



SAPIENZA
UNIVERSITÀ DI ROMA

Ontology-Based Information Extraction

experiences, framework, algorithms and tools

Department of Computer, Control and Management Engineering “Antonio Ruberti”

Dottorato di Ricerca in Ingegneria Informatica – XXXIII Ciclo

Candidate

Federico Maria Scafoglieri

Thesis Advisor

Prof. Domenico Lembo

Co-Advisors

Prof. Maurizio Lenzerini

Prof. Massimo Mecella

June 2021

Thesis defended on 17 September 2021
in front of a Board of Examiners composed by:

Prof. Alessandro Saetti (chairman)

Prof. Emanuele Panizzi

Prof. Marco Maggini

Ontology-Based Information Extraction *experiences, framework, algorithms and tools*
Ph.D. thesis. Sapienza – University of Rome

© 2021 Federico Maria Scafoglieri. All rights reserved

This thesis has been typeset by L^AT_EX and the Sapthesis class.

Author's email: scafoglieri@diag.uniroma1.it

*Fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza.*

*Ye were not form'd to live the life of brutes,
But virtue to pursue and knowledge high.*

*from The Divine Comedy
Hell, Canto XXVI, Verses 119-120
Dante Alighieri*

Acknowledgments

First of all, I would like to express my sincere gratitude to my advisor, Prof. Domenico Lembo, for his constant support and enthusiastic encouragement during my Ph.D. journey. The numerous and lengthy conversations I had with him were a constant source of inspiration and wide perspectives. Without his help, this work would never have been accomplished. My debt to him goes beyond this dissertation. I also wish to thank the “Data Management and Service-Oriented Computing” research group at the “Department of Computer, Control and Management Engineering” of “Sapienza University of Rome”. I feel very lucky and honored to have been part of it. During my Ph.D. career, I have joined the IBM Almaden research center as a visiting scholar. I am grateful to all those who made me feel at home during that time. I am also thankful to my mentors at IBM: Dr. Ronald Fagin, Prof. Phokion Kolaitis, Dr. Yunyao Li, Dr. Lucian Popa. It was truly a privilege for me to work with these outstanding researchers. Special thanks go to my father Michele, my sister Giulia, and especially to my mother Innocenza, to whom I dedicate this thesis.

Federico Maria Scafoglieri

Abstract

A significant portion of the information collected by enterprises and organizations resides in text documents and is thus inherently unstructured. Turning it into a structured form is the aim of Information Extraction (IE). Depending on the approach, the output of an IE process can fill forms, populate relational tables, or even be presented through an ontology. This last approach, known in the literature under the name of Ontology Based Information Extraction (OBIE), is particularly interesting, since ontologies may facilitate the integration with other corporate and external data and enable data management and governance at an abstract, conceptual level. However, despite OBIE has been so far the subject of several investigations, how to exploit the reasoning abilities offered by an ontology to improve the extraction process has not yet been specifically studied. This thesis is intended to be a first step in that direction.

Starting from our *experience* gained from implementing OBIE systems via open-source technologies, and with the intent to address the encountered weaknesses, we propose a formal *framework* for OBIE, called Ontology Based Document Spanning (OBDS). We devise our proposal by revisiting the Ontology Based Data Access (OBDA) paradigm, a sophisticated form of semantic data integration from relational databases, and leveraging the investigation on Document Spanners, a recent formal study of rule-based information extraction that follows the database principles.

The reasoning service of main interest in OBDS, as usual in ontology based data management approaches, is Query Answering (Q. A.). We provide an analysis of this service in different settings and propose *algorithms* for Q. A., in the spirit of OBDA. Right here we show how the ontology plays a major role by mediating the extraction of information from text.

To demonstrate the applicability of our approach in practice, we illustrate MASTRO SYSTEM-T, an OBDS *tool* that we have implemented using robust industrial technologies and experimented on large document datasets.

Last but not least, we formally treat the problem of the Entity Resolution (ER), which is recurrent in the OBIE context, as in general in information integration approaches.

Keywords: Ontology, Information Extraction, Natural Language Processing, Theoretical Computer Science, Document Spanners, Entity Resolution.

List of Publications

The results presented in this thesis are part of my doctoral research work findings. Many of such results have been already published in the following scientific publications¹:

- [134] D. Lembo, Y. Li, L. Popa, K. Qian, and F. M. Scafoglieri. Ontology mediated information extraction with MASTRO System-T. In *Proceedings of the 19th International Semantic Web Conference (ISWC) Demos and Industry Tracks*, 2020. Best Demo Paper
- [135] D. Lembo, Y. Li, L. Popa, and F. M. Scafoglieri. Ontology mediated information extraction in financial domain with Mastro System-T. In *Proceedings of the 6th International Workshop on Data Science for Macro-Modeling (DSMM)*, pages 1–6, 2020.
- [143] D. Lembo and F. M. Scafoglieri. Ontology-based document spanning systems for information extraction. *International Journal of Semantic Computing*, 14(01):3-26, 2020.
- [188] D. Caltabiano, E. Catoni, A. Fabrizi, D. Lembo, M. Minenna, M. Punchina, G. Ronconi, M. Ruzzi, F. M. Scafoglieri. Semantic Technologies for the Production and Publication of Open Data in ACI - Automobile Club d’Italia. In *Proceedings of the 18th International Semantic Web Conference (ISWC) Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas)*, pages 307–308, 2019.
- [140] D. Lembo and F. M. Scafoglieri. Coupling ontologies with document spanners. In *Proceedings of the 32nd International Workshop on Description Logics (DL)*, 2019.
- [191] D. Lembo and F. M. Scafoglieri. A formal framework for coupling document spanners with ontologies. In *Proceedings of the 2nd IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 155–162, IEEE, 2019.
- [98] G. Ganino, D. Lembo, M. Mecella, and F. M. Scafoglieri. Ontology population for open-source intelligence: A GATE-based solution. *Software: Practice and Experience*, 48(12):2302-2330, 2018.

¹Authors are listed in alphabetical order as the norm in theoretical computer science.

- [99] G. Ganino, D. Lembo, M. Mecella, and F. M. Scafoglieri. Ontology population for open-source intelligence (discussion paper). In *Proceedings of the 26th Italian Symposium on Advanced Database Systems (SEBD)*, 2018.
- [100] G. Ganino, D. Lembo, and F. M. Scafoglieri. Ontology population from raw text corpus for open-source intelligence. In *Current Trends in Web Engineering – ICWE 2017 International Workshops (selected revised papers)*, pages 173-186. Springer, 2017

Some contributions reside in unpublished works that are listed below:

- [88] R. Fagin, P. Kolaitis, D. Lembo, L. Popa, F. M. Scafoglieri. Entity Resolution in Description Logics: A theoretical prespective. *Manuscript*
- [145, 192, 136, 190] D. Lembo, F. Medda, F. M. Scafoglieri. Ontology Population from Key Information Documents to uncover Financial Market Manipulation. *Manuscript*

Contents

1	Introduction	1
1.1	Thesis Contributions	5
1.2	Structure of the Thesis	7
2	Background	9
2.1	Ontologies	9
2.1.1	OWL	10
2.1.2	GRAPHOL	12
2.1.3	SPARQL	13
2.2	Metrics	14
3	Ontology Population a GATE-based Approach	17
3.1	General Architecture for Text Engineering	19
3.2	Approach and Architecture	20
3.2.1	Semantic annotation	20
3.2.2	Ontology population	27
3.3	Case Study: design and development	29
3.3.1	Crawling Phase	30
3.3.2	Domain Ontology	31
3.3.3	Gazetteer	33
3.3.4	JAPE Rules in the Semantic Annotation Phase	33
3.3.5	Multi-Lingual Noun Phrase Extractor (MuNPEX)	38
3.4	Case Study: tests and results	38
3.4.1	Performance Evaluation	39
3.4.2	Discussion	39
3.5	Simplifying Gazetteer Lists Generation: design and development	41
3.6	Simplifying Gazetteer Lists Generation: Tests and Results	43
3.7	Final Remarks	49

4	Financial Market Supervision through Information Extraction	53
4.1	CONSOB Domain	55
4.1.1	Key Information Document	56
4.1.2	Ontology	58
4.2	Tool	59
4.2.1	Data Preparation	60
4.2.2	Annotation	62
4.2.3	Exporter	65
4.3	Evaluations and results	67
4.3.1	First Dataset	68
4.3.2	Second Dataset	69
4.3.3	Third Dataset	70
4.3.4	Execution time performances	71
4.4	Final Remarks	72
5	Theoretical Background	73
5.1	Relational Databases	73
5.1.1	Query Answering in Relational Databases	73
5.2	Description Logics	74
5.2.1	Query answering in Ontologies	77
5.3	Computational Complexity	79
5.4	Ontology Based Data Access	80
5.4.1	Mapping Assertions	80
5.4.2	Semantics and Query Answering	81
5.5	Document Spanners	82
5.5.1	Strings and spans	82
5.5.2	Spanner representation	83
5.5.3	An algebra over spanners	84
6	Linking Text Documents to Ontologies	87
6.1	Ontology-based document spanning Framework	89
6.2	Complexity of query answering in OBDS systems	92
6.3	Query Answering via Query Rewriting in <i>DL-Lite</i>	94
6.3.1	GAV Extraction Assertions	95
6.3.2	GLAV Extraction Assertions	99
6.4	Final Remarks	101

7	Mastro System-T	103
7.1	MASTRO	105
7.2	SYSTEMT	107
7.3	MASTRO SYSTEM-T	109
7.3.1	System Overview	110
7.3.2	Query Answering	111
7.4	Case Studies	113
7.4.1	EDGAR	113
7.4.2	CONSOB	115
8	Entity Resolution	121
8.1	Preliminaries	124
8.1.1	Equivalence Classes	124
8.1.2	Ontologies with Concrete Domains	125
8.2	KER systems	126
8.2.1	Terminological component of a KER system	126
8.2.2	Assertional component of a KER system	127
8.2.3	Entity resolution component of a KER system	128
8.2.4	Semantics of a KER system	129
8.3	Universal models	131
8.4	Query answering	132
8.5	Computing a universal model	136
8.6	Adding functionalities	146
8.6.1	Functionalities on attributes as matching dependencies	148
8.6.2	Universal Models	151
8.6.3	Query answering in the presence of functional attributes	151
8.6.4	Revisiting the Chase	153
8.7	Final Remarks	155
9	Related Work	157
9.1	OBDA	157
9.2	Declarative Information Extraction	159
9.3	Ontology-Based Information Extraction	161
9.4	Entity Resolution	162

10 Conclusion	165
10.1 Discussion	165
10.2 Future works	167
A Consob Appendix	189

Chapter 1

Introduction

A huge portion of information is nowadays spread in free-text documents, like reports, e-mails, web pages, articles, etc. These documents are obviously tailored for the human reading, but it is often desirable, within an organization, that relevant data contained therein are extracted and integrated with other corporate data.

Information Extraction (IE) is a subdiscipline of Natural Language Processing (NLP) that studies how to automatically extract data from text and turn them into a structured format, typically a spreadsheet, database, or even a knowledge base [117]. IE has been intensively studied starting from the late '80s [105], and since then, several extraction methods have been proposed, which, in broad terms, can be classified as either ML-based (machine learning-based) or rule-based [189]. In the former case, IE is based on probabilistic models, e.g., probabilistic classifiers or sequence models, (e.g., [109, 95]). Instead, rule-based (a.k.a. pattern-based) approaches encode specific extraction tasks into rules (pattern), mostly corresponding to finite-state transducers (e.g., [68, 55, 197]). Although an ML-based approach can lead to saving the design effort required for pattern definition and extraction rule development, a rule-based approach is adopted in this dissertation for three main reasons. First, it tends, in some scenarios like the financial ones studied in this thesis [57], to yield higher performances, because human expertise in these contexts usually results in very accurate patterns and extraction rules. Second, in the applications of this dissertation, the design effort required by a rule-based approach is expected to be less than the effort required for manually annotating a sufficiently large portion of training data, required by ML-based approach. Third, rule-based approaches (when fully declarative) are the best candidate to be suitably coupled with ontologies, which is the ultimate goal of this thesis.

Indeed, in several contexts, like the ones described in the following chapters, it is often desirable that the extracted data are organized according to an ontology, i.e., a formal conceptualization of the domain of interest [106]. This choice typically simplifies and empowers data

governance and sharing, since ontologies allow for shifting data management at the conceptual level, as well as for reasoning over the representation they provide. Research on Ontology Based Information Extraction (OBIE) [186] attempts to satisfy this need.

Although different formalisms were proposed, in the most general sense ontologies represent knowledge in the form of defined ‘entities’ and their properties. For this reason, the OBIE literature has converged in defining two main subtasks, i.e., the extraction of entities, along with their classification, and the retrieval of their properties (relations with other entities or with values). These two subtasks are defined below.

- **Named Entity Recognition (NER)**, also known as entity identification, entity extraction, or entity chunking, amounts to classify a contiguous set of word tokens contained in unstructured text into pre-defined categories. Consider, for instance the following sentence:

President Joe Biden lives in Washington D.C.

The result of NER should annotate the text as follows:

President [Joe Biden]_{PERSON} lives in [Washington D.C.]_{LOCATION}

i.e., it should produce an annotation PERSON for ‘Joe Biden’, and LOCATION for ‘Washington D.C.’.

- **Relation extraction (RE)** recognizes relationships between two entities or between an entity and a value, starting from evidences in the text. In continuation of the previous example, RE should annotate the document as follows:

President [Joe Biden]_{PERSON} $\xrightarrow{\text{LIVES_IN}}$ [Washington D.C.]_{LOCATION}

that is, Joe Biden, that was previously annotated as a PERSON, and Washington D.C., previously annotated as a LOCATION, are now linked through the relation LIVE_IN

Both NER and RE techniques have seen some advancements over the years. However, rule-based OBIE approaches typically suffer the lack of a formal framework, based on a clear semantics and with a well-defined connection between the rule-based extraction mechanisms and the rules specifying the ontology. At the same time, and mainly because of the above mentioned lack, reasoning over the ontology has not been so far really exploited to support and empower the IE process, so that ontologies in OBIE have been essentially treated until now as simple conceptual models. Through this dissertation, we aim at moving a step forward to fill this gap.

To this aim, we have looked at some prominent open-source rule-based IE tools, such as GATE [70] and coreNLP [156], and used them through the mediation of a domain ontology. However, from our experiences, it turned out that the coupling of ontologies and extraction rules we realized continued to suffer from some of the problems we complained about, in particular, because of the lack of declarative semantics at the basis of the rule languages adopted by these tools, which affects the clear assessment of their expressive power, as also remarked in [87]. At the same time, the use of such languages resulted often involved, and the specification of extractor components particularly time-consuming, also due to the need of having to frequently complement rules with custom pieces of programming code.

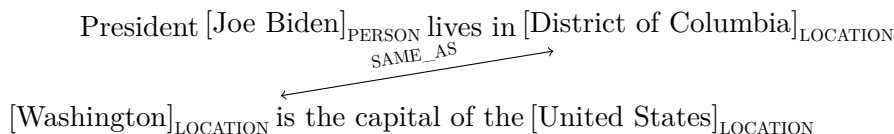
We thus have shifted our attention to a more formal treatment of rule-based IE. In this respect, Fagin et al. have carried out in the last years a foundational study on the topic, and have introduced a formal framework based on the notion of (*document*) *spanner* [86, 87]. A *spanner* is a function that maps a given string to a relation, i.e., a set of tuples, over its spans. A *span* is a pair of indices that identify substrings of a given string. For example, given the string President Joe Biden, the spans [10, 12) and [13, 17) identify the substrings Joe and Biden, respectively. Fagin et al. studied possible representations of spanners and analyzed how the use of some algebraic operations on the relations returned by the spanners evaluation influences the expressiveness of spanner-based extractors. In particular, they considered spanners defined by regular expressions with capture variables (a.k.a. “*regex formulas*”), and manipulated through some *algebraic* operators. Intuitively, regex formulas are regular expressions allowing for mapping sub-matches of regular expressions, in the form of spans, to variables. Algebraic operators considered in [86], are the relational operators union, projection, and join, plus string-equality selection, which allows to select tuples of spans identifying certain wanted substrings. Spanners represented by regex formulas and combined with the above operators are called core spanners, denoted $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$, and are those providing the expressiveness suited to the aim of our investigation, as we will see in the following chapters of this dissertation.

Inspired by the above work, in this thesis we construct a formal framework, called Ontology Based Document Spanning (OBDS), for coupling spanners with ontologies. Through this framework we intend to structure information extracted from text documents according to the terminology provided by an ontology. To this aim, we adapt the well-known Ontology Based Data Access (OBDA) framework, in which an ontology is mapped to an external source database through declarative mappings specifying the semantic relationship between the ontology vocabulary and the data at the sources [48, 212]. OBDA is a powerful paradigm for data access, integration, and governance. In OBDA, however, ontologies have been used so far only on top of relational databases, with very few exceptions (as, e.g., [30]). In this dissertation we thus enrich OBDA with the capability of accessing unstructured information contained in text documents.

Within OBDS systems, we study the problem of query answering, which is the service of main interest in ontology based data management systems, and show that the ontology plays a leading role in mediating information extraction. Interestingly, the task we solve is a form of on-the-fly information extraction, in the spirit of query answering in OBDA.

We finally complement the above investigation through the study of Entity resolution (ER), another core task useful to populate ontologies from text with clean data, and that our initial experiences on OBIE highlighted as a serious issue in IE. ER amounts to identify different entities that describe the same real-world object [129]. It is a fundamental process in data management [85], that affects data of any kind, from structured ones [91], as in relational databases [129, 20], to semistructured information, as in the Semantic Web context [166, 201], to unstructured data, as for entities that have to be resolved starting from text documents.

In particular, **Entity Resolution** in NLP, also known in the literature as Record Linkage, is the task of finding tokens in free-text that refer to the same entity, as in the following example.



Here, District of Columbia and Washington, both previously annotated with LOCATION, clearly refer to the same city of the real-world and then should be linked to each other (denoted through a SAME_AS relation in the example).

Our investigation on ER led us to the definition of a formal framework, called KER, in which we couple ontologies with expressive entity resolution rules, allowing for specifying the conditions under which two entities have to be interpreted as the same real-world object. We define tailored semantics for KER systems to properly deal with entity resolution rules in the presence of ontologies, and study query answering under such semantics. We point out that in the KER framework we do not consider mechanisms to link ontologies to external data sources, e.g., text documents, as in OBDS, and that our results are obtained through chase-based techniques. Thus, to apply them to OBDS (or even OBDA) systems it is necessary to first populate the ontology by materializing the result of the extraction phase.

We finally point out that the experimental results shown in this thesis are reproducible, and all the material used, including datasets, code, ontologies, etc., can be found in the following GitHub repository <https://github.com/Scafooo/Phd-Thesis>.

1.1 Thesis Contributions

This thesis provides both practical and theoretical results on the themes we have discussed so far. In the following, we provide a more precise account of all its contributions.

- I We address OBIE from a practical perspective. We consider the most widely used open-source IE technologies, GATE and CoreNLP, and adapt them to populate domain ontologies. Properly, we describe, by detailing the approach followed and highlighting the issues encountered, the pipelines that we have designed, and the ad-hoc implementations that we have carried out. Our results have been validated in two main scenarios, one (approached through GATE) from the Open Source Intelligence context, considering text documents crawled from the web, and one (faced with coreNLP) concerning with extraction from financial documents, investigated in the context of a collaboration with CONSOB (Italian Companies and Exchange Commission). For both experimentations we designed domain ontologies and executed fully reproducible tests on large datasets.
- II We introduce the notion of *Ontology Based Document Spanning (OBDS) system*. In an OBDS system, an ontology is linked to text documents through *extraction assertions*, which act similarly as mapping assertions in OBDA, and in which document spanners are associated to queries over the ontology.
- III We study *query answering over an OBDS system*, i.e., how to answer a user query specified over the ontology by retrieving the answers from the text documents mapped to the ontology. We consider the case in which (i) the ontology is specified in the Description Logics $DL-Lite_{\mathcal{R}}$ or $DL-Lite_{\mathcal{F}}$ (see Section 5.2), (ii) user’s queries are conjunctive queries (CQs), (iii) spanners in extraction assertions belong to the class $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$ (see Section 5.5), (iv) queries in the head of extraction assertions are CQs. We show that *query answering is in PTIME in data complexity* (i.e., the complexity computed only with respect to the size of the underlying documents). We remark that $DL-Lite_{\mathcal{R}}$ and $DL-Lite_{\mathcal{F}}$ are the two most popular ontology languages used in OBDA to deal with large datasets, CQs are the most expressive queries for which query answering over ontologies has been shown to be decidable, and spanners in $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$ are among the most expressive document spanners considered in [86]. We note also that extraction assertions we define resemble GLAV mapping assertions used in data integration and in OBDA, i.e., the most expressive form of mappings adopted in these contexts [146, 78, 48, 89].
- IV We investigate *query rewriting in OBDS systems*, i.e., whether it is possible to answer a query by first rewriting it and then evaluating the rewriting over the data layer. Our aim

is to understand whether we can reduce query answering to the execution of a document spanner of the same kind of those used in the extraction assertions. We positively answer the above question for the case in which ontologies are specified in $DL-Lite_{\mathcal{R}}$ and extraction assertions have the full expressive power allowed by our framework, and for the case of $DL-Lite_{\mathcal{F}}$ ontologies when we adopt some restrictions on the form of extraction assertions (still obtaining, however, a practically interesting case). We indeed provide an algorithm that rewrites every CQ issued over an OBDS system (i.e., over its ontology) into a spanner belonging to $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$.

V We present MASTRO SYSTEM-T, a tool for OBDS. We describe its architecture and the components that realize its query answering process. Through the support of an experimentation within a real-world financial application domain, we show the benefits our tool can provide in IE. We make a performance comparison between MASTRO SYSTEM-T and CoreNLP. Namely, we re-examine the CONSOB scenario mentioned above, and use MASTRO SYSTEM-T to perform the information extraction task.

VI We formally define a framework for Entity Resolution over ontologies, which we call KER. Namely, we describe the components of a KER system, and in particular the form of entity resolution rules it allows for. We give the semantics of KER systems in terms of special interpretations, which, besides the domain of interpretation and the interpretation function, are characterized by an *equivalence relation* allowing to interpret ontology concepts with sets containing equivalence classes that gather together resolved entities, i.e., entities that are denoting the same real-world object.

VII We study KER systems without and with functional dependencies in the ontology language. Whereas functionalities on roles are assimilable to entity resolution rules, functionalities on attributes may enforce equalities on different values (e.g., different numbers or strings), which would cause a KER system to be inconsistent. To cope with this situation, we interpret functionalities on attributes as *matching dependencies* [23, 92], i.e., rules establishing conditions under which values have to be matched (rather than equated). We resolve matching through a general function that takes the union of the values to be matched, in the spirit of the *union class of match and merge functions* considered in [18]. We show that also in the presence of functional attributes conjunctive query answering can be reduced to query evaluation over a tailored chase built specifically for KER system that can be constructed in polynomial time respect to the size of facts reside in the ontology.

1.2 Structure of the Thesis

The thesis is organised in ten chapters, whose content is briefly summarised below:

- Chapter 1 is the current introduction
- Chapter 2 is a background where basic notions about OWL ontologies, SPARQL queries, and the metrics used to evaluate the quality of IE systems are presented. Some of these notions are here given informally and precisely formalized later, in Chapter 5.
- Chapter 3 illustrates an approach for the automatic population of predefined ontologies with data extracted from text, and discusses the design and realization of a pipeline based on the General Architecture for Text Engineering (GATE) system. We provide an experimental validation of our approach, on a specific domain namely “Mafia Capitale”, that shows its performances in terms of the quality of the information we are able to extract.
- Chapter 4 reports the activity we have been carried out within a joint project between Sapienza University and CONSOB, consisting into an IE application applied to financial documents, to support monitoring actions aimed to uncover financial wrongdoing.
- Chapter 5 is a theoretical background introducing Description Logic ontologies, the framework of Document Spanners, and the Ontology Based Data Access.
- Chapter 6 illustrates the formal framework of Ontology Based Document Spanning (OBDS). The problem of query answering over OBDS is studied considering different settings. We will provide both algorithms and complexity results.
- Chapter 7 presents MASTRO SYSTEM-T, an OBDS tool born from a joint collaboration between the Sapienza University and IBM Research Almaden, and its application in financial domains. The chapter also reports about a comparison in terms of performance with the technology used in Chapter 4.
- Chapter 8 introduces KER systems, i.e., a formal framework for entity resolution over ontologies. In this chapter, we study query answering over KER systems.
- Chapter 9 reviews some literature that is closely related to this thesis.
- Chapter 10 concludes the thesis with a brief discussion and possible directions for future work.

Chapter 2

Background

This chapter contains some of the basics needed to understand the following chapters of this thesis. Here, we introduce ontologies, their definitions through textual and visual representations, and the SPARQL query language. We dedicate the last part to talk about the metrics used to evaluate the results of the approaches that will be shown in this dissertation. This chapter is intended to be a brief introduction to such matters, while an exhaustive treatment of them is out of our scopes. For further background we refer the reader to [160, 173]

2.1 Ontologies

In Computer Science, an ontology is commonly defined as a specification of a conceptualization, that is, a formal description of an abstract, simplified view of a certain portion or aspect of the world [106, 107]. According to this definition, an ontology provides a conceptual representation of a domain of interest, and thus it abstracts from aspects typical of logical or physical data modeling and storage. Furthermore, an ontology is formal, which means that the description of the world it provides is not ambiguous. It is usually given in some mathematically based language with precise syntax and clear semantics, commonly rooted in some logic. Another essential characteristic of ontologies is that they are shared, i.e., they are agreed upon by all their users. Therefore, ontologies are considered an excellent way to represent knowledge on the Web, where they are mainly used to add semantics to data. This also allows for the usage of powerful reasoning mechanisms that ontologies are usually equipped with [9]. The importance of ontologies to interpret and structure Web data is also demonstrated by the huge standardization effort carried out by the W3C, which led to the definition of OWL, the standard Web Ontology Language¹ [110].

¹<https://www.w3.org/TR/owl2-primer/>

2.1.1 OWL

As it is typical in ontologies and data modeling, in OWL, we distinguish between *intensional* and *extensional* knowledge. Intensional knowledge is given in terms of logical axioms involving classes (a.k.a. concepts) and properties, which are of two types, *object properties* (a.k.a. binary relationships or roles) and *data properties* (a.k.a. attributes). Classes denote sets of individuals (a.k.a. objects), object properties denote binary relations between individuals, whereas data properties denote binary relations between individuals and values from predefined datatypes. Each of these elements in OWL is generally defined as resource, and it is identified by a sequence of characters called *Uniform Resource Identifier* (URI). When the URI is identifying an individual, we also refer to it as entity. Among the various concrete syntaxes proposed to express ontologies in OWL, in this thesis, we use the RDF/Turtle² one. At the syntactic level we thus treat an OWL ontology as an RDF dataset, i.e., a set of triples, each consisting of a subject, a predicate, and an object, and express it in Turtle, according to which elements in a triple are separated by whitespaces, each triple is terminated by a ‘.’ (dot), and prefixes can be used to make URIs more compact (we will not use more advanced features of Turtle).

Under the above RDF view, an OWL ontology can be readily seen as a *Knowledge Graph*, i.e., an oriented graph where the nodes are the resources that appear in the triples in the position of subject and object, and for each triple there is a labeled edge, whose label is the triple predicate, and whose direction goes from the node representing the subject to the node representing the object.

OWL is a very expressive language composed of a wide vocabulary used to define its resources. For the sake of simplicity, we introduce its most common constructs through an example, and we refer the reader to [104] for further details.

Example 2.1. Consider the following ontology which captures a part of the domain described

²<https://www.w3.org/TR/turtle/>

later in Chapter 3 and acts also as a running example for this chapter:

Intensional Level

```

 $\theta_1$ ) :Person a owl:Class .
 $\theta_2$ ) :Politician_Company a owl:Class .
 $\theta_3$ ) :City a owl:Class .
 $\theta_4$ ) :first_name a owl:DataProperty .
 $\theta_5$ ) :last_name a owl:DataProperty .
 $\theta_6$ ) :lives_in a owl:ObjectProperty .
 $\theta_7$ ) :Politician rdfs:subclassOf :Person .
 $\theta_8$ ) :Journalist rdfs:subclassOf :Person .
 $\theta_9$ ) :Journalist owl:disjointWith :Politician .
 $\theta_{10}$ ) :first_name rdfs:domain :Person .
 $\theta_{11}$ ) :last_name rdfs:domain :Person .
 $\theta_{12}$ ) :lives_in rdfs:domain :Person .
 $\theta_{13}$ ) :lives_in rdfs:range :City .

```

Extensional Level

```

 $\alpha_1$ ) :#Biden a :Politician .
 $\alpha_2$ ) :#Washington a :City .
 $\alpha_3$ ) :#Biden :first_name 'Joe' .
 $\alpha_4$ ) :#Biden :last_name 'Biden' .
 $\alpha_5$ ) :#Biden :lives_in :#Washington .

```

Here `owl:` and `rdfs:` are two standard prefixes reserved to identify resources belonging to the OWL vocabulary (which also includes terms from the RDFS vocabulary). Instead `:` (the empty prefix) is used to specify the resources belonging to this specific ontology (as a convention in this thesis, when `:` is followed by the symbol `#`, the URI refers to an entity).

The intensional level ($\theta_1 - \theta_{11}$) declares the classes `:Person` (θ_1), `:Politician` (θ_2) and `:City` (θ_3), the attributes `:first_name` (θ_4) and `:last_name` (θ_5) and the role `:lives_in` (θ_6). θ_7 and θ_8 assert that every politician and every journalist is also a person. Triple θ_9 says that if someone is a journalist she/he cannot be a politician. Triple θ_{10} and θ_{11} specify the domain of the attributes `:has_first_name` and `:has_last_name`, respectively, which is `:Person`. Finally θ_{12} and θ_{13} specify that the domain of the role `:lives_in` is `:Person` and the range is `:City`.

The extensional level ($\alpha_1 - \alpha_5$) states that the entities `:#Biden` and `:#Washington` are instances of `:Politician` and `:City` (through α_1 and α_2 , respectively). The name of `:#Biden` is 'Joe' and its last name is 'Biden'. Finally the last triple (α_5) states that `:#Biden` lives in `:#Washington`. □

We would also like to point out that in OWL it is also possible to assert that two entities in fact denote the same individual (notice that OWL does not adopt the unique name assumption, i.e., does not impose that different entities denote different objects of the domain of discourse). This is done through the use of the `owl:sameAs`. For instance, the as-

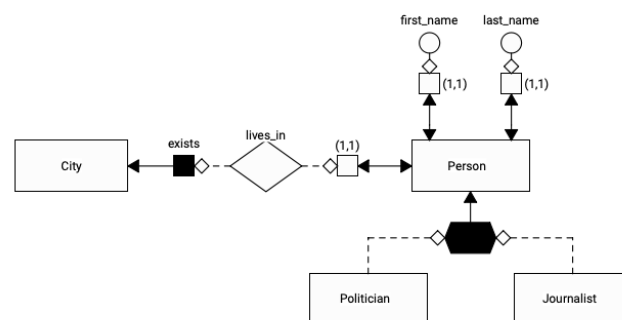
sertion `:#District_of_Columbia owl:sameAs :#Washington` states that the two individuals `:#District_of_Columbia` and `:#Washington` have to be interpreted as the same object.

As mentioned above, the purpose of an ontology is not limited to the mere representation of knowledge, but provides a suitable structure over which complex reasoning tasks can be performed. For this reason, the W3C consortium has defined the current version of OWL, also referred to as OWL 2, which is completely rooted in Description Logics, and it has designed different tractable profiles³ (also called fragments or sub-languages). Each profile trades some expressive power, placing restrictions on the structure of OWL 2 ontologies, in order to improve the efficiency of the reasoning services.

2.1.2 Graphol

We would like to mention that, besides standard textual syntaxes used to specify ontologies⁴ (e.g., the RDF/Turtle syntax that was used in the previous example), various attempts have been made to devise graphical representations of OWL ontologies, to make the understanding of ontologies and their specifications easier for non-experts in logic or formal languages. Among them, GRAPHOL⁵ is a recent diagrammatic language that is completely graphical, i.e., it does not require to complement diagrams with logical formulas, and that is equivalent to OWL 2, the current version of the OWL standard [138, 139]. Thanks to this equivalence, GRAPHOL completely preserves the OWL ontology specification, but at the same time allows for a more intuitive ontology comprehension. This is the reason why in this thesis we prefer sometimes to present the ontologies using this graphic formalism rather than through textual syntax.

Example 2.2. Consider the following GRAPHOL ontology which contains the intensional level of the OWL ontology in Example 2.3



Classes are represented through labeled rectangles, objectproperties through labeled diamonds, and datatypes through labeled circles (as in Entity-Relationship diagrams). To

³<https://www.w3.org/TR/owl2-profiles/>

⁴<https://www.w3.org/TR/owl2-syntax/>

⁵<https://www.obdasystems.com/graphol>

denote the first (domain) and the second (range) component of a property, a white and a black square are respectively used, connected to the property they refer to with dashed arrows ending with a small diamond (called input edges). A black hexagon indicates a union of classes that are also disjoint, whereas for plain union an hexagon labeled with “or” is used. Other operators allowed in hexagons are, for instance, “and” (to capture intersection of classes), and “not” (to capture the complement of a class). Each class involved in the union (or in the intersection, complement, etc.) is connected to the hexagon with an input edge. Labels associated to the domain and range of properties are used to specify restrictions, such as existential restrictions (label `exists`), or cardinality restrictions (label `x,y`, where `x` and `y` are non-negative integers, with $y \geq x$ and $y \neq 0$). Cardinality restrictions are similar to cardinality constraints in Entity-Relationship or UML class diagrams. Solid arrows denote inclusions between classes (or properties), even general ones, which are obtained through the use of operators like the ones mentioned above (see, e.g., the arrow going from the black hexagon to the class `:Person`, or the one going from the range of `:lives_in` to the class `:City`).

2.1.3 SPARQL

Among the various reasoning services over ontologies, one of the most studied that will also be treated later in this thesis is query answering. Several query languages have been designed for this purpose, including `OntoQL` [115], `RDQL` [195], `SeRQL` [31]. In this thesis, we consider the `SPARQL` language [173] standardized in 2008 by the W3C and supported by most RDF triple stores, thus considered the standard language for querying ontologies.

The main query type in `SPARQL` is the `SELECT` query, which has two main components: a list of selected variables, and a `WHERE` clause for specifying the basic graph patterns, i.e., a conjunction of triples, specified in RDF/Turtle like syntax, except that each of the subjects, predicate, and object may be a variable.

The evaluation of these queries on (the RDF graph corresponding to) the ontology is performed through a graph pattern matching mechanism. The result of a `SELECT` query is the set of all pattern matches, which are usually represented as a table having one column for each selected variable and one row for each pattern match.

Example 2.3. Consider the following `SPARQL` query posed over the ontology in Example 2.3, which is asking for the first names and last names of the politicians living in Washington.

```
SELECT ?X ?Y
WHERE {
    ?Z has_first_name ?X
    ?Z has_last_name ?Y
```

```

    ?Z a :Politician .
    ?Z :lives_in :#Washington .
}

```

The result of the query is the following table:

?X	?Y
'Joe'	'Biden'

SPARQL also provides several operators for combining graph patterns such that optional patters, union of patterns and filters. We refer the reader to [108] for a complete list and description of them.

