DBpedia Spotlight.
  - http://ns.inria.fr/covid19/graph/entityfishing: named entities identified by Entity-fishing in articles title/abstract.
  - http://ns.inria.fr/covid19/graph/entityfishing/body: named entities identified by Entity-fishing in articles bodies.
  - http://ns.inria.fr/covid19/graph/bioportal-annotator: named entities identified by Bioportal Annotator in articles titles/abstracts.
  - http://ns.inria.fr/covid19/graph/acta: argumentative components and PICO elements extracted by Argumentative Clinical Trial Analysis platform (ACTA) from articles titles/abstracts.

- COVID-19 Literature Knowledge Graph:
  - https://www.kaggle.com/group16/covid19-literature-knowledge-graph: Information about the generation of this Knowledge Graph. https://github.com/GillesVandewiele/COVID-KG.
- CORD-19-on-FHIR: A FHIR RDF dataset for COVID-19 research: 6 graphs.
- Other resources.

We used these graphs with different combination to query the dataset from the SPARQL endpoint.
Another problem is that if the query is like this:

```
SELECT DISTINCT ?prop
FROM <http://ns.inria.fr/covid19/graph/articles>
FROM <http://ns.inria.fr/covid19/graph/metadata>
FROM <http://ns.inria.fr/covid19/graph/entityfishing>
FROM <http://ns.inria.fr/covid19/graph/dbpedia-spotlight>
FROM <http://ns.inria.fr/covid19/graph/bioportal-annotator>
FROM <http://ns.inria.fr/covid19/graph/acta>
WHERE {
?sub ?prop ?obj.
FILTER ( ?prop != rdf:type ) }
```

the server returns a *"Proxy Error"* or *"S1T00 Error SR171: Transaction timed out"*, because the machine used is not powerful enough to support the load or because the timeout is over (the problem was then solved with a new server for the endpoint).
The solution found was to write in a file the name of the graphs, read the file using the *FileReader* object and for each line read (graph) make a query like this:
SELECT DISTINCT ?prop
FROM < graph read >
WHERE {
?sub ?prop ?obj.
FILTER (?prop != rdf:type)
}
It is possible in this way to have duplicate classes or properties, but the problem is easily solved by adding in the data structure the class/property only if it is not already there, and only in this case the other queries are made.

A starting point for future works

In the last period of the internship, once I have acquired more knowledge in this area and once the endpoint has been replaced, we tried to use more specialized queries for the classes: thanks to the entity fishing technique we can extrapolate from Wikidata the entities of the Covid-On-The-Web dataset.
In practice, the queries made previously work well with datasets where all classes and properties are defined using the classic statements. In our case, however, the elements we want are defined as Wikidata entities.
What the following query does is take from the graph concerning the articles, all the entities with the *hasBody* property, which has as its object the IRI of the entity on which the body of the article is centered.
Then, by querying the Wikidata graph, it associates each entity of the body to a Wikidata class through the Wikidata property wdt: P31 (instance of) and it returns these classes. For each class there are other 3 queries for the individuals, the subclasses and the superclasses.
The property wdt:P31 is used because Wikidata has its own property to define the class of an instance, and so the use of the statement rdf:type is not possible.
The same thing happens for the statement rdfs:subClassOf, replaced by the property

wdt:P279.
These are the queries to find the classes with their associated elements:

- **SELECT distinct ?classBody**
  FROM <http://ns.inria.fr/covid19/graph/entityfishing>
  FROM named <http://ns.inria.fr/covid19/graph/wikidata-named-entities-full>
  FROM <http://ns.inria.fr/covid19/graph/articles>
  WHERE {
  ?x oa:hasBody ?body.
  GRAPH <http://ns.inria.fr/covid19/graph/wikidata-named-entities-full>
  { ?body wdt:P31 ?classBody }
  }

- For each class:

  1. SELECT distinct ?ind
     WHERE {
     GRAPH <http://ns.inria.fr/covid19/graph/wikidata-named-entities-full> {
     ?ind wdt:P31 <*IRI class* > }
     }
  2. SELECT distinct ?subClass
     WHERE {
     GRAPH <http://ns.inria.fr/covid19/graph/wikidata-named-entities-full> {
     ?subClass wdt:P279 <*IRI class* > }
     }
  3. SELECT distinct ?superClass
     WHERE {
     GRAPH <http://ns.inria.fr/covid19/graph/wikidata-named-entities-full> {
     ?superClass wdt:P279 <*IRI class* > }
     }

This type of query is most useful for finding satisfactory results for this type of dataset.
This approach could therefore be better developed in the future.
In the next Chapter, we will see some of the results with both the approaches.

Algorithm 1: Data structures about classes

Result: One list containing the classes and their individuals, and two HashMaps containing respectively the pairs (class, superclasses) and the pairs (class, subclasses)

```
1  IRI: IRIclass;
2  String: individual, resultClass, resultIndividual;
3  ArrayList: frequentConcepts, allIndividuals, allIRI, individuals;
4  HashMap: conceptIsSubsumed, conceptSubsumes;
5  begin
6      Connection with the SPARQL Endpoint;
7      Execute the query which returns all the classes;
8      for class in the results of the query do
9          IRIclass ← resultClass ;
10         if allIRIs doesn't contain the IRIclass then
11             allIRI ← IRIclass;
12             Execute the query which returns all the individuals of the IRIclass;
13             for individual in the results of the query do
14                 allIndividuals ← individual ;
15                 individuals ← individual;
16             end
17             frequentConcepts ← (IRIclass, individuals);
18             Execute the query which returns all the superclasses;
19             for superclass in the results of the query do
20                 superClasses ← superclass ;
21             end
22             conceptIsSubsumed ← (IRIclass, superclasses);
23             Execute the query which returns all the subclasses;
24             for subclass in the results of the query do
25                 subClasses ← subclass ;
26             end
27             conceptSubsumes ← (IRIclass, subclasses);
28         end
29     end
30 end
```

Algorithm 2: Data structures about properties

Result: One list containing the properties and their individuals, and 4 HashMaps containing respectively the pairs (property, superproperties), (property, subproperties), (property, domain classes), (property, range classes),

IRI: *IRIproperty*;

String: *subject*, *object*,*resultProperty*;