2.2 Metrics

In the following chapters of this thesis, we will show the quality of the proposed approaches through a series of tests. These are performed, taking into account the standard parameters for evaluating IE tasks. Properly they are the Correct Annotations (a.k.a. True Positives), i.e., the annotations we have identified that turned out to be correct, the Spurious Annotations (a.k.a False Positives), i.e., the annotations we have found that were indeed wrong, and the Missing Annotations (a.k.a. False Negatives), i.e., those annotations that we were not able to find. These three parameters allowed us to calculate:

- *Precision*, i.e. the fraction of the correct annotations over the total number of identified annotations. It is formally defined as follows:

$$Precision = \frac{Correct}{Correct + Spurious} \quad (2.1)$$

- *Recall*, i.e the fraction of the correct annotations over the total amount of annotations It is formally defined as:

$$Recall = \frac{Correct}{Correct + Missing} \quad (2.2)$$

- *F-measure*, i.e the harmonic mean of *Precision* and *Recall*. It is formally defined as:

$$F\text{-measure} = \frac{Precision * Recall}{0.5 * (Precision + Recall)} \quad (2.3)$$

Example 2.4. In the following example, the entity Joe Biden is not correctly annotated, because

we missed the annotation over the token Joe. Instead, Washington D.C. can be defined as a correct annotation.

President Joe [Biden]_{PERSON} lives in [Washington D.C.]_{LOCATION}

For the rest of the thesis, in the tables concerning the tests carried out, if not differently specified, we indicate with C.A. the correct annotations, with S.A. the spurious annotations and, with M.A. the missing annotations.

Chapter 3

Ontology Population a GATE-based Approach

In this chapter we report about a first experience on Information Extraction, whose final outcome has been the construction of the instance level of an OWL ontology by using data extracted from text documents crawled from the Web. To this aim we have used GATE (General Architecture for Text Engineering) and some specific third-party components tailored to ontology population. This experience has been carried out in the context of Open-Source INTelligence (OSINT) for security applications.

OSINT is intelligence, i.e., information gathering, based on publicly available sources such as news sites, blogs, forums, etc. OSINT is nowadays used in many application scenarios, for instance for security (e.g., identifying lone wolves and weak signals on the Web [11]), market intelligence (understanding users' profiles and trends), or statistics (to cross-check and complement data collected with traditional methods [193]). A major issue in using *Internet as a data source* is that Web data come mainly in the form of free text, thus with no structure and formal semantics. This means that two problems have to be faced. First, how to select structured information from unstructured texts, and, second, how to interpret the selected information according to precise semantics.

For a comprehensive solution we have investigated how to populate a domain ontology using the information extracted from a given cluster of textual documents crawled from the Web. Structuring Web data according to the predicates and axioms defined in an ontology turns out to be particularly effective for our purposes, even in the light of the reasoning abilities ontologies allow for [9].

Among various existing open-source tools for information extraction (e.g., LingPipe¹ or

¹<http://alias-i.com/lingpipe/>

OpenNLP²), we decided to use GATE³, given the flexibility it allows to customize its underlying architecture, and to incorporate other external components developed by third parties. The extraction activity in GATE is typically carried out through different stages, each depending on the contingent needs of the user, who can, for instance, adopt existing dictionaries (a.k.a. Gazetteers) for NER (Named Entity Recognition), or create new ones, and/or specify tailored extraction rules through the use of the *Java Annotation Pattern Language* (JAPE) [69]. GATE has become popular in the last years, especially in relation to information extraction from English documents. To some extent, it also supports other languages, primarily thanks to the dictionaries created and then shared on the platform by its many users.

The main contributions of this chapter are:

- (i) to show how to build a complete ontology population pipeline step-by-step, presenting all the relevant methods, techniques, and design choices, entirely based on open source tools;
- (ii) to provide an experimental validation of this approach that shows its performances in terms of the quality of the information we are able to extract;
- (iii) to describe how to exploit semantic technologies to reduce the manual workload needed in some components of our pipeline (in particular, for the definition of Gazetteer lists).

Our techniques have been tested within the XASMOS and RoMA projects, involving the ‘Leonardo’ company (formerly Selex), and the ‘Sapienza’ Research Center on Cyber Intelligence and Information Security. The projects focused on OSINT for security applications. Namely, we considered the case study ‘Mafia Capitale’, from the name of an important 2015 investigation that received quite a lot of attention from the Italian media, and, thus, turned out to be a valid testbed (for both number of Web documents available and significance of the domain). For our case study specific dictionaries and JAPE rules for Italian were created to instantiate a domain ontology with the information extracted from Web documents. We present our case study to provide some insight on the possibilities offered by such an approach. In particular, we successfully applied it to more than 2600 documents crawled from the Web. The results we obtained were encouraging in terms of number of extracted instances and quality of the information extracted.

In our case study we experienced that some of the tasks we had to deal with, such as dictionaries or JAPE rule definitions, were rather domain specific and time-consuming since they required a lot of manual work. We, thus, started to investigate how to refine our approach so that the time needed to perform these tasks could be reduced and the solution adopted could be

²<https://opennlp.apache.org/>

³<https://gate.ac.uk/>

easily reused in different contexts. In particular, we focused on the dictionary construction task, and faced it with a more general approach, which relies on a simple extraction of dictionaries through SPARQL queries issued over the open knowledge base Wikidata ⁴.

The chapter is structured as follows:

- Section 3.2 describes our approach for ontology population and introduces the modules that were used in the GATE system;
- Section 3.3 presents our case study and explains the specific actions we had to take for the application at hand;
- Section 3.4 reports the evaluations and results for our case study.
- Section 3.5 illustrates the Wikidata-based approach for the generation of Gazetteer lists;
- Section 3.6 discusses the evaluations and results for this approach.

3.1 General Architecture for Text Engineering

GATE is an architecture, a framework and a development environment for Language Engineering (LE) [70]. Since it is an 'architecture', it defines the organization of an LE system and the assignment of responsibilities to different components, and ensures that the interactions of the components satisfy the system's requirements. As a 'framework', it provides a reusable design for an LE software system and a set of preset software building blocks that language engineers can use, extend and customize for their specific needs [72].

GATE has a component-based model which allows for coupling and decoupling of the processors, thereby facilitating the comparison of alternative configurations of the system or different implementations of the same module (e.g., different parsers). GATE comprises a core library and a set of reusable LE modules. The framework implements the architecture and provides (amongst other things) facilities for processing and visualizing sources, including representation, import and export of data. The provided reusable modules are able to perform basic language processing tasks such as POS (Part-Of-Speech) and semantic tagging. This eliminates the need for users to keep recreating the same kind of components, and provides a good starting point for new applications.

GATE components may be implemented through a variety of programming languages, but they are always represented to the system as Java classes. A class may simply call the underlying program or provide an access layer to a database; alternatively it may implement the whole component.

⁴<https://www.wikidata.org/>

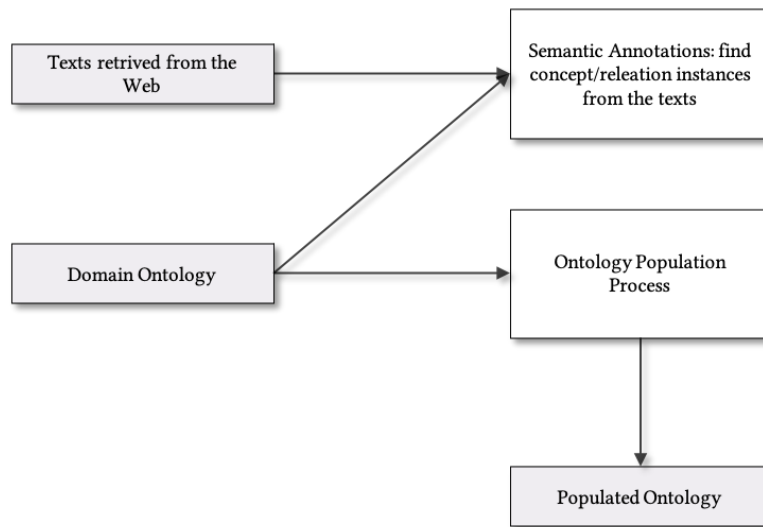


Figure 3.1. The process for ontology population.

3.2 Approach and Architecture

Our work builds, by leveraging GATE, a pipeline for the extraction of information from text documents in a specific language. We assume to have as input a reference ontology that has been designed by analysts and domain experts, and that needs to be populated with instances of classes and properties. To achieve this goal through the components in GATE, our approach proceeds through two main phases, see Figure 3.1:

1. *Semantic annotation*: in this phase, we create annotations, i.e., metadata that indicate properties of the text contained in the analyzed documents. At the end of this phase, the annotations will allow us to identify in the text those entities that are indeed instances of the classes and properties of the ontology given as input to our pipeline.
2. *Ontology population*: in this phase, we extract and classify instances of classes and properties described in the reference ontology starting from the output of the previous phase.

In the following, we detail all the processing resources (PRs) used to create our pipeline in GATE.

3.2.1 Semantic annotation

Entity identification, entity disambiguation, and text annotation are the three main tasks that semantic annotation of resources has to deal with. In our proposed GATE-based approach,

this phase relies on several PRs, which are available as GATE plugins, possibly provided by third-party organizations⁵.

In the following, we describe such resources, through an ongoing example in which we analyze a short text, and we show the output produced by each component, through some screenshots. In such screenshots, we have three main areas (cf. Figure 3.2): the left-upper part of the window shows the text that is analyzed, the right-hand side of the window reports the types of annotations available, and which the user can select, whereas the bottom part of the window describes the selected annotations. This area indicates the *Type* of the annotation, the portion of the text that the annotation refers to (*Start* and *End* characters), the *Id* of the annotation (which is unique), and the *Features* of the annotation (which depend on the type)⁶.

The semantic annotation components used in our approach are the following ones:

1. *Document Reset*: this component allows to reset the annotations that have been added to a document, and it is useful when different applications are executed on the same corpus. This resource has been inserted in each pipeline before any other PR so that annotations added by a previous application on the text documents do not influence the results obtained in the current execution [69].
2. *GATE Unicode Tokeniser*: this component is used to split the text in *Tokens* and *SpaceTokens*; the latter ones denote spaces among single terms, whereas the former ones are of four kinds, i.e., number, punctuation, symbol, and word. The use of this component is essential for introducing in the document those annotations that will be exploited in next phases by JAPE rules, which we describe later. In Figure 3.2, we show the output of the GATE Unicode Tokeniser for our ongoing example. For instance, the first row in the bottom part describes an annotation of type *Token* that refers to a string starting at character 0 and ending at character 3. Among the features, we can read that the *Token* is of kind *word*, contains 3 characters, the first character is an upper letter, and the string is “The”. Similarly for the other rows.
3. *RegEx Sentence Splitter*: this component divides the processed document into *sentences*, which are chunks of text that make sense taken in isolation. It is essentially language-independent, in the sense that it can be used as it is for very many common languages, such as English, German, Italian, etc. It is an alternative component to the ANNIE⁷ Sentence Splitter, a splitter component, completely based on JAPE, provided by GATE. In particular, RegEx Sentence Splitter outperforms ANNIE in terms of execution time

⁵<https://gate.ac.uk/gate/doc/plugins.html>

⁶The column Set refers to specific annotation sets, but is not used in this example [69].

⁷<https://gate.ac.uk/sale/tao/splitch6.html#chap:annie>

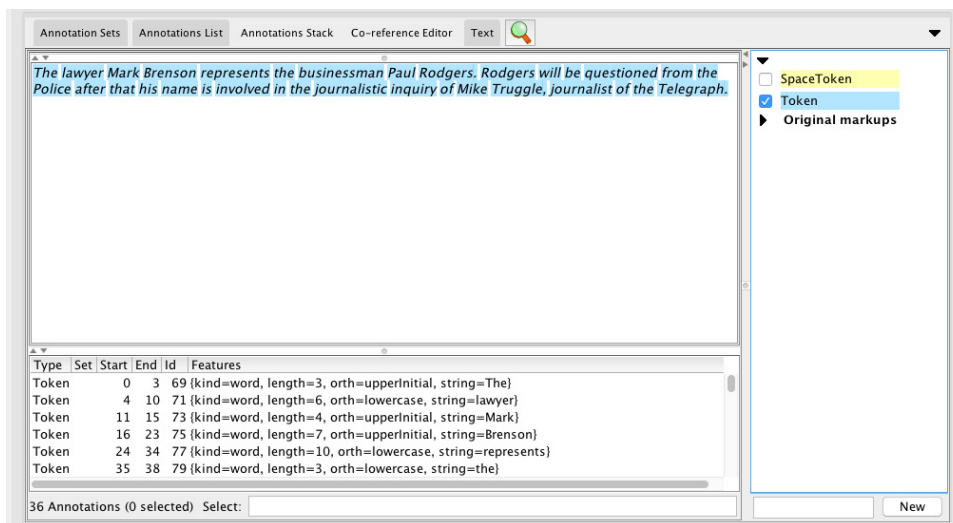


Figure 3.2. An example output of GATE Unicode Tokeniser.

and for its abilities to deal with several languages and irregular inputs. Moreover, this component is completely implemented in Java, and, as the name itself says, it is based on regular expressions that define the syntactic rules for sentence identification. At the end of this phase, two new annotations are added to the document, i.e., *Sentences* and *Splits*. As shown in Figure 3.3, no particular features are assigned to sentence annotations, whereas split annotations can be *(i)* internal, i.e., splits among two sentences, *(ii)* external, i.e., the split that closes the document, or *(iii)* non-splits (not shown in the example), which are fragments similar to splits, but not really splits (such as punctuations used for abbreviations).

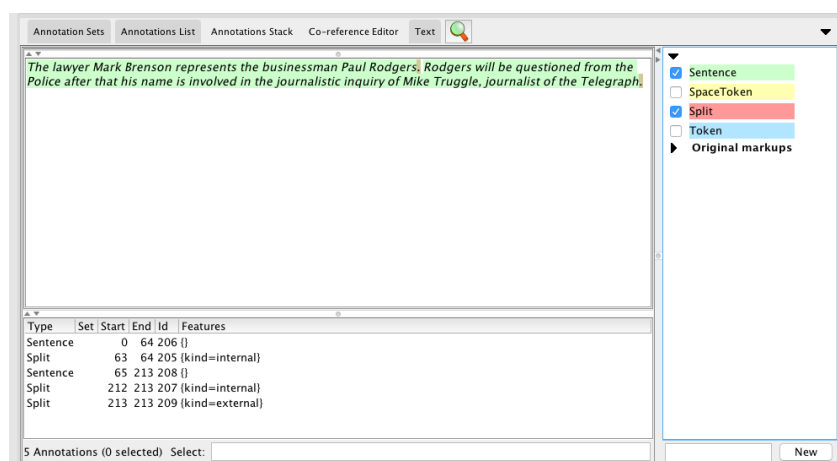


Figure 3.3. An example output of RegEx Sentence Splitter.

4. *TreeTagger Part-of-Speech (TreeTagger POS)*⁸: it is a component for document annotation with POS and lemma information, developed at the Institute for Computational Linguistics of the University of Stuttgart. It is a Markov Model tagger that makes use of a decision tree to get more reliable estimates for contextual parameters [194]. It can be used with various languages, provided that it is fed with an input parameter file specific for the language. Despite the fact that several POS taggers exist that could be used in this phase, the TreeTagger POS turned out to be the best one for the Italian language (which is the language considered in our case study). Indeed, we tested other taggers (such as TagPro⁹) on various documents in Italian, and TreeTagger POS provided best performances for both number of correct POS annotations and lemma information. In particular, information on lemmas, which are canonical forms of set of words (e.g., “represent” is the lemma of “represents”, “representing”, etc.), is a peculiar characteristic of TreeTagger POS. Similarly to the tokenization obtained through the GATE Unicode Tokeniser, at the end of this phase we have a set of annotations associated to each token; in this case, the column *Features* provides POS and lemma information. This is shown in Figure 3.4, where we called such annotations *SemanticTokens*, and where each annotation reports, in particular, a category and a lemma. We notice that the category assumes one among several POS values, specific for the language at hand (e.g., for the English language, DT is a determiner, NN is a common name, NP is a proper singular name, VVZ, is verbe present tense, 3rd person singular).

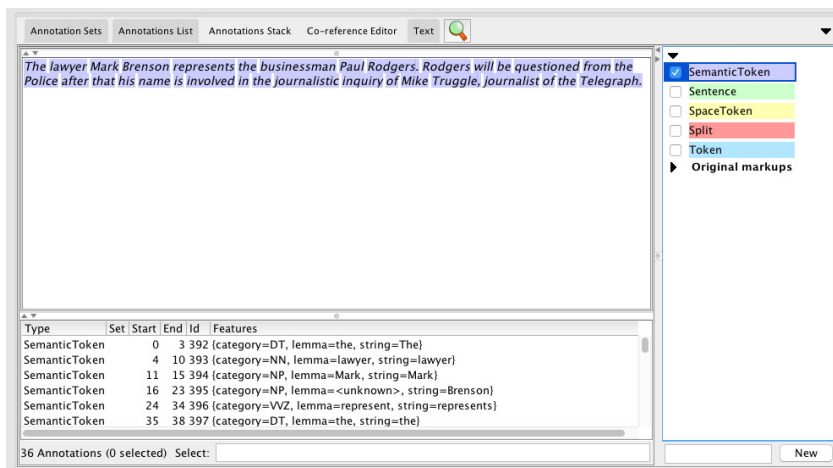


Figure 3.4. An example output of TreeTagger POS.

5. *Gazetteer* [69]: this component annotates the documents on the basis of a set of lists

⁸<http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger>

⁹<http://hlt-services2.fbk.eu/textpro/?p=89>

containing names of entities, such as countries or organizations, but also names or abbreviations for types of companies (e.g., ltd., Corp., Inc.), for political offices (e.g., President, Prime Minister, Senator), etc. Each list can be associated with a so-called major and minor type. Usually, at least the major type is specified. Intuitively, these types correspond to categories, such that minor types are more specific than the corresponding major types. If the document contains a string matching with an element of a Gazetteer list, the component annotates the string with the major and minor type of this list. If the string has more than one match, major and minor types of all the matching lists are added.

As an example, let us consider two different lists, called *day.list* and *month.list*, respectively, the former containing the days of the week, the latter containing the months of the year. We associated as major type to both lists the value *time*, whereas we set as minor type *days* for *day.list* and *month* for *month.list*. If a document contains the string “Monday”, it will be annotated with a *Lookup* annotation having major type *time* and minor type *day*. This allows to exploit such annotations in next phases (in particular through JAPE rules) at different level of abstraction, e.g., considering the string as a time fragment or as a day, depending whether the major or minor type is accessed.

The effectiveness of this phase is completely dependent on the quality of the information contained in the lists. The construction of such lists is a non-trivial task, but once created, a list can be used in several applications.

In Figure 3.5, we provide the output of the Gazetteer PR applied to the text of our ongoing example. We used a list with some job names, a list with some proper names of persons, and a list with some names of organizations. For instance, in the fifth row, we annotate the string “Police” with minor type *defence*, and major type *organization*.

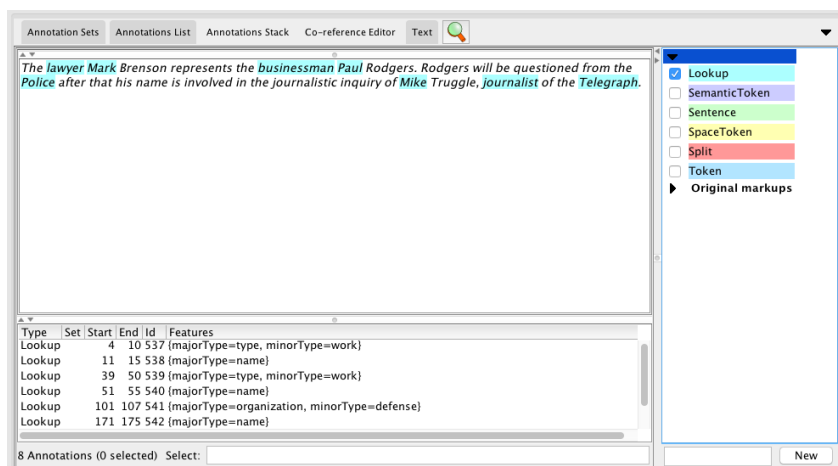


Figure 3.5. An example of output of the Gazetteer PR.

6. *JAPE Transducer (Semantic)*: this component completes the phase of semantic annotation and is used to import the user-written JAPE rules into the GATE platform, so that they can be used to create or modify text annotations on documents. The JAPE language allows to recognize regular expressions among the annotations previously produced. Once the expression is matched, a further annotation referring to the searched patterns/entities is added to the document [69].

The JAPE Transducer executes a JAPE grammar, which is a set of phases organized in a precise order, each one composed by a set of patterns and action rules. These phases are performed in a sequential way to create a cascade of finite-state transducers. Each of these transducers takes as input the output and the related annotations of the previous phase. The running order of the phases is defined in an index JAPE file.

Consider the following example of a phase in JAPE:

```
1 Phase: University
2 Input: Lookup Token
3 Options: control = appelt
4 Rule: University1
5 Priority:80
6 (Token.string == "University" Token.string == "of" Lookup.minorType == city):orgName
7 →
8 :orgName.Organisation = {rule = "University1", kind = "university"}
```

From line 1 to line 3 we have the header that contains the attributes:

- *Phase*: it indicates the name of the phase.
- *Input*: it indicates the kind of data, obtained from previous operations, on which the user wants to perform actions. In the example, we consider Lookup annotations created by the Gazetteer.
- *Options*: it indicates the kind of matching style, which defines how we deal with annotations that overlap, or where multiple matches are possible for a particular sequence. There are 5 options:
 - *brill*: if one or more rules identify a match in the same portion of the document, they are performed all together. More annotations on the same piece of text can be identified. All the rules are executed starting from the same position and the next matching will start from the position in which the longest match ends;
 - *all*: it is similar to *brill*, but the rule continues the search of a match from the

- next offset with respect to the currently found one;
- first: when a rule finds a match, it is activated regardless of a possible longest match;
- once: when a rule has been activated, the entire phase ends after the first match;
- appelt: only one rule can be activated in the same part of the text, according to precise rules of priority:
 - * among all the rules that have a match with the identical initial position, it will be activated only the one that corresponds to the longest match;
 - * if one or more rules have a match on the same portion of the document, it will be activated the one with highest priority;
 - * if there is more than one rule with the same priority, it will be activated the rule that was defined earlier.

From line 4 to the end we have the description of the actions in case of matching:

- *Rule*: the fourth line indicates the name of the first rule, the presence of more than one rule in a single phase is allowed but it is good to pay attention to the order in which they are written and performed, to prevent unexpected results caused by the control options.
- *Priority (optional)*: the fifth line indicates the priority of a rule. The user can declare an optional parameter of priority associated to each rule that is usually a positive integer. A higher number corresponds to a higher priority. If the priority is not declared, by default all rules have priority -1 .
- *LHS (Left Hand Side)*: it is everything before the arrow and it consists in the description of the pattern that the rule must follow to find the match. In the example, it corresponds to the sixth row.
- *RHS (Right Hand Side)*: it describes the actions to be performed on the annotations when there is a matching with the pattern defined by LHS. In the example, it corresponds to the eighth row. Information about the text span is transferred from the LHS of the rule using a label, and it is annotated with the entity type (which follows it). Finally, attributes and their corresponding values are added to the annotation. Alternatively, the RHS of the rule can contain Java code to create or manipulate annotations.

The phase shown in the example behaves as it follows: each time “University of <name of a city>” is found, e.g., “University of Bari”, an annotation of type *organization* is created

over the whole expression, and this new annotation has two attributes, namely *rule* with value *University1* (to indicate the rule producing it) and *kind* with value *university*.

At the end of this phase, through the use of specific JAPE rules, we can have new annotations, such as *Lawyer* depicted in Figure 3.6.

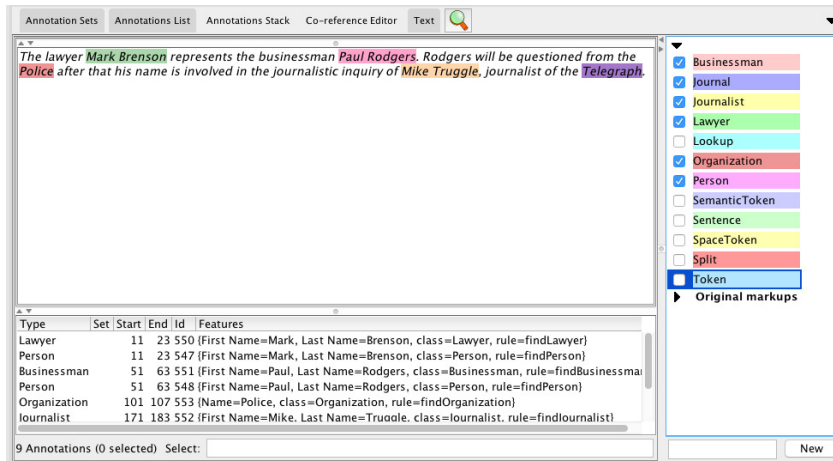


Figure 3.6. An example output of our JAPE Transducer (Semantic) rules.

3.2.2 Ontology population

In the ontology population phase of our approach, all the previous PRs are created in order to use the *OwlExporter*¹⁰ component; it can populate OWL ontologies by using the Protegé-OWL libraries. The main features of the OwlExporter are:

- *Exporting individuals*: creating OWL individuals using entities of a corpus;
- *Exporting dataproperties*: creating OWL dataproperties using information from a corpus;
- *Exporting objectproperties*: creating OWL objectproperties among OWL individuals by exploiting information obtained from a corpus;
- *Exporting coreference chains*: creating coreference chains using `owl:sameAs` for entities that re-appear in different parts of a corpus.

The three PRs that compose the ontology population phase are the following:

1. *JAPE Transducer (MuNPEX)*¹¹: this component is implemented in JAPE, and is used for (multi-lingual) noun phrase (NP) extraction, i.e., identification of elements in a sentence

¹⁰<http://www.semanticsoftware.info/owlexporter>

¹¹<http://www.semanticsoftware.info/munpex>

having a noun as head word, which is the word determining the syntactic function of the phrase. It is needed for managing the natural language processing specific for the OwlExporter, as it will be further clarified in the case study. MuNPEX requires a POS tagger to work and can additionally use detected named entities to improve chunking performance [209]. Currently the supported languages are English, German, and French, with additional Spanish support in beta. Thus we had to adapt it for the Italian language. For each detected NP, an annotation *NP* is added to the document, as depicted in Figure 3.7.

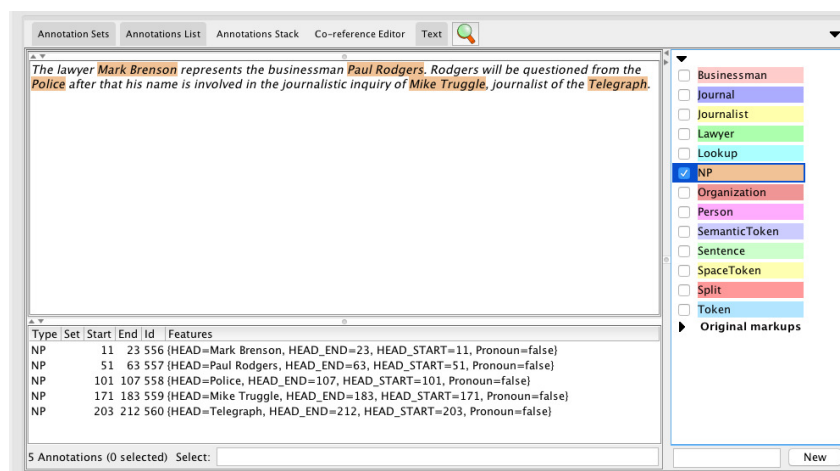


Figure 3.7. An example output of JAPE Transducer (MuNPEX).

2. *JAPE Transducer (Mapping)*: the annotations created by the semantic annotation phase are not in a compatible format to be an input to OwlExporter. It is necessary to convert them through JAPE rules; the user needs to create the following two new annotations declaring the mapping among the NLP annotations (created during processing), the external NLP and domain ontologies (created by an ontology engineer) [211]:

- *OwlExportClass*: this annotation records which document annotations need to be exported, and to which ontology class;
- *OwlExportRelation*: this annotation defines the export of roles between entities, which are recorded using OWL objectproperties.

At the end of this phase, there will be two new annotations: *OwlExportClassDomain* and *OwlExportRelationDomain*, as depicted in Figure 3.8.

3. *OwlExporter*: this is a component that can be included as part of a GATE pipeline. It allows to export document annotations created by the previous PRs to individuals

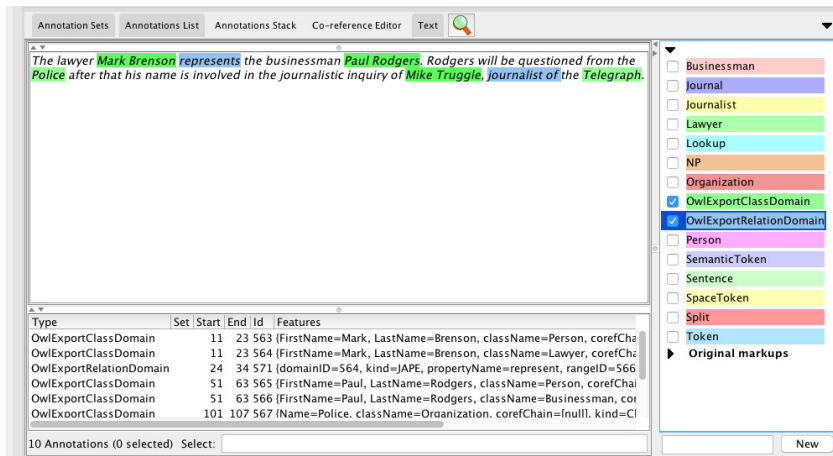


Figure 3.8. An example output of JAPE Transducer (Mapping).

in OWL. The OwlExporter manages two ontologies, a domain-specific and a domain-independent one. This allows to link domain-specific entities detected in a text to their lexical representation, e.g., paragraphs, sentences, or noun phrases. The first ontology is a domain specific ontology that models concepts and relationships that are relevant for the given domain; the latter ontology is a domain independent NLP ontology that contains concepts commonly used in language engineering [210]. When OwlExporter terminates, we obtain the population of the reference ontology with the information obtained from the analysed text documents.

3.3 Case Study: design and development

In this section, we describe our case study, focusing in particular on the design and development phases.

Our solution has been tested in a project in which we considered the domain of “Mafia Capitale”, from the name of a judicial inquiry that involved the city council of Rome in 2015, according to which, alleged crime syndicates misappropriated money destined for city services. This investigation received a lot of attention by the media, thus it allowed us to retrieve a large number of newspaper articles. The information retrievable was however completely unstructured and difficult to analyse in an automatic way. At the same time, manually processing it was definitely prohibitive.

We started from the Web crawling phase, from the definition of an ontology to represent some concepts of interest related to our domain, and we created specific Gazetteer lists and JAPE rules used for document annotation by GATE PRs described in Section 3.2. Many

technical details on all the phases of the case study, very useful for practitioners and for the reproducibility of the whole case study, are provided online at <https://tinyurl.com/Ontology-population-via-GATE>, where there are an online appendix with all configuration instructions, the Gazetteer, the JAPE rules, the ontology and the crawled documents, i.e., all the artefacts needed for re-creating the case study and the experimental results. This initial pipeline is referred in the following as OS/HK – open-source/human-knowledge, to mean that the needed artefacts are produced manually, based on human knowledge by also possibly exploiting publicly (open) available information sources (papers, blogs, etc.), especially when creating the Gazetteer lists. We will also experiment a further pipeline in which Gazetteer lists are built semi-automatically by using Wikidata (cf. Section 3.5).

We remark that documents that we extracted from the Web are all written in the Italian language, and thus we could not resort either to the many English tailored Gazetteer lists available in the literature, or to the JAPE rules commonly used for the English language. Instead, we had to define new Gazetteer lists and sets of JAPE rules specifically for the present project. At the same time, however, these outcomes are reusable resources, that is, they are general enough to be exploited also in other domains where the text to be annotated is in Italian. The interested reader and practitioner can find them in the online appendix mentioned above. We now overview the main phases of our project.

3.3.1 Crawling Phase

Initially, we have chosen different Web sources to retrieve articles from online newspapers as raw data. We had to carry out crawling operations on newspaper websites, for a period ranging from 16 June 2015 to 29 February 2016. We made use of Web Content Extractor ¹², a software that is able to crawl web sites also in the presence of robot.txt files; we have configured it accurately through various settings, like the initial seed, which is the URL from which the crawling action starts, and the depth of research. Through these two parameters this crawling software is able to create an SQL table with the following columns:

- *ID*, which is the number of the entry
- *TITLE*, which is the title text of the article
- *ARTICLE*, which is the content of the article
- *URL*, which is the URL where to find the article

Once this table is populated, it is possible to export the extracted text through SQL queries in XML format, which is the format accepted by GATE.

¹²<http://www.newprosoft.com/web-content-extractor.htm>

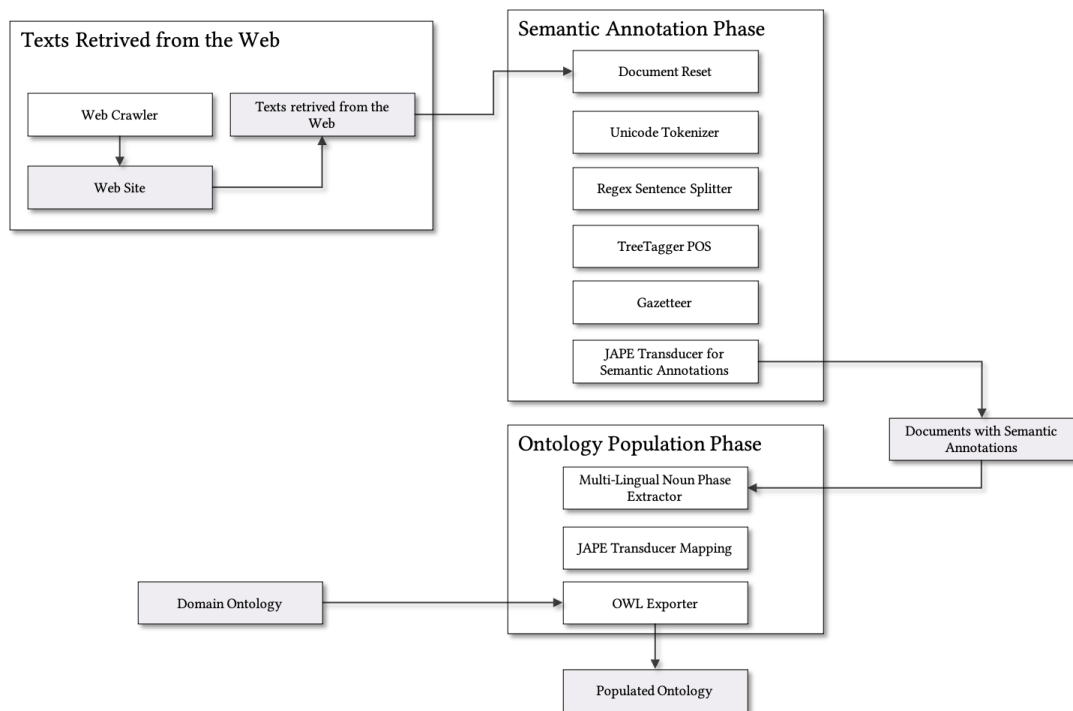


Figure 3.9. The description of our Pipeline.

To obtain articles related to “Mafia Capitale”, for every newspaper web site we selected, we set the initial seed to the result of the keyword search we executed over the newspaper web site using the words “Mafia Capitale”. For example, for the web site of “La Repubblica”, a major Italian newspaper, we have set the initial seed to “*http://ricerca.repubblica.it/ricerca/repubblica?query = Mafia + Capitale*”. The crawling phase has generated, in about 3 hours, the amount of 2657 articles, distributed as indicated in the following Table 3.1:

Newspaper	Number of articles
La Repubblica	938
Il Messaggero	777
Libero	842
Il Fatto Quotidiano	100

Table 3.1. Number of documents in the case study

3.3.2 Domain Ontology

The (intensional level of the) ontology is depicted in Figure 3.10, through the GRAPHOL ontology language – as anticipated in Section 2.1.2, it provides a visual representation for OWL ontologies.

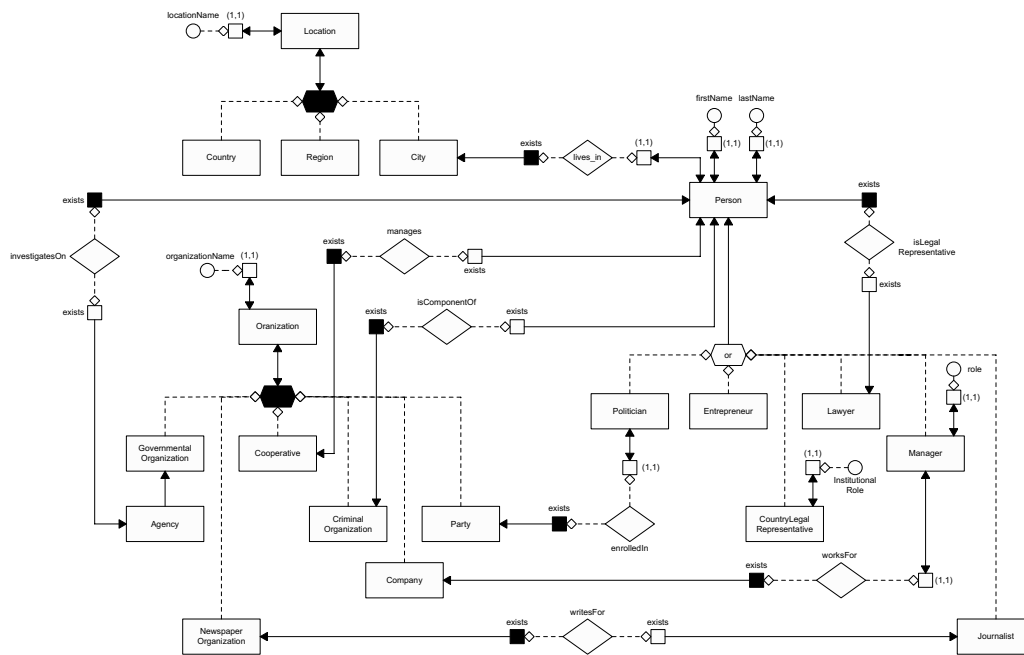


Figure 3.10. The reference ontology in the case study

The ontology conveys that Politician, Entrepreneur, Lawyer, Manager, CountryLegalRepresentative, and Journalist are non-disjoint sub-classes of Person. They inherit `firstName` and `lastName` from Person, as well as the fact that each instance `livesIn` in a City. All such properties are mandatory and functional, as indicated by the (1,1) labels in the diagram. In addition, each Manager has also a role (exactly one), and a CountryLegalRepresentative has an institutionalRole (exactly one). A Location can be a Country, Region, or City. Each Location has necessarily a name. Organizations are instead partitioned into GovernmentalOrganizations, Cooperatives, CriminalOrganizations, NewspaperOrganizations, Company(ie)s, and Party(ie)s. An Agency is a special type of GovernmentalOrganization. Each Organization has exactly one name. An Agency can `investigatesOn` Persons, whereas a Person can `manage` Cooperatives, or can be a component of (cf. `isComponentOf`) a CriminalOrganization. A Politician is `enrolledIn` exactly one Party, a Manager `worksFor` exactly one Company and a Journalist `writesFor` NewspaperOrganizations. All objectproperties are typed in both the domain and range (indicated by the arrows denoting inclusions from the domain or the range of the property to the class it is typed on), which means that no individuals other than those among the instances of the typing classes can instantiate the domain and the range of the objectproperty. Similarly for the domain of dataproperties (whereas for simplicity we do not indicate the typing of their ranges).

3.3.3 Gazetteer

As discussed in Section 3.2, the Gazetteer PR annotates the documents on the basis of a set of lists containing names of entities. These lists are used to find occurrences of these names in text (e.g., for the task of NER). In our project, it was crucial to create all the Gazetteer lists used for document annotation, because all the articles we analyzed are written in Italian, and when we started our project no reusable support for Gazetteer lists for Italian was available.

To create the lists related to the domain entities of “Mafia Capitale”, we used two methods:

- based on *human knowledge*: this approach is used to write the lists containing words that identify certain categories to which entities belong, e.g., containing all the words identifying lawyers. In order to create this list, we have read some articles from newspapers about the domain of interest, to better understand the terms identifying a specific category, as “Lawyer”, “Advocate”, “Barrister” and “Attorney” for the category of “Lawyer”. Then, we wrote the terms in the Gazetteer lists in both uppercase and lowercase letter, as GATE has a case sensitive approach.
- based on *open data sources*: this approach is used to find the first and last name of individuals belonging to a specific category, e.g., the members of the City Council of Rome, with the information of their role as well. To this aim, we have downloaded specific lists of interest from open data sites, e.g., OpenPolitici¹³ for the lists of Politicians currently in charge, and also by other data sources such as DBpedia, etc.

The construction of such lists was not trivial and required a significant amount of time to obtain good results; in our case, the effort is quantifiable in 7 person-days. This takes into account a first phase of search of the information on the Web and a subsequent cleaning, which has led to a translation and transformation of the extracted resources into the format accepted by GATE for the of Gazetteer. The value of 7 person-days can be subject to important variations as it depends strictly on both the language and the domain of interest. A richer ontology in a less widely spoken language could lead to a greater search effort and also to a greater manual writing of resources.

3.3.4 JAPE Rules in the Semantic Annotation Phase

The JAPE Transducer completes the semantic annotation phase and it is used to import the user-written JAPE rules into the GATE platform, so they can be used to create or modify text annotations on documents. JAPE allows users to recognise regular expressions in annotations on documents by using the annotations previously produced from other PRs.

¹³<http://politici.openpolis.it/>

Next we describe the process of building annotations through JAPE rules. We have created many JAPE rules to annotate different entities based on the respective classes of the reference ontology. As an example, in the following we discuss this process for the *Lawyer* annotation. It has *First Name* and *Last Name* as attributes in the ontology, so we need them when we annotate a *Lawyer* entity in text documents. In a nutshell, we go through the text and search for expressions (i) referring to a legal profession, and (ii) containing the first name and last name of a person qualified with the term under investigation; as an example, when we want to create an annotation *Lawyer* for the entity *Person* called “Carlo Taormina” given the following expression “L’avvocato Carlo Taormina ha difeso l’imputato davanti alla Corte.”¹⁴, the first step is to create a *Person* annotation identifying humans from the attributes *first name* and *last name*. We assume that, since it is very difficult to create a list of all possible surnames, we can not use a Gazetteer list to find such an annotation. Nevertheless, it is possible to write a Gazetteer list containing all the most common first names. Accordingly, we have created a Gazetteer list that creates a *Lookup* annotation when there is a matching between an entry in the *first name* list and a word in the analysed texts. Moreover, since we want to find the *first name* and *last name* to identify a *Person* entity, the first name only is obviously not enough; therefore, starting from the assumption that usually a first name is followed by a surname with a capital letter, we can adopt the annotation *Token* created by the Unicode Tokeniser that has an orth-attribute indicating if a word commences with a capital or lowercase letter. Our project is based on Italian language, so we distinguished six cases for the construct *first name* and *last name*:

1. FirstName Surname, as in Antonio Bianchi;
2. FirstName SecondName Surname, as in Stefano Andrea Bianchi;
3. FirstName FirstSurname SecondSurname, as in Dante Delli Priscoli;
4. FirstName SecondName FirstSurname SecondSurname, as in Marco Antonio Della Valle;
5. FirstName FirstSurname ’ SecondSurname, as in Riccardo Dell’Aquila (note the apostrophe separating the two parts of the surname);
6. FirstName SecondName FirstSurname ’ SecondSurname, as in Alessandro Maria Dell’Acqua.

Now we have *Person* annotations on the text and we can use them to find the humans that are lawyers, by searching the sentence with annotations *Person* near an expression that contains

¹⁴The translation of the sentence in English is “The lawyer Carlo Taormina has defended the accused in front of the Court.”

a reference to the legal profession. This annotation is produced by the Lawyer JAPE rules ¹⁵

```

Phase: Lawyer//it is the name of the Phase
Input: Token Lookup Person //they are the input annotation of the Phase
Options: control = Brill
//it indicates the kind of matching. In this case we use Brill: If one or more rules
    identify a match in the same portion of the document, they are performed all at
    once.

Rule:FindLawyerCaseOne
//it indicates the name of the first rule. In a Phase we can have many rules.
((({Lookup.minorType==lawyer}):jobOne// it is a word that is in Lawyer.lst(list of
    terms that referred to the lawyer profession) annotated by Gazetteer; it is a
    word that identify the lawyer job and we add a label jobOne on it
({Person}):personWorkOne) // it is an annotation Person composed by First Name and
    Surname found by the JAPE rule Person described above and we add a label
    personWorkOne on it
):personWork // it is the label of the entire regular expression that is composed by
    the construct (Lookup)(Person)
-->
//here the RHS part starts
{gate.FeatureMap features = Factory.newFeatureMap();
// this command is used to create a new type of Annotation
gate.AnnotationSet firstNameSet = (gate.AnnotationSet)bindings.get("personWorkOne");
// through this command we can take the collection of regular expression found in the
    text that matching the regular expression with label personWorkOne defined in LHS
gate.Annotation firstNameAnn = (gate.Annotation)firstNameSet.iterator().next();
// this command is used to take a single matching of person
features.putAll(firstNameAnn.getFeatures());
// this command is used to export all the attributes present in the matching
    annotation Person to the new Annotation Lawyer
features.put("class","Lawyer");
// attach the new attribute class to the new annotation created with value the string
    'Lawyer' that represents the specific concept of the reference ontology. This
    attribute is useful for the Ontology Population Phase
features.put("rule", "FindLawyerOne");
// attach a new attribute rule to the new annotation created with value the string
    'FindLawyer' that represents the name of the rule that finds the annotation
outputAS.add(firstNameSet.firstNode(), firstNameSet.lastNode(), "Lawyer",features);

```

¹⁵The Lawyer rule in JAPE is translated in English for the reader, but in our work the JAPE rules are written in Italian.

```

// create the new annotations Lawyer and their related attributes
}

Rule:FindLawyerCaseTwo// it indicates the name of the second rule

((({Person}):personWorkTwo
// it is an annotation Person composed by First Name and Surname found by the JAPE
  rule Person described above; we add a label personWorkTwo on it
({Token.string == ","}) // it is a comma that follows the first Person annotation
({Lookup.minorType==lawyer}):jobTwo)
// it is a word that is in Lawyer.lst(list of terms that referred to the lawyer
  profession) annotated by the Gazetteers; it is a word that identify the lawyer
  job; we add a label jobTwo on it
):personWork// it is the label of the entire regular expression that is composed by
  the construct (Person) , (Lookup)
-->
//the description of the RHS in this case is equal to the description of the RHS of
  FindLawyerCaseOne
{
gate.FeatureMap features = Factory.newFeatureMap();
gate.AnnotationSet firstNameSetDue =
  (gate.AnnotationSet)bindings.get("personWorkTwo");
gate.Annotation firstNameAnnDue = (gate.Annotation)firstNameSetDue.iterator().next();
features.putAll(firstNameAnnDue.getFeatures());
features.put("class", "Lawyer");
features.put("rule", "FindLawyerTwo");
outputAS.add(firstNameSetDue.firstNode(), firstNameSetDue.lastNode(),
  "Lawyer",features);}

```

This rule can annotate as *Lawyer* the human occurring in this two types of sentence:

- *Lookup(with minor type equals to lawyer) Person*, as in “L’avvocato Carlo Taormina ha difeso l’imputato”¹⁶: the rule is activated, so we have an annotation *Lawyer* on the *Person* identified by Carlo Taormina.
- *Person, Lookup(with minor type equals to lawyer)*, as in “Carlo Taormina, avvocato di fama internazionale”¹⁷: the rule is activated, so we have an annotation *Lawyer* on the *Person* identified by Carlo Taormina.

We decided to use a short range between the annotation *Person* and *Lookup* on the basis of

¹⁶The translation of the sentence in English is “The lawyer Carlo Taormina has defended the accused”.

¹⁷The translation of the sentence in English is “Carlo Taormina, the internationally famous lawyer”.

an analysis on text documents; in particular, the following LHS part of the rule shows that the range can be maximum two *Tokens*:

```

({Lookup.minorType==lawyer}):jobOne
// it is a word that is in Lawyer.lst (list of terms that refers to the lawyer
  profession) annotated by Gazetteer; it is a word that identifies the lawyer job;
  we add a label jobOne on it
(({Token})?) // a generic token that can be present
(({Token})?)// a generic token that can be present
({Person}):personalavorauno
// it is an annotation Persona composed by Name and Surname found by the JAPE rule
  Person described above; we add a label personWorkOne on it

```

Finally, we want to annotate *Lawyer* in a special case where the category lawyer is referred to two persons. Let us consider the sentence “Gli avvocati Carlo Taormina e Mario Rossi sono appena entrati in aula”¹⁸. This states that both Carlo Taormina and Mario Rossi are lawyers, but there is only one word referring to the legal profession carried out by both the individuals. In order to put a correct *Lawyer* annotation on Carlo Taormina and Mario Rossi, the following DoubleLawyer JAPE rule needs to be adopted and produces a matching if there is a plural word referring to the legal profession followed by two annotations Person separated by the conjunction “e” (“and” in English):

```

//we show only the Regular expression to identify this special case of Lawyer
Rule: AvvocatoTipoDouble
(
  ({Lookup.minorType==lawyerMultiple}):jobOne
  // it is a word that is in Lawyers.lst (list of terms that referred to the lawyer
    profession in plural case) annotated by Gazetteer
  ({Person}):lawyerOne
  (({Token.kind == word,Token.string == "e"}))
  ({Person}):lawyerTwo
)
):personJob // the regular expressions finds the following case (Lookup) (Person) "e"
  (Person)

```

¹⁸The translation of the sentence in English is “The lawyers Carlo Taormina and Mario Rossi just arrived in the courtroom”.

3.3.5 Multi-Lingual Noun Phrase Extractor (MuNPEX)

We have introduced in Section 3.2 MuNPEX as a multi-lingual noun phrase extraction component. Currently, this component does not support the Italian language. We have implemented an Italian support for MuNPEX based on the previously described TreeTagger POS. To this aim, we have modified the standard classes of the MuNPEX French version to adapt them to TreeTagger parameter files used for the Italian language. For each detected noun phrase, an annotation “NP” is added to the document, which includes several features [209]:

- *DET*, the determiner of the NP;
- *MOD*, a list of modifiers of the NP;
- *HEAD*, the head noun of the NP;
- *Pronoun*, boolean value (true, false) indicating a pronoun NP;
- *MOD2*, NP modifiers that appear after the HEAD noun.

MuNPEX contains language-specific and language-independent files and it is implemented as a set of grammars running in a multi-phase JAPE Transducer. For our Italian version, we have implemented the file `it-np main.jape` that contains the transducer definition with five phases:

- `it-np-parts.jape`, which contains the language-specific definitions for the determiner, modifier, and head slots;
- `np-entities.jape`, which is a language-independent file defining which named entities to use for the HEAD slot of an NP;
- `check.jape`, which is a language-independent file that handles the special cases of differences between the input and output of the transducer;
- `np.jape`, which is a language-independent file that constructs annotations from the constituents detected in the previous phases;
- `clean.jape`, which cleans up temporary annotations.

3.4 Case Study: tests and results

Here, we describe the evaluation of the approach previously described in Sections 3.2 and 3.3. To test our techniques, we have used an amounts of 2657 articles generated by the crawling phase. We evaluated the approach on two different types of test:

1. In the first one, we have chosen 10 documents, in a random way among those produced by the crawling phase, and have asked a domain expert to annotate them, according to the domain ontology we specified. The manual annotations aimed at identifying all and only the instances of the ontology predicates that can be extracted by the documents (on the basis of the document content and the knowledge of the expert). We have then processed these documents through our GATE-based pipeline, and we have compared the annotations produced by our pipeline with the manual annotations of the domain expert.
2. In the second one, we have processed in our pipeline all the 2657 articles obtained by the crawling phase. Due to the large number of documents, obviously no previous reference annotation could be done, and also checking Precision for all annotations has been impossible. We thus used this test to measure the total number of class, objectproperty and dataproperty membership assertions, produced and to verify the Precision on a portion of the analyzed documents.

3.4.1 Performance Evaluation

The results of the above tests are shown in Table 3.2 and in Table 3.3, respectively, where *Words* indicate the number of words contained in the documents, and *CA*, *OPA* and *DPA* respectively denote the number of class, objectproperty, and dataproperty assertions extracted by our pipeline, *S. Ann.* and *M. Ann.* is the number of spurious annotations and missing annotations, respectively. In Table 3.3, the execution time represents the running time of the entire pipeline in order to process 2657 documents. Both the experiments were performed on a MacBook Pro 9.2, with Intel Core i7 2,90 GHz and 8GB RAM.

Words	Total Annotations	S. Ann.	M. Ann.	Precision	Recall	F-measure
6.084	414	11	98	97,3	78,7	87%

Table 3.2. Results of test #1

Words	CA	OPA	DPA	Exec.time (sec.)
1.285.290	38.452	419	99.145	1.948,88 (~30 mins)

Table 3.3. Results of test #2

3.4.2 Discussion

We soon notice excellent Precision and good Recall, a very high number of dataproperty assertions added to the ontology, a good number of class assertions, but few objectproperty assertions.

As for the Precision, we point out that the errors we obtained were mainly due to the wrong annotations associated to the word “Marino”, which is both a city (close to Rome) and the last name of a Rome ex-mayor¹⁹. Indeed, in our pipeline, disambiguation is done through the JAPE rules we defined, which however are able to identify the correct annotation only when the last name is coupled in the text with the first name of the mayor, which is often not the case in the selected documents.

The above problem can be mitigated by adding to our pipeline two additional PRs, namely the *Pronominal Coreference* and the *OrtoMatcher* (we in particular expect the latter to be able to solve the previous issue). We observe that the use of such PRs would allow us to also substantially increase the number of objectproperty assertions retrieved, as the example discussed in the following show.

- *Pronominal Coreference (PC)*: this PR provides an annotation on pronouns that refer to an already annotated entity. For example, in the sentence “Marco Rossi is a successful lawyer. He lives in Milan”, our current solution recognizes that an individual named Marco Rossi is a lawyer, and also that Milan denotes a city, but it is not able to understand that “He” refers to the individual previously annotated and to extract the objectproperty assertion representing the fact that Marco Rossi lives in Milan. By using the PC we can instead obtain this missing annotation. Currently GATE provides this PR only for the English language, and thus we could not directly introduce it in our pipeline.
- *OrtoMatcher*: it provides an annotation on an entity that is indeed denoted in the text with an abbreviation of an already annotated expression. For instance, in the sentence “Marco Rossi is a successful lawyer. Rossi lives in Milan”, our current solution is not able to understand that Marco Rossi lives in Milan, since the full name is abbreviated into the last name only. OrtoMatcher would be able instead to find this annotation. However, the OrtoMatcher provided by GATE needs a quite exhaustive list of abbreviations for each non-abbreviated denotation of an entity, similar to Gazetteer lists. This approach is clearly impossible to pursue for cases with very many possible abbreviations (as for a person). In the current release of our approach, this resource is actually used only for the most common abbreviations for the names of some famous organizations in the world.

We point out that the insertion in our pipeline of the above resources would also allow to augment the recall in our experiments, since its current value is mainly due to missed objectproperty assertions.

¹⁹Ignazio Marino was the mayor of Rome at the time of the investigation, and therefore his name appears many times in the news articles used for our case study.

In the second test, we were not aiming at measuring Precision and Recall for all documents, but our main goal was to evaluate the impact of our approach on a large text corpus, and get an idea of the computation time it requires on real-world cases. Nonetheless, to get an idea of the quality of the result, we measured the Precision on the 1% of the data in the output and we have got a value of 94%.

3.5 Simplifying Gazetteer Lists Generation: design and development

As explained in Section 3.3.3, the construction of Gazetteer lists is in general non-trivial and requires a significant amount of time to obtain good quality results.

We thus investigated alternative methods for Gazetteer lists generation that might reduce the required manual effort, and might be more easily replicable in different domains, thus augmenting the generality of our approach. To this aim, we considered the knowledge base Wikidata as source for Gazetteer lists production.

The aim of presenting this variant is twofold: *(i)* to demonstrate, on the one side, how the pipeline we have built is flexible and different phases can be modified and incrementally refined in order to improve them; and, on the other side, *(ii)* to show how semantic technologies can be used also internally to the pipeline in order to improve some phases.

Wikidata is a collaboratively edited *Knowledge Graph*, which organizes a large amount of data in a structured way according to a general reference ontology. It is an openly accessible resource, following the Semantic Web standards for exporting, interconnecting and querying data, which can be edited and read by both machines and humans.

In Wikidata every resource has its own URI based on the following convention. Classes and individuals are identified by the Wikidata prefix `:wd` followed by ‘Q’ and a sequence of numbers (for example `wd:Q47729` identifies the *democratic party*)²⁰. For property identifiers (both object and data properties) the pattern is similar to that of classes and instances except that the prefix is `:wdt` and ‘P’ is used instead of ‘Q’ (for example `wd:P17` identifies the the property *country of belonging*).

Our idea has been therefore to exploit such resource to partially automate the process of Gazetteer lists generation.

We have thus downloaded a Wikidata RDF dump (specifically we used the version of March 3, 2017), and have loaded it on a graph database management system with RDF/SPARQL

²⁰The use of a common identifier for both classes and instances is due to the meta-modelling of the knowledge base [148].

support, namely Blazegraph²¹. This has enabled us to access Wikidata information through standard SPARQL queries.

Through this approach, it is possible to avoid tedious and time-consuming development of ad-hoc solutions for each required Gazetteer list. Indeed, once the setup of the system is completed, a user just needs to define and execute a set of SPARQL queries to obtain the lists of interest.

For example, to obtain the list of the names of Italian political parties, it is possible to use the property `wdt:P31`, which represents *instance of* (corresponding to the predicate *rdf:type*, typically abbreviated into *a*), and the property `wdt:P279`, which represents *subclass of* (corresponding to the predicate *rdfs:subclassOf*), and to reference the class `wd:Q7278`, which represents *political party*. Finally we need to filter this statement through the property `wdt:P17`, which represents *country of belonging*, referencing the class `wd:Q7278`, which represents *Italy*. In the following, we can observe the exact structure of the SPARQL query that returns the names of Italian political parties from Wikidata:

```
SELECT ?element
WHERE {
    ?item wdt:P31/wdt:P279* wd:Q7278 .
    ?item wdt:P17 wd:Q38 .
    ?item rdfs:label ?element
}
```

The result of the SPARQL query is exported as CSV file, which has to be simply renamed, so that its extension is `.lst`, in order to be processed by the Gazetteer.

We conclude this section by providing a qualitative evaluation of the effort required by the Wikidata approach. In Table 3.4 we give the costs in person-days for the various tasks we went through in our experiment.

Task	Person-days
System setup	0.5
Wikidata ontology comprehension and analysis	0.5
Query specification	0.5
Query execution	0.3

Table 3.4. Person effort for the generation of the Wikidata-based Gazetteer lists

The total effort in this case has been of 1,8 person-days, to be compared with the 7 person-days of the OS/HK approach.

²¹<https://www.blazegraph.com/>

The task of system setup can be avoided using the SPARQL online endpoint²² provided by the Wikidata Query Service, if the queries to be executed do not exceed the execution time limit imposed by the system. This also allows to always use the most up-to-date data. Moreover, once the practitioner has learned the most common relations used within Wikidata and has understood the structures of the portion of interest, future applications of this approach will no longer involve an in-depth analysis of Wikidata ontology, thus allowing to save more effort/time. As for query specification, the time taken for the generation of our 12 queries, was 0,5 person-days. These queries are easy to write, and as can be seen from the previous example, they have an almost standard structure with variations only with respect to the Wikidata class. Their execution time is low, in fact the time to perform was about 0,3 person-days (a few hours).

In the next section, we test the quality of the lists obtained through the Wikidata approach with respect to the other ones defined in Section 3.3, comparing the quality of annotations produced through them.

3.6 Simplifying Gazetteer Lists Generation: Tests and Results

In this section, we compare the Wikidata approach for the generation of Gazetteer lists, described in Section 3.5, with the open source (OS) and human knowledge (HK) approach described in Section 3.3.

To this aim, we consider five lists containing words identifying *Politicians*, *Journalists*, *First Names*, *Criminal Organizations* and *Political Parties*, respectively. Each list has been produced and used in two versions, i.e., the OS/HK-based one and the Wikidata-based one. The comparison has been done on 15 articles, randomly selected among those returned by the crawling phase described in Section 3.3.1, and following the same criteria of the first test discussed in Section 3.4: a hand-based annotation has been performed by a domain expert; then, the articles have been annotated through the Gazetteer component of our pipeline by using the two different sets of lists generated with the two approaches; we have finally computed Precision, Recall and F-measure in the two cases, by using the manual annotation of the expert as ground truth.

We have checked the results produced by this experiment in two different points of our pipeline: just after the annotation done by the Gazetteer PR, and at the end of the entire pipeline. The first check (test #1) is aimed to highlight the contribution given to the semantic annotation by the Gazetteer PR, alimeted either with the OS/HK-based lists or with the Wikidata-based ones, and thus without the “adjustments” provided by the JAPE rules. Therefore the results obtained in this case are produced by a pipeline composed only by the following PRs: Document

²²<https://query.wikidata.org/>

Reset, Unicode Tokeniser, RegEx Sentence Splitter, TreeTagger POS and Gazetteer. Such results are given in Table 3.5. Notice that this “reduced” pipeline does not actually produce OWL assertions corresponding to the instances of the ontology, but returns documents annotated by the PRs that have been used in the pipeline (and in particular containing the annotations obtained through the Gazetteer lists). Thus, Precision, Recall and F-measure are computed with respect to the corresponding annotations done by the domain expert. In this table, the columns “Total Ann.”, “S. Ann.”, and “M. Ann.” refer to the overall number of annotations produced by our pipeline, the number of false positives, and the number false negatives, respectively.

In the second check (test #2) we instead look at the final results obtained after both the semantic annotation and the ontology population phases of our pipeline. In this case we aim to measure the overall impact of substituting in our entire extraction process the OS/HK-based Gazetteer lists with the Wikidata-based ones. The results are given in Table 3.6, where Precision, Recall and F-measure are given with respect to the OWL assertions produced through the manual annotation of the domain expert. The columns “OWL Assertions”, “S. OWL Ass.” and “M. OWL Ass.” refer to the total number of OWL assertions returned by our pipeline, the number of false positives, and the number of false negatives, respectively. Notice also that OWL assertions we consider refer to instances of classes of the ontology. For example, the assertions considered in the first row are of the form (p a :*Person*), where p is an instance of the class *Person* (cf. Section 3.3.2) that the Gazetteer list *First Name Open Source*, together with the use of the JAPE rules (as described in Section 3.3), allowed to identify. Similarly, the assertions considered in other rows refer to the ontology classes *CriminalOrganization*, *Politician*, *Journalist*, and *Party*.

List	Entries	Total Ann.	S. Ann.	M. Ann.	Precision	Recall	F-measure
First Names Open Source	8913	322	83	0	74,2%	100%	85,2%
First Names Wikidata	2517	262	31	5	88,1%	97,8%	92,7%
Criminal Organizations Human Knowledge	25	37	21	0	43,2%	100%	60,3%
Criminal Organizations Wikidata	68	20	18	14	10%	12,5%	11%
Criminal Organizations Wikidata Mod.	273	30	24	10	20%	37,5%	26%
Politicians Open Source	1083	11	0	40	100%	21,6%	35,4%
Politicians Wikidata	7778	17	0	34	100%	33,3%	50%
Journalists Open Source	369	4	0	4	100%	50%	66,6%
Journalists Wikidata	3727	8	0	0	100%	100%	100%
Political Parties Open Source	131	23	0	2	100%	92%	95,8%
Political Parties Wikidata	447	4	0	21	100%	16%	27,6%
Political Parties Wikidata Mod.	1293	26	7	6	73%	76%	74,5%

Table 3.5. Results of test #1

We first discuss the results given in Table 3.5. We initially notice that the number of entries in the lists created with Wikidata is in total larger than the number of entries obtained through the OS/HK approach. In particular, if we do not consider the list containing Italian first names,

List	OWL Assertions	S. OWL Ass.	M. OWL Ass.	Precision	Recall	F-measure
First Names Open Source	313	5	59	98,4%	83,9%	90,5%
First Names Wikidata	305	4	66	98,6%	82,0	89,5%
Criminal Organizations Human Knowledge	16	0	0	100%	100%	100%
Criminal Organizations Wikidata	2	0	14	100%	2,5%	22,2%
Criminal Organizations Wikidata Mod.	9	1	8	88,8%	50%	64,0%
Politicians Open Source	22	0	29	100%	43,1%	60,2%
Politicians Wikidata	23	0	28	100%	45,1%	62,2%
Journalists Open Source	4	0	4	100%	50%	66,6%
Journalists Wikidata	8	0	0	100%	100%	100%
Political Parties open source	23	0	2	100%	92%	95,8%
Political Parties Wikidata	4	0	21	100%	16%	27,6%
Political Parties Wikidata Mod.	26	7	6	73%	76%	74,5%

Table 3.6. Results of test #2

which is the only one for which the OS/HK approach outperforms Wikidata in terms of amount of returned entries, this number is on average 9 times larger for the Wikidata lists. On the other hand, the Wikidata-based lists contain various entries with typos and flaws. For example, the *Political Parties Wikidata* list contains the entry ““PSDI”” (with wrong additional quotation marks), which does not allow to correctly annotate the word PSDI (which is an acronym of an Italian political party). The presence of more errors in the Wikidata-based lists was somehow expected. Our experiments however show that the Precision value for annotations with the Wikidata lists is the same or better than the Precision obtained with OS/HK lists in all cases but the Criminal Organizations one. We point out that this list in the OS/HK approach has been accurately produced through a manual process based on human knowledge, since, before putting Wikidata into the loop, we could not find any open data source providing such information in the form of a dictionary list. Nonetheless we could not avoid some errors in annotations, since the names of such organizations are inherently ambiguous and may easily lead to production of false positives. In this respect, we could not expect this problem to be mitigated by using a list automatically extracted by Wikidata.

For several Wikidata lists, also the Recall is close to the one measured for the corresponding OS/HK lists. For the cases of *Criminal Organizations* and *Political Parties* we instead got initially very low values for the Recall. This was mainly due to problems with uppercase/lowercase letters, since GATE is case sensitive. We thus refined the two lists by adding for each entry the three versions: all uppercase letters, all lowercase letters, and all capitalized words. In Table 3.5 and Table 3.6 these new lists are denoted as *Wikidata Mod.* (modified).

We can observe that with this fix the value of Recall for both the lists has become greater than that of the original Wikidata lists. However, for the *Political Party Wikidata Mod.* list, this approach has caused a decrease in Precision. This is due to the introduction of new entries that have more than one meaning, such as the word *si* (i.e., the lowercase version of SI, which is the acronym of an Italian political party). This word is indeed used in Italian as reflexive

pronoun, but with the use of the modified lists it has been sometime wrongly annotated as a political party.

We remark that the *Political Party Wikidata list* (i.e., the non-modified one) does not contain the all lowercase version of the entries, and thus this problem did not arise by using the original version of this list. Despite the decrease of Precision, we notice that the modification of this list produced an increment of the Recall, which reached the 79%, and of the F-measure, which changed from 26,7% to 74,5%, that thus largely compensates the loss of Precision.

Wrapping up our discussion on test #1, we can say that the overall results obtained with the Wikidata approach are quite good, as highlighted in Figure 3.11, where for the sake of presentation we show the values of the F-measure in the various cases in the form of a bar graph.

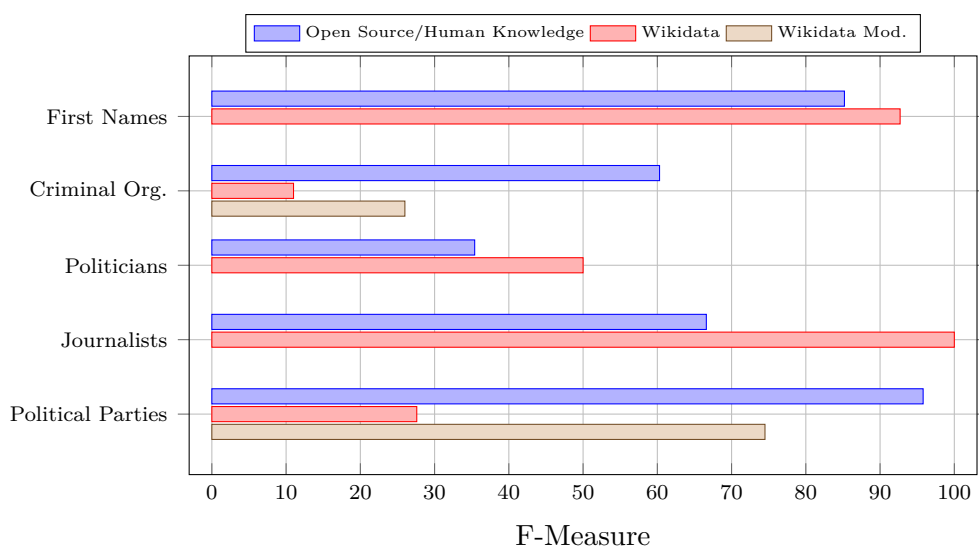


Figure 3.11. F-Measures for test #1

The question now arises about which is the effect of executing our entire pipeline (that in particular means introducing in the annotation process the contribution of the JAPE rules we have specified). We thus now focus on the results given in Table 3.6. As done for the previous test, in Figure 3.12 we also show as a bar graph the F-measures given in this table. By comparing the F-measures of the two tests, we soon notice that there is an improvement for the *First Names*, *Criminal Organizations* and *Politicians* lists, both in the OS/HK-based and in the Wikidata-based approach, whereas for the other lists Precision, Recall, and thus F-measure remain the same as before. This confirms that the use of JAPE rules in general improve the quality of the extracted information. In particular, for all the versions of the *Criminal Organizations* list we notice a great increment in Precision, which testifies that in this case the JAPE rules have primarily contributed to eliminate many of the spurious annotations introduced by the Gazetteer PR. A similar behaviour can be seen for both versions of the *First*

Names lists. Instead, for the two versions of the *Politicians* list, the gain we get is on the Recall (being the Precision the same as in Table 3.5), which means that in this case the JAPE rules have helped to identify relevant cases that were not captured in the previous test. We also notice that the improvement is almost always more substantial for the OS/HK-based approach. This is not surprising, since to some extent the JAPE rules have been specified to work on the annotations provided by these Gazetteer lists. Nonetheless, these JAPE rules contribute to improve our results also for the case of Wikidata-based Gazetteer lists. Indeed, even if the OS/HK approach in test #2 outperforms the Wikidata approach in 3 cases over 5 (thus at a first glance overturning the result of test #1), in the overall the two approaches produce in fact results of very similar quality, if we also take into account the number of annotations produced through each single Gazetteer list. This can be seen by computing the weighted average of the F-measures in the two cases, where the weight for the F-measure of each list is the fraction of the number of annotations obtained for this list over the total number of annotations, which returns 89,3% for the OS/HK approach and 86,4% for the Wikidata approach. Formally, the weighted average of the F-measures is:

$$\frac{\sum_{k=1}^n x_i \cdot w_i}{\sum_{k=1}^n w_i}$$

where each x_i is the F-measure of the i -th list, and its weight w_i is equal to $\frac{a_i}{\sum_{i=1}^n a_i}$, with a_i the number of annotations produced for the i -th Gazetteer list. For the sake of completeness, we show the various x_i , w_i , and the weighted F-measure in Table 3.7 and Table 3.8.