ArrayList: *frequentRoles*, *allIndividuals*, *roleIndividuals*, *allIRIProperties*, *domainClasses*, *rangeClasses* ;

HashMap: *roleIsSuperProperty*, *roleIsSubProperty*, *roleDomains*, *roleRanges*;

```
begin
    Execute the query which returns all the properties;
    for property in the results of the query do
        if frequentRoles doesn't contain the IRIproperty then
            allIRIProperties ← IRIproperty;
            Execute the query which returns all the subj and obj of the
             IRIproperty;
            for (subject, object) in the results of the query do
                allIndividuals ← subject ;
                allIndividuals ← object ;
                roleIndividuals ← (subject, object);
            frequentConcepts ← (IRIclass, roleIndividuals);
            Execute the query which returns all the super properties of the
             IRIproperty;
            for super property in the results of the query do
                superProperties ← superproperty ;
            roleIsSubProperty ← (IRIproperty, superProperties);
            Execute the query which returns all the sub properties;
            for sub property in the results of the query do
                subProperties ← subproperty ;
            roleIsSuperProperty ← (IRIproperty, subProperties);
            Execute the query which returns all the domain's classes ;
            for domain class in the results of the query do
                domainClasses ← domainclass ;
            roleDomains ← (IRIproperty, domainClasses);
            Execute the query which returns all the range's classes;
            for range class in the results of the query do
                rangeClasses ← rangeclass ;
            roleRanges ← (IRIproperty, rangeClasses);
```

Generated rules

4.1 Results of the queries

After querying the dataset from the endpoint, the recorded results are in three different files:

- One containing all the IRI of the classes, and everything related to them,
- One containing the IRI of the properties and everything related to them,
- The last containing all the individuals (and so all the individuals of the classes and all the subjects/objects of the properties).

Thanks to these results the algorithm can take all the information needed.

4.2 Results of the algorithm

The parameters setting are the same used in the original algorithm by Minh:

- There are 200 generations.
- Each population generated has maximum 5000 patterns.
- The minimum fitness value acceptable is 0.

It would have been better to increase the number of generations to have a large number of rules discovered. However, it has not always been possible to do so: the dataset was very large and due to the limits of my machine, already with 200 generations it took many hours to do the tests.

Moreover, as we said, there are some limits to the query about the individuals of the classes and the properties, in particular, we can save only up to 500 individuals per class/property.

The generated rules are saved in a file called $COVID-19$ and the rules are entered in descending order of fitness value.
After having finished implementing the queries avoiding problems and errors, we did many tests to see if it was possible to generate interesting rules, as close as possible to the context regarding the COVID-19. So, we tested the algorithm on different graphs.
An example of a generated rule could be this one:

http://schema.org/ScholarlyArticle(?c) <=
http://purl.org/ontology/bibo/AcademicArticle(?c),

where **SholarlyArticle** and **AcademicArticle** are classes belonging to the graph related to the articles, **?c** is the variable which represents the possible individual of these classes and <= is the symbol which represents the implication.
The meaning of this rule is:
If any individual c **is an Academic Article, then it is also a Scholarly Article**.
The rule that states the opposite was also found and it says:
If any individual c **is a Scholarly Article, then it is also an Academic Article**.
These are interesting rules, even if not related to the COVID-19, as they relate two different ontologies: in this case, the algorithm has learned that the two classes, even if they are part of two different ontologies, are equivalent.
Therefore, thanks to these two rules, a new axiom of equivalence has been discovered that aligns two ontologies.
Another possible rule could be like this one:

http://purl.org/ontology/bibo/AcademicArticle(?c) <=
http://purl.org/vocab/frbr/core#partOf(?z1, ?c) &
http://purl.org/spar/fabio/Abstract(?z1).

In this case **Abstract** is a class, **partOf** is a property, **(?z1, ?c)** are the variables

(any individual) of the property and **?c** is also the variable (any individual) of the class, the operator & means that the two condition must occur together.
In this case the meaning of this rule is:
If any individual $z1$ is an Abstract AND $z1$ is part of c, then c is an Academic Article.

There could be another type of rules, which contain, for example, only properties or with more than 2 atoms in the body (up to 10).

4.2.1 Check consistecy program

Once the rules are generated, however, it may be that they are not consistent with the knowledge base. Having entered the limit of 500 individuals for each class to avoid memory problems, the rules generated are not always consistent.
But even if we could rely on all the individuals present in the RDF base, we could hardly be certain that a rule is correct or wrong, because an RDF base may not be complete (of the triples that should be there are not there) or contain noise (contain triples that should not be there, because they are incorrect).
The consistency check in the original algorithm was managed using a reasoner who went to check the truthfulness of the rules in the knowledge base (see Section 2.2.2).
In our case it is not possible to use the reasoner, or in any case, it would become a very long work.
The solution founded was to create a program that read the rules from the file generated with the Genetic Algorithm and, for each generated rule, went to query the dataset with a new query, depending on the form of the rule.

Let's take as an example with one of the rules mentioned above.
To understand if the rule ***http://schema.org/ScholarlyArticle(?c)*** <= ***http://purl.org/ontology/bibo/AcademicArticle(?c)*** is consistent with the dataset we must query it and check if exists some individual who is an Academic Article, but **not** a Scholarly Article. So, we want to find in the dataset some individual which confirms the contrary of the rule, if any.
In the rule the variable is only one, so the query we have to do takes in consideration only one variable and two Concept Atoms. So, we can make a query like this:

SELECT DISTINCT ?c
WHERE {
?c a <http://purl.org/ontology/bibo/AcademicArticle>.
FILTER (NOT EXISTS ?c a <http://schema.org/ScholarlyArticle>)
}
The NOT EXISTS clause eliminates from the results all those that satisfy the condition of the clause.
Thus, the results of this query are all the individuals which are Academic Article but not Scholarly Article.
In this case there is no individual returned by the query, which means that every Academic Article is also a Scholarly Article and the rule is true.