List	x_i	w_i	Wheighted F-Measure
First Names Open Source	90.5%	0.82	75%
Criminal Organizations Human Knowledge	100%	0.042	4.23%
Politicians Open Source	60.2%	0.058	3.5%
Journalists Open Source	66.6%	0.01	0.7%
Political Parties Open Source	95.8%	0.06	5.8%
Total			89.2%

Table 3.7. The weighted average of the F-measures for the OS/HK approach

We conclude this section by having a look at the OWL dataproperty assertions returned by our entire pipeline in the two approaches considered in this section. We first notice that the OWLExporter component often introduces several individuals (i.e., identifiers given in the form of IRIs²³) to denote the same entity. In fact, the OWLExporter reconciles these individuals through `owl:sameAs` assertions, which say that two different URIs are “equivalent”. We can

²³<https://www.w3.org/TR/owl2-syntax/>

List	x_i	w_i	Wheighted F-Measure
First Names Wikidata	89.5%	0.82	73.6%
Criminal Organizations Wikidata	64.0%	0.01	1.55%
Politicians Wikidata	62.2%	0.03	3.85%
Journalists Wikidata	100%	0.02	2.15%
Political Parties Wikidata	74.5%	0.07	5.22%
Total			86.4%

Table 3.8. The weighted average of the F-measures for the Wikidata approach

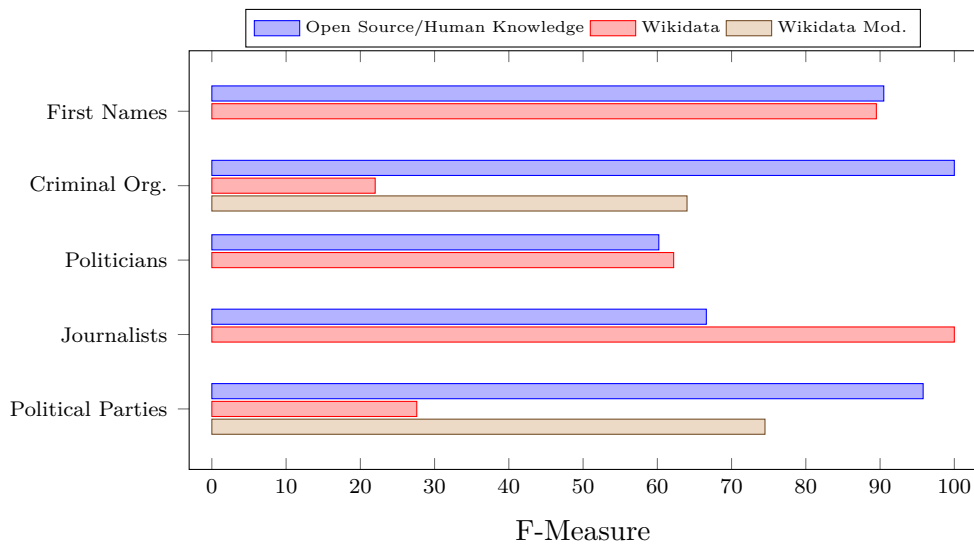


Figure 3.12. F-Measures for test #2

thus interpret them with the same object of the domain. For example, the OWL assertion ($A \text{ owl:sameAs } B$) is indeed saying that A and B are two different identifiers for the same object (intuitively, they act as synonyms). As a consequence, every property of A is clearly also a property of B . Coming back to the OWL dataproperty assertions returned by our pipeline, we notice that under the adoption of OS/HK-based Gazetteer lists we obtain a larger number of such assertions with respect to those obtained by making use of the Wikidata-based Gazetteer lists. This however does not really results in a loss of this kind of assertions. Indeed, in the first approach data properties are in fact returned in a “closed form with respect to the `owl:sameAs` assertions”, i.e., in the resulting OWL file a data property for an individual A (e.g., ($A \text{ :name "Ignazio"}$)) is also explicitly associated to all the individuals related to A through `owl:sameAs` (e.g., also ($B \text{ :name "Ignazio"}$) is present in the OWL file). Instead, this does not happen when we use Wikidata-based Gazetteer lists. This is mainly due to the way in which these lists are written down in files and the way in which JAPE rules use such annotations (remember that we have designed the JAPE rules primarily to couple them with the OS/HK-based Gazetteer

lists).

To verify the above behaviour, and to show that it is indeed possible and easy to recover dataproperty annotations missing from the OWL file produced by the pipeline equipped with the Wikidata-based Gazetteer list, we resorted to the use of an OWL reasoner, and used it to generate the dataproperty assertions that were implicit in the OWL file (i.e., not asserted but implied by the ontology according to the `owl:sameAs` relation). We did this by considering only the two Gazetteer lists *Politicians Wikidata* and *Journalists Wikidata*, since are those for which we measured the most substantial (apparent) loss of dataproperties. The results of this further test are given in Table 3.6, where “DPA”, “M. DPA”, and “DPAR” refer to the dataproperty assertions returned by our pipeline, those missed by our pipeline, and those obtained after the use of the OWL reasoner. For the sake of completeness, we also report the number of Class assertions (“CA”) computed by the pipeline. As shown in the table, in both cases the reasoner allowed us to recover all the missing assertions. As OWL reasoner we used HermiT [101], through the plugin available for Protégé²⁴. To obtain the missing assertions it has been sufficient to open our populated ontology in Protégé, start the reasoner and then export the result. We point out that HemiT computed the missing assertions in few seconds (this is indeed an easy task, which essentially amounts to compute the transitive closure of the `owl:sameAs` relation, which is polynomial in the number of assertions).

Lists	CA	DPA	M. DPA	DPAR	Recover
Politicians Wikidata	23	21	32	53	100%
Journalists Wikidata	8	0	16	16	100%

Table 3.9. Results of test #3

Realistically, we would obtain results similar to those commented in this section when the number of analysed articles increases. We believe that this confirms that resorting to Wikidata for producing lists to be used by the Gazetteer PR allows for both time savings, in our case 1.8 person days instead of 7 person days, and annotations of a good quality.

3.7 Final Remarks

In this chapter, we have proposed a working pipeline based on GATE that allows to populate a reference ontology from Italian text documents through the use of specific PRs. We have discussed a pipeline, that consists of the following main steps:

²⁴<https://protege.stanford.edu/>

- *Web crawling*, which is not specific of our approach and for which any standard technique can be applied.
- *Ontology definition*, which, as web crawling, is not specific of our approach, and for which usual methods and best practices for ontology and conceptual modeling can be adopted (cf., e.g., [200]). We only require the ontology be specified in the standard OWL syntax.
- *Semantic Document Annotation via GATE*. We have precisely shown how these activity should be carried out, by explaining the specific GATE configuration we had to create for Information Extraction from documents written in Italian.
- *Ontology population*: We have shown how the ontology is finally populated with instances extracted from the semantic annotated documents and exported in OWL. In particular, we have show how to customize the OwlExporter component, a third party PR, to be used on Italian text documents in order to make the ontology population phase easier.

After presenting the above pipeline, we have also discussed a second *partially-automated* pipeline, in which we have exploited available open knowledge bases (namely, Wikidata) to simplify Gazetteer lists generation and provide a more definite procedure to approach their definition (thus increasing the replicability of our solution). This second pipeline consists of the same phases of the previous one, and makes use of exactly the same components. However, the generation of Gazetteer lists is faced with a completely different approach, based on querying, through the standard SPARQL language, an open ontology available on the Web, i.e., Wikidata, in order to create such lists without a specific design (as done in the first pipeline). We have shown through additional experiments that such an approach produces results of quality that is comparable to the one reached in the previous pipeline (cf. Table 3.6), where the Gazetteer lists are produced through a less structured approach requiring more manual effort.

We recall that many technical details of our approach, particularly useful for practitioners and for the reproducibility of our solution, are available through online resources accessible at <https://tinyurl.com/Ontology-population-via-GATE>.

We conclude this part by talking about the problems encountered in our implementation, which also serve as motivation for the following chapters.

Although GATE, being a rich environment for Language Engineering, supports general development and provides good debugging mechanisms, implementing ad-hoc solutions through the definition of new modules imposes a thorough knowledge of the APIs and data structures underlying the framework, not always supported by adequate documentation. This aspect dramatically affects the usability of this technology, which requires an initial significant effort to be understood.

Regarding JAPE rules, the main criticism that can be raised concerns with its semantics, which is not declarative [87]. The expressive power of this extraction language or fragment thereof is not completely formalized, and is thus difficult to make comparisons with other rule-based information extraction languages, such as [203]. Another important question that remains open concerns the complexity of evaluating these rules.

We finally notice that the role of the ontology in the experimentation we have just described has been essentially limited to the structuring of the information. In the IE activities, we only used the vocabulary of the ontology to annotate text so that next phases in the pipeline could suitably instantiate the corresponding ontology predicates. Of course, once the ontology is populated, one can take advantage of reasoning services it allows for, as well as exploit the conceptual representation it provides to support data sharing and integration. However, extraction rules and ontology axioms in our GATE-based experience remained basically independent with one another, and no specific ontology reasoning ability has been used in the extraction tasks. As we will see in following chapters (in particular in Chapter 6), devising an approach in which extraction rules and ontology axioms are more integrated, and the interaction between such languages, as well as reasoning over them, is well-understood, is instead the main objective of the investigation carried out in this thesis.

Chapter 4

Financial Market Supervision through Information Extraction

In this chapter we describe an IE application we developed to support monitoring actions aimed to uncover financial wrongdoing that can be identified through the inspection of documents describing financial instruments, such as securities, stocks, or derivatives. The activities we report here have been carried out within a joint project between Sapienza University and CONSOB¹ (Italian Companies and Exchange Commission).

Before delving in the details of our development and experimentation, we point out that, in the literature, the use of rule-based IE techniques in the financial and economic context [119, 186] has proven to be particularly effective, and represents nowadays the preferred choice in industrial IE applications by virtue of the interpretability and maintainability of extraction rules [57, 203]. We thus adopt such an approach, and, as done in the project described in the previous chapter, we use rules to populate a domain ontology specifically designed to provide a formal conceptualization of the information of interest, to facilitate data extraction, integration, and sharing. However, as detailed in the following, we use here different tools, towards the identification of more easy-to-use information extraction means. Also, as said, we focus on the different scenario of financial market.

The financial market is where the trading and exchange of financial instruments of various kinds occur. In recent years, financial news stories have brought to the fore the problem of market manipulation, i.e., market abuse aimed at mystifying reality, e.g., inflating or deflating the price of a security, to influence the behavior of the market for personal gain. In this chapter we will focus on market manipulations that may derive from false or misleading information contained in documents designed to help investors to understand investment products behaviour

¹Commissione Nazionale per le Società e la Borsa – <https://www.consob.it>

and support comparison with other similar products.

In Italy, supervising the financial market to detect illicit activities is entrusted to CONSOB. CONSOB daily receives from the creators of financial instruments (a.k.a. financial manufacturers) helpful documentation that should validate the non-existence of malfeasance. On this, CONSOB carries out an activity that can be summarized in two phases. The first one consists in searching for certain specific data in the documents, namely Key Information Documents (KIDs), transmitted by financial manufacturers. In the second one, retrieved data are analyzed through comparison with other information sources, in order to detect irregularities and, if necessary, alert the competent authorities.

There are however some critical issues related to the first phase: (*i*) the result of this phase consists of a set of textual reports that do not structure information gathered from the documents, nor prepare it for automatic processing; (*ii*) the search task is essentially carried out manually, and thus it is particularly time-consuming.

For the above reasons, the monitoring activity carried out by CONSOB is not made on every document but is done on a sample basis, with a negative impact on the performance of the supervision tasks.

In this chapter, we discuss the solution we have adopted to address the above issues. Regarding (*i*), we have modeled the information that CONSOB wants to extract from the KIDs through a dedicated domain ontology. This ontology is written in a vocabulary shared with CONSOB and compliant with European regulations that establish which data have to be included in the KIDs and how they have to be structured. About (*ii*), we built an IE tool based on CoreNLP [156], an open-source toolkit for IE by Stanford University, where, we compiled in a set of rule-based extraction rules the manual process of searching information performed by CONSOB operators.

The choice to use CoreNLP instead of GATE, which we have used in the project described in Chapter 3, lies in the possibility offered by this tool to realize lean and fast solutions to implement NLP systems, supported by reusable components that are particularly efficient and allow to solve the most common IE tasks.

The main contributions of this chapter are:

- The formalization of a rich financial domain ontology, including a rich taxonomy of products.
- The development of a pipeline for extracting data from financial documents, implemented with open-source technologies.
- Experiments of the approach on a large dataset, validated by the experts of the domain.

We point out that some of the resources presented in this chapter are reusable for other purposes. For example, the ontology can be exploited by other institutions that carry out similar market monitoring activities. Also, although our tool has been implemented for the Italian language, with a simple translation of the extraction rules and a minimal change in the settings of the other components, it can be reused for other languages. Moreover, the annotated dataset we produced can be used for other purposes, i.e., it can be adopted as training set for statistical IE approaches. We conclude this introduction by remarking that we ran a massive test validated by CONSOB on a large dataset with compelling results.

In order to facilitate the reproducibility of the experiments, we provide the ontology, the data and the tool in the following repository accessible through the link <https://github.com/Scafooo/Phd-Thesis/tree/main/Chapter%204>. For presentation purposes, we have translated into English some portions of the material presented in this chapter, which may slightly differ from their counterparts in the repository.

The structure of this chapter is summarized below:

- In Section 4.1, the KIDs analyzed by CONSOB are described. We discuss the main information contained therein and their formalization in the domain ontology.
- In Section 4.2, we describe the approach and the architecture adopted to solve the problems previously mentioned, also focusing on the technologies used to implement our solution. Properly, we briefly describe the Stanford CoreNLP library, and we focus on one of its components, namely `TOKENSREGEX`, useful to define the extraction rules.
- In Section 4.3, we show the evaluation of the obtained results. Namely, we evaluate the performances of our solution in terms of precision and recall, and discuss how the validation was carried out.


4.1 CONSOB Domain

As said, the domain of interest for the project described in this chapter (which we simply call CONSOB domain) concerns with the information context described through Key Information Documents (KIDs).

In the remainder of this section, we first describe the main characteristics of KIDs, and then we talk about the portion of the ontology that formalizes the information that we want to extract from the KIDs. In this chapter we only describe part of the development we have carried out for information extraction from KIDS.

4.1.1 Key Information Document

A KID is a document that contains the most relevant information related to a so-called Packaged Retail Investment and Insurance product (PRIIP). Figure 4.1 shows a portion of a KID acting as a running example for this chapter.



CREDIT SUISSE
termini indicativi

Documento informativo

Intento

Il presente documento contiene informazioni chiave relative a questo prodotto d'investimento. Non si tratta di un documento promozionale. Le informazioni, prescritte per legge, hanno lo scopo di aiutarvi a capire le caratteristiche, i rischi, i costi, i guadagni e le perdite potenziali di questo prodotto e di aiutarvi a fare un raffronto con altri prodotti d'investimento.

Prodotto

Nome del prodotto / ISIN:	100% ProNote con Partecipazione in USD su Thomson Reuters Gl. Resource Prot. Select Index, ISIN: CH0524993752 (il prodotto)
Ideatore del prodotto:	Credit Suisse AG , il nostro sito web: www.credit-suisse.com/derivatives , per ulteriori informazioni chiamare il numero +41 (0)44 335 76 00 .
Emittente:	Credit Suisse AG, Zurigo, tramite la sua succursale di Londra, UK
Autorità competente:	L'autorità di controllo competente

Il presente documento è stato creato il febbraio 27, 2020, 06:51 CET.

State per acquistare un prodotto che non è semplice e può essere di difficile comprensione.

Cos'è questo prodotto?

Tipo: Diritti valori (Wertrechte) disciplinati dal diritto svizzero.

Obiettivi: Il prodotto è uno strumento finanziario complesso collegato ad un sottostante (Thomson Reuters Gl. Resource Prot. Select Index (Indice azionario), il **sottostante**, si veda la tabella seguente). Investendo nel prodotto, è possibile partecipare ad una percentuale dell'eventuale performance positiva del sottostante. Nella data di rimborso finale l'investitore riceverà l'importo di rimborso finale, pari al 100% del taglio. Tale importo è denominato importo di rimborso minimo e non dipende dalla performance del sottostante. Inoltre, nella data di rimborso finale è possibile ricevere un importo di pagamento in caso di performance del sottostante favorevole per l'investitore. In caso di performance del sottostante sfavorevole per l'investitore, l'importo di pagamento può essere pari a zero.

Figure 4.1. KID by CreditSuisse. The full document can be found in Appendix

A recent European regulation² governs the content and the presentation of the KIDs to enhance investor protection standards for retail clients and increase transparency in the market. More in detail, through a well-defined template (reported in Figure 4.2), all PRIIPs producers are imposed not only what information to provide in the document, but also how to present it. According to the template, the KID has to be split in several sections, each intended to provide certain types of information.

In this chapter we focus only on a portion of the information contained in a KID, which corresponds to the data we extracted, and for which we evaluated recall and precision of our approach.

Specifically, we consider some information contained in the ‘Purpose’, ‘Product’, and ‘What is this product?’ sections of the KID. The data we are interested in are described in Table 4.1:

²https://www.eiopa.europa.eu/sites/default/files/publications/pdfs/jc_2017_49_jc_priips_qa_update_april_2019.pdf

Key Information Document	
Purpose This document provides you with key information about this investment product. It is not marketing material. The information is required by law to help you understand the nature, risks, costs, potential gains and losses of this product and to help you compare it with other products.	What happens if [PRIIP Manufacturer] is unable to pay out? Information on whether there is a guarantee scheme, the name of the guarantor or investor compensation scheme operator, including the risks covered and those not covered.
[Alert (where applicable)] You are about to purchase a product that is not simple and may be difficult to understand	What are the costs? Costs over time Template and narratives according to Annex VII
Product [Name of Product] [Name of PRIIP manufacturer] [where applicable ISIN] [website for PRIIP manufacturer] [Call [telephone number] for more information] [Competent Authority of the PRIIP Manufacturer in relation the PRIIP] [date of production of the KID]	Composition of Costs Template and narratives according to Annex VII Narratives on information to be included on other distribution costs
What is this product? Type Objectives Intended retail investor Insurance benefits	How long should I hold it and can I take money out early? <div style="background-color: #ccc; padding: 2px; border: 1px solid #000; margin: 5px 0;"> Recommended [required minimum] holding period: [x] </div> Information on whether one can disinvest before maturity, the conditions on this, and applicable fees and penalties if any. Information on the consequences of cashing-in before the end of the term or before the end of the recommended holding period
What are the risks and what could I get in return? Risk indicator Description of the risk-reward profile Summary Risk Indicator SRI template and narratives as set out in Annex III on possible maximum loss: can I lose all invested capital? Do I bear the risk of incurring additional financial commitments or obligations? Is there capital protection against market risk?	How can I complain?
Performance Scenarios Performance Scenario templates and narratives as set out in Annex V including where applicable information on conditions for returns to retail investors or built-in performance caps, and statement that the tax legislation of the retail investor's home Member State may have an impact on actual payout	Other relevant information

Figure 4.2. KID Template according to the European regulations

Information	Description	Section
PRODUCT NAME	Name of the Product	Product
MANUFACTURER NAME	Legal name of PRIIPS manufacturer	Product
IDENTIFICATION CODE	Unique identification code of the PRIIP, if applicable	Product
WEB SITE	Website referring to the PRIIP's manufacturer	Product
PRODUCTION DATE	Day month and year of product issue	Product
ALERT	A well-defined phrase that identifies products that are particularly difficult to understand	Purpose
TYPE TEXT	It identifies the type of product through a brief description	What is this product?
CURRENCY	The currency used to provide economic data about the product	What is this product?

Table 4.1. Information contained in the 'Purpose', 'Product' and 'What is this product?' sections of KIDs.

Although a rigid template for KIDs exists, this is often interpreted by financial manufacturers more as a guideline rather than a legal obligation, and several variants for the structure of the

document, and even for the names of the sections, can actually be found among the hundreds of documents received by CONSOB.

For example, taking into consideration the excerpt of KID showed in Figure 4.1, we can see that the purpose section is identified by the header ‘Intento’³ but the KIDs of other manufacturers usually adopt the header ‘Scopo’, which is the compliant choice respect to the EU regulations for documents in Italian. Other aspects on which discrepancies can be frequently found among different KIDs, and with respect to the KID template, concern with the order of the various sections, which is not always the same, and the different ways used to provide the same content (tables, bullet points, lists).

This heterogeneity of presentation has an evident impact on the information extraction phase, which will be discussed in Section 4.2.

We finally point out that financial manufacturers are required by law to transmit to CONSOB the KIDs associated to the PRIIPs they produce. The current mechanism adopted for transmission, is to send KIDs in PDF format via certified e-mails (only one KID per email).

In order to keep track of this flow of information, CONSOB uses a cataloguing system called DEMACO. This system stores the emails and it manages the entire lifecycle of the documents, from their receipt to their deletion, or preservation over time.

4.1.2 Ontology

In this section, we briefly describe a portion of the domain ontology. The entire ontology is composed of 52 classes, 10 relationships, and 26 attributes, an is given in Appendix A. The excerpt that we describe here is given in Figure 4.3 as a Graphol diagram.

The two cornerstone concepts of the ontology are **Document** and **Product**, that represent PRIIPs and KIDs, respectively. Among the various attributes of **Product**, `identification_code`, `product_name`, `type`, and `issue_date` correspond to some of the fields given in Table 4.1. **Financial_Entity** is specialized in two disjoint classes, i.e., **Regulator** and **Manufacturer**. Every individual instance of the class **Regulator** is responsible of the product to which it is associated via the relation `supervise`. Every **Manufacturer** produces the **Product** and writes its associated **Document**. Then, every **Document** has a `receipt_date`, a `protocol_number`, along with its ID. Every **Document** can be a **Start** or an **Update** document. Each KID of type **Update** updates another document. This pattern allows to keep track of updates (update occurs when a new KID is provided for a certain PRIIP, since some of the PRIIP characteristics have been changed).

We conclude this section by discussing the classification of financial products. One of the key

³‘Scopo’ and ‘Intento’ have the same meaning of ‘Purpose’ in English

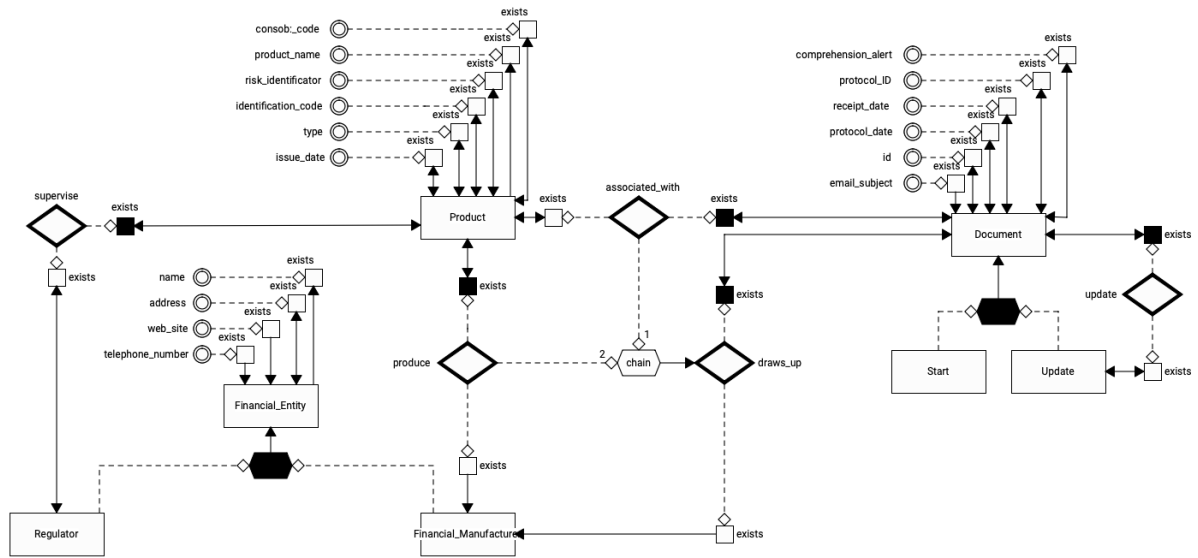


Figure 4.3. Ontology excerpt

requests from CONSOB operators has been to classify PRIIPS into several classes. This task plays a crucial role in the identification of market manipulation attempts, as it allows for an initial skimming of products to be supervised. For this reason, we have formalized in the ontology a three-level taxonomy of products. The taxonomy, built with the support of CONSOB and following the standards in the sector, is composed by five main classes (*Securities*, *IBIP*, *MOP*, *CIS*, *Structured Deposit*), having in turn an overall number of 26 subclasses. The Graphol version of the taxonomy is given in Appendix X, whereas in the excerpt given in Figure 4.3 the classification of products is captured simply through the attribute *type*, which we use to simplify the presentation.

4.2 Tool

In this section, we focus on the IE tool we implemented to extract the necessary information from the KIDs. We will discuss the technologies used, the approach followed, and the issues encountered.

The tool, whose architecture is shown in Figure 4.4, is organized in three macro modules, each of them dealing with a specific task. The tasks are briefly described below and discussed in more detail later in the chapter.

- **Data Preparation:** the KIDs are extracted from the emails stored in the DEMACO system and undergo a series of pre-processing steps, that transform them from PDF files

into pieces of annotable text.

- **Annotation:** By means of rule-based extractors, the annotations useful to provide the final output are generated. The classification of the products is also carried out at this stage.
- **Export:** The output of the tool is arranged in the form of the extensional level of the ontology (i.e., an ABox), or, upon request, as a CSV file.

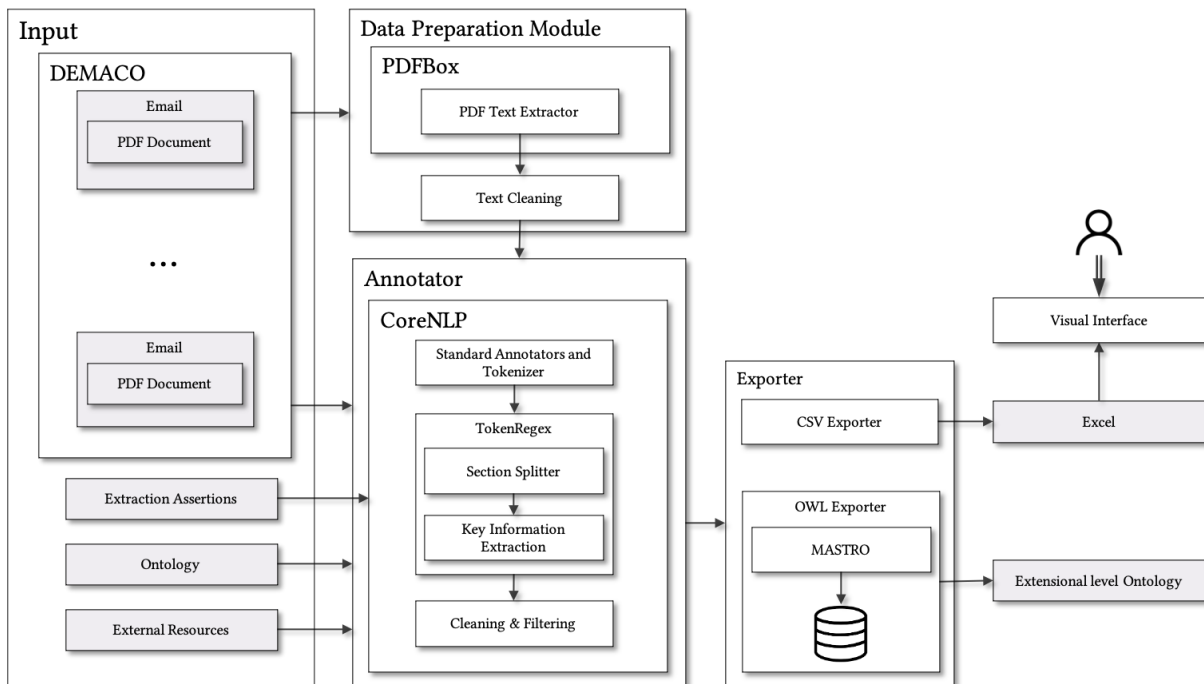


Figure 4.4. Architecture of the Tool

4.2.1 Data Preparation

Data Preparation is carried out by the Data Preparation Module. As said, this module transforms the PDF contained in the emails sent to CONSOB by financial manufacturers into plain text⁴, and clean it from errors.

This step is very critical, and if the PDF-to-text transformation produces bad quality results, many errors do actually occur in the subsequent phases (these errors are indeed caused by malformed sentences in the resulting text documents, rather than by the inability of the tool

⁴We tried transforming directly the PDF into a more structured format like XML, but the results were unusable for our purposes.

to process the language). For this reason, we performed it using PDFBox⁵ by Apache, a state-of-the-art and highly customizable library for working with PDF documents. PDFBox presents a large number of options to manage such transformation. These include dealing with different types of characters, normalization of the fonts, managing spaces, reading modes, etc.

In the various tests carried out, we noticed how the misreading of documents depended on the financial manufactures that had sent them. Properly, the style and the form of PRIIP presentation in the KIDS produced by some financial manufacturers caused to PDFBox, under default settings, transformation problems. To overcome these problems, the PDFBox parameters are adapted according to who sent the KID. This allowed us to drastically reduce the problems and consequently increase the precision and recall of the annotation phase.

Example 4.1. An example of the output of the *Data Preparation* task, where the input is the KID in Figure 4.1, is reported below:

Documento informativo termini indicativi

Intento

Il presente documento contiene informazioni chiave relative a questo prodotto d'investimento.

Non si tratta di un documento promozionale . Le informazioni , prescritte per legge , hanno lo scopo di aiutarvi a capire le caratteristiche , i rischi , i costi , i guadagni e le perdite potenziali di questo prodotto e di aiutarvi a fare un raffronto con altri prodotti d'investimento .

Prodotto

Nome del prodotto / ISIN : 100 % ProNote con Partecipazione in USD su Thomson Reuters Gl . Resource Prot . Select Index ,
ISIN : CH0524993752 -LRB- il prodotto -RRB-

Ideatore del prodotto : Credit Suisse AG ,

il nostro sito web : www.credit-suisse.com/derivatives , per ulter ori informazioni chiamare il numero +41 -LRB- 0 -RRB- 44 335 76 00 .

Emittente : Credit Suisse AG , Zurigo , tramite la sua succursale di Londra , UK

Autorit'a competente : L'autorit'a di controllo competente

Il presente documento 'e stato creato il febbraio 27 , 2020 , 06:51 CET .

State per acquistare un prodotto che non 'e semplice e pu' essere di difficile comprensione .

⁵<https://pdfbox.apache.org>

Cos ’ ‘e
...

Here some of the text structure in the original PDF file is lost during the transformation, but punctuation and new lines (useful, e.g., to identify the various sections) allow to reconstruct it .

□

4.2.2 Annotation

The annotation task, carried out by the Annotation Module, is the heart of the tool and is based on CoreNLP. CoreNLP [156] is a JAVA toolkit for natural language processing whose first release dates back to 2006 and is still maintained and evolving. The principles upon which it is based and which gave rise to its implementation are (i) make an easy annotation on text, (ii) easy to learn, (iii) minimal setup, (iv) provide a lightweight framework using plain Java objects, (v) transparent communication between components via a common interface. Stanford CoreNLP is currently one of the most widely used systems for building text processing tools. Its popularity is due to the fact that in order to use this toolkit, it is necessary to know only JAVA and not specific languages and components management, as it happens, for example, with GATE. Other success factors of CoreNLP have been: complete documentation that favors its immediate use; the software distributed through an open-source license; an active community that has enormously contributed to the project. Although the software was written in JAVA, the community has created several wrappers for other programming languages, including Python, Perl, Ruby, Scala, and Javascript(Node.js). Regarding the supported natural languages, the initial version of the software was able to process only English, German, Chinese, Arabic, and French. Over the years, the support has been extended to other languages, mainly by external contributors. For example, as far as Italian is concerned, the extension took place through the CoreNLP-it module [4]. This toolkit, written in JAVA, provides core NLP services accessible through APIs that can be combined to generate a pipeline for text annotation. Among these services, we mention those related to tokenization, lemmatization, POS tagging, and sentence splitting.

Another core component of CoreNLP used in our tool is TOKENSREGEX [53], which extends traditional regular expressions on strings by working on tokens instead of characters and defining pattern matching via a stage-based application. These extensions of regular expressions allow writing the extraction rules, i.e., rule-based extractors matching on additional token-level features, such as part-of-speech annotations, named entity tags, and custom annotations. In a similar way to the JAPE rules showed in Chapter 3, these characteristics bring concise rules at a higher level than just matching against the individual words.

We introduce the syntax of a rule written in TOKENSREGEX through the following example and we remind the reader to [53] for a complete treatment.