To understand if the rule **http://purl.org/spar/fabio/ResearchPaper(?c) <= http://purl.org/dc/terms/identifier(?z1, ?c) & http://purl.org/spar/fabio/ResearchPaper(?z1)** is consistent we take in consideration two variables, one Concept Atom and one Role Atom. So, the query made is:
SELECT DISTINCT ?c
WHERE {
?z1 <http://purl.org/dc/terms/identifier> ?c.
?z1 a <http://purl.org/spar/fabio/ResearchPaper>
FILTER (NOT EXISTS ?c a <http://purl.org/spar/fabio/ResearchPaper>
)
}
The result of the query is composed by 50101 different individuals.
This means that there are in the dataset 50101 individuals which identified Research Papers, but there are not Research Papers themselves.
The number of individuals who don't satisfy the rule are too big to consider the rule true, then it is discarded by the program.

In general, queries change based on whether atoms are classes or properties and care must be taken to correctly save variables.
If the query result is empty or with few individuals which don't satisfy the rule, then the rule is accepted and placed in the rules' file. Otherwise, if there are many results,

the rule is removed and not considered valid, since it is not consistent with the dataset.

4.2.2 Tests and results

To do most of the tests, we used the queries presented at the beginning of Section 3.2.1. Only in the final test we used the latest version of the queries made in the last period of the internship, presented at the and of Section 3.2.1. We will see its results in Section 4.2.4.

To make the tests the most specific graphs for the COVID-19 were chosen, and we produced many results, even if most of the tests did not lead to the discovery of interesting rules in the context of the Corona Virus.
Below are the tests carried out:

1. Run the algorithm with all the graphs belonging to the CORD-19 Named Entities Knowledge Graph:

   - http://ns.inria.fr/covid19/graph/articles
   - http://ns.inria.fr/covid19/graph/metadata
   - http://ns.inria.fr/covid19/graph/entityfishing
   - http://ns.inria.fr/covid19/graph/entityfishing/body
   - http://ns.inria.fr/covid19/graph/dbpedia-spotlight
   - http://ns.inria.fr/covid19/graph/bioportal-annotator
   - http://ns.inria.fr/covid19/graph/acta
   - https://www.kaggle.com/group16/covid19-literature-knowledge-graph
   - https://github.com/fhircat/CORD-19-on-FHIR/pubtator
   - https://github.com/fhircat/CORD-19-on-FHIR/cord-19_comm_use_subset_RDF
   - https://github.com/fhircat/CORD-19-on-FHIR/cord-19_custom_license_RDF
   - https://github.com/fhircat/CORD-19-on-FHIR/cord-19_noncomm_use_subset
   - http://www.openkg.cn/dataset/covid-19-research

   With these graphs we tried these tests:

- We tried with the only limit of 500 individuals per class/property: This test produced 4416 classes and 667 properties. Finally, the rules generated are only 5 and not very interested.
- Run the algorithm with the limit of at most 100 classes for each graph and 400 individuals per class/property.
  We did it to try to have more interesting results with these few classes / properties since the algorithm was originally designed to work with few classes and few properties.
  The result produced 354 classes and 667 properties, with only 10 rules generated [Fig. 4.1]
  Then, the rules have been tested thanks to the algorithm mentioned above (Section 4.2.1) and 5 consistent rules have been found. [Fig. 4.2]

```
1. http://schema.org/ScholarlyArticle(?c) <= http://purl.org/ontology/bibo/AcademicArticle(?c)

2. http://purl.org/ontology/bibo/AcademicArticle(?c) <= http://schema.org/ScholarlyArticle(?c)

3. http://purl.org/dc/terms/identifier(?x, ?y) <= http://purl.org/dc/terms/identifier(?z1, ?y) &
        http://purl.org/dc/terms/identifier(?z1, ?x) & http://purl.org/dc/terms/identifier(?z2, ?z1) &
        http://purl.org/dc/terms/identifier(?z1, ?z2)

4. http://www.w3.org/ns/oa#TextPositionSelector(?c) <= http://www.w3.org/ns/oa#TextQuoteSelector(?c)

5. http://purl.org/dc/elements/1.1/publisher(?x, ?y) <= http://schema.org/publisher(?x, ?y)

6. http://schema.org/publisher(?x, ?y) <= http://purl.org/dc/elements/1.1/publisher(?x, ?y)

7. http://www.w3.org/ns/oa#TextQuoteSelector(?c) <= http://www.w3.org/ns/oa#TextPositionSelector(?c)

8. http://purl.org/dc/terms/identifier(?x, ?y) <= http://purl.org/dc/terms/identifier(?y, ?x)

9. http://purl.org/dc/terms/identifier(?x, ?y) <= http://purl.org/dc/terms/identifier(?y, ?z1)  &
        http://purl.org/dc/terms/identifier(?y, ?x) & http://purl.org/dc/terms/identifier(?z1, ?x)

10. http://www.openkg.cn/COVID-19/research/property/P2(?x, ?y) <= http://www.openkg.cn/COVID-19/research/property/P13(?x, ?y)
```

Figure 4.1: Rules generated with the graph related to the Covid and limit 100 to the class of each graph.

```
http://schema.org/ScholarlyArticle(?c) <= http://purl.org/ontology/bibo/AcademicArticle(?c)

http://purl.org/ontology/bibo/AcademicArticle(?c) <= http://schema.org/ScholarlyArticle(?c)

http://purl.org/dc/elements/1.1/publisher(?x, ?y) <= http://schema.org/publisher(?x, ?y)

http://schema.org/publisher(?x, ?y) <= http://purl.org/dc/elements/1.1/publisher(?x, ?y)

http://www.w3.org/ns/oa#TextQuoteSelector(?c) <= http://www.w3.org/ns/oa#TextPositionSelector(?c)

Consistent Rules: 5
```

Figure 4.2: Consistent rules found starting from the 10 found with the graph related to the Covid and limit 100 to the class of each graph.

- Run the algorithm with 500 individuals per class/property. For each graph there is a limit of at most 30 classes and 50 properties, since there are hundreds of classes and properties in "github" graphs and it takes more time to compute.

  In particular, the limit on classes is placed only on the following graphs:

  - https://github.com/fhircat/CORD-19-on-FHIR/cord-19_comm_use_subset_RDF,
  - https://github.com/fhircat/CORD-19-on-FHIR/cord-19_custom_license_RDF,
  - https://github.com/fhircat/CORD-19-on-FHIR/cord-19_noncomm_use_subset.

  And the limit on the properties is placed only in the graph:

  - https://github.com/fhircat/CORD-19-on-FHIR/pubtator.

  The result produced 22 rules and 7 of these are consistent [Fig. 4.3].

```
http://www.w3.org/ns/oa#Annotation(?c) <= http://www.w3.org/ns/prov#Entity(?c)

http://schema.org/ScholarlyArticle(?c) <= http://purl.org/ontology/bibo/AcademicArticle(?c)

http://www.w3.org/ns/oa#Annotation(?c) <= http://www.w3.org/ns/prov#Entity(?c)

http://schema.org/ScholarlyArticle(?c) <= http://purl.org/ontology/bibo/AcademicArticle(?c)

http://www.w3.org/ns/oa#TextQuoteSelector(?c) <= http://www.w3.org/ns/oa#TextPositionSelector(?c)

http://purl.org/ontology/bibo/AcademicArticle(?c) <= http://schema.org/ScholarlyArticle(?c)

http://purl.org/dc/terms/identifier(?x, ?y) <= http://purl.org/dc/terms/identifier(?z1, ?y) &
       http://purl.org/dc/terms/identifier(?x, ?z1)

http://schema.org/Dataset(?c) <= http://www.w3.org/ns/dcat#Dataset(?c)

http://rdfs.org/ns/void#Dataset(?c) <= http://www.w3.org/ns/dcat#Dataset(?c)

Consistent Rules: 7
```

Figure 4.3: Consistent rules found starting from the 22 found with at most 30 classes and 50 properties.

- Run the algorithm with 400 individuals per class/property, without using the "github" graphs.
  The result produced 29 rules and 12 of these are consistent [Fig. 4.4].
  Also in this case the consistent rules are more or less the same of the other results.

```
http://schema.org/datePublished(?x, ?y) <= http://www.w3.org/ns/prov#wasGeneratedAtTime(?x, ?y)

http://schema.org/ScholarlyArticle(?c) <= http://purl.org/ontology/bibo/AcademicArticle(?c)

http://www.w3.org/ns/oa#Annotation(?c) <= http://www.w3.org/ns/prov#Entity(?c)

http://purl.org/ontology/bibo/AcademicArticle(?c) <= http://schema.org/ScholarlyArticle(?c)

http://www.w3.org/ns/oa#TextQuoteSelector(?c) <= http://www.w3.org/ns/oa#TextPositionSelector(?c)

http://schema.org/publisher(?x, ?y) <= http://purl.org/dc/elements/1.1/publisher(?x, ?y)

http://purl.org/dc/terms/licence(?x, ?y) <= http://schema.org/licence(?x, ?y)

http://purl.org/dc/terms/identifier(?x, ?y) <= http://purl.org/dc/terms/identifier(?z1, ?y) &
        http://purl.org/dc/terms/identifier(?x, ?z1)

http://schema.org/Dataset(?c) <= http://www.w3.org/ns/dcat#Dataset(?c)

http://rdfs.org/ns/void#Dataset(?c) <= http://www.w3.org/ns/dcat#Dataset(?c)

http://purl.org/dc/terms/identifier(?x, ?y) <= http://semanticscholar.org/cv-research/pubtator(?x, ?y)

http://semanticscholar.org/cv-research/pubtator(?x, ?y) <= http://semanticscholar.org/cv-research/pubtator(?z1, ?y) &
        http://purl.org/dc/terms/identifier(?x, ?z1)

Consistent Rules: 12
```

Figure 4.4: Consistent rules found starting from the 29 found without using the "github" graphs.

2. In this tests, the query about the classes was modified in order to take only the classes defined by the RDF schema.
   So, to search this type of classes, in the WHERE clause, instead of using *?ind a ?class*, we used *?class a rdfs:Class*.

   Using all the graphs from the CORD-19 Named Entities Knowledge Graph (from github, only the pubtator graph), we found 15 rules and only 3 of them are consistent, so it is not an interesting result.

   Using only the graph from the Inria website and one from Openkg the gener-

ated rules are 44 and 16 of them were consistent [Fig 4.5].
These graphs are:

- http://ns.inria.fr/covid19/graph/metadata,
- http://ns.inria.fr/covid19/graph/entityfishing,
- http://ns.inria.fr/covid19/graph/entityfishing/body,
- http://ns.inria.fr/covid19/graph/dbpedia-spotlight,
- http://ns.inria.fr/covid19/graph/bioportal-annotator,
- http://ns.inria.fr/covid19/graph/articles,
- http://www.openkg.cn/dataset/covid-19-research.

```
http://schema.org/licence(?x, ?y) <= http://purl.org/dc/terms/licence(?x, ?y)
http://purl.org/dc/elements/1.1/publisher(?x, ?y) <= http://schema.org/publisher(?x, ?y)
http://purl.org/spar/amo/proves(?x, ?y) <= http://rdfs.org/sioc/argument#supports(?x, ?y)
http://schema.org/publisher(?x, ?y) <= http://purl.org/dc/elements/1.1/publisher(?x, ?y)
http://purl.org/dc/terms/licence(?x, ?y) <= http://schema.org/licence(?x, ?y)
http://www.w3.org/ns/prov#wasGeneratedAtTime(?x, ?y) <= http://schema.org/datePublished(?x, ?y)
http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?x, ?y)
http://schema.org/datePublished(?x, ?y) <= http://www.w3.org/ns/prov#wasGeneratedAtTime(?x, ?y)
http://www.w3.org/ns/prov#wasQuotedFrom(?x, ?y) <= http://www.w3.org/ns/prov#wasQuotedFrom(?z1, ?y) &
        http://rdfs.org/sioc/argument#supports(?z1, ?x) &
http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?x, ?y) &
        http://rdfs.org/sioc/argument#supports(?y, ?x)
http://purl.org/spar/amo/proves(?x, ?y) <= http://purl.org/spar/amo/proves(?y, ?x) &
        http://rdfs.org/sioc/argument#supports(?x, ?y)
http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?y, ?x) &
        http://rdfs.org/sioc/argument#supports(?y, ?x) & http://purl.org/spar/amo/proves(?x, ?y)
http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?y, ?x) &
        http://purl.org/spar/amo/proves(?x, ?y)
http://purl.org/spar/amo/proves(?x, ?y) <= http://rdfs.org/sioc/argument#supports(?x, ?y) &
        http://rdfs.org/sioc/argument#supports(?y, ?x)
http://purl.org/vocab/frbr/core#partOf(?x, ?y) <= http://ns.inria.fr/covid19/property/hasTitle(?y, ?x)
http://purl.org/vocab/frbr/core#partOf(?x, ?y) <= http://ns.inria.fr/covid19/property/hasBody(?y, ?x)

Consistent Rules: 16
```

Figure 4.5: Consistent rules found starting from the 44 found using only the graph from the Inria website and one from Openkg.