Example 4.2. Consider the following TOKENSREGEX extraction rule useful for extracting International Securities Identification Numbers (ISINs) from (the plain text version of) KIDS:

```
ENV.defaults["stage"] = 3
```

```
$StartISIN = (
    /ISIN/ /:/ |
    /CODICE|Codice/ /IDENTIFICATIVO|Identificativo/ /ISIN/ /:/* /Code/* /:/* |
    /Identificatori|Identificatore/ /del/ /presente/* /prodotto/ /:/* |
    /Codice/ /del/ /Prodotto|prodotto/ /:/ |
    /Codice/ /titolo/ /:/ |
    ...
)

$EndISIN = (
    / /
    ...
)

$code = "/([A-Za-z] [A-Za-z] [0-9]{10})/"

{
    ruleType: "tokens",
    pattern: (
        ($StartISIN) (? $CodeISIN [{word:$code} & {SECTION:"SECTION_PRODUCT"}]+?)
        ($EndISIN)
    ),
    action: ( Annotate($CodeISIN, ISIN, "ISIN"))
}
```

The ISIN is an alphanumeric sequence of 12 characters identifying a PRIIP. This sequence begins with two characters, identifying the country code of the product, followed by ten numbers. In our TOKENSREGEX rule, this characteristic is compiled into the regex identified by `$code`. Then, `$StartISIN` and `$EndISIN` represent the set of sequences of tokens (separated from each other through the symbol `|`) preceding or following the ISIN code, respectively.

The main part of the rule lies in the JSON-like structure at the end of it. In a nutshell:

- `ruleType` specifies how the pattern should be used. In our case, the rule works on tokens specified by `"tokens"` value.

- `pattern` contains the pattern to be matched over the text, built over the tokens using groups like the ones in POSIX (but also with names such as `?$CodeISIN`). In our example, our pattern specifies how to find the token such that there is a matching with the regex `$code` among all the tokens in the product section (i.e. annotated with `SECTION_PRODUCT`), and such that it is preceded by the tokens in `$StartISN` and followed by the ones in `$EndISIN`.
- `action` describes what should happen when the pattern is matched, precisely what annotation to apply. In our example, we annotate the token identified by the group `$CodeISIN` with the annotation `ISIN`.

We point out that the code `ENV.defaults["stage"] = 3` in the first line of the rule is an optional field describing the order of applications and the priority of the extraction rule with respect to the others.

The result of the application of this rule over the text in Example 4.1 is the annotation of `CH0524993752` with `ISIN`. □

After standard annotation steps such as POS tagging, sentence splitting etc., which are based on CoreNLPit, we rely to `TOKENSREGEX` component the following tasks: *(i)* reconstruct sections of the text; *(ii)* annotate fields useful for populating the ontology; *(iii)* clean and filter annotations; *(iv)* classify financial products.

Steps *(i)* and *(ii)* are carried out using rules like the one in Example 4.2. In particular, in *(i)*, taking into the due account punctuations and new lines introduced by the PDF-to-text transformation, we generate annotations that effectively reconstruct the KID sections. Step *(iii)* is needed since extracted data sometimes requires to be cleaned and/or standardized (mainly due to misreading of the PDF content, or due to different formats used to represent the same information, e.g., dates). This step is carried out through either special `TOKENSREGEX` rules that, when matched, remove the annotations over the tokens, or through custom `JAVA` code. As an example of cleanup task we describe the one performed on manufacturers. As said, manufacturers are required to report their legal name in the KIDs. For this reason, we have created a module that, given a dictionary containing the manufacturers' entries, removes everything that is not part of its legal name⁶. As said, standardization is, for instance, applied to dates, which we transform, through custom `JAVA` code, into the standard European date format "DD.MM.YYYY".

We finally discuss step *(iv)*. As already pointed out, document classification allows us to identify an initial set of financial products on which to perform market control actions. This

⁶For example in Figure 4.1 near the name of manufacturer (near the line 'Emittente') there is also the legal address

classification is done by a multistage rule-based classifier, whose features are the tokens annotated by the previous phases. For each subclass of the concept **Product**, we specified three annotation rules. The first analyzes the annotations related to the product name. This has the highest priority and is the first to be executed. If it fails in identifying the type of the product, the rule that processes the tokens related to the product type is triggered. In case also this second rule is not able to accomplish the task, the third rule is executed. This acts on the tokens present in the product section. We give below an example of classification rule analyzing the annotations on product name.

Example 4.3. Consider the following extraction rule.

```
# SECURITIES - CERTIFICATES - BENCHMARK

$BenchmarkName = (
  [{word:/benchmark/} & {nameProduct:"NAME_PRODUCT"}] |
  [{word:/Tracker/} & {nameProduct:"NAME_PRODUCT"}]
)

{
  ruleType: "tokens",
  pattern: (
    (?$ClassificationOnName $BenchmarkName)
  ),
  action: ( Annotate($ClassificationOnName, BenchmarkCertificates,
    "SECURITIES:CERTIFICATES:BENCHMARK")),
  result: "BenchmarkCertificates"
}
```

In words, if the product name contains the words **benchmark** or **tracker** then the annotation "SECURITIES:CERTIFICATES:BENCHMARK" will be applied. □

We finally point out that rules have been designed iteratively with support of CONSOB, alternating between phases of testing, writing and validation.

4.2.3 Exporter

The last task is carried out by the Exporter module. The purpose of this module is to prepare the output in the desired form. It transforms text annotations into two possible formats (*i*) ontology extensional level, (*ii*) CSV (Excel) file.

Regarding (*i*), the population of the ontology cannot be done using annotations on the text directly. This is because sometimes some information comes from other sources and cannot be

incorporated directly into the rules. Namely, we cannot express directly in `TOKENSREGEX` rules how to build the URIs denoting certain individuals, such as URIs relative to the documents, which have to be built upon their names and file paths (which are not information contained in KIDs). This has an obvious impact also in the instantiation of the ontology attributes and relationships that exist between the domain objects obtained by extraction.

To overcome this problem we put all the extracted information (annotation results, information in the subject of the email enclosing the KID, and the data in the DEMACO system) in tabular form, and stored it into as a relational database. Then, with the support of MASTRO [42], a well-known tool for semantic access to relational sources, we integrated the resources at our disposal in order to generate the ontology extensional level.

We report in Table 4.2, the annotations and respective OWL assertions generated using them.

Annotation	OWL Assertion
PRODUCT_NAME	P_ID :has_name 'PRODUCT_NAME'
MANUFACTURER_NAME	M_ID :has_name 'MANUFACTURER_NAME'
CODE	P_ID :has_code 'CODE'
WEBSITE	M_ID :has_web_site 'WEBSITE'
DATE	P_ID :has_production_date 'DATE'
ALERT	D_ID :has_name 'ALERT'
TYPE TEXT	P_ID :has_name 'TYPE TEXT'
CURRENCY	P_ID :currency 'CURRENCY'
CLASSIFICATION	P_ID a :Securities P_ID a :Derivatives ...

Table 4.2. Correspondence Annotations and ontology assertions

In the above table, `P_ID` and `M_ID` are two URIs identifying a product and a manufacturer, respectively. Intuitively, the assertions under the **OWL Assertion** column are generated from the portions of text identified by the annotations in the **Annotation** column. For example using the `TOKENSREGEX` extraction rule in Example 4.2, and the text in Example 4.1 we can generate the OWL Assertion relative to the Product Code `:2def9bb012344 :has_code CH0524993752` where `:2def9bb012344` is an identification of the Product built upon the file path of the document. As for the export option (ii), we remark that this has been a requirement by CONSOB, being CSV the format that can be read by most of the data analysis programs used by CONSOB. To facilitate the iteration of CONSOB experts with the tool, we also created a graphical interface shown in Figure 4.5.

Figure 4.5. User Interface of the Tool

4.3 Evaluations and results

In this section, we report the performance of the approach described in Section 4.2. We show the Precision, Recall, and F-measure obtained on three different datasets of KIDs.

The the first two datasets served as the main seed for designing the extraction rules we use in our tool, and the last one is a very large document base used to carry out a massive test and provide our final assessment on the system performances.

In the following, for each dataset we also provide (i) the total number of documents it contains, (ii) how many different most specific product types occur in the dataset⁷, (i.e., the number of unique product classes),

(iii) how many different financial manufacturers occur in the dataset (i.e., the total number of financial manufactures that have produced at least one document in the dataset). The above metadata characterize both the size and the variety present in the dataset. This last aspect is particularly important, since KIDs describing PRIIPs of different types have some differences in the way in which information is described (and may contain slightly different data), and KIDs produced by different manufacturers may be diversely formatted and may use distinct terminologies (as already discussed).

⁷With most specific type we mean the most specific class of the taxonomy the product is an instance of. For a product type to occur in the dataset there must be at least a KID describing a product of that type.

Before describing the datasets, we remark that CONSOB experts confirmed the tool’s performance through a manual inspection of the annotations we have produced with our tool. All datasets can be found here [URL](#).

4.3.1 First Dataset

The first dataset is composed of KIDs specifically selected by CONSOB operators as representative of the variety of the total set of documents to be analyzed. As said, this set has been used as first document base for testing the initial set of extraction rules we designed on the basis of experts specifications. The characteristics of the dataset are given below.

Number of KIDs	31
Number of PRIIP’s manufacturers	17
Number of most specific product types	16

Table 4.3. First Dataset

The results in terms of precision and recall relative to the kind of annotations listed in Table 4.2

Annotation	T. A.	C. A.	S. A.	M. A.	Precision	Recall	F-Measure
Product Name	31	27	1	4	96.2%	87.1%	91.4%
Manufacture	31	31	0	0	100%	100%	100%
ISIN	24	21	1	3	95.4%	87.5%	91.3%
Web Site	22	22	0	0	100%	100%	100%
Production Date	31	31	0	0	100%	100%	100%
Alert	24	24	0	0	100%	100%	100%
Type Text	31	28	1	3	96.5%	90.3%	93.3%
Currency	31	28	1	3	96.5%	90.3%	93.3%
Classification	31	28	1	3	96.5%	90.3%	93.3%

Table 4.4. Performance of the Approach on the First Dataset

In Table 4.4, columns **T. A.**, **C. A.**, **S. A.**, **M. A.** refer to total, correct, spurious, missing annotations, respectively. The cases in which 100% of F-measure is not reached are due to an incorrect transformation of the PDF into text (occurred in 3 documents). Despite some programming efforts to mitigate the above problem, in the best transformation we obtained some sentences

are still mixed together, making it impossible to correctly split the document into the right sections, which affects the correct extraction of some fields.

4.3.2 Second Dataset

The KIDs contained in the second dataset were chosen randomly by CONSOB, with the aim of creating a dataset 20 times larger than the first dataset. The metadata of the second dataset are shown in Table 4.5.

Number of KIDs	701
Number of PRIIP's manufacturers	88
Number of most specific product types	22

Table 4.5. Second Dataset

The system performance using the extraction rules written for the previous dataset is shown below.

Field	T.A.	C. A.	S. A.	M. A.	Precision	Recall	F-Measure
Product Name	701	470	40	231	92.1%	67.0%	77.6%
Manufacture	701	415	94	56	81.5%	59.2%	68.6%
ISIN	528	510	0	18	100%	96.5%	98.2%
Web Site	630	597	0	33	100%	94.7%	97.2%
Production Date	701	454	190	247	70.5%	64.7%	67.5%
Alert	531	531	0	0	100%	100%	100%
Type Text	701	415	56	286	88.1%	59.2%	70.8%
Currency	610	478	86	132	83.2%	78.3%	80.7%
Classification	701	378	96	323	79.7%	53.9%	64.3%

Table 4.6. Performance of the Approach on the Second Dataset

As shown in Table 4.6, the recall dropped considerably concerning the tests on the first dataset due to the increment of the KIDs heterogeneity. The rules that have failed the most are those aimed to identify the KID sections. Consequently, it was impossible to extract the fields associated with them. To overcome the problem, we had to implement some changes on the rules to correct them. The new system performance is reported in Table 4.7. The difference in terms of F-measure with the previous set of rules is reported in Chart 4.6.

Field	T.A.	C. A.	S. A.	M. A.	Precision	Recall	F-Measure
Product Name	701	695	2	16	99.7%	99.1%	99.4%
Manufacture	701	656	0	55	100%	93.5%	96.6%
ISIN	528	510	0	18	100%	96.5%	98.2%
Web Site	630	597	0	33	100%	100%	100%
Production Date	701	670	0	31	100%	95.5%	97.7%
Alert	531	531	0	0	100%	100%	100%
Type Text	701	671	13	30	98.1%	95.7%	96.9%
Currency	610	579	40	31	93.5%	94.9%	94.2%
Classification	701	701	40	0	94.6%	100%	97.2%

Table 4.7. Performance of the Approach on the Second Dataset

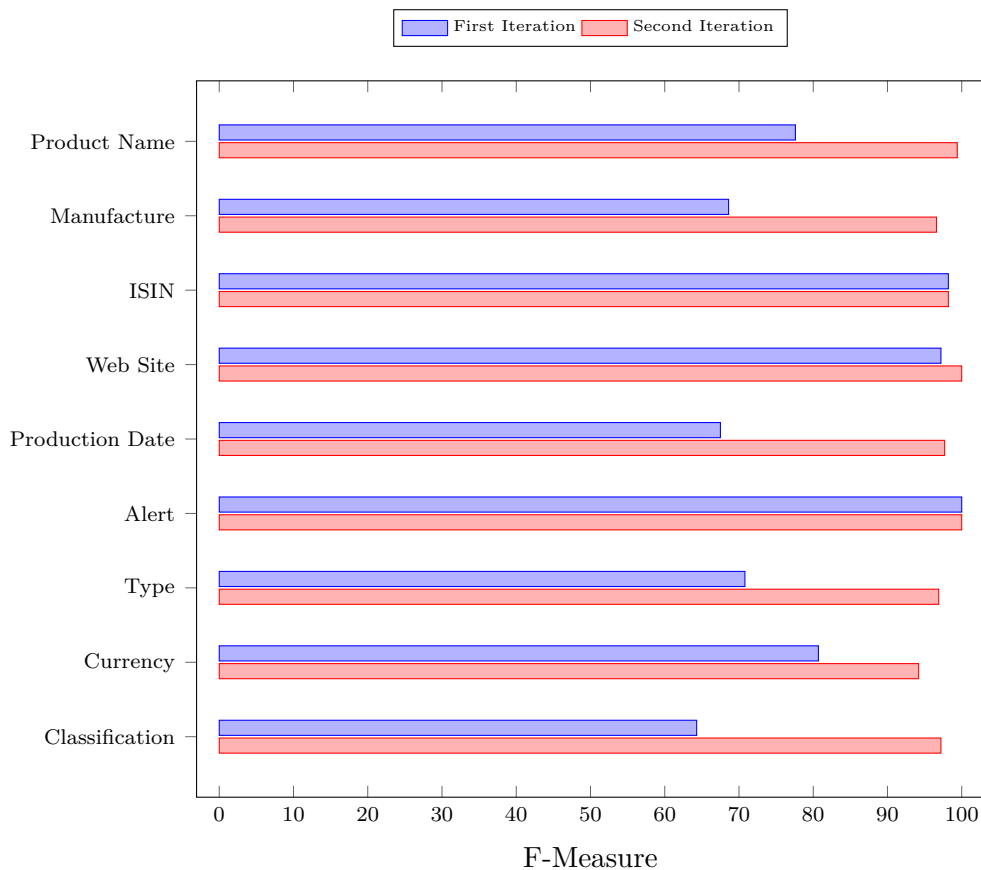


Figure 4.6. F-Measures difference between the non-fixed rules and the fixed ones applied to the second dataset

4.3.3 Third Dataset

This last test is the most representative among those we performed. We have conducted a massive experiment, which is indicative of the performance of the tool in production, on over

14000 documents, for a total size of 995MB. The characteristics of the dataset are given below.

Number of KIDs	14606
Number of PRIIP's manufacturers	182
Number of most specific product types	24

Table 4.8. Third Dataset

The rules compared to the tests on the second dataset have undergone some minor changes, mainly related to the cleaning of some fields and classification. The final results, which are very encouraging, are reported below.

Field	T.A.	C. A.	S. A.	M. A.	Precision	Recall	F-Measure
Product Name	14242	14117	125	50	99.1%	99.6%	99.3%
Manufacture	14025	13865	160	91	98.8%	99.0%	98.9%
ISIN	14242	14242	0	0	100%	100%	100%
Web Site	13485	13485	351	0	97.4%	100%	98.7%
Production Date	13762	13611	151	196	98.9%	99.2%	99.0%
Alert	12561	12561	0	23	100%	99.2%	99.6%
Type Text	13777	13537	240	159	98.2%	98.8%	98.5%
Currency	12471	11566	905	1390	92.7%	89.2%	90.9%
Classification	14504	13887	617	136	95.7%	98.9%	97.3%

Table 4.9. Performance of the Approach on the Second Dataset after extraction rules fixing

Note that the performance on the third dataset is higher with respect to the first and second datasets. This is attributable to the fact that the documents that generate errors on a large dataset are also the ones that appear the less.

4.3.4 Execution time performances

We want to conclude this section by talking about the tool's performance in terms of execution time. The processing time for 14606 documents (third dataset) was 8.32 hours, with an average of 2.05 seconds per document. On average, the data preparation took ~ 1.03 seconds per document. That is, it took the tool around 4.2 hours to transform all PDF documents in plain text. Obviously, this cost could be avoided if documents were produced in different native format. At the same time, we believe that this can be reduced by adopting different transformation tools, which we could not properly investigate in the timeframe of the project.

4.4 Final Remarks

In this chapter, we have reported an experience on information extraction from financial documents, aimed to structure gathered data according to a domain ontology.

We exploited the CoreNLP technology to build a rule-based tool able to extract key information from KIDs. In our development we could experience that the Stanford toolkit, compared to GATE, simplifies some annotation steps and allows to write NLP modules more comfortably, based on its components. Through a series of massive tests, supported by validation from CONSOB, we have also shown the quality of our solution, which lead to obtain particularly encouraging results.

In our discussion, we focused on `TOKENSREGEX`, a CoreNLP component useful for defining extraction rules. This component, similar to `JAPE` in GATE, allows to annotate portions of text using a rule-based language acting over the tokens. This language shares with `JAPE` not only its usefulness but unfortunately also the fact that, to the best of our knowledge, no study on its formal characteristic has been so far carried out. As a consequence it is not completely clear which is the expressive power of the language, and the computational complexity of evaluating `TOKENSREGEX` rules over a document.

In conclusion, this language, as `JAPE`, does not seem adequate for an effective coupling with popular ontology languages for data management, where the trade-off between expressiveness and complexity is particularly important, considered their use over large datasets.

Chapter 5

Theoretical Background

This chapter is an introduction of some theoretical notions useful for the next chapters of this thesis. We first recap some aspects of relational databases [2]. Next we provide a brief introduction to Description Logic (DL) ontologies [10, 46]. Then, after recalling some basics of Complexity Classes [152, 172, 81], we turn our attention to Ontology Based Data Access (OBDA), a sophisticated form of Information Integration [176, 213]. Finally we describe the formal framework of Document Spanners proposed by Fagin et al. in [86, 87] for rule-based IE. This chapter is intended to be a brief introduction to such matters, while an exhaustive treatment of them is out of our scopes. For further background we refer the reader to the literature cited above.

5.1 Relational Databases

A relational database schema (or simply schema) \mathcal{S} is a finite set of predicate symbols, each with a specific arity, and a set of integrity constraints. Given a schema \mathcal{S} , an \mathcal{S} -database DB is a finite set of facts $s(\vec{c})$, where s is an n -ary predicate symbol of \mathcal{S} , and $\vec{c} = (c_1, \dots, c_n)$ is an n -tuple of constants. For the rest of this thesis, when we refer to a relational database we always consider it as defined above, also known in literature as complete database [153], i.e., without null values.

5.1.1 Query Answering in Relational Databases

Relational databases were born with the purpose of storing information and making it accessible through query mechanisms. In this subsection we start with a general notion of query in first-order logic (FOL), and then we move to the definition of queries over relational databases.

A *query* is a function-free FOL open formula, which we denote as:

$$\{\vec{x} \mid \exists \vec{y}. \phi(\vec{x}, \vec{y})\} \quad (5.1)$$

where $\exists \vec{y}. \phi(\vec{x}, \vec{y})$ called the body of the query is a FOL formula with free variables \vec{x} , also called the target list of the query, and existentially quantified variables \vec{y} , possibly containing constants. The number of variables in \vec{x} is the *arity* of the query. Among FOL queries, we in particular consider *conjunctive queries* (CQs), i.e., queries in which $\exists \vec{y}. \phi(\vec{x}, \vec{y})$ is a conjunction of the form $\exists \vec{y}. p_1(\vec{x}_1, \vec{y}_1) \wedge \dots \wedge p_n(\vec{x}_n, \vec{y}_n)$, where each $p_i(\vec{x}_i, \vec{y}_i)$ is an *atom*, $\vec{x} = \cup_{i=1}^n \vec{x}_i$ and $\vec{y} = \cup_{i=1}^n \vec{y}_i$.

A *union of conjunctive query* (UCQ) is a FOL query of the form:

$$\{\vec{x} \mid \exists \vec{y}_1. \phi_1(\vec{x}, \vec{y}_1) \vee \dots \vee \exists \vec{y}_n. \phi_n(\vec{x}, \vec{y}_n)\} \quad (5.2)$$

such that each $\{\vec{x} \mid \exists \vec{y}_i. \phi_i(\vec{x}, \vec{y}_i)\}$ is a CQ. To simplify notation, we can write a FOL query $\{\vec{x} \mid \exists \vec{y}. \phi(\vec{x}, \vec{y})\}$ as the formula $\exists \vec{y}. \phi(\vec{x}, \vec{y})$, and a UCQ as a set of CQs.

A query over a relational database has the form (5.1) where each atom is an n -ary predicate in the signature of \mathcal{S} . Given a relational database schema \mathcal{S} , an \mathcal{S} -database DB and a conjunctive query q of arity n over \mathcal{S} , an n -tuple \vec{c} of constants is an *answer to q on DB*, if $\text{DB} \models \exists \vec{y}. \phi(\vec{c}, \vec{y})$. The answers to q can be obtained by standard evaluation of q over DB.

In the following, sometimes we use $q(\vec{x})$ to denote a query of the form (5.1) with free variables \vec{x} , and $q(\vec{c})$ to denote $\exists \vec{y}. \phi(\vec{c}, \vec{y})$.

5.2 Description Logics

Description Logics (DLs) [9] are decidable fragments of first-order logic that are largely recognized as one of the best means to specify ontologies, being them formally well-understood and equipped with powerful mechanisms to reason upon the representations they allow to specify. DLs model the domain of interest in terms of *concepts*, that are abstractions for sets of objects, and *roles*, that denote binary relations between objects. They are widely used in the context of the Semantic Web, and indeed are at the basis of OWL 2, the W3C standard for specifying ontologies [67] (as said in Chapter 2, in OWL 2 concepts are called classes and roles are called objectproperties).

Formally a DL ontology \mathcal{O} is defined as a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where:

- \mathcal{T} , called TBox, is the *terminological component*, which contains assertions (i.e., closed formulas of the logic, a.k.a. sentences) representing intensional knowledge, and
- \mathcal{A} , called ABox, is the *assertional component*, which contains assertions representing extensional knowledge.

For the rest of this section we assume to have a fixed infinite countable alphabet Γ of names for concepts (called atomic concepts), roles (called atomic roles), and constants (which we also call entities, as done in previous sections) The formal semantic of a DL language is given in terms of FOL interpretations. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ over Γ consists of a non-empty set $\Delta^{\mathcal{I}}$ of objects (the interpretation domain) and an interpretation function $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each role R a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$, and to each entity e an object $o \in \Delta^{\mathcal{I}}$.

An interpretation \mathcal{I} is a model of an ontology \mathcal{O} if \mathcal{I} satisfies all the assertions in \mathcal{T} (denoted with $\mathcal{I} \models \mathcal{T}$) and \mathcal{A} (denoted with $\mathcal{I} \models \mathcal{A}$). The satisfaction of an assertion is defined in the standard way for DLs [9]. We denote with $Mod(\mathcal{O})$ the set of all models of an ontology \mathcal{O} . We say that \mathcal{O} is satisfiable if $Mod(\mathcal{O}) \neq \emptyset$, unsatisfiable, otherwise. We also say that \mathcal{O} entails a FOL sentence ψ , denoted $\mathcal{O} \models \psi$, if $\psi^{\mathcal{I}}$ evaluates to true in every $\mathcal{I} \in Mod(\mathcal{O})$, where $\psi^{\mathcal{I}}$ denote the standard interpretation of a FOL sentence [2].

In this thesis we will mainly focus on ontologies expressed in *DL-Lite*, a family of DLs particularly suited for specifying ontologies on top of large data repositories [46, 176], and that is at the basis of OWL 2 QL, one of the tractable profiles of OWL 2 [159], already mentioned in Chapter 2. More specifically, we will consider the two basic members of this family, i.e., *DL-Lite_R* and *DL-Lite_F*.

In *DL-Lite_R*, the TBox is a finite set of assertions having the following forms:

$$\begin{aligned} B_1 \sqsubseteq B_2 \quad Q_1 \sqsubseteq Q_2 & \quad (\text{concept/role inclusion}) \\ B_1 \sqsubseteq \neg B_2 \quad Q_1 \sqsubseteq \neg Q_2 & \quad (\text{concept/role disjointness}) \end{aligned}$$

where: each Q_i , with $i \in \{1, 2\}$, is a basic role, i.e., an atomic role $R \in \Gamma$ or its inverse R^- ; each B_i , with $i \in \{1, 2\}$, is a basic concept, i.e., an atomic concept $C \in \Gamma$, or a concept of the form $\exists R$ or $\exists R^-$, i.e., unqualified existential restrictions, which denote the set of individuals occurring as first argument (a.k.a. domain) or second argument (a.k.a. range) of R , respectively.

In *DL-Lite_F*, inclusions and disjointnesses between roles are not allowed but it is possible to specify functionalities, which are assertions of the form:

$$(\text{funct } Q) \quad (\text{role functionalities})$$

where Q is a basic role.

In both the above logics, the ABox is a finite set of membership assertions of the form $C(e_1)$ or $R(e_1, e_2)$, where C and R are an atomic concept and an atomic role, respectively, and e_1 and e_2 are entities.

Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, the semantics of *DL-Lite_R* and *DL-Lite_F* constructs is

as follows:

$$\begin{aligned}
C^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
R^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
(R^-)^{\mathcal{I}} &= \{(o_2, o_1) \mid (o_1, o_2) \in R^{\mathcal{I}}\} \\
(\exists Q)^{\mathcal{I}} &= \{o \mid \exists o' . (o, o') \in Q^{\mathcal{I}}\} \\
(\neg B)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}} \\
(\neg Q)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus Q^{\mathcal{I}}
\end{aligned}$$

where C is an atomic concept, R an atomic role, Q a basic role, and B a basic concept. Then, \mathcal{I} satisfies an inclusion $B_1 \sqsubseteq B_2$ if $B_1^{\mathcal{I}} \subseteq B_2^{\mathcal{I}}$, and satisfies $B_1 \sqsubseteq \neg B_2$ if $B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}} = \emptyset$, and analogously for assertions on roles (only for $DL\text{-Lite}_{\mathcal{R}}$). Furthermore, \mathcal{I} satisfies (funct Q) (only for $DL\text{-Lite}_{\mathcal{F}}$) if there are no $o_1, o_2, o_3 \in \Delta^{\mathcal{I}}$ such that both (o_1, o_2) and (o_1, o_3) belong to $Q^{\mathcal{I}}$. Finally, \mathcal{I} satisfies an ABox assertion $C(e_1)$ (resp. $R(e_1, e_2)$), if $e_1^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $(e_1^{\mathcal{I}}, e_2^{\mathcal{I}}) \in R^{\mathcal{I}}$).

We finally note that $DL\text{-Lite}$ logics adopt the Unique Name Assumption, that is, in every interpretation different entities are interpreted with different objects.

Example 5.1. Consider the atomic concepts *Professor* and *Course*, the atomic roles *teaches* and *expert_in*, and the following $DL\text{-Lite}_{\mathcal{R}}$ TBox:

$$\begin{aligned}
\theta_1 : \textit{Course} &\sqsubseteq \neg \textit{Person} & \theta_2 : \textit{Professor} &\sqsubseteq \textit{Person} \\
\theta_3 : \textit{teaches} &\sqsubseteq \textit{expert_in} & \theta_4 : \exists \textit{teaches}^- &\sqsubseteq \textit{Course}
\end{aligned}$$

Such a TBox states that a course is not a person (θ_1), every professor is a person (θ_2), whoever teaches (a course) is an expert (about it) (θ_3), and that everything that is taught (i.e., occurs in the range of *teaches*) is a course (θ_4).

Instead, we obtain a TBox in $DL\text{-Lite}_{\mathcal{F}}$ if we substitute θ_3 with the assertion

$$(\text{funct } \textit{teaches}^-)$$

specifying that each course can be taught by at most one professor.

The following assertions are an example of ABox (for both $DL\text{-Lite}_{\mathcal{R}}$ and $DL\text{-Lite}_{\mathcal{F}}$).

$$\begin{aligned}
\alpha_1 : & \textit{Professor}(\textit{Einstein}) \\
\alpha_2 : & \textit{teaches}(\textit{Einstein}, \textit{Physics})
\end{aligned}$$

Such an ABox states that *Einstein* is a Professor (α_1) and that *Einstein* teaches Physics (α_2). □

Observe that $DL\text{-Lite}$ is an extension of the ontology language $DL\text{-Lite}_{RDFS}$ [13]. Specifically,

a $DL-Lite_{RDFS}$ is a finite set of assertions of the form:

$$B \sqsubseteq C \quad R_1 \sqsubseteq R_2$$

where B is a basic concepts, C an atomic concepts, and R_1, R_2 atomic roles.

We will also consider a slight extension of the DL ontology language $DL-Lite_{RDFS}$, namely $DL-Lite_{RDFS}^-$, which also allows for the concept/role disjointness assertions expressible in $DL-Lite_{\mathcal{R}}$.

We further note that both $DL-Lite_{\mathcal{R}}$ and $DL-Lite_{\mathcal{F}}$ are extensions of $DL-Lite_{core}$ i.e., the minimal logic of the $DL-Lite$ family. A $DL-Lite_{core}$ ontology is a finite set of assertions of the form:

$$B_1 \sqsubseteq B_2 \quad B_1 \sqsubseteq \neg B_2$$

Observe that $DL-Lite_{core}$ and $DL-Lite_{RDFS}$ (respectively, $DL-Lite_{RDFS}^-$) are incomparable fragments of $DL-Lite_{\mathcal{R}}$.

5.2.1 Query answering in Ontologies

A query over an ontology has the form (5.1), where each atom predicate is either an atomic concept or an atomic role from the ontology signature. Unlike relational databases, answering a query over an ontology amounts to computing the so-called *certain answers*, i.e., those answers that hold in all models of the ontology. Formally, given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a query q of arity n over \mathcal{O} , an n -tuple $\vec{c} = (c_1, \dots, c_n)$ of constants is a *certain answer to q* with respect to \mathcal{O} if $\mathcal{O} \models \exists \vec{y}. \phi(\vec{c}, \vec{y})$, i.e., the sentence obtained by substituting in q each variable in \vec{x} with the corresponding constant in \vec{c} is entailed by \mathcal{O} (or, equivalently, $(c_1^{\mathcal{I}}, \dots, c_n^{\mathcal{I}}) \in q^{\mathcal{I}}$ for each $\mathcal{I} \in Mod(\mathcal{O})$). The set of certain answers to q with respect to \mathcal{O} is denoted by $cert(q, \mathcal{O})$.

We notice that query answering over an unsatisfiable ontology is meaningless, since computing the certain answers to a query amounts to get all tuples of constants having the same arity of the query. For these reasons, in this thesis we will consider only query answering over satisfiable ontologies. We recall also that, for both $DL-Lite_{\mathcal{R}}$ and $DL-Lite_{\mathcal{F}}$, establishing whether an ontology is *satisfiable* can be reduced to query answering over a satisfiable ontology as shown in [46, 47].

Computing the certain answers to a query q with respect to a (satisfiable) ontology \mathcal{O} can be solved by *query rewriting*, i.e., by first computing the so called perfect reformulation q_r of q with respect to the TBox \mathcal{T} , and then evaluating q_r over the ABox \mathcal{A} (that is, evaluating it over an interpretation that is isomorphic to \mathcal{A} , which intuitively corresponds to consider \mathcal{A} a

relational database instance). Formally, given a query q and a TBox \mathcal{T} , a *perfect reformulation* of q with respect to \mathcal{T} is a query q_r such that, for every ABox \mathcal{A} , $\text{cert}(q, \langle \mathcal{T} \mathcal{A} \rangle) = \text{cert}(q_r, \mathcal{A})$ (notice that $\text{cert}(q_r, \mathcal{A})$ indeed corresponds to evaluate q_r over \mathcal{A} seen as a database).

Calvanese et al. proposed in [46] a prototypical algorithm, called **PerfectRef**, for computing the perfect reformulation of a UCQ Q with respect to a *DL-Lite_R* or *DL-Lite_F* TBox. At the basis of the algorithm there is a property saying that to compute the certain answers via rewriting over satisfiable *DL-Lite* ontologies, only concept/role inclusions (also called positive inclusions) need to be used in the reformulation process.

According to **PerfectRef**, such inclusions are used as rewriting rules, from right to left, to repeatedly rewrite atoms in the queries in Q (seen as a set of CQs). When an atom is rewritten, a new CQ is added to the result, as long as a fix point is reached. The final rewriting is indeed a UCQ. For example, given a TBox assertion $C_1 \sqsubseteq C_2$, and a query $\{x \mid C_2(x)\}$ the atom $C_2(x)$ is rewritten into $C_1(x)$ and the query $\{x \mid C_1(x)\}$ is added to the result. Notice however that for an atom to be rewritten according to an inclusion assertion in \mathcal{T} its terms must respect some syntactic conditions [46]. Moreover, when atoms in the query unify, **PerfectRef** performs such unification, which may then trigger some further atom rewriting¹.

For more details on **PerfectRef** we refer the reader to [46]. Below, we simply provide an example to intuitively show how it works.

Example 5.2. Consider the following query q expressed over the ontology \mathcal{O} of Example 5.1:

$$q = \{x \mid \exists y. \text{Person}(x) \wedge \text{teaches}(x, y) \wedge \text{Course}(y)\}$$

asking for persons who teach a course.

The certain answers to q in \mathcal{O} are given by the evaluation of the UCQ Q produced by the algorithm **PerfectRef** over \mathcal{A} . Q is a set consisting of the following CQs:

$$\begin{aligned} q &: \{x \mid \exists y. \text{Person}(x) \wedge \text{teaches}(x, y) \wedge \text{Course}(y)\} \\ q_1 &: \{x \mid \exists y. \text{Professor}(x) \wedge \text{teaches}(x, y) \wedge \text{Course}(y)\} \\ q_2 &: \{x \mid \exists y. \text{Person}(x) \wedge \text{teaches}(x, y)\} \\ q_3 &: \{x \mid \exists y. \text{Professor}(x) \wedge \text{teaches}(x, y)\}. \end{aligned}$$

The query q_1 is obtained from q by rewriting $\text{Person}(x)$ into $\text{Professor}(x)$, according to inclusion θ_2 . the CQ q_2 is obtained from q after rewriting $\text{Course}(y)$ into $\exists z. \text{teaches}(z, y)$ (according to inclusion θ_4) and after unifying $\text{teaches}(x, y)$ and $\text{teaches}(z, y)$. Similarly for q_3 ,

¹We note that, as a consequence of unification operations, the target list of a query in the set of CQs returned by **PerfectRef** may also contain constants, and that the target lists of the CQs in the returned set may also not be equal to one another.

which is derived from q_1 . Since \mathcal{O} is satisfiable, we can obtain the certain answers to q in \mathcal{O} by evaluating Q over the ABox \mathcal{A} , which returns the set $\{Einstein\}$. \square

Interestingly, the evaluation of the UCQ returned by `PerfectRef` can be delegated to a relational DBMS in charge of managing the data in the ABox, thus making *DL-Lite* logics particularly suited for efficient ontology-based data management. We say that both *DL-Lite_R* and *DL-Lite_F* enjoy UCQ rewritability of conjunctive query answering, since for every TBox \mathcal{T} expressed in such languages and every CQ q , the perfect reformulation of q with respect to \mathcal{T} can be always expressed as a UCQ. We notice that UCQ-rewritability is a special case of the more general *first-order-rewritability* of query answering, which is already enough for query answering to be solved through query rewriting and query evaluation over a relational DBMS. Notably, logics of the *DL-Lite* family are essentially the maximal DLs enjoying first-order-rewritability of conjunctive query answering [47]. Other logics for which such property holds have been proposed in the context of existential rules [162, 163] or Datalog+/- [36].

5.3 Computational Complexity

We assume familiarity with the basics about computational complexity, as defined in standard textbooks [152, 172]. In particular, we consider the following complexity classes:

$$\text{AC}^0 \subsetneq \text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{EXPTIME}$$

We have depicted the known relationships between these complexity classes. In particular, it is known that AC^0 is strictly contained in LOGSPACE , while it is open whether any of the other inclusions is strict [152]. However, it is known that $\text{PTIME} \subsetneq \text{EXPTIME}$. We will also mention the complexity class coNP , which is the class of problems that are the complement of a problem in NP . We only comment briefly on the complexity classes AC^0 and LOGSPACE , which readers might be less familiar with. AC^0 is the class of problems that can be solved by a uniform family of circuits of constant depth and polynomial size, with unlimited fan-in AND gates and OR gates. An example of problem in AC^0 is the evaluation of a first-logic query over an interpretation. LOGSPACE is the class of problems that can be decided by a two-tape (deterministic) Turing machine that receives its input on the read-only input tape and uses a number of cells of the read/write work tape that is at most logarithmic in the length of the input. A prototypical problem that is in LOGSPACE (but not in AC^0) is undirected graph reachability [182].

We will consider computational complexity of query answering, and in particular *data*

complexity, which is the complexity of evaluating a query in a language as a function of the size of the underlying data instance only, e.g., the ABox for query answering over a DL ontology (that is, the size of the query and the schema are treated as fixed constants) [204]. The other type of complexity typically considered for the query answering problem is *combined complexity*, which considers the query, the schema, and the database instance (e.g., the TBox and the ABox for query answering over a DL ontology) as input variables. The combined complexity of a query language is typically one exponential higher than data complexity.

We notice that, as said before, the perfect rewriting returned by the algorithm `PerfectRef` we described above is a UCQ (that is, a first-order query), which thus shows that conjunctive query answering in *DL-Lite* logics is tractable in data complexity, and more precisely, in AC^0 . As for combined complexity, it is NP-complete, since evaluating a UCQ over an interpretation (and thus over a database instance or over an ABox) is an NP-complete problem.

5.4 Ontology Based Data Access

Linking data to ontologies and providing (efficient) reasoning services over them is the main objective of *Ontology-based Data Access (OBDA)* [213, 60]. In OBDA the ontology, properly its intensional level, is coupled with external databases through a mapping, which declaratively specifies the semantic relationship between the ontology and the data. A user interacts only with the ontology, e.g., by posing queries, which are automatically processed by sophisticated algorithms that return the answers to the user by reasoning on the ontology and the mapping. The OBDA paradigm resorts to a three-level architecture, consisting of the ontology, some existing data sources relevant for an organization, and the mapping between the two.

From a more formal perspective, an OBDA system Ξ is expressed as a triple $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where:

- \mathcal{T} is a DL Tbox.
- \mathcal{S} is a relational database schema, also called *source schema*.
- \mathcal{M} is a mapping, i.e., a finite set of mapping assertions relating \mathcal{S} to \mathcal{T} .

An OBDA instance is a pair $\langle \Xi, \text{DB} \rangle$, where $\Xi = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ is an OBDA system, and DB is an \mathcal{S} -database, also called source database for Ξ .

5.4.1 Mapping Assertions

Assertions constituting a mapping \mathcal{M} relating a relational schema \mathcal{S} to a TBox \mathcal{T} are of the form

$$\forall \vec{x}. (\exists \vec{y}. \phi_{\mathcal{S}}(\vec{x}, \vec{y}) \rightarrow \exists \vec{z}. \phi_{\mathcal{T}}(\vec{x}, \vec{z})) \quad (5.3)$$

where $\phi_{\mathcal{S}}(\vec{x}, \vec{y})$ and $\phi_{\mathcal{T}}(\vec{x}, \vec{z})$ are finite conjunctions of atoms, as those allowed in CQs, over \mathcal{S} and \mathcal{T} , respectively [146, 79]². Mapping assertions of the above form are also called GLAV (Global-and-Local-as-View) mapping assertions. Special cases of GLAV mapping assertions are GAV (Global-as-View) and LAV (Local-as-View) mapping assertions.

A GAV mapping assertion is a GLAV mapping assertion in which the right-hand side of the implication does not make use of existential variables, i.e., it is an assertion of the form $\forall \vec{x}.(\exists \vec{y}.\phi_{\mathcal{S}}(\vec{x}, \vec{y}) \rightarrow \phi_{\mathcal{T}}(\vec{x}))$.³

A LAV mapping assertion is a GLAV mapping assertion in which the left-hand side of the implication is simply an atom without constants or repeated variables, such that all its variables appear in the right-hand side of the assertion, i.e., it is of the form $\forall x_1, \dots, x_n.(s(x_1, \dots, x_n) \rightarrow \exists \vec{z}.\phi_{\mathcal{T}}(x_1, \dots, x_n, \vec{z}))$, where s is an n -ary predicate symbol of \mathcal{S} , and x_1, \dots, x_n are pairwise different variables.

5.4.2 Semantics and Query Answering

We now give the semantics of an OBDA system by defining its models. An interpretation \mathcal{I} is a *model* of an OBDA instance $\langle \Xi, \text{DB} \rangle$, denoted by $\mathcal{I} \models \langle \Xi, \text{DB} \rangle$, if (i) $\mathcal{I} \models \mathcal{T}$ and (ii) \mathcal{I} satisfies the mapping \mathcal{M} with respect to DB , which is defined as follows: for every mapping assertion $m \in \mathcal{M}$ of the form (5.3), and every tuple \vec{c} in the evaluation of the query $\exists \vec{y}.\phi_{\mathcal{S}}(\vec{x}, \vec{y})$ over DB , $(\exists \vec{z}.\phi_{\mathcal{T}}(\vec{c}, \vec{z}))^{\mathcal{I}}$ evaluates to true in \mathcal{I} . The set of models of an OBDA instance $\langle \Xi, D \rangle$, denoted by $\text{Mod}(\Xi, \text{DB})$, is the set of interpretations \mathcal{I} for $\langle \Xi, \text{DB} \rangle$ such that $\mathcal{I} \models \langle \Xi, \text{DB} \rangle$. *Entailment* of a FOL sentence ψ from an OBDA instance $\langle \Xi, \text{DB} \rangle$, denoted $\langle \Xi, \text{DB} \rangle \models \psi$ is naturally defined as the task of verifying whether $\psi^{\mathcal{I}}$ evaluates to true in every $\mathcal{I} \in \text{Mod}(\Xi, \text{DB})$.

Several reasoning services have been proposed for OBDA systems [60, 63, 66] but the main service of interest is query answering, i.e., computing the certain answers to queries posed over the ontology of the system [176]. Given an OBDA instance $\langle \Xi, \text{DB} \rangle$ and a query q of the form (5.1) and of arity n , an n -tuple $\vec{c} = (c_1, \dots, c_n)$ of constants is a *certain answer* to q with respect to $\langle \Xi, \text{DB} \rangle$ if $\langle \Xi, \text{DB} \rangle \models \exists \vec{y}.\phi(\vec{c}, \vec{y})$ (or, equivalently, $(c_1^{\mathcal{I}}, \dots, c_n^{\mathcal{I}}) \in q^{\mathcal{I}}$ for each $\mathcal{I} \in \text{Mod}(\Xi, \text{DB})$).

Similar to the setting of a single ontology, we give the notion of perfect reformulation for OBDA systems: given an OBDA system $\Xi = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ and a query q , a *perfect reformulation* of q with respect to Ξ is a query q_r such that, for every source database DB for Ξ , $\text{cert}(q, \Xi, \text{DB})$ coincides with the evaluation of q_r over DB . We notice that when \mathcal{T} is expressed in a logic enjoying first-order rewritability of conjunctive query answering, as *DL-Lite_R* or *DL-Lite_F*, and

²Note that $\exists \vec{z}.\phi_{\mathcal{T}}(\vec{x}, \vec{z})$ may also contain variable object terms used to construct entities that instantiate ontology predicates from values returned by the query $\exists \vec{y}.\phi_{\mathcal{S}}(\vec{x}, \vec{y})$. An explanation on the role of object terms is given in Section 6.1

³We remark that also more expressive forms of GAV mappings have been often studied, namely allowing the query in the left-hand side to be a generic SQL query over \mathcal{S} [176].

\mathcal{M} is GAV, when q is a CQ, q_r is always expressible as a first-order query. This is typically done through the so-called *unfolding* of the perfect reformulation q' of q with respect to \mathcal{T} . Intuitively, the unfolding of q' through the mapping \mathcal{M} amounts to substitute each predicate S in q' with the union of queries in the body of the mapping assertions in \mathcal{M} having S in their heads.

5.5 Document Spanners

Recently, Fagin et al. [86, 87] have initiated a foundational study on rule-based IE, and proposed a new framework for it constructed on the notion of (*document*) *spanner*. In a nutshell, a spanner is a program that extracts from a text document **Doc** (i.e., a string) a relation containing tuples of *spans*, which are pairs of indices identifying substrings of **Doc**. For example if **Doc** is the string `Albert_Einstein_from_Ulma`, the span $[8, 16)$ selects the substring `Einstein`, which is the slice of **Doc** going from the eighth to the fifteenth character in **Doc** (by definition, a span $[i, j)$ goes from position i to position $j-1$, included).

Fagin et al. have in depth investigated how to represent spanners and how to combine them through algebraic operators. In particular, they have studied spanners defined by regular expressions with capture variables and operators adapted from *relational algebra*.

P	r	o	f	e	s	s	o	r	_	E	i	n	s	t	e	i	n	_	t	a	u	g	h	t	_	p	h	y	s	i	c	s	.
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
T	h	e	_	P	r	o	f	e	s	s	o	r	_	w	o	n	_	a	_	n	o	b	e	l	.								
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60								

Figure 5.1. Document **Doc**^{ex}

We now recall the definitions of spans and spanners, discuss a way of representing spanners, and present an algebra, through which the spanners of interest in this dissertation are defined. Our presentation is necessarily concise. For more details we refer the reader to [86].

5.5.1 Strings and spans

We fix a finite alphabet Σ of symbols, which we assume totally ordered. In the following examples Σ is composed by the lower and capital letters of the English alphabet, the full stop (“.”), and the underscore (“_”), which stands for the space character. We denote by Σ^* the set of all finite strings, called also *documents* over Σ . Thus, a document **Doc** $\in \Sigma^*$ is such that **Doc** = $\sigma_1 \dots \sigma_n$, with $n \geq 0$ and $\sigma_i \in \Sigma$ for $i \in \{1, \dots, n\}$.

A *span* identifies a substring of **Doc** by specifying its bounding indices. Formally a *span* of **Doc** has the form $[i, j)$, where $1 \leq i \leq j \leq n + 1$. If $[i, j)$ is a span of **Doc**, then **Doc** _{$[i, j)$} denotes the substring $\sigma_i \dots \sigma_{j-1}$. Note that **Doc** _{$[i, i)$} is the empty string, and that **Doc** _{$[1, n+1)$} is **Doc**. Two spans $[i, j)$ and $[i', j')$ are equal if and only if $i = i'$ and $j = j'$.

We denote by $\text{Spans}(\mathbf{Doc})$ the set of all possible spans of \mathbf{Doc} .

Example 5.3. Consider the document \mathbf{Doc}^{ex} given in Figure 5.1, and the span $[11, 19)$. It identifies the substring `Einstein`, i.e., $\mathbf{Doc}^{\text{ex}}_{[11,19)} = \text{Einstein}$. \square

We assume to have a fixed and infinite set SVars of *variables*, disjoint from Σ^* . Given a finite set $V \subseteq \text{SVars}$ and a document $\mathbf{Doc} \in \Sigma^*$, a (V, \mathbf{Doc}) -*tuple* is a mapping $\mu : V \rightarrow \text{Spans}(\mathbf{Doc})$ that assigns a span of \mathbf{Doc} to each variable in V . When V is clear from the context, we simply call the above tuple a (\mathbf{Doc}) -*tuple*. A (V, \mathbf{Doc}) -*relation* is a set of (V, \mathbf{Doc}) -*tuples*.

A *document spanner* (or simply *spanner*) is a function P over V that maps a document \mathbf{Doc} to a (V, \mathbf{Doc}) -*relation*. We use $\text{SVars}(P)$ to denote the set of variables of a spanner P . The cardinality of $\text{SVars}(P)$ is the *arity* of P . We may also use $P(v_1, \dots, v_n)$ to denote a spanner P over variables $V = v_1, \dots, v_n$. Furthermore, given a document \mathbf{Doc} , we write $\text{eval}(P, \mathbf{Doc})$ to denote the (V, \mathbf{Doc}) -*relation* returned by P with \mathbf{Doc} as input.

Example 5.4. In Figure 5.2 we provide an example of (V, \mathbf{Doc}) -*relation*, for the spanner $\llbracket \gamma_{\text{tok}} \rrbracket$, such that $\text{SVars}(\llbracket \gamma_{\text{tok}} \rrbracket) = \{x\}$. (V, \mathbf{Doc}) -*tuples* in this figure correspond to the words of \mathbf{Doc}^{ex} from Figure 5.1 (we discuss below how to represent such spanner in formulas). \square

$\text{eval}(\llbracket \gamma_{\text{tok}} \rrbracket, \mathbf{Doc}^{\text{ex}})$	
	x
μ_1	$[1, 10)$
μ_2	$[11, 19)$
μ_3	$[20, 26)$
μ_4	$[27, 34)$
μ_5	$[35, 38)$
μ_6	$[39, 48)$
μ_7	$[49, 52)$
μ_8	$[53, 54)$
μ_9	$[55, 60)$

Figure 5.2. Spanner $\llbracket \gamma_{\text{tok}} \rrbracket$ applied to the document in Figure 5.1

5.5.2 Spanner representation

Among the possible ways of representing spanners [86], in this thesis we use so-called *regex* formulas. In order to characterize them, we first give the definition of *variable regex*, which is an extension of a regular expression with capture variables. Its grammar is defined as follows:

$$\gamma := \emptyset \mid \epsilon \mid \sigma \mid (\gamma \vee \gamma) \mid (\gamma \cdot \gamma) \mid \gamma^* \mid x\{\gamma\} \quad (5.4)$$

The symbol \emptyset defines the empty set, ϵ is the empty string, and $\sigma \in \Sigma$. The \vee , \cdot , and $*$ symbols denote disjunction, concatenation, and the Kleene-star operators, respectively. $x\{\gamma\}$ instead indicates that the match obtained through the variable regex γ is mapped (in the form of a span) to the variable $x \in \text{SVars}$. Parenthesis may be used to specify precedence between operators.

We denote by $\text{SVars}(\gamma)$ the set of variables that occur in γ . We use γ^+ as abbreviations $\gamma \cdot \gamma^*$, and $[\sigma_i\text{-}\sigma_j]$ as a shortcut for the disjunction of all characters $\sigma \in \Sigma$ such that $\sigma_i \leq \sigma \leq \sigma_j$.

In this thesis we consider only variable regex expressions that are *functional*, i.e., such that in a matching over a document each variable is associated with one span. A functional variable regex is called *regex formula*. The class of regex formulas is denoted by RGX .

Example 5.5. Consider the following (simplified) set of regex formulas:

- $\gamma_{tok} = (\epsilon \vee (\Sigma^* \cdot (\cdot \vee _))) \cdot x_1 \{[a-zA-Z]^+\} \cdot ((\cdot \vee _) \cdot \Sigma^*)$,
i.e., a regex formula assigning to x_1 the words in a document (that is, every non-empty sequence of alphabetic characters preceded by either a space or an empty string, and followed by either a fullstop or a space);
- $\gamma_{cap} = (\epsilon \vee (\Sigma^* \cdot (\cdot \vee _))) \cdot x_1 \{[A-Z] \cdot \Sigma^*\} \cdot ((\cdot \vee _) \cdot \Sigma^*)$,
i.e., a regex formula assigning to x_1 the words that begin with a capital letter;
- $\gamma_{aft_prof} = (\Sigma^* \cdot _) \cdot (\text{Professor} \cdot _) \cdot x_1 \{\Sigma^+\} \cdot (_ \cdot \Sigma^*)$,
i.e., a regex formula assigning to x_1 the words that follow the word Professor (plus a space).

□

A regex formula γ naturally represents a spanner, and by $\llbracket \gamma \rrbracket$ we denote the spanner that is represented by γ . Then, with $\llbracket \text{RGX} \rrbracket$ we denote the class of all spanners represented by regex formulas.

5.5.3 An algebra over spanners

We now present an algebra over spanners. This algebra extends the class of spanners that are represented by regex formulas, i.e., $\llbracket \text{RGX} \rrbracket$, with the following operators: union (\cup), projection (π), (natural) join (\bowtie), and string-equality selection ($\zeta^=$). The set of spanners represented by formulas in the class RGX closed under \cup , π , \bowtie and $\zeta^=$ is denoted by $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$. Formally, let P , P_1 and P_2 be spanners and let \mathbf{Doc} be a document, the above operators are defined as follows [86]:

- **Union.** The union $P_1 \cup P_2$ is defined when P_1 and P_2 are *union compatible*, that is, $\text{SVars}(P_1) = \text{SVars}(P_2)$. In that case, $\text{SVars}(P_1 \cup P_2) = \text{SVars}(P_1)$ and $\text{eval}(P_1 \cup P_2, \mathbf{Doc}) = \text{eval}(P_1, \mathbf{Doc}) \cup \text{eval}(P_2, \mathbf{Doc})$.

- **Projection.** If $\mathbf{v} \subseteq \text{SVars}$, then $\pi_{\mathbf{v}}(P)$ is the spanner such that $\text{SVars}(\pi_{\mathbf{v}}(P)) = \mathbf{v}$ and $\text{eval}(\pi_{\mathbf{v}}(P), \mathbf{Doc})$ is obtained from $\text{eval}(P, \mathbf{Doc})$ by restricting the domain of each (\mathbf{Doc}) -tuple to \mathbf{v} .
- **(Natural) Join.** The join between spanners is defined as $P_1 \bowtie P_2$. It holds that $\text{SVars}(P_1 \bowtie P_2) = \text{SVars}(P_1) \cup \text{SVars}(P_2)$, and $\text{eval}(P_1 \bowtie P_2, \mathbf{Doc})$ consists of all (\mathbf{Doc}) -tuples μ that agree with some $\mu_1 \in \text{eval}(P_1, \mathbf{Doc})$ and $\mu_2 \in \text{eval}(P_2, \mathbf{Doc})$.
- **String selection.** Let x and y be two variables in $\text{SVars}(P)$, the string-equality selection operator is defined as $\zeta_{x,y}^- P$. We have that $\text{SVars}(\zeta_{x,y}^- P) = \text{SVars}(P)$, and $\text{eval}(\zeta_{x,y}^- P, \mathbf{Doc})$ consists of all (\mathbf{Doc}) -tuples μ in $\text{eval}(P, \mathbf{Doc})$ such that $\mathbf{Doc}_{\mu(x)} = \mathbf{Doc}_{\mu(y)}$.

Example 5.6. Using the regex formula defined in Example 5.5 we can define, using the spanner algebra, the following more expressive and complex $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$ -spanner.

- $\llbracket \rho_{prof} \rrbracket = \llbracket \gamma_{cap} \rrbracket \bowtie \llbracket \gamma_{aft_prof} \rrbracket$, i.e., the spanner represented by a regex formula that assigns to the variable x each word that both begins with a capital letter and follows the string `Professor_`. The result of applying $\llbracket \rho_{prof} \rrbracket$ to the document \mathbf{Doc}^{ex} in Figure 5.1 is shown in Figure 5.3. The extracted span is $[11, 19)$ corresponding to the substring `Einstein`. \square

$\text{eval}(\llbracket \rho_{prof} \rrbracket, \mathbf{Doc}^{\text{ex}})$	
	x_1
μ_1	$[11, 19)$

Figure 5.3. Result of spanner $\llbracket \rho_{prof} \rrbracket$ applied to the document in Figure 5.1

In our framework, which we introduce in the next chapter, we will consider only spanners belonging to $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$. It is worthwhile to remind the reader that spanners in such class are called *core spanners*, for being them able to capture the core of AQL, the declarative language used in SystemT, the IBM rule-based IE tool [55].

Chapter 6

Linking Text Documents to Ontologies

One of the main lessons we learned from the experiences we have described in Chapter 3 and Chapter 4 is that state-of-the-art solutions for IE are not specifically designed for an adequate exploitation of the presence of a domain ontology, which essentially remains a terminology through which organize the extracted data. Above all, we have suffered the lack of an organic framework where rule-based extraction mechanisms could be smoothly coupled with the rules specifying the domain ontology, so that reasoning services over the ontology could be suitably exploited to support and enrich the entire IE process. Furthermore, introducing the ontology in the extraction pipeline required in both our projects some ad-hoc manual operations, specific for the application at hand and difficult to frame in a general and well-regulated approach.

In this chapter we fill some of the above gaps and propose a formal framework for coupling ontologies and rules for IE. Our starting point is Ontology-based Data Access (OBDA). As recalled in Section 5.4, in OBDA the ontology is coupled with external databases through mapping assertions, which declaratively specify the semantic relationship between the ontology and the data. In OBDA, however, ontologies have been essentially used so far only on top of relational databases, with very few exceptions (as, e.g., [29, 54]), and how to access unstructured data, like those contained in text documents, using the ontologies as in OBDA is still unexplored.

Clearly, for OBDA systems to be able to access text documents, classical mapping assertions have to be modified. It is however crucial that the declarative nature of the mapping is preserved, and that the languages used to specify it are well-understood and precisely formalized. To this aim, we looked at the framework of document spanners (see Section 5.5), which lays foundational groundwork for rule-based IE.

In the following we thus propose a formal framework for coupling ontologies with spanners

for IE from documents. Within this framework, we focus on the problem of query answering and provide some complexity results and practical algorithms for the case when the ontologies are specified in some languages of the *DL-Lite* family of Description Logics and are coupled with expressive spanners.

More in detail, the contributions of this chapter can be summarized as follows:

- We introduce the notion of *Ontology-based document spanning (OBDS) system*. In an OBDS system, an ontology is linked to text documents through *extraction assertions*, which act similarly as mapping assertions in OBDA. Roughly speaking, an extraction assertion associates a document spanner P to a query q over the ontology, with the intended meaning that the tuples of strings corresponding to the spans returned by P evaluated over a text document must be among the answers to q evaluated over the ontology. An extraction assertion can be thus seen as a rule, where P is the body and q is the head.
- We study *query answering over an OBDS system*, i.e., how to answer a user query specified over the ontology by retrieving the answers from the text documents mapped to the ontology. We consider the case in which (i) the ontology is specified in either *DL-Lite_R* or *DL-Lite_F* (see Section 5.2), (ii) user's queries are CQs, (iii) spanners in the body of extraction assertions belong to the class $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$, i.e., are defined as regex formulas extended with the relational algebra operators union, projection, join, and string selection (see Section 5.5), (iv) queries in the head of extraction assertions are CQs. We show that *query answering is in PTIME in data complexity* (i.e., the complexity computed only with respect to the size of the underlying documents). We remark that *DL-Lite_R* and *DL-Lite_F* are the two most popular ontology languages used in OBDA to deal with large datasets, CQs are the most expressive queries for which query answering over ontologies has been shown to be decidable, and spanners in $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$ are among the most expressive document spanners considered in [86]. We note also that extraction assertions we define resemble GLAV mapping assertions used in data integration and in OBDA, i.e., the most expressive form of mappings adopted in these contexts [146, 78, 48, 89].
- We investigate *query rewriting in OBDS systems*, i.e., whether it is possible to answer a query by first rewriting it and then evaluating the rewriting over the data layer. Our aim is to understand whether we can reduce query answering to the execution of a document spanner of the same kind of those used in the extraction assertions. We positively answer the above question for the case in which ontologies are specified in *DL-Lite_R*. We indeed provide an algorithm that rewrites every CQ issued over an OBDS system (i.e., over its ontology) into a spanner belonging to $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$. We also show that the same holds when the ontology is expressed in *DL-Lite_F* and extraction assertions are GAV,

i.e., they have in their heads only CQs without existential variables. We believe that these results have an interesting practical fallout, since in these cases it is possible to delegate the evaluation of the rewriting to same engine that is in charge of evaluating the spanners in the body of the extraction assertions. We notice that this behaviour is similar to what happens in OBDA when query answering is first-order rewritable: source data are managed by a relational DBMS which is able to process both the queries in the body of mapping assertions and the perfect reformulations computed when solving query answering by rewriting (see also Section 5.4.2).

Interestingly, in our OBDS framework, as IE engine we can use an off-the-shelf tool like IBM SystemT [55], whose AQL language allows for expressing spanners belonging to $[[\text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}}]]$. Also, the modular nature of our rewriting technique seems to streamline the incorporation of OBDS capabilities into current OBDA engines (e.g., [43, 75]) as it will be discussed in Chapter 7.

We conclude this section by observing that, to the best of our knowledge, none of the previous works on ontology-based IE has proposed a formal declarative framework, nor has studied the problem of query answering, with the exception of [144], which this chapter is based on.

The rest of the chapter is organized as follows. In Section 6.1 we introduce our OBDS framework, and in Section 6.2 we establish our complexity results on query answering. Then, in Section 6.3, we provide our query rewriting algorithms for OBDSs systems equipped with $DL\text{-Lite}_{\mathcal{R}}$ or $DL\text{-Lite}_{\mathcal{F}}$ ontologies.

6.1 Ontology-based document spanning Framework

In this section we present our framework for coupling documents to ontologies, which we call *Ontology-based document spanning* (OBDS) framework. In the last part of the section, we also describe the problem of query processing in OBDS, which we will then study in depth in the next sections of this chapter.

Before delving into the details of the framework, we discuss how to deal with the following problem: when mapping documents to ontologies, it is likely that the text does not directly contain the identifiers that are used at the ontology level to denote the objects that are instances of the predicates of the ontology (in other terms, URI identifying entities may not be contained explicitly in the documents). Rather, the strings that are extracted from the document should more correctly interpreted as values. Our basic idea to deal with this problem is to use the same technique adopted in OBDA to construct entities from values, that is, consider object identifiers formed by (logic) terms built though the string values extracted from the documents [176]. To formally describe this mechanism we recall the notions of object term and variable term. An

object term has the form $\mathbf{f}(\vec{d})$ where \vec{d} is an m -tuple of either constants or variables and \mathbf{f} is a function symbols of arity m . If \vec{d} does not contain constants, $\mathbf{f}(\vec{d})$ is called variable object term. If instead \vec{d} is a tuple of only constants, $\mathbf{f}(\vec{d})$ is called ground object term.

We now turn to the framework definition. The three ingredients for an OBDS system are the ontology, a set of extraction assertions, and a source text document.

Definition 6.1. An OBDS System \mathcal{Y} is a pair $\langle \mathcal{T}, \mathcal{R} \rangle$, where

- \mathcal{T} is a DL TBox.
- \mathcal{R} is a set of *extraction assertions* of the form

$$P(\vec{x}) \rightsquigarrow \Psi(\vec{x}) \quad (6.1)$$

where

- $P(\vec{x})$ (the left-hand side of the assertion) is a $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$ -spanner.
- $\Psi(\vec{x})$ (the right-hand side of the assertion) is a CQ¹ over \mathcal{T} with free variables in \vec{x} , possibly using variable object terms $\mathbf{f}(\vec{w})$, such that $\vec{w} \subseteq \vec{x}$, as arguments of its atoms. Note that $\Psi(\vec{x})$ may contain also existentially quantified variables.

In the following, when the TBox of an OBDS system \mathcal{Y} is specified in a DL language \mathcal{L} we say that \mathcal{Y} is an \mathcal{L} OBDS system.

Example 6.1. Let $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ be an OBDS system where \mathcal{T} is as in Example 5.1, and let $\llbracket \rho_{prof} \rrbracket$ and $\llbracket \gamma_{teaches} \rrbracket$ be two spanners, where $\llbracket \rho_{prof} \rrbracket$ is as defined in Example 5.6, whereas the regex formula representing $\llbracket \gamma_{teaches} \rrbracket$ is:

$$\gamma_{teaches} = (\epsilon \vee (\Sigma^* \cdot _)) \cdot x_2 \{ \Sigma^+ \} \cdot (_ \cdot \text{taught} \cdot _) \cdot y_2 \{ \Sigma^+ \} \cdot ((_ \vee _) \cdot \Sigma^*)$$

i.e., a regex assigning to x_2 the words before the word **taught**, and to y_2 the words after **taught**.

The set of extraction assertions \mathcal{R} is as follows:

$$\begin{aligned} m_1 : \llbracket \rho_{prof} \rrbracket(x_1) &\rightsquigarrow \text{Professor}(\mathbf{prof}(x_1)) \\ m_2 : \llbracket \gamma_{teaches} \rrbracket(x_2, y_2) &\rightsquigarrow \text{teaches}(\mathbf{prof}(x_2), \mathbf{course}(y_2)) \end{aligned}$$

Notice that both **prof** and **course** are function symbols of arity 1 used to construct entities from the string returned by the spanners. □

¹In extraction assertions we consider CQs of the form (5.1) to be simply written as $\exists \vec{y}. \phi(\vec{x}, \vec{y})$. Notice that the right-hand side of extraction assertions have the same form of the right-hand side of mapping assertions of OBDA systems (cf. Equation (5.3))

The semantics of an OBDS system $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ is defined with respect to a document **Doc**. Given one such document, an interpretation \mathcal{I} is a *model for \mathcal{Y} with respect to **Doc*** if:

- \mathcal{I} is a model for \mathcal{T} , and
- for each extraction rule of the form (6.1) in \mathcal{R} , $(\Psi(\mathbf{Doc}_{\mu(x_1)}, \dots, \mathbf{Doc}_{\mu(x_n)}))^\mathcal{I}$ evaluates to true in \mathcal{I} for each $\mu \in \text{eval}(P, \mathbf{Doc})$.

We use $Mod(\mathcal{Y}, \mathbf{Doc})$ to denote the set of models of \mathcal{Y} with respect to **Doc**. The notion of entailment naturally extends to OBDS systems, i.e., given a sentence ψ we write that $\langle \mathcal{Y}, \mathbf{Doc} \rangle \models \psi$ if $\psi^\mathcal{I}$ evaluates to true in every $\mathcal{I} \in Mod(\mathcal{Y}, \mathbf{Doc})$.

In a way similar to what happens for mappings in the context of data integration [146] and OBDA, we can have two types of extraction assertions, i.e., GAV and GLAV. GLAV assertions are exactly of the kind we discussed so far. Instead, in a GAV *extraction assertion* there are no existentially quantified variables in its right-hand side. In this case, $\Psi(\vec{x})$ in assertions of type (6.1) is in the form $p_1(\vec{x}_1) \wedge \dots \wedge p_k(\vec{x}_k)$, with $\cup_{i=1}^k \vec{x}_i = \vec{x}$. It is easy to see that a GAV extraction assertion is equivalent to the set of following assertions:

$$\begin{aligned} P(\vec{x}_1) &\rightsquigarrow p_1(\vec{x}_1) \\ &\dots \\ P(\vec{x}_k) &\rightsquigarrow p_k(\vec{x}_k), \end{aligned}$$

that is, the right-hand side of each assertion is a single-atom query without existential variables. Therefore, from now on, we always assume that GAV *extraction assertions* have the form above (unless otherwise specified).

We conclude this section by talking about query answering, which is the task of computing the certain answers to a query posed on the ontology of the OBDS system.

Definition 6.2. Let $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ be an OBDS system, let q be a query, and let **Doc** be a document. A tuple of constants and ground object terms \vec{t} is a *certain answer* to q with respect to \mathcal{Y} and **Doc** if for every model $\mathcal{I} \in Mod(\mathcal{Y}, \mathbf{Doc})$ it holds that $(q(\vec{t}))^\mathcal{I}$ evaluates to true in \mathcal{I} .

The set of certain answers to q with respect to \mathcal{Y} and **Doc** is denoted by $\text{cert}(q, \mathcal{Y}, \mathbf{Doc})$.

For example, the set of the certain answers to the query $\{x \mid \text{Person}(x)\}$ in the OBDS system \mathcal{Y} of Example 6.1 with respect to the document **Doc**^{ex} in Figure 5.1 is $\{\text{prof}(\text{Einstein})\}$.

² $(q(\vec{t}))^\mathcal{I}$ is the interpretation in \mathcal{I} of the sentence $q(\vec{t})$, which possibly contains ground object terms. Each such term $\mathbf{f}(\vec{c})$ is interpreted exactly as a constant, i.e., $(\mathbf{f}(\vec{c}))^\mathcal{I} \in \Delta^\mathcal{I}$ and no two different terms are interpreted with the same object in $\Delta^\mathcal{I}$ (i.e., we adopt the unique name assumption on terms, too).

6.2 Complexity of query answering in OBDS systems

To establish computational complexity of query answering in our framework we show how to reduce this problem to query answering in an OBDA system.

Intuitively, given an OBDS system $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ and a document **Doc**, we can construct an OBDA system $\Xi = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and a source database DB for Ξ such that:

- \mathcal{T} is the same TBox of \mathcal{Y} ;
- \mathcal{S} is a source schema which contains a relation schema T_P for each spanner P occurring in \mathcal{R} , such that the arity of T_P coincides with the number of variables in $\text{SVars}(P)$ (in other terms, \mathcal{S} is the schema “produced” by the spanners in \mathcal{R});
- \mathcal{M} is a mapping containing an assertion m for each extraction assertion r in \mathcal{R} , such that m and r have the same right-hand side, and, let P be spanner of r such that $|\text{SVars}(P)|$ is n , the left-hand side of m is the query $T_P(x_1, \dots, x_n)$ (in other terms, \mathcal{M} contains the same assertions of \mathcal{R} , modulo a substitution of the spanners with the corresponding relation symbol in \mathcal{S});
- DB is an \mathcal{S} -database obtained by evaluating each spanner in \mathcal{R} over the document **Doc**, which returns tuples of spans, and by extracting the substrings of **Doc** identified by such spans.