In this case in the generated rules there are many rules regarding specific property or classes from the research graph, but all these rules are deleted with the consistency check.

3. In the last experiment we tried to use even less graphs in two different way, to see if the type of rule found changed with fewer classes and properties available, in the first case and with many concepts in the second case.

In the first test we tried with only these two graphs, considering them the most specific for the topic:

- http://ns.inria.fr/covid19/graph/articles,
- http://www.openkg.cn/dataset/covid-19-research.

In fact the number of classes and properties has decreased: 26 classes and 71 properties.
We obtained 34 rules, but only 10 of them are consistent, but very similar to the other results.

In the second test we tried with the same graphs, plus other two:

- https://www.kaggle.com/group16/covid19-literature-knowledge-graph,
- https://github.com/fhircat/CORD-19-on-FHIR/pubtator.

We found many classes, 2479, most concepts taken from DBpedia, and 1565 properties, also taken from DBpedia (the majority).
This is the only test which produced the maximum number of rules possible, 5000, given that the fitness value never reached zero and with many atoms in the body of the rules, reaching even 10 atoms per rule.
From these 5000 rules almost all of them passed the consistency check, reaching 4805 rules in one test and 4863 in another.
We put here only a few rules taken from those found [Fig. 4.6].

16. http://dbpedia.org/class/yago/WikicatLawSchoolsInDelhi(?c) <= http://dbpedia.org/class/yago/WikicatUniversitiesAndCollegesInDelhi(?c) &
    http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c) & http://dbpedia.org/class/yago/Group100031264(?c)
17. http://dbpedia.org/class/yago/WikicatLivingPeople(?c) <= http://dbpedia.org/class/yago/Person100007846(?c) &
    http://www.wikidata.org/entity/Q3918(?c)
18. http://www.wikidata.org/entity/Q24229398(?c) <= http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Agent(?c)
19. http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c) <= http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Agent(?c)
20. http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Agent(?c) <= http://www.wikidata.org/entity/Q24229398(?c)
21. http://dbpedia.org/class/yago/Object100002684(?c) <= http://dbpedia.org/class/yago/PhysicalEntity100001930(?c)
22. http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c) <= http://www.wikidata.org/entity/Q24229398(?c)
23. http://dbpedia.org/class/yago/Abstraction100002137(?c) <= http://dbpedia.org/class/yago/Group100031264(?c)
24. http://dbpedia.org/class/yago/WikicatLivingPeople(?c) <= http://dbpedia.org/class/yago/Person100007846(?c) &
    http://dbpedia.org/class/yago/PhysicalEntity100001930(?c)
25. http://schema.org/Organization(?c) <= http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Agent(?c)
26. http://dbpedia.org/class/yago/SocialGroup107950920(?c) <= http://dbpedia.org/class/yago/Institution108053576(?c)
27. http://dbpedia.org/class/yago/WikicatOrganizationsEstablishedIn1838(?c) <= http://dbpedia.org/class/yago/Abstraction100002137(?c) &
    http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c) & http://dbpedia.org/class/yago/WikicatMethodistUniversitiesAndC
28. http://dbpedia.org/class/yago/Institution108053576(?c) <= http://dbpedia.org/class/yago/SocialGroup107950920(?c) & 2.0
29. http://dbpedia.org/class/yago/WikicatOrganizationsBasedInBinghamton,NewYork(?c) <= http://dbpedia.org/class/yago/PhysicalEntity100001930(?c
    http://dbpedia.org/class/yago/WikicatUniversitiesAndCollegesInNewYork(?c) & http://www.wikidata.org/entity/Q3918(?c) &
    http://www.wikidata.org/entity/Q24229398(?c) & http://dbpedia.org/class/yago/YagoPermanentlyLocatedEntity(?c)
30. http://schema.org/Organization(?c) <= http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#Agent(?c) &
    http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c)
31. http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c) <= http://schema.org/Organization(?c) &
    http://www.wikidata.org/entity/Q24229398(?c)
32. http://dbpedia.org/class/yago/WikicatOrganizationsBasedInTaiwan(?c) <= http://dbpedia.org/class/yago/SocialGroup107950920(?c) &
    http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c) & http://www.wikidata.org/entity/Q327333(?c) &
    http://www.w3.org/2002/07/owl#Thing(?c) & http://dbpedia.org/class/yago/School108276720(?c)
33. http://dbpedia.org/class/yago/LawEnforcementAgency108348815(?c) <= http://dbpedia.org/class/yago/Abstraction100002137(?c) &
    http://dbpedia.org/class/yago/PhysicalEntity100001930(?c) & http://schema.org/Organization(?c) &
    http://dbpedia.org/class/yago/SocialGroup107950920(?c) & http://schema.org/EducationalOrganization(?c) &
    http://dbpedia.org/class/yago/WikicatParksInIndianapolis,Indiana(?c) &
    http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c)
34. http://dbpedia.org/class/yago/WikicatComputerScienceInstitutes(?c) <= http://dbpedia.org/class/yago/WikicatUniversitiesInCatalonia(?c) &
    http://dbpedia.org/class/yago/Institution108053576(?c) & http://umbel.org/umbel/rc/EducationalOrganization(?c) &
    http://www.wikidata.org/entity/Q3918(?c)
35. http://dbpedia.org/class/yago/WikicatParksInIndianapolis,Indiana(?c) <= http://dbpedia.org/class/yago/Abstraction100002137(?c) &
    http://dbpedia.org/class/yago/Unit108189659(?c) & http://dbpedia.org/class/yago/PhysicalEntity100001930(?c) &
    http://umbel.org/umbel/rc/EducationalOrganization(?c) & http://schema.org/Organization(?c) &
    http://dbpedia.org/class/yago/Object100002684(?c) & http://dbpedia.org/class/yago/YagoLegalActorGeo(?c)
36. http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#SocialPerson(?c) <= http://schema.org/Organization(?c)
37. http://dbpedia.org/class/yago/SocialGroup107950920(?c) <= http://dbpedia.org/class/yago/Organization108008335(?c)

Figure 4.6: Consistent rules found starting from the 5000 found with articles, research, literature and pubtator graphs.