In Figure 6.1, we give an algorithm, called `obds2obda`, that taken as input a set of extraction assertions \mathcal{R} returns \mathcal{M} , \mathcal{S} , and DB as described above.

Algorithm `obds2obda`

Input: A set of extraction assertions \mathcal{R} , a document DB

Output: A mapping \mathcal{M} , a relational schema \mathcal{S} and a relational database DB

begin

$\mathcal{S} \leftarrow \emptyset$;

DB $\leftarrow \emptyset$;

$\mathcal{M} \leftarrow \emptyset$;

for each $r \in \mathcal{R}$, where $r = P(\vec{x}) \rightsquigarrow \Psi(\vec{x})$, **do**

$\mathcal{S} \leftarrow \mathcal{S} \cup \{T_P \mid \text{such that } T_P \text{ is a fresh relation schema of the same arity of } P\}$;

DB $\leftarrow \text{DB} \cup \{T_P(\text{Doc}_{\mu(x_1)}, \dots, \text{Doc}_{\mu(x_n)}) \mid \mu \in \text{eval}(P, \text{Doc})\}$;

$\mathcal{M} \leftarrow \mathcal{M} \cup \{T_P(\vec{x}) \rightsquigarrow \Psi(\vec{x})\}$;

return \mathcal{M} , \mathcal{S} , and DB

end

Figure 6.1. The `obds2obda(\mathcal{R}, Doc)` algorithm

The following lemma shows the semantic relation between an OBDS system and the corresponding OBDA system constructed with the algorithm `obds2obda`.

Lemma 6.1. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{R} \rangle$ be Given an OBDS system and \mathbf{Doc} be a document. Let \mathcal{M} , \mathcal{S} and \mathbf{DB} be respectively the mapping, the relational schema, and the database returned by $\text{obds2obda}(\mathcal{R}, \mathbf{Doc})$, and let $\Xi = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be an OBDA system. Then, $\text{Mod}(\mathcal{E}, \mathbf{Doc}) = \text{Mod}(\Xi, \mathbf{DB})$.*

Proof. Let us assume that there exists $\mathcal{I} \in \text{Mod}(\mathcal{Y}, \mathbf{Doc})$ such that $\mathcal{I} \notin \text{Mod}(\Xi, \mathbf{DB})$. Since \mathcal{I} is a model for \mathcal{Y} with respect to \mathbf{Doc} , then \mathcal{I} satisfies \mathcal{T} . Thus, if \mathcal{I} is not a model of Ξ with respect to \mathbf{DB} , \mathcal{I} does not satisfy \mathcal{M} . This means that there must be an assertion $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{x})$ belonging to \mathcal{M} such that there exists a tuple of constants \vec{c} in the evaluation of $\Phi(\vec{x})$ over \mathbf{DB} for which $(\Psi(\vec{c}))^{\mathcal{I}}$ evaluates to false in \mathcal{I} . However, by construction of \mathbf{DB} , $\Phi(\vec{x}) = T_P(\vec{x})$, and every tuple \vec{c} is such that $\vec{c} = (\mathbf{Doc}_{\mu(x_1)}, \dots, \mathbf{Doc}_{\mu(x_n)})$ for some $\mu \in \text{eval}(P, \mathbf{Doc})$, and since \mathcal{I} satisfies \mathcal{R} (by hypothesis), it holds that $(\Psi(\vec{c}))^{\mathcal{I}}$ evaluates to true in \mathcal{I} . This leads to a contradiction and thus shows that $\text{Mod}(\mathcal{Y}, \mathbf{Doc}) \subseteq \text{Mod}(\Xi, \mathbf{DB})$. The fact that $\text{Mod}(\Xi, \mathbf{DB}) \subseteq \text{Mod}(\mathcal{Y}, \mathbf{Doc})$ can be proved in an analogous way, thus finally showing the thesis. \square

The theorem below follows from Lemma 6.1 and the fact that computing the certain answers to a CQ over an OBDA system whose TBox is specified in either $DL\text{-Lite}_{\mathcal{R}}$ or $DL\text{-Lite}_{\mathcal{F}}$ is in AC^0 in data complexity [46].

Theorem 6.1. *Let $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ be either a $DL\text{-Lite}_{\mathcal{R}}$ or $DL\text{-Lite}_{\mathcal{F}}$ OBDS system, \mathcal{R} be a set of GLAV extraction assertions, \mathbf{Doc} be a document, and q be a CQ over \mathcal{Y} . Then computing $\text{cert}(q, \mathcal{E}, \mathbf{DB})$ can be solved in time polynomial in the size of \mathbf{DB} .*

Proof. From Lemma 6.1 it follows that $\text{cert}(q, \mathcal{Y}, \mathbf{Doc}) = \text{cert}(q, \Xi, \mathbf{DB})$, where $\Xi = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, and \mathcal{M} , \mathcal{S} and \mathbf{DB} are returned by $\text{obds2obda}(\mathcal{R}, \mathbf{Doc})$. Thus the data complexity of computing $\text{cert}(q, \mathcal{Y}, \mathbf{Doc})$ is equal to the execution cost of obds2obda , with respect to the input document \mathbf{Doc} , and the cost of computing $\text{cert}(q, \Xi, \mathbf{DB})$. It is also easy to verify that obds2obda runs in polynomial time in the size of \mathbf{Doc} . Indeed, the only steps of obds2obda that depend on \mathbf{Doc} concern with the construction of \mathbf{DB} , which is obtained by the evaluation of all the spanners in \mathcal{R} over \mathbf{Doc} , and the subsequent extraction of the substrings of \mathbf{Doc} identified by the spans returned by such evaluations, which clearly are tasks polynomial in \mathbf{Doc} (see also [86]). As for the cost of computing $\text{cert}(q, \Xi, \mathbf{DB})$, we recall that conjunctive query answering in OBDA systems having either $DL\text{-Lite}_{\mathcal{R}}$ or $DL\text{-Lite}_{\mathcal{F}}$ TBoxes and GAV mappings is in AC^0 in data complexity [176]. This result extends also to GLAV mappings, for $DL\text{-Lite}_{\mathcal{R}}$ TBoxes, as shown in [74]. When mappings are GLAV and TBoxes are in $DL\text{-Lite}_{\mathcal{F}}$ the complexity rises to PTIME, which follows from the results in [82] and [46].

Thus, in all cases the problem can be solved in time polynomial in the size of \mathbf{Doc} . \square

Obviously, since $DL-Lite_{core}$, $DL-Lite_{RDFS}$, and $DL-Lite_{RDFS}^-$ are languages contained in $DL-Lite_{\mathcal{R}}$, query answering over OBDS systems where the TBox is expressed in one of the previous logics is polynomial in the size of **Doc**.

We notice that the above technique that reduces query answering over OBDS systems to query answering over OBDA systems is obviously general and can be used also when the TBox is specified in other DL languages. In all cases, however, we need to pay the cost of constructing a source database for the OBDA system Ξ by evaluating the spanners in \mathcal{R} over the document **Doc** (which is polynomial).

From practical perspective, however, the approach of “materializing” the result of spanner evaluation may have some drawbacks. Indeed, the source document is independent from the ontology, and thus it may happen that, during the lifetime of an OBDS system, its content is modified (in other terms, the system can be coupled with a new document, still using the same extraction assertions). This would clearly require to set up a mechanism for keeping the database created via spanner execution up-to-date with respect to the document “evolution”. Furthermore, this is not in the spirit of virtual data integration, which is typically performed through OBDA systems. To overcome such problems, in the next section we propose a different approach to query answering, which we base on query rewriting.

6.3 Query Answering via Query Rewriting in *DL-Lite*

In this section we study query rewriting over OBDS systems, i.e., how to answer a CQ q posed over one such system \mathcal{Y} by transforming q into a spanner whose evaluation over an underlying document **Doc** returns the certain answers to q in \mathcal{Y} with respect to **Doc**.

We start by considering OBDS systems equipped with GAV extraction assertions, and show that, in this case, CQ answering in both $DL-Lite_{\mathcal{R}}$ and $DL-Lite_{\mathcal{F}}$ OBDS systems is reducible to the evaluation of a $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$ -spanner over a document **Doc**, i.e., a spanner of the same expressiveness of those allowed in the extraction assertions.

Then we tackle the general case of GLAV extraction assertions. For this setting, we show that the above result still holds for $DL-Lite_{\mathcal{R}}$ (and its fragments) OBDS systems, and actually, we can use the same technique of the GAV case, modulo an easy transformation of the extraction assertions. We also show that, instead, this technique does not work for $DL-Lite_{\mathcal{F}}$ OBDS systems. For this case, we envisage that input queries should be rewritten in an algebra over regex formulas allowing for recursion, i.e., that allows for expressing spanners that go beyond $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$ (e.g. a spanner in RGXlog [202]).

6.3.1 GAV Extraction Assertions

Given a GAV *DL-Lite* _{\mathcal{R}} or *DL-Lite* _{\mathcal{F}} OBDS system $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ and a query q over \mathcal{Y} , we rewrite q in three steps, which we call *rewriting based on the ontology*, *rewriting based on the extraction assertions* and *reformulation into document spanners*. The first step is aimed at compiling the TBox into the query. The second is aimed at rewriting the query obtained in the first step (which is still a query expressed over the ontology) according to the assertions in \mathcal{R} . The result produced at this step is a set U of queries, having an “intermediate syntax” between CQs and spanners in $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$. The final step transforms the queries in U into document spanners in the class $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$.

Rewriting based on the ontology. For the first step we adopt the algorithm PerfectRef presented in [46], which we have briefly described in Sec. 5.2.1.

Rewriting based on the extraction assertions. The second step is through an *unfolding* method, similar to the one described in [176]. Roughly, the unfolding procedure substitutes, in all possible ways, each atom α in each query returned by PerfectRef with the spanners occurring in the left-hand side of extraction assertions referring to the predicate occurring in α . To this aim, we use the procedure *Unfolding*, which takes as input a UCQ Q and a set of extraction assertions \mathcal{R} . This procedure, for each CQ $q \in Q$, each atom $p_i(\vec{t}_i)$ in q (where \vec{t}_i is a tuple of terms, i.e., variables and/or constants), and each extraction assertion $P(\vec{v}_i) \rightsquigarrow p_i(\vec{v}_i)$, computes the most general unifier σ between $p_i(\vec{t}_i)$ and $p_i(\vec{v}_i)$, and, if such a σ exists, substitutes $p_i(\vec{t}_i)$ with $P(\vec{v}_i)$ and applies σ to the obtained formula. Note that only queries having all atoms that unify with at least one extraction assertion are completely unfolded and returned by *Unfolding*. After this step, the returned set U contains queries of the form $\{\vec{t} \mid \exists \vec{y}_1, \dots, \vec{y}_n. P_1(\vec{t}_1, \vec{y}_1) \wedge \dots \wedge P_n(\vec{t}_n, \vec{y}_n)\}$, where the target list \vec{t} may contain variables, constants, and object terms³, each P_i is a spanner, each \vec{y}_i is a (possibly empty) sequence of variables, and each \vec{t}_i is a (possibly empty) sequence of variables occurring also in \vec{t} . An example of unfolding is given in Example 6.2.

Reformulation into document spanners. The last step is carried out by the *Transform* algorithm. Roughly speaking, such an algorithm transforms the body of each query $f \in U$ into a document spanner in $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$, and returns this spanner together with the target list of f , suitably modified on the basis of certain variable substitutions needed for the transformation. In particular, *Transform* converts each join between values (expressed by multiple occurrences of a variable in the body of f) and each selection (specified through the occurrence of a constant in the body of f) into a Cartesian product between spanners (i.e., a \bowtie between spanners with no common variables), to which a string selection (i.e., $\zeta^=$) is applied. More precisely,

³With a little abuse, we continue to call \vec{t} target list, even though it does not contain only variables (as defined in Sec. 5.2.1).

Transform operates in three steps. First of all, it substitutes each constant c occurring in f with a fresh (existentially quantified) variable, say w , and adds to the conjunction in f the atom $P_c(w)$, where $P_c = \llbracket \Sigma^* \cdot w\{c\} \cdot \Sigma^* \rrbracket$, i.e., P_c is the spanner represented by a regex formula that assigns to the variable w only the spans matching with the constant c . For example, given the query $\hat{f} = \{y \mid \exists x. P_1(x, y) \wedge P_2(y, c)\}$, Transform, in its first step, reformulates \hat{f} into $\hat{f}' = \{y \mid \exists x, w. P_1(x, y) \wedge P_2(y, w) \wedge P_c(w)\}$. In the second step, for each variable z that appears more than once in the query body, Transform substitutes each occurrence of z with a fresh variable, and adds to the query body a conjunction of equalities specifying that all such fresh variables are equal to one another. If z occurs in the target list (as free variable or as argument of object terms), it is substituted with any of the newly introduced variables. In our ongoing example, \hat{f}' is reformulated into $\hat{f}'' = \{y_1 \mid \exists x, y_2, w_1, w_2. P_1(x, y_1) \wedge P_2(y_2, w_1) \wedge P_c(w_2) \wedge y_1 = y_2 \wedge w_1 = w_2\}$. In its third step, Transform iteratively applies the following rule, as long as it is applicable: let f'' be the query computed after the second step of Transform, let β be a conjunction of atoms of the form $\alpha_1 \wedge \alpha_2 \wedge x = y$ occurring in f'' , such that x occurs in α_1 and y occurs in α_2 , substitute β in f'' with $\zeta_{x,y}^{\leftarrow}(\alpha_1 \bowtie \alpha_2)$ ⁴. Finally, Transform adds a projection (i.e., π) to the query in order to project out only the variables occurring in the target list of the query, eliminates the existential quantification to obtain a syntactically correct span representation, and returns both the target list and the computed spanner. In our example, the body of \hat{f}'' is thus finally transformed into $\pi_{y_1}(\zeta_{y_1, y_2}^{\leftarrow}(P_1(x, y_1) \bowtie (\zeta_{w_1, w_2}^{\leftarrow}(P_2(y_2, w_1) \bowtie P_c(w_2))))))$, whereas the target list returned by Transform is simply constituted by the variable y ⁵.

The rewriting algorithm for the GAV case, which put together the three functions we have just described is given below. A complete example of the entire rewriting process is given in Example 6.2 (see Eq. 6.3).

⁴Note that, except for the first iteration, in subsequent applications of the rule α_1 and α_2 can be sub-formulas computed in previous iterations.

⁵In this simple example, Transform could even not explicitly return the target list, but in general the target list conveys information crucial to construct object terms that may occur in the certain answers (see Example 6.2).

Algorithm OBDS_Rewriting(\mathcal{Y}, q)

Input: OBDS $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$, such that \mathcal{T} is either a *DL-Lite* $_{\mathcal{R}}$ or a *DL-Lite* $_{\mathcal{F}}$ TBox
and \mathcal{R} is a set of GAV extraction assertions,

CQ q

Output: Sequence of terms T (i.e., a target list),
Document spanner $P \in \llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$

begin

$Q = \text{PerfectRef}(\mathcal{T}, q)$

$U = \text{Unfolding}(Q, \mathcal{R})$

$(T, P) = \text{Transform}(U)$

return (T, P)

end

Example 6.2. Consider the setting of Example 6.1, and the following query q that asks for the persons who teach a course:

$$q = \{x \mid \exists y. \text{Person}(x) \wedge \text{teaches}(x, y) \wedge \text{Course}(y)\} \quad (6.2)$$

as shown in Example 5.2, the result of $\text{PerfectRef}(\mathcal{T}, q)$ is the set Q containing the following CQs:

$$q : \{x \mid \exists y. \text{Person}(x) \wedge \text{teaches}(x, y) \wedge \text{Course}(y)\}$$

$$q_1 : \{x \mid \exists y. \text{Professor}(x) \wedge \text{teaches}(x, y) \wedge \text{Course}(y)\}$$

$$q_2 : \{x \mid \exists y. \text{Person}(x) \wedge \text{teaches}(x, y)\}$$

$$q_3 : \{x \mid \exists y. \text{Professor}(x) \wedge \text{teaches}(x, y)\}.$$

After the execution of PerfectRef , $\text{Unfolding}(Q, \mathcal{R})$ unfolds the queries in Q by using the extraction assertions in \mathcal{R} . In our example only q_3 can be completely unfolded.

For q_3 , the atom $\text{Professor}(x)$ unifies with the atom $\text{Professor}(\mathbf{prof}(x_1))$ in the extraction assertion m_1 through the unifier $\sigma' = \{x \rightarrow \mathbf{prof}(x_1)\}$, and then the atom $\sigma'(\text{teaches}(x, y)) = \text{teaches}(\mathbf{prof}(x_1), y)$ unifies with the atom $\text{teaches}(\mathbf{prof}(x_2), \mathbf{course}(y_2))$ in the extraction assertion m_2 with the unifier $\sigma'' = \{x_1 \rightarrow x_2, y \rightarrow \mathbf{course}(y_2)\}$. The unfolding will thus produce the following query⁶:

$$\{\mathbf{prof}(x_2) \mid \exists y_2. (\llbracket \rho_{prof} \rrbracket(x_2) \wedge \llbracket \gamma_{teaches} \rrbracket(x_2, y_2))\} \quad (6.3)$$

⁶Note that the application of the unifiers actually renames the variables used in the spanners.

In the above query (6.3), we are slightly abusing the notation, since, after the unfolding, the variables denote spans, and not directly the strings we are looking for. Thus, when we write $\mathbf{prof}(x_2)$ we in fact mean $\mathbf{prof}(\mathbf{Doc}_{x_2})$, where \mathbf{Doc} denotes the underlying text document. In other words, $\mathbf{prof}(x_2)$ indicates that the answer to the query consists of ground object terms with function symbol \mathbf{prof} and as argument the strings identified by the spans returned through x_2 , when the spanner represented by the regex formula in (the body of) the query (6.3) is evaluated.

Afterwards, the algorithm $\mathbf{Transform}(U)$ rewrites the above query in the spanner syntax in order to obtain a document spanner ready to be evaluated over the underlying document. $\mathbf{Transform}(U)$ first produces the following query, where no variable occurs more than once (see the description of the second step of $\mathbf{Transform}$):

$$\{\mathbf{prof}(z_1) \mid \exists y_2, z_2. (\llbracket \rho_{prof} \rrbracket(z_1) \wedge \llbracket \gamma_{teaches} \rrbracket(z_2, y_2) \wedge z_1 = z_2)\} \quad (6.4)$$

Then, it produces a representation of (the body of) the above query in the $\llbracket \mathbf{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$ syntax. More precisely, it computes the spanner, which we denote $\llbracket \rho_{transform} \rrbracket$, defined as follows:

$$\llbracket \rho_{transform} \rrbracket = \pi_{z_1}(\zeta_{z_1, z_2}^=(\llbracket \rho_{prof} \rrbracket \bowtie \llbracket \gamma_{teaches} \rrbracket)) \quad (6.5)$$

The above spanner is returned together with the target list $T = \mathbf{prof}(z_1)$. Note that, in Eq. (6.5), $\llbracket \rho_{prof} \rrbracket$ and $\llbracket \gamma_{teaches} \rrbracket$ are the spanners as defined in Examples 5.5 and 5.6, but in which the variables have been renamed by the functions $\mathbf{Unfolding}$ and $\mathbf{Transform}$. More in detail, the original variable x_1 in $\llbracket \rho_{prof} \rrbracket$ is now z_1 , and the original variable x_2 of $\llbracket \gamma_{teaches} \rrbracket$ is now z_2 . \square

We show in the following that the algorithm $\mathbf{OBDS_Rewriting}$ can be used to obtain the certain answers to a CQ q . It is indeed sufficient to evaluate over the underlying document the spanner returned by the algorithm, extract the strings corresponding to the spans produced by such an evaluation, and use them to bind the variables in the target list returned by $\mathbf{OBDS_Rewriting}$. To formalize this last aspect, we need to introduce the function \mathbf{res} . Given a document \mathbf{Doc} , a spanner P such that $\mathbf{SVars}(P) = V = v_1, \dots, v_m$, a target list $T = t_1, \dots, t_n$, such that set of variables occurring in T coincides with V , and given a (V, \mathbf{Doc}) -tuple $\mu \in \mathbf{eval}(P, \mathbf{Doc})$, we define $\mathbf{res}(T, \mu)$ as the function that returns a tuple of constants and ground object terms c_1, \dots, c_n such that each c_i is obtained as follows:

- if t_i is a constant, $c_i = t_i$;
- if $t_i = v_j$, where $1 \leq j \leq m$, $c_i = \mathbf{Doc}_{\mu(v_j)}$;
- if $t_i = \mathbf{f}_i(v_{j_1}, \dots, v_{j_k})$, where $1 \leq j_i \leq m$ for $i \in \{1, \dots, k\}$, $c_i = \mathbf{f}_i(\mathbf{Doc}_{\mu(v_{j_1})}, \dots, \mathbf{Doc}_{\mu(v_{j_k})})$.

We are now ready to provide the main result of this section.

Theorem 6.2. *Let $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ be either a $DL-Lite_{\mathcal{R}}$ or $DL-Lite_{\mathcal{F}}$ OBDS system, such that \mathcal{R} is a set of GAV extraction assertions, let \mathbf{Doc} be a document, let q be a CQ over \mathcal{Y} , and let T and P be the target list and spanner returned by $\text{OBDS_Rewriting}(\mathcal{Y}, q)$, respectively. Then, $\text{cert}(q, \mathcal{Y}, \mathbf{Doc}) = \bigcup_{\mu \in \text{eval}(P, \mathbf{Doc})} \text{res}(T, \mu)$. Furthermore, $P \in \llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$.*

Proof. The result follows from the following facts: (i) $\text{PerfectRef}(q, \mathcal{T})$ returns the perfect rewriting of a CQ q with respect to a $DL-Lite_{\mathcal{R}}$ or $DL-Lite_{\mathcal{F}}$ TBox \mathcal{T} , i.e., given an ABox \mathcal{A} , the certain answers to q over $\langle \mathcal{T}, \mathcal{A} \rangle$ coincide with the evaluation of q over \mathcal{A} , seen as a database [46]; (ii) the soundness of the procedure Unfolding to rewrite queries in GAV OBDA systems, as shown in [176], and (iii) the correctness of the algorithm Transform , which performs a purely syntactic/symbolic conversion. As for this last point, Transform simply converts CQs whose atoms use (symbols denoting) document spanners as predicates, into spanners represented in $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$. It is not difficult to see that, for each CQ q in U , the first and second step of $\text{Transform}(U)$ produce a CQ query that is equivalent to q . Then the algorithm simply turns joins in the CQ (which are expressed through equalities between variables) into Cartesian products between spanners (i.e., natural joins between spanners with no common variables), which are in fact expressed over spans. The semantics of the joins between values is then obtained through the string-selection operator applied to the result of the natural joins between spans. As a final step, the algorithm simply re-expresses the projection specified in the query through the target list by using the projection operator π . It is then easy to see that T and P respect the pre-conditions of the function res , i.e., that the set of variables occurring in T coincides with $\text{SVars}(P)$. Then, by construction we get that P belongs to the class $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^-\}} \rrbracket$. \square

Obviously, the results of Theorem 6.2 also apply to OBDS systems where the TBox is expressed in $DL-Lite_{\text{core}}$, $DL-Lite_{RDFS}$, or $DL-Lite_{RDFS}^{\neg}$.

Example 6.3. In continuation of Example 6.2, we execute $\text{eval}(\llbracket \rho_{\text{transform}} \rrbracket, \mathbf{Doc}^{\text{ex}})$, where \mathbf{Doc}^{ex} is the document in Fig. 5.1, and we obtain the span $[11, 19)$. Then, $\text{cert}(q, \mathcal{Y}, \mathbf{Doc}^{\text{ex}}) = \{\text{prof}(\text{Einstein})\}$. \square

6.3.2 GLAV Extraction Assertions

We now consider the case in which we do not pose any restriction on the extraction assertions, i.e., they are GLAV. We first consider $DL-Lite_{\mathcal{R}}$ OBDS systems, and show that one such system \mathcal{Y} with GLAV extraction assertions can be transformed into a system \mathcal{Y}' having GAV extraction assertions only and an analogous behaviour for query answering. That is, the set of certain answers to a CQ q in \mathcal{Y} with respect to a document \mathbf{Doc} coincides with the set of certain

answers to q in \mathcal{Y}' with respect to **Doc**. To this aim we exploit a transformation technique from GLAV to GAV OBDA systems presented in [74]. For the sake of completeness, we describe below the transformation from [74] (slightly adapted to the OBDS setting).

First thing, we recall that a GLAV extraction assertion r has the form $P(\vec{x}) \rightsquigarrow \Psi(\vec{x})$ where $\Psi(\vec{x})$ is a CQ, i.e., an expression of the form $\exists \vec{y}. \phi(\vec{x}, \vec{y})$, and $P(\vec{x})$ is a document spanner. Given an OBDS system $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$, we can thus turn it into a system having only GAV assertions by transforming each assertion $r \in \mathcal{R}$ as follows:

Substituting each y_i in the right-hand side of r with the term $f_i(\vec{x})$, such that f_i is a fresh function symbol, i.e., it is different from all function symbols used in the assertions in \mathcal{R} , it is different from all other fresh function symbols used to transform other extraction assertions, and it is such that $f_i \neq f_j$, for each $i, j \in 1, \dots, n$, where n is the number of variables in \vec{y} ;

We denote with $\tau(r)$ the GAV extraction assertion obtained from a GLAV assertion r through the above procedure. Given a set of GLAV extraction assertions \mathcal{R} , we define $\tau(\mathcal{R}) = \{\tau(r) \mid r \in \mathcal{R}\}$. The following theorem rephrases in the OBDS setting the analogous theorem given in [74] for OBDA systems.

Theorem 6.3. *Let $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ be a $DL\text{-Lite}_{\mathcal{R}}$ OBDS system, let q be a CQ, let **Doc** be a document, and let $\mathcal{Y}_{\tau} = \langle \mathcal{T}, \tau(\mathcal{R}) \rangle$. Then $\text{cert}(q, \mathcal{Y}, \mathbf{Doc}) = \text{cert}(q, \mathcal{Y}_{\tau}, \mathbf{Doc})$.*

With the above result in place we are thus able to compute the certain answers to a conjunctive query q in a general (i.e., GLAV) OBDS system $\mathcal{Y} = \langle \mathcal{T}, \mathcal{R} \rangle$ when the TBox is specified in $DL\text{-Lite}_{\mathcal{R}}$. It is indeed sufficient to apply the transformation τ to the set of extraction assertions \mathcal{R} , thus obtaining $\mathcal{Y}_{\tau} = \langle \mathcal{T}, \tau(\mathcal{R}) \rangle$, and then proceed with the query rewriting method described in Sec. 6.3.1, i.e., execute $\text{OBDS_Rewriting}(\mathcal{Y}_{\tau}, q)$, modulo a trivial split of each extraction assertion in such a way that the resulting set of extraction assertions contains only assertions with a single atom query in their right-hand side (as described in Sec. 6.1). Certain answers are thus obtained through the evaluation over the underlying document of the spanner returned by $\text{OBDS_Rewriting}(\mathcal{Y}_{\tau}, q)$ and the use of the coupled target list that this algorithm also returns (see Theorem 6.2), provided that tuples containing object terms constructed with the fresh function symbols introduced by the transformation τ are excluded from the answer (these tuples are indeed not certain answers, because the object terms produced by τ are denoting only the existence of individuals, which may be different in the various models).

Let us now consider $DL\text{-Lite}_{\mathcal{F}}$ OBDS systems. According to [74], Theorem 6.3 does no longer hold when the TBox is specified in that logic. This is related to the fact that functionalities present in $DL\text{-Lite}_{\mathcal{F}}$ OBDS systems (or OBDA systems) induce equalities on existential variables

which can never be satisfied by object terms introduced by the transformation τ due to the Unique Name Assumption adopted on $DL-Lite_{\mathcal{F}}$ ontologies.

For this case, we think that input queries should be rewritten in spanners specified in an algebra over regex formulas allowing for recursion, in the same spirit of rewriting algorithms for answering conjunctive queries in data integration systems in the presence of (G)LAV mappings, such as the one proposed in [82].

6.4 Final Remarks

This chapter introduced the notion of Ontology-based Document Spanning systems, which is inspired by the well-known OBDA framework, but allows to link ontologies to text documents. We studied the problem of query answering on OBDS systems in different settings, and provided algorithms and complexity results. Properly, we have shown that for both $DL-Lite_{\mathcal{R}}$ and $DL-Lite_{\mathcal{F}}$ OBDS systems with GLAV extraction assertions having $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}} \rrbracket$ -spanners in the left-hand side, answering conjunctive queries is polynomial in data complexity (i.e., in the size of the documents). Moreover, we have provided a query rewriting algorithm, namely `OBDS_Rewriting`, that allows us to reduce conjunctive query answering to the evaluation of a spanner of the same expressiveness as those allowed in the extraction assertions. This algorithm is sound and complete for $DL-Lite_{\mathcal{R}}$ OBDS systems with GLAV extraction assertions and for $DL-Lite_{\mathcal{F}}$ systems and GAV extraction assertions.

The above results obviously apply also to OBDS systems whose TBox is expressed in sublanguages of $DL-Lite_{\mathcal{R}}$ and $DL-Lite_{\mathcal{F}}$, such as $DL-Lite_{core}$, $DL-Lite_{RDFS}$ and $DL-Lite_{RDFS}^-$. Interestingly, it is straightforward to show that Theorems 6.1 and 6.2 are also valid for $DL-Lite_{\mathcal{A}}$ [177], an extension of both $DL-Lite_{\mathcal{F}}$ and $DL-Lite_{\mathcal{R}}$ with the possibility of using attributes (i.e., dataproperties in OWL jargon), and of expressing functionalities on roles and attributes together with inclusions between roles and attributes, provided that a certain syntactic restriction is satisfied (intuitively, functional roles or attributes cannot occur in the right-hand side of positive inclusions between roles or attributes). The extension to attributes clearly says that the above theorems also hold for TBoxes specified in OWL 2 QL.

The question remains about the feasibility of this approach from a practical point of view. We will deal with this topic in the next chapter, where we integrate two state-of-the-art industrial solutions to realize a tool for OBDS system management and query answering over them.

Chapter 7

Mastro System-T

In this chapter we present MASTRO SYSTEM-T, a tool for ontology-based IE developed within this thesis. MASTRO SYSTEM-T is designed according to the OBDS framework described in Chapter 6, and is able to manage OBDS systems whose TBox is expressed in one of the main *DL-Lite* logics (including *DL-Lite_A* and OWL 2 QL), and extraction assertions are GAV. It can process conjunctive queries specified over OBDS systems through a variant of the algorithm *OBDS_Rewriting* presented in Section 6.3.1.

As it can be guessed from its name, MASTRO SYSTEM-T has its foundations on two existing tools, namely SYSTEMT¹, a rule-based IE system by IBM [56], and MASTRO², a state-of-the-art OBDA engine from Sapienza University and OBDA Systems [76]. Indeed, our tool integrates the extraction abilities of SYSTEMT and the query rewriting services of MASTRO, by leveraging the modular nature of the OBDS framework, and the possibility it offers, for the *DL-Lite* case, to reduce conjunctive query answering to the execution of a spanner of the class $[[\text{RGX}^{\{\cup, \pi, \bowtie, \zeta^=\}}]$. As said at the end of Section 5.5, spanners of this class can be expressed through AQL, the core language allowed in SYSTEMT to express information extractors. Therefore, execution of such extractors is actually demanded by MASTRO SYSTEM-T to a standard SYSTEMT installation.

The thesis contributions reported in this chapter can be summarized as follows:

- We present MASTRO SYSTEM-T, describing its architecture and the components that realize its query answering process.
- Through the support of an experimentation within a real-world financial application domain, we show the benefits of the OBDS approach from a practical perspective.
- We make a performance comparison between MASTRO SYSTEM-T and CoreNLP. Namely, we re-examine the CONSOB scenario seen in Chapter 4, and use MASTRO SYSTEM-T to

¹https://researcher.watson.ibm.com/researcher/view_group.php?id=1264

²<https://www.obdasystems.com/mastro>

perform the information extraction tasks.

As mentioned above, we will discuss two different experimentations. In the first one we performed extraction over a set of real-world financial text documents from a context in which we believe our approach can bring several benefits. Specifically, we considered the Electronic Data Gathering, Analysis and Retrieval system (EDGAR)³, which is a U.S. government open repository of data about companies who are required by law to file information with the Securities and Exchange Commission (SEC). EDGAR, among its objectives, aims to increase transparency of the security market and to favour the sharing of public company's financial information. In this respect, ontologies can play a crucial role, for their ability of supporting data interpretation and sharing. At the same time, EDGAR contains terabyte of data, mostly unstructured and subject to continuous changes (about 3,000 filings per day). For these reasons it is unfeasible to process them manually. Thus, our approach can bring great benefits to the EDGAR system, for its ability of enabling automatic extraction of structured information in front to on-the-fly user requests and exploiting reasoning over ontologies to retrieve complete answers.

To this aim, we initialized MASTRO SYSTEM-T providing an ad-hoc ontology written for this domain and a series of extractors to populate the ontology. We finally selected a set of queries to evaluate the effectiveness of our approach and demonstrate the advantages that can be achieved in this real-world financial application.

In particular, our experiments show that reasoning over the ontology is crucial to increase the recall in IE, properly +9.8% from the baseline, since reasoning over the ontology allows us to trigger extraction assertions that would be overlooked in query answering without reasoning.

In the second experimentation, we mainly aimed at evaluating MASTRO SYSTEM-T in terms of execution time. For this purpose, we carried out some more extensive tests related to the CONSOB scenario (see also Section 4.1). Specifically, by selecting a set of user queries, posed over the CONSOB ontology, and measuring their execution time, we show how MASTRO SYSTEM-T significantly increases the speed of the information extraction process compared to the CoreNLP-based implementation described in Chapter 4.

The rest of the chapter is organized as follows:

- In Section 7.1, we briefly describe the MASTRO tool for OBDA.
- In Section 7.2, we introduce SYSTEMT, whose theoretical foundation lies in the framework of Document Spanners [123], summarized in Section 5.5. In this chapter, we in particular focus on the Annotation Query Language of SYSTEMT.
- In Section 7.3, we present the novel MASTRO SYSTEM-T tool, introducing its architecture and its main components, and showing the query answering workflow.

³<https://www.sec.gov/edgar/searchedgar/companysearch.html>

- In Section 7.4, we show an application of MASTRO SYSTEM-T over the financial domain EDGAR. Then, we discuss the more extensive experimentation we carried out on the CONSOB domain.

7.1 Mastro

MASTRO is a Java tool for ontology-based data access formerly developed at Sapienza Università di Roma, and currently carried on by OBDA Systems s.r.l., a Sapienza start-up company.

MASTRO is able to manage OBDA systems (cf. Section 5.4) whose TBox is specified in one of the *DL-Lite* languages or in OWL 2 QL, the OWL 2 counterpart of *DL-Lite_R*. The mapping in MASTRO is given as a set of GAV assertions, which use object terms in their right-hand side (cf. Section 5.4.1), and whose left-hand side contains SQL queries posed over a relational database, managed by an external and autonomous DBMS. Several concrete syntaxes have been proposed in the literature to express mappings. MASTRO uses a proprietary one based on XML, but can also take as input and export mappings written in the W3C standard R2RML syntax [164]. For the sake of presentation, in this chapter we write mappings in a simplified syntax, according to which the right-hand side of mapping assertions is given in terms of SPARQL basic graph patterns (cf. Section 2.1.3) that make use of R2RML templates to construct object identifiers (which correspond to object terms described in Section 6.1).