Many other tests have been done but with more or less the same results as those just presented.

4.2.3 Final rules

After doing all the tests, it was noticed that most of the rules found among the consistent ones were repeated for each test.
It was therefore possible to define a set of total rules, excluding those generated by the last test performed.
The rules obtained are therefore 38 and are as follows:

1. http://schema.org/ScholarlyArticle(?c) <=
   http://purl.org/ontology/bibo/AcademicArticle(?c)

2. http://purl.org/ontology/bibo/AcademicArticle(?c) <=
   http://schema.org/ScholarlyArticle(?c)

3. http://schema.org/ScholarlyArticle(?c) <= http://purl.org/vocab/frbr/core#partOf(?z1, ?c) & http://ns.inria.fr/covid19/property/hasBody(?c, ?z1)
4. http://purl.org/ontology/bibo/AcademicArticle(?c) <= http://ns.inria.fr/covid19/property/hasBody(?c, ?z1) & http://purl.org/vocab/frbr/core#partOf(?z1, ?c)
5. http://purl.org/ontology/bibo/AcademicArticle(?c) <= http://purl.org/vocab/frbr/core#partOf(?z1, ?c) & http://purl.org/spar/fabio/Abstract(?z1)
6. http://purl.org/ontology/bibo/AcademicArticle(?c) <= http://purl.org/dc/terms/abstract(?c, ?z1) & http://purl.org/spar/fabio/Abstract(?z1)
7. http://www.w3.org/ns/oa#Annotation(?c) <= http://www.w3.org/ns/prov#Entity(?c)
8. http://www.w3.org/ns/oa#TextQuoteSelector(?c) <= http://www.w3.org/ns/oa#TextPositionSelector(?c)
9. http://schema.org/licence(?x, ?y) <= http://purl.org/dc/terms/licence(?x, ?y)
10. http://purl.org/dc/terms/identifier(?x, ?y) <= http://purl.org/dc/terms/identifier(?z1, ?y) & http://purl.org/dc/terms/identifier(?x, ?z1)
11. http://purl.org/dc/terms/identifier(?x, ?y) <= http://semanticscholar.org/cv-research/pubtator(?x, ?y)
12. http://purl.org/dc/terms/licence(?x, ?y) <= http://schema.org/licence(?x, ?y)
13. http://www.w3.org/ns/dcat#Dataset(?c) <= http://rdfs.org/ns/void#Dataset(?c) & http://schema.org/Dataset(?c)
14. http://www.w3.org/ns/prov#wasGeneratedAtTime(?x, ?y) <= http://schema.org/datePublished(?x, ?y)
15. http://www.w3.org/ns/prov#wasQuotedFrom(?x, ?y) <= http://www.w3.org/ns/prov#wasQuotedFrom(?z1, ?y) & http://rdfs.org/sioc/argument#supports(?z1, ?x)
16. http://schema.org/datePublished(?x, ?y) <= http://www.w3.org/ns/prov#wasGeneratedAtTime(?x, ?y)

17. http://schema.org/Dataset(?c) <= http://www.w3.org/ns/dcat#Dataset(?c)

18. http://schema.org/Dataset(?c) <=
    http://rdfs.org/ns/void#Dataset(?c) & http://www.w3.org/ns/dcat#Dataset(?c)

19. http://rdfs.org/ns/void#Dataset(?c) <= http://www.w3.org/ns/dcat#Dataset(?c)
    & http://schema.org/Dataset(?c)

20. http://rdfs.org/ns/void#Dataset(?c) <= http://www.w3.org/ns/dcat#Dataset(?c)

21. http://semanticscholar.org/cv-research/pubtator(?x, ?y) <= http://semanticscholar.org/cv-research/pubtator(?z1, ?y) & http://purl.org/dc/terms/identifier(?x, ?z1)

22. http://purl.org/spar/fabio/hasPubMedId(?x, ?y) <= http://purl.org/ontology/bibo/pmid(?x, ?y)

23. http://purl.org/ontology/bibo/pmid(?x, ?y) <=
    http://purl.org/spar/fabio/hasPubMedId(?x, ?y)

24. http://purl.org/spar/fabio/hasPubMedId(?x, ?y) <=
    http://purl.org/spar/fabio/hasPubMedId(?x, ?z1) & http://purl.org/ontology/bibo/pmid(?x, ?y) & http://purl.org/ontology/bibo/pmid(?x, ?z1)

25. http://purl.org/dc/terms/abstract(?x, ?y) <= http://purl.org/vocab/frbr/core#partOf(?y, ?x) & http://purl.org/spar/fabio/Abstract(?y)

26. http://purl.org/vocab/frbr/core#partOf(?x, ?y) <= http://schema.org/ScholarlyArticle(?y)
    & http://ns.inria.fr/covid19/property/hasTitle(?y, ?x)

27. http://purl.org/vocab/frbr/core#partOf(?x, ?y) <=
    http://ns.inria.fr/covid19/property/hasTitle(?y, ?x)

28. http://purl.org/vocab/frbr/core#partOf(?x, ?y) <=
    http://ns.inria.fr/covid19/property/hasBody(?y, ?x)

29. http://purl.org/vocab/frbr/core#partOf(?x, ?y) <=
    http://purl.org/vocab/frbr/core#partOf(?z1, ?y) &
    http://ns.inria.fr/covid19/property/hasBody(?y, ?x) &
    http://ns.inria.fr/covid19/property/hasBody(?y, ?z1)

30. http://purl.org/dc/elements/1.1/publisher(?x, ?y) <= http://schema.org/publisher(?x, ?y)
31. http://schema.org/publisher(?x, ?y) <= http://purl.org/dc/elements/1.1/publisher(?x, ?y)
32. http://purl.org/spar/amo/proves(?x, ?y) <= http://rdfs.org/sioc/argument#supports(?x, ?y)
33. http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?x, ?y)
34. http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?x, ?y) & http://rdfs.org/sioc/argument#supports(?y, ?x)
35. http://purl.org/spar/amo/proves(?x, ?y) <= http://purl.org/spar/amo/proves(?y, ?x) & http://rdfs.org/sioc/argument#supports(?x, ?y)
36. http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?y, ?x) & http://rdfs.org/sioc/argument#supports(?y, ?x) & http://purl.org/spar/amo/proves(?x, ?y)
37. http://rdfs.org/sioc/argument#supports(?x, ?y) <= http://purl.org/spar/amo/proves(?y, ?x) & http://purl.org/spar/amo/proves(?x, ?y)
38. http://purl.org/spar/amo/proves(?x, ?y) <= http://rdfs.org/sioc/argument#supports(?x, ?y) & http://rdfs.org/sioc/argument#supports(?y, ?x)

As we can see, the rules discovered seem to be highly plausible but not relevant. Anyway, the 38 rules discovered could complete the dataset Covid-On-The-Web, even if they are fairly obvious and they don't seem to add much to the knowledge base. They could be useful to improve data extraction or find inconsistencies between them.
However, among the 4863 rules discovered with the DBpedia graph, it seems that some of that are potentially interesting, but not directly related to the COVID-19.
It might be useful to use them to enrich bibliographic metadata.

4.2.4 Rules discovered with the last version of queries

As explained in Section 3.2.1 in the last period of the internship we tried a new approach with the queries.
This is made because, as we can see in Section 4.2.3 above, the rules generated with the other approach are potentially interesting to enrich the metadata or to improve the data extraction, but they are not relevant for the topic of the dataset. So, we tried to discover more specific rules, trying to define different classes.
Of course, the consistency check has also been adapted to these rules (using the wdt: P31 property seen in Section 3.2.1).
The use of these queries can be improved, having only been introduced at the end of the internship, but the results obtained seem to be better.
The rules found with this approach are 2000 and after the consistency check, 1549 rules have remained.

We show below only some of these rules:

1. http://www.wikidata.org/entity/Q18975918(?c) <=
   http://www.wikidata.org/entity/Q18603649(?c) &
   http://www.wikidata.org/entity/Q9332(?c) &
   http://www.wikidata.org/entity/Q21076236(?c) &
   http://www.wikidata.org/entity/Q182672(?c).

   The translation of this rule is:
   C is an Orthomyxoviridae infectious disease (Q18975918) if it is a causal factor (Q18603649), a behavior (Q9332), a human activity (Q21076236) and a zoonosis (Q182672).

2. **IF c is an artificial geographic entity (Q27096235), a system (Q58778), a local government (Q6501447) and statistical territorial entity (Q15042037) THEN c is a human settlement (Q486972)**.

3. **IF c is a perceptible object (Q337060), contained a region of space-time (Q58415929) THEN it is a concrete object (Q4406616)**.

4. **IF c is an agents that affect mitosis of cells (Q50430330) THEN it is a**

plant hormone (Q192949).

5. **IF c is a factor (Q21076236) THEN it is a causal factor (Q18603649)** (AND VICEVERSA).

6. **IF c is a chemical component (Q20026787) and it is a portion of something (Q15989253) THEN it is a physical substance (Q28732711)**.

There are many other rules, some relevant, others less so. The peculiarity of these rules is that they refer directly to concepts associated with articles regarding COVID-19.
By filtering the types of concepts in the queries, we could get even more specific rules.

Conclusion and future perspectives

In Section 1.4 the main goals of my internship are listed. Here, we analyze the progress made.

1. In the first phase, I did an accurate study of the contest and of the state of the art. This was very useful for the work of the following months.

2. Thanks to the first phase, it was possible to refactor the codes, in particular, the works of Duc Minh Tran [3] [4] has been analyzed and modified by adding comments, the JavaDoc and by improving parts of the code making them more understandable.
   In this part of the internship, I understood better the work done by Minh and I also improved my programming skills, having as purpose the quality of the code and not just the usability (which had already been verified by Minh).
   This could be very useful for my future career.

3. In the last phase, the aim was to adapt the refactored code for its use on RDF graphs: the purpose of the new algorithm is to create SWRL rules through genetic algorithms, starting from RDF graphs, so through endpoints, without having to load *.owl files (as was done by Minh).
   To do this, a connection was created with the SPARQL endpoint of the considered graph and through queries in SPARQL, it was possible to extract the classes and properties with their associated elements from the named graphs.
   The connection with the endpoint and the execution of the queries was done directly in the code (see Algorithm 1, Algorithm 2) using the Apache Jena API (see Section 3.1.1). So, it could be possible to use this code for any RDF graph. In our case, the Covid-On-The-Web [1] dataset was used. The rules obtained, even if

correct, were not of particular interest for the topic covered by the dataset.
In any case, it is possible to integrate these rules into the metadata, to enrich knowledge and make data extraction easier.
To get better rules we tried to change the queries (as explained at the end of Section 3.2.1), trying to get more specific classes. Since the Covid-On-The-Web dataset is really complex, we need to increase the complexity of the queries, in order to have better results. In fact, the rules obtained by the latter queries seem to be more interesting (Section 4.2.4). The new approach could be the starting point for future researches.
This phase suffered some slowdown due to problems with the server on which the endpoint is created.

The algorithm produced some plausible rules and we tried it with some other RDF datasets: it works quite well. This means that the method created is valid, but it definitively needs some improvements for the dataset in analysis.

The main result we have is that now it is possible to apply the algorithm created by Duc Minh Tran to any RDF dataset and to consider only entities of the desired namespaces. It is a good result, since the use of this algorithm directly on RDF graphs makes the generated rules constantly updated, without the need to download the updated dataset or the ontology every time.