An example of OBDA specification managed by MASTRO is provided below (the ontology is given as a set of OWL 2 QL triples).

Example 7.1. Consider the following MASTRO TBox:

```

 $\theta_1$ ) :Company a owl:Class .
 $\theta_2$ ) :PublicCompany a owl:Class .
 $\theta_3$ ) :PrivateCompany a owl:Class .
 $\theta_4$ ) :comp_name a owl:DataProperty .
 $\theta_5$ ) :has_revenue a owl:DataProperty .
 $\theta_6$ ) :PublicCompany rdf:subClassOf :Company .
 $\theta_7$ ) :PrivateCompany rdf:subClassOf :Company .
 $\theta_8$ ) :has_name rdf:domain :Company
 $\theta_9$ ) :has_revenue rdf:domain :Company

```

In this ontology there are three classes :Company(θ_1), :PublicCompany(θ_2) and :PrivateCompany(θ_3). We have also two attributes :comp_name(θ_4) and :has_revenue(θ_5). θ_6 and θ_7 assert that a :Company can be a :PrivateCompany or a :PublicCompany. Finally θ_8 and θ_9 assert that the attributes :comp_name and :has_revenue refer to the entities of the class :Company.

The external database to which the TBox is linked contains information on U.S. institutional offices, gathered for simplicity in the following table:

TAB table

ID	NAME	REVENUE	...	TYPE
P1	McKinsey & Company	\$10.5 B	...	Private
P2	Bloomberg	\$12.5 B	...	Private
P3	Apple	\$274.2 B	...	Public
...
P341	IBM	\$5.6 B	...	Public

The linkage is done through two mapping assertions. The first one is:

```

SELECT TAB.ID as id, TAB.NAME as name,      :{id}#ID a :PublicCompany,
       TAB.REVENUE as rev                    ~> :{id}#ID :has_name name,
FROM TAB                                     :{id}#ID :has_revenue rev
WHERE TAB.TYPE = 'Public';

```

The above mapping assertion virtually populates the class `:PublicCompany` and its attributes `:has_name` and `:has_revenue`, by using the SQL query in the left-hand side. In the right-hand side of the assertion, `:{id}#ID` is a *template* [73] used to specify how to construct objects from the tuples of values retrieved by the SQL query. Intuitively, for every value assigned to `id` by the SQL query, an object identifier is constructed by substituting `{id}` with the value in `TAB.ID`.

The second mapping assertion instead virtually populates the class `:PrivateCompany`, as described below:

```

SELECT TAB.ID as id, TAB.NAME as name,      :{id}#ID a :PrivateCompany,
       TAB.REVENUE as rev                    ~> :{id}#ID :has_name name,
FROM TAB                                     :{id}#ID :has_revenue rev
WHERE TAB.TYPE = 'Private';

```

□

The main reasoning service in MASTRO is query answering. Answering unions of conjunctive queries over OBDA instances managed by MASTRO is performed through a query rewriting technique similar to the one described in Section 5.2.1, coupled with an unfolding step (in fact, MASTRO implements an optimized version of PerfectRef based on the Presto algorithm [184], plus several additional optimizations aimed at reducing the size of the perfect reformulations it compute).

We show the MASTRO query answering process through the following example (UCQs taken as input by MASTRO are specified in SPARQL):

Example 7.2. Consider the following SPARQL query on the MASTRO system in Example 7.1:

```

SELECT ?X
WHERE {
    ?X a :Company
}

```


First of all MASTRO translate the query into a union of SPARQL queries, reasoning on the fact that each `:PublicCompany` and each `:PrivateCompany` is a `:Company`. Then, MASTRO generates the following SQL through an unfolding step based on the mapping specification.

```

SELECT TAB.ID as id
FROM TAB
WHERE TAB.TYPE = 'Public'

UNION

SELECT TAB.ID as id
FROM TAB
WHERE TAB.TYPE = 'Private'

```

The tuples resulting from the evaluation of this query on the database are then translated, through the templates, in entities that instantiate the ontology. In the end, MASTRO returns the following result:

?x
: 'P1' #ID
: 'P2' #ID
: 'P3' #ID
...
: 'P341' #ID

□

For more documentation about MASTRO, we refer the reader to [149, 62, 76, 42]

7.2 SystemT

Declarative Information Extraction explores the relationship between the extractors, i.e., rules that produce tuples of spans (intervals of character indexes within the document), and their manipulation using relational algebra operators [198] (see also the framework of document spanners described in Section 5.5).

We recall that a *span* is a pair of indexes that identify sub-strings of a given document. For example, given the string `IBM Almaden`, the spans `[0, 3)` and `[4, 11)` identify the substrings `IBM` and `Almaden`, respectively. The extractors usually process the text via transducers (e.g., regular expressions with capture variables or a dictionary lookup), and produce a set of spans. They can be combined together using relational algebra operators (usually union, projection, and join) to formulate new extractors.

The annotation query language (AQL), the language at the basis of SYSTEMT [131, 56], implements this idea by allowing to define extractors using a friendly declarative language with SQL-like syntax. In addition to the classic relational algebra operators working over the spans, SYSTEMT allows to use operators specifically defined for IE tasks. These include:

- *Regular Expression matcher.* Given a regular expression, the matcher provides as output the set of spans corresponding to the matches.
- *Dictionary matcher.* Given a dictionary consisting of a set of words or phrases, it produces as output a span for each match of dictionary entries found in the text.
- *Consolidate.* It allows to choose how to handle match duplicates or overlapping spans. This choice is generally made by deleting annotations or by merging the spans.

SYSTEMT is highly optimized through both a mechanism to rewrite AQL extractors into smaller ones and an execution plan made specifically for declarative information extraction.

We introduce the syntax of System-T through the following example of AQL Extractor.

Example 7.3. Consider the following document **Doc**:

```
The revenue of the company Apple grows fast.
The new CEO of the company IBM is Arvind Krishna
```

Now let's consider the following AQL extractor

```
create dictionary Comp_dict as('company');

create view View_Comp_prefix as
extract dictionary 'Comp_dict'
  on D.text as Comp_prefix
from _Document D;

create view ViewComp as
extract pattern (<I.Comp_prefix>) (<C.token>)
  return group 0 as Comp
  return group 1 as prefix
  and group 2 as comp_name
from View_Comp_prefix I,
Generic_Export.token C;
```

This extractor is composed by two views, namely `View_Comp_prefix` and `ViewComp`. Each view in AQL is responsible to map spans to variables. Specifically, `View_Comp_prefix` assigns to the `Comp_prefix` variable all spans that match the word 'company', using the dictionary `Comp_dict`. `ViewComp` uses the pattern `(<I.Comp_prefix>) (<C.token>)`, built using the previous view, that matches all the pairs of tokens where the first word is 'company'. Then, `ViewComp`, through a syntax for groups similar to the one of the POSIX regex, maps the full match of the pattern

to the variable `Comp`, the first group of the pattern (`<I.Comp_prefix>`) to the variable `prefix`, and the second group of the pattern (`<C.token>`) to the variable `comp_name`.

The result of evaluation of the `ViewComp` over the document `Doc` is reported in the following table:

Comp	prefix	comp_name
[19, 32)	[19, 26)	[27, 32)
[60, 71)	[60, 67)	[68, 71)

□

For more technical details about the architecture, the optimization, the syntax of AQL and the theory behind SYSTEMT, we refer the reader to [56] and [86].

7.3 Mastro System-T

MASTRO SYSTEM-T is a Java tool for OBDS developed within this thesis and enabled by a joint study agreement between IBM Almaden and Sapienza University of Rome.

MASTRO SYSTEM-T leverages the results given in Chapter 6, and thus manages OBDS systems in which the TBox is expressed in a *DL-Lite* logics (including OWL 2 QL) and the extraction assertions are GAV. As said, under this setting, conjunctive query answering can be solved through query rewriting (cf. the algorithm `OBDS_Rewriting` of Section 6.3).

Since MASTRO SYSTEM-T is based on MASTRO, it also inherits from MASTRO the concrete syntax used for the definition of some of its components. In particular, UCQs issued by users are encoded in SPARQL and the TBox is expressed through any of the standard OWL 2 syntaxes [161]. As for extraction assertions, their concrete syntax is very similar to the syntax used in MASTRO for the mapping. The main difference is that the body does no longer contain an SQL query but rather an AQL extractor.

Example 7.4. Consider again the TBox defined in Example 7.1. The following one is an example of extraction assertion which imports the view `ViewComp` of Example 7.3:

```
import view ViewComp from module m;
create view_Comp as
  select GetText(C.comp_name) as name, ~> :{n}#ID a :Company,
         C.comp_name as n                :{n}#ID :has_name name
  from m.ViewComp C;
```

This assertion specifies how to generate instances of the class `:Company` and its attribute `:has_name` starting from the spans returned by the AQL extractor evaluated on a given document. □

In the following we describe the architecture of our tool and the process of query answering by rewriting implemented in MASTRO SYSTEM-T.

7.3.1 System Overview

The architecture of the tool is illustrated in Figure 7.1. We soon notice that whereas MASTRO

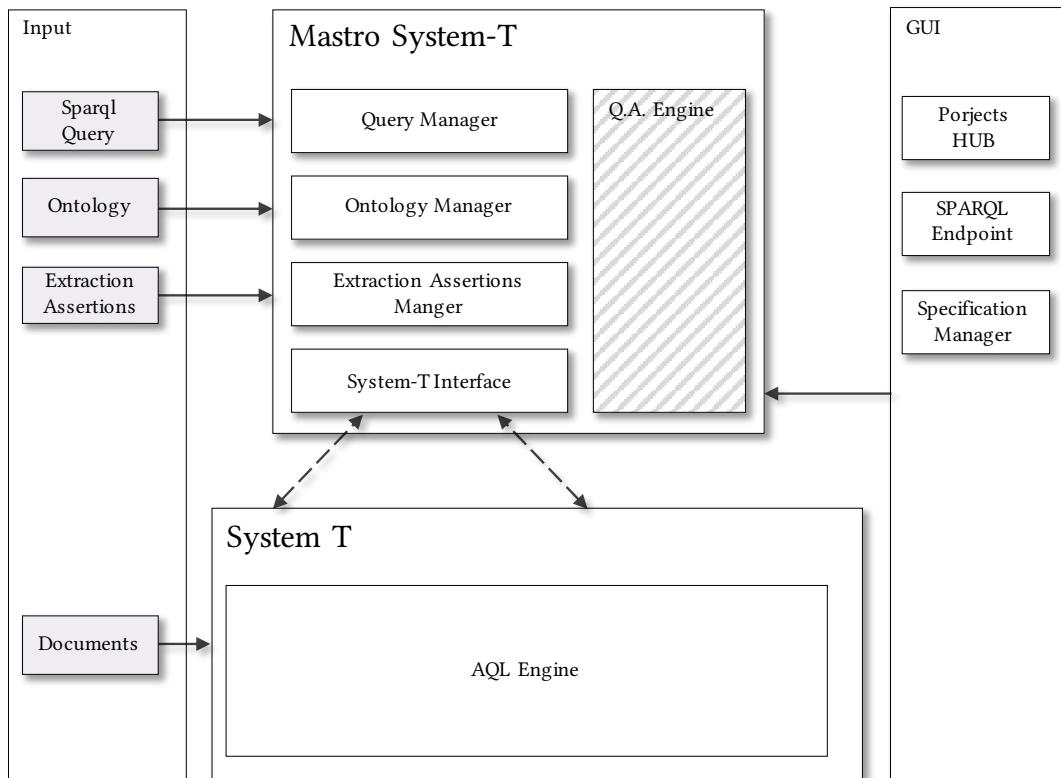


Figure 7.1. Architecture of MASTRO SYSTEM-T

SYSTEM-T takes as input an OBDS specification (i.e., the Ontology and the extraction assertions), as well as a user query, documents are instead given as input to an instance of SYSTEMT, which is external to MASTRO SYSTEM-T and suitably interfaced with it through a dedicated software module (System-T Interface). Indeed, MASTRO SYSTEM-T does not directly manipulate the source documents, but, for each user query, it produces the specification of the AQL extractors that will be then executed by SYSTEMT to obtain the spans from which to construct the query answers, as described in more details later on. Ontology axioms are serialized in an internal in-memory structure by the ‘Ontology Manager’ module. SPARQL queries are instead parsed and managed by the ‘Query Manager’ module. Both components are based on the Apache Jena framework⁴, and are indeed inherited by the system MASTRO. Extraction assertions are encoded

⁴<https://jena.apache.org/>

in a tailored concrete syntax and handled by the ‘Extraction Assertions Manager’.

MASTRO SYSTEM-T also provides a purpose-built graphical interface coded in Vue.js⁵ to manage projects, inspect the specification, and query the system through a SPARQL endpoint (see Figure 7.2).

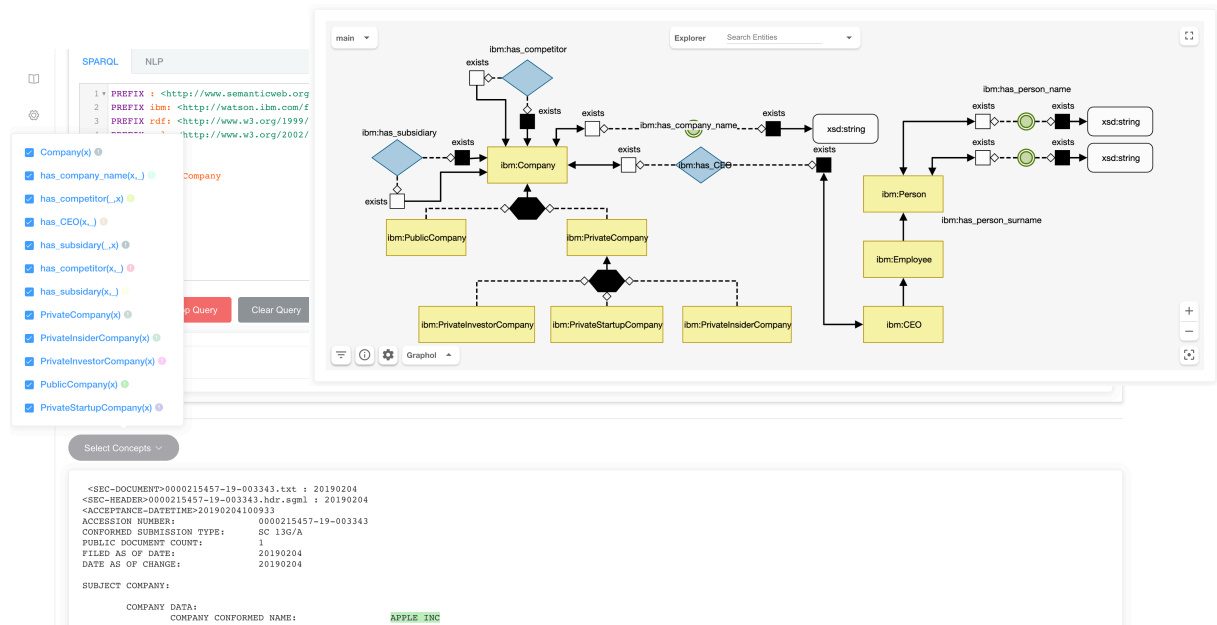


Figure 7.2. User Interface

The ‘QA Engine’ is the core component of MASTRO SYSTEM-T which implements the query answering service offered by our tool. It is in turn composed by submodules devoted to specific phases of the query answering process. Such modules, as well as the workflow for query answering are described in more detail below.

7.3.2 Query Answering

MASTRO SYSTEM-T computes answers to user queries posed over the ontology by transforming them into AQL extractors and delegating their execution to SYSTEMT. With this approach there is no need to apply every extraction assertion in the OBDS specification to the document in order to materialize all the facts instantiating the ontology. MASTRO SYSTEM-T triggers only the extraction assertions useful to generate the answers to the user query at hand and returns always the most updated answer possible. This is particularly suited for dynamic scenarios,

⁵<https://vuejs.org>

like the EDGAR one, where source documents change frequently and query answers cannot be computed on the basis of outdated materializations.

In a nutshell, the query transformation process realized by the ‘QA Engine’ includes an ontology-based query rewriting phase, and a further reformulation step that uses extraction assertions to transform the query over the ontology into a set of extractors to be executed over the text documents. The complete workflow is illustrated in Figure 7.3 and briefly explained in the following.

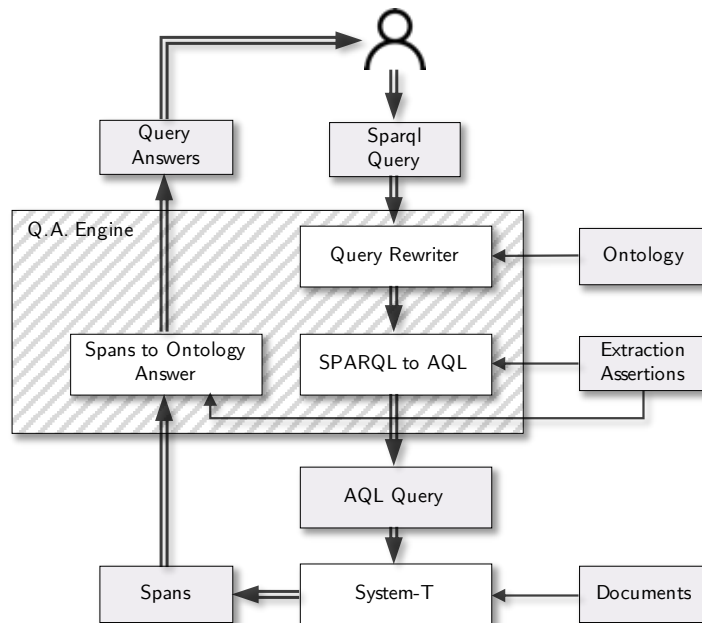


Figure 7.3. Query Answering Workflow

(Ontology-based) Query Rewriting. Given a SPARQL query q and a TBox \mathcal{T} , the first step is based on a compilation of \mathcal{T} into q , so that after the rewriting the ontology can be disregarded in the next query processing phases. This process is carried out by the module ‘Query Rewriter’ and the result is a UCQs. ‘Query Rewriter’ is in fact a module already existing in MASTRO, which we could reuse as is, by virtue of the modular nature of our query processing algorithm (cf. Section 6.3). As already said, ontology-based query rewriting in MASTRO (and thus in the module ‘Query Rewriter’) is realized through an optimization of the PerfectRef algorithm called Presto (see also Section 7.1).

Translation into AQL. The second step consists in translating the UCQs resulting from the *Query Rewriting* phase into a set of AQL extractors. Roughly, the module ‘SPARQL to AQL’ substitutes, in all possible ways, each atom a in each query returned by the previous step with the extractors occurring in the left-hand side of extraction assertions referring to the ontology predicate occurring in a . This translation takes into account all possible selections and joins

inside the queries and transform them into string selections and join between spans, using the special AQL predicate *Equals*. Notice that the module ‘SPARQL to AQL’ essentially implements the *Unfolding* and *Transform* functions of the algorithm *OBDS_Rewriting*.

AQL extractor Evaluation. Now the AQL extractor is given as input to *SYSTEMT*. The AQL extractor goes through a series of optimization steps in order to increase its performance in terms of execution speed. The results returned by *SYSTEMT* is a set of tuples of spans, together with the strings that they identify on the underlying document.

From Spans to Ontology answers. Given the set of spans and associated string values returned by previous phase, the module ‘Span to Ontology answers’ computes the objects instantiating the ontology by using the templates declared in the extraction assertions, and gives back the answers to the user.

7.4 Case Studies

In this section we describe our experiments. We first talk about the one carried out on the EDGAR scenario, and then we focus on the CONSOB one.

7.4.1 EDGAR

Electronic Data Gathering, Analysis, and Retrieval system (EDGAR) is a public platform where companies acting in U.S. are required by law to enter a range of information for government controls. EDGAR is mainly composed by a large amount of raw text subject to significant updates over time. Since human effort is not sufficient to process this amount of data, there is the need for a mechanism that can automate the extraction phase by always providing the most update information and allowing data sharing and standardization. To carry out our experiments, we have designed a financial domain ontology tailored for the EDGAR system.

The ontology contains 75 classes, 56 object properties, 214 data properties, and 907 axioms. Here we show a small excerpt of the ontology, built around the concept *Company*, and reported in Figure 7.4 in *GRAPHOL*. The figure shows 9 classes (concepts), 3 objectproperties (roles), and 3 dataproperties (attributes). In words, the ontology is saying that a company can be either public or private. Also, investor, startup, and insider companies are private companies (and therefore are companies), and are pairwise disjoint. Each company has a name (that is a string), and a CEO, and can have a subsidiary or a competitor company. A CEO is always associated to the company she/he directs, and is an *Employee*, which is a *Person*, and as such has a name and a surname (two strings).

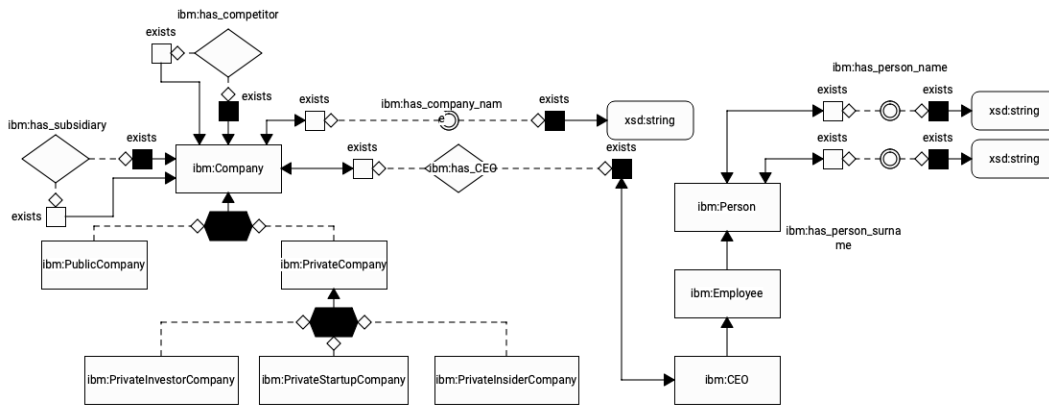


Figure 7.4. Ontology

As for the underlying dataset, we have selected 249 text documents⁶ containing data about 5 companies in the Fortune top 500, for a total dataset size of 250 MB.

We have written 30 extraction assertions linked to an equal number of ad-hoc AQL extractors. Finally, in order to test the performance of our tool, with the main goal of highlighting the role of reasoning in the extraction phase, we selected 3 simple but representative queries described below.

The first query asks MASTRO SYSTEM-T to retrieve all the companies.

```
SELECT ?X WHERE {
    ?X a :Company}
```

For this query, we obtained the most interesting results, reported in Table 7.1, where we give the values of precision and recall that we obtained by executing the query either without or with reasoning, respectively, shown under the columns ‘Base’ and ‘Reasoning’. Values without reasoning have been obtained by disabling the ontology-based query rewriting phase discussed in Section 7.3.2. In this way, we did not include in the results answers inferred by virtue of the axioms of the ontology, which has been thus considered in this case as a flat schema.

	Base	Reasoning	Gap
Precision	81.82%	82.71%	+0.89%
Recall	66.8%	76.26%	+9.46%
F-Measure	73.59%	79.35%	+5.76%

Table 7.1. Company query results

We note that the increase in recall in this case is due to the fact that the rewriting of the

⁶At the following <https://www.sec.gov/Archives/edgar/data/51143/000104746918001117/a2233835z10-k.htm>, there is an example of document regarding the IBM Company

query is quite large, and includes, among other queries, also the queries

```
SELECT ?X WHERE {?X :has_competitors ?Y}
SELECT ?X WHERE {?X :has_subsidary ?Y}
```

We could test that the extractors associated by the extraction assertions to the roles `:has_subsidary` and `:has_competitors` perform particularly well in terms of amount of answers retrieved, but also of precision, which remains almost the same as the base case.

We obtained a similar behaviour for the query

```
SELECT ?X WHERE {
  ?X :has_CEO ?Y}
```

This query asks for the individuals participating in the domain of the relationship `:has_ceo`. The execution of the query without reasoning makes use only of the extractor connected to `:has_ceo`, which in our dataset extracts only 5 individuals. By enabling the reasoning, the number of extracted individuals increases to 2582.

The last query we show requests all the persons:

```
SELECT ?X WHERE {
  ?X a :Person}
```

In this case, the reasoning does not produce significant improvements, indeed we obtained only the 0.1% increment in recall with respect to the base case. This is due to the fact that the extractor mapped to the concept `Person` is already able to populate this class in an almost complete manner. That is, it looks also for strings matching patterns suited to identify special types of persons, i.e., those instantiating also (complex) classes subsumed by `Person`.

On the other hand, defining “complete extractors” means to manually encode in them the reasoning that could be automatically done over the ontology, and this is clearly not always possible. In particular, in our ontology, `:Person` has a less articulated hierarchy and connections with other classes with respect to the class `:Company`, and this is why extraction assertions are already able to directly provide almost all its instances.

7.4.2 CONSOB

In this part of the chapter we describe our second experiment, i.e., the one in which we have used MASTRO SYSTEM-T for the CONSOB domain described in Chapter 4.

In order to compare the time performance of MASTRO SYSTEM-T with that of our CoreNLP-based implementation described in Chapter 4, we translated into AQL extractors the extraction rules that we have defined for CoreNLP. Then, based on such AQL extractors, we built the

MASTRO SYSTEM-T extraction assertions and linked them to the predicates of the domain ontology showed in Figure 4.3.

As an example, we report below the AQL extractor (Example 7.5) and the extraction assertion (Example 7.6) used to map the attribute `:identification_code` of the CONSOB ontology.

Example 7.5. The following AQL extractor is used to extract from text documents the ISIN of financial products (i.e., the code that identifies a product).

```

module ISIN;
import view _Document from module IEWTDocument as _Document;
import view Prodotto from module section as sectionProdotto;

create view ISINToBeConsolidated as
  extract
    regex /([A-Za-z][A-Za-z][A-Z0-9]10)/
    on sP.sectionProdotto as isin
    from sectionProdotto sP;

create view ISIN as
  select Min(i.isin) as isin
  from ISINToBeConsolidated i;

export view ISIN;

```

Note that AQL provides particularly useful constructors for IE tasks that are not expressible in CoreNLP, effectively making SystemT's language more expressive than Stanford's one. In fact, the above extractor, compared to its counterpart in TOKENSREGEX, uses an AQL construct called `Min` that allows to select the first occurrence of the match, thus avoiding the search in a more specific portion of the text. □

Example 7.6. Consider now the following extraction assertion:

```

import view ISIN from module ISIN;
import view ProductID from module IDs;
create view _populateISIN ~> :{id}#ID :identification_code isin
  select s.isin as isin, p.id as id
  from ISIN s, ProductID as p;

```

As said, this extraction assertion specifies how to virtually populate the attribute `:identification_code`, which relates every product with its ISIN, obtained using the AQL extractor in Example 7.5. □

Regarding the dataset, we used the third one considered in the CONSOB scenario (i.e., the largest one), whose characteristics are summarized in Table ??.

We then defined ed executed four queries. In Table 7.2, we report such queries, a description for each of them, the execution time, and the number of tuples obtained from their evaluations. We point out that, also in this experiment on the CONSOB scenario, the bottleneck is the

Query ID	Query	Description	Time of execution	N. tuples
Q_1	<pre>SELECT ?X ?Y WHERE { ?Z :web_site ?X ?Z a :Financial_Entity ?Z :name ?Y }</pre>	Return the website and the name of all financial institutions (i.e., instances of the class <code>Financial_Entity</code>)	$\sim 23m$	14001
Q_2	<pre>SELECT ?X ?Y WHERE { ?Z a :IBIP ?Z :has_name ?X ?Z :identification_code ?Y }</pre>	Return the names and the identification code of IBIP products	$\sim 16m$	518
Q_3	<pre>SELECT ?X ?Y WHERE { ?X :issue_date ?Y ?X a :Derivatives }</pre>	Return all the Derivatives and their issue dates	$\sim 15m$	39
Q_4	<pre>SELECT ?X ?Y WHERE { ?X a :Document ?X :alert :Y ?Z :associated_with ?X ?Z a :Securities }</pre>	Return all the documents and their alerts if they are associated with a Security product	$\sim 45m$	12542

Table 7.2. Execution time query

transformation of the original PDF documents into plain text, as previously mentioned in Chapter 4 for the coreNLP-based implementation. The time taken by this task is the same for all the queries and is around 4 hours. We have not included this amount of time in the figures given in Table 7.2, where we thus exhibit times that correspond to query evaluation as if the source documents (i.e., the KIDs) were directly in plain text format. We want to also point out that the query Q_4 is the one that takes more time. The explanation resides in the fact that, among the four queries we executed, Q_4 is the one that triggers the greatest number of AQL extractors and returns as output a high number of tuples.

Overall, the times we have obtained may seem considerable and not suitable for an interactive usage. However, these times must be compared with the total extraction time required in a traditional IE approach, which actually requires populating the entire ontology before running the queries (in the tested scenario total extraction time is in the order of hours, as discussed later on). Whereas, from the one hand, it is true that over the populated ontology not one but all queries can be executed, on the other hand, as in all materialized approaches to data integration, one has to be aware that queries are not run over fresh data, and refresh costs have to be taken into account. We further remind the reader that the database of documents considered in our

experiment is particularly large (more than 14000 KIDs). With smaller document datasets, response times of less than one minute can be easily achieved. At the same time, it has to be considered that our experiment is one of the first of its kind, and that MASTRO SYSTEM-T is still susceptible to several optimizations, which are currently under study.

We now highlight the main practical differences we have experienced with respect to the coreNLP-based implementation.

- We first remark that MASTRO SYSTEM-T allows for directly processing conjunctive queries over the ontology, whereas our implementation based on coreNLP does not offer this functionality. To obtain the same service in that case, we have to give the ontology we have populated through our pipeline (both the TBox and the ABox returned by the extractor component of our tool) as input to a system for ontology-mediated query answering and OBDA, such as MASTRO, and rely on its query processing functionalities. It is clear that the entire approach is more involved, and that the two tools it is based on (i.e., MASTRO and the coreNLP-based extractors) are not really integrated with one another, as instead is realized in MASTRO SYSTEM-T. Furthermore, and above all, this approach is not virtual, that is, information extraction cannot be done at query time.
- We then point out that MASTRO SYSTEM-T can be even used to materialize the ABox instantiating the TBox of the OBDS system. This can be done by triggering all extraction assertions. We asked MASTRO SYSTEM-T to compute such ABox for the CONSOB case, and our tool accomplished the task in 5 hours and 25 minutes. Compared with the time needed by the coreNLP-based implementation to materialize the result of the extraction (which, notice, is not exactly the ABox materialized by MASTRO SYSTEM-T), we measured a reduction of more than 2 hours and 30 minutes (for both cases we are also considering the time to transform PDF files into plain text files). This testifies that the solution we developed may allow for important gains in time performances.
- The advantages of MASTRO SYSTEM-T with respect to coreNLP concern also with the usability of the system. Properly, by virtue of the high expressive power of the AQL language, MASTRO SYSTEM-T allows us to considerably simplify the writing of the extraction rules and to strongly reduce the use of custom (e.g., Java) code to go beyond the abilities of the language (to which instead we had to often resort in the coreNLP-based approach).

To make the last observation more concrete, we provide below an example of extraction rule in AQL.

Example 7.7. Consider the following (portion of) AQL extractor:

```
import view nameProduct from module nameProduct as nameProduct;
...

create dictionary SECURITIES_CERTIFICATI_BENCHMARK
from file 'dictionaries/SECURITIES:CERTIFICATES:BENCHMARK.dict';
...
create view classificationName as
select
case
...
when ContainsDict ('SECURITIES_CERTIFICATI_BENCHMARK', np.nameProduct)
    then 'SECURITIES:CERTIFICATI:BENCHMARK'
...
else 'Unknown'
as classificationName
from nameProduct np;
```

This extractor using the dictionary `SECURITIES_CERTIFICATI_BENCHMARK` creates annotations useful to class to classify the products i.e. to generate the instances of the class `Benchmark`, that is a subclass of the concept `Product`. The AQL code shown is a portion of a more general one that on the basis of some dictionaries classifies all documents and not just benchmarks. We refer the reader to the following links to view the entire extractor (<https://github.com/Scafooo/Phd-Thesis/tree/main/Chapter%205/SystemT/src/tipologiaNome>) and its counterpart in CoreNLP (https://github.com/Scafooo/Phd-Thesis/blob/main/Chapter%204/Tool/resources/rules/10_tipologiaNome.rule.txt).

□

Chapter 8

Entity Resolution

In this chapter we deal with entity resolution (a.k.a. entity matching, entity reconciliation, record linkage, duplicate detection), i.e., the problem of establishing whether different entities in the data refer in fact to the same real-world object [18, 130, 170, 52]. Entity resolution is a very relevant issue in data management, especially when the information is integrated from different heterogeneous sources, possibly semi-structured or unstructured, e.g., extracted from text documents. This latter scenario turns out to be particularly challenging, since names of entities have to be resolved from free text, and thus the task is hampered by the multiple ways an entity can be described, even in the same document. Thus, we believe that the study described in this chapter nicely complements the investigation on Ontology-based IE carried out in this thesis, since it deals with an issue that is particularly critical in the context of IE.

In the following, we investigate entity resolution on DL ontologies equipped with concrete domains and attributes (i.e., binary predicates representing relations between concepts and values from a predetermined value-domain), such as *age* or *height*. In other terms, we here explicitly deal with the logical counterpart of OWL datatype, differently from what we have done in Chapter 6, where attributes have not been considered in the formal treatment, but extension of the results we obtained to languages allowing for attributes, as OWL 2 QL, resulted straightforward.

In our investigation, we abstract away from possible mappings connecting the ontology to external data sources, as in Ontology-based Data Access (see Section 5.4) or in Ontology-based Document spanning systems (Chapter 6). We thus assume that data directly instantiate ontology predicates, i.e., are stored as facts in the ontology ABox. We however pursue a declarative approach, which can be easily generalized to the presence of linking mechanisms between the ontology and the data sources (e.g., by materializing the result of evaluating mappings or extraction rules over the sources).

To incorporate an entity resolution mechanism in our framework, we introduce the notion of

KER system, i.e., a system extending the standard TBox and ABox components of a DL ontology with a third component containing *entity resolution rules*, i.e., logical horn rules specifying the conditions under which different entities have to be considered the same individual. As formally described in the following, the rules we introduce correspond to a form of equality-generating dependencies (egds) [2]. Then, we study *query answering* on KER systems and propose a foundational approach to address this problem.

More in detail, in the following sections we will proceed as follows:

- We first study the case of TBoxes expressed through tuple-generating dependencies (tgds). This allows us to define a very general framework that captures typical positive inclusion assertions used in DLs. We formally define the components of a KER system, and in particular the form of entity resolution rules we couple with the TBox.
- We give the semantics of KER systems in terms of special interpretations, which, besides the domain of interpretation and the interpretation function, are characterized by an *equivalence relation* allowing to interpret ontology concepts with sets containing equivalence classes