5.1 Repository DiscoverSWRLRulesFromKB

The algorithms are published as Open Source in this repository `https://github.com/ValeBellu/DiscoverSWRLRulesFromKB` with the license GNU GPLv3, which lets people do almost anything they want with the project, except distributing closed source versions.
It contains:

- SWRLRulesFromOntology: this is the algorithm by Minh (explained in Chapter 2), refactored by me, used to generate SWRL rules from ontology loaded.
  - CheckConsistencyRules: this is the algorithm used to check if the rules generated from the algorithm above are consistent with the ontology. It uses

10 threads to test 10 rules at the same time. This algorithm was refactored during the internship.

- SWRLRulesFromRDF: this is the algorithm modified (explained in Chapter 3), thanks to which it is possible to query directly the SPARQL endpoint of any RDF graphs.
  - CheckConsistencyRulesRDF: this is the new algorithm created (explained in Section 4.2.1) to check if the rules generated from the SWRLRulesFromRDF are consistent with the RDF graph. It is integrated into the algorithm SWRLRulesFromRDF.

5.2 Future perspectives

During the last phases of the internship various ideas and developments were proposed, but it was not possible to complete them due to lack of time. It might be interesting to develop such ideas in the future.

- We should investigate why the rules generated are not inherent to COVID-19.
  It could be that the dataset is constructed in a way that does not facilitate the creation of such rules with that algorithm.
  In the future, it could interested to continue the work with the queries described at the end of Section 3.2.1, trying to make more and more specific queries for this dataset.

- My work has been to improve the loading of the dataset in the Minh's algorithm, which has a focus on specific types of axioms, such as subsumption and disjunction axioms and SWRL rules.
  In the future, it might be useful to extend the software portfolio to include support for additional types of axioms, like equivalence axioms, property domain/range axioms, keys axioms.

- Another possible improvement for being able to work with big data would be to use models like MapReduce to leverage frameworks like Hadoop or Spark, so that we can perform big data analytics.

Bibliography

[1] Wimmics Team. *Covid-on-the-Web Dataset*. Available at `https://github.com/Wimmics/CovidOnTheWeb`.

[2] Claudia d'Amato, Steffen Staab, Andrea G. B. Tettamanzi, Tran Duc Minh, and Fabien Gandon. "Ontology Enrichment by Discovering Multi-Relational Association Rules from Ontological Knowledge Bases". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. SAC '16. Pisa, Italy: Association for Computing Machinery, 2016, pp. 333–338. ISBN: 9781450337397. DOI: `10.1145/2851613.2851842`. URL: `https://doi.org/10.1145/2851613.2851842`.

[3] Claudia d'Amato, Andrea G. B. Tettamanzi, and Tran Duc Minh. "Evolutionary Discovery of Multi-relational Association Rules from Ontological Knowledge Bases". In: *Knowledge Engineering and Knowledge Management*. Ed. by Eva Blomqvist, Paolo Ciancarini, Francesco Poggi, and Fabio Vitali. Cham: Springer International Publishing, 2016, pp. 113–128. ISBN: 978-3-319-49004-5.

[4] Minh Duc Tran, Claudia d'Amato, Binh Thanh Nguyen, and Andrea G. B. Tettamanzi. "Comparing Rule Evaluation Metrics for the Evolutionary Discovery of Multi-relational Association Rules in the Semantic Web". In: *Genetic Programming*. Ed. by Mauro Castelli, Lukas Sekanina, Mengjie Zhang, Stefano Cagnoni, and Pablo García-Sánchez. Cham: Springer International Publishing, 2018, pp. 289–305. ISBN: 978-3-319-77553-1.

[5] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. "DL-FOIL Concept Learning in Description Logics". In: *Inductive Logic Programming*. Ed. by Filip Železný and Nada Lavrač. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 107–121. ISBN: 978-3-540-85928-4.

[6] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. "AMIE: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases". In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 413–422. ISBN: 9781450320351. DOI: 10.1145/2488388.2488425. URL: https://doi.org/10.1145/2488388.2488425.

[7] Peter Haase and Johanna Völker. "Ontology Learning and Reasoning — Dealing with Uncertainty and Inconsistency". In: *Uncertainty Reasoning for the Semantic Web I*. Ed. by Paulo Cesar G. da Costa, Claudia d'Amato, Nicola Fanizzi, Kathryn B. Laskey, Kenneth J. Laskey, Thomas Lukasiewicz, Matthias Nickles, and Michael Pool. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 366–384. ISBN: 978-3-540-89765-1.

[8] I3S. *Presentation of the I3S laboratory*. Available at https://www.i3s.unice.fr/en/node/311.

[9] Jens Lehmann. "DL-Learner: Learning Concepts in Description Logics". In: *J. Mach. Learn. Res.* 10 (Dec. 2009), pp. 2639–2642. ISSN: 1532-4435.

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

[11] Thu Huong Nguyen and Andrea G. B. Tettamanzi. "Learning Class Disjointness Axioms Using Grammatical Evolution". In: *Genetic Programming*. Ed. by Lukas Sekanina, Ting Hu, Nuno Lourenço, Hendrik Richter, and Pablo García-Sánchez. Cham: Springer International Publishing, 2019, pp. 278–294. ISBN: 978-3-030-16670-0.

[12] Qudamah K. Quboa and Mohamad Saraee. "A State-of-the-Art Survey on Semantic Web Mining". In: *Intelligent Information Management* Vol.05No.01 (2013), p. 8. URL: 10.4236/iim.2013.51002.

[13] *Introduction to the Semantic Web*. Available at http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html.

[14] Gerd Stumme, Andreas Hotho, and Bettina Berendt. "Semantic Web Mining: State of the art and future directions". In: *Journal of Web Semantics* 4.2 (2006). Semantic Grid –The Convergence of Technologies, pp. 124–143. ISSN: 1570-8268. DOI:

https://doi.org/10.1016/j.websem.2006.02.001. URL: http://www.sciencedirect.com/science/article/pii/S1570826806000084.

[15] Andrea G.B. Tettamanzi, Catherine Faron Zucker, and Fabien Gandon. *RDF Mining: Ideas for research*. 2015.

[16] Duc Minh Tran. "Discovering multi-relational association rules from ontological knowledge bases to enrich ontologies". PhD thesis. France, Vietnam, 2018.

[17] Johanna Völker and Mathias Niepert. "Statistical Schema Induction". In: *The Semantic Web: Research and Applications*. Ed. by Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 124–138. ISBN: 978-3-642-21034-1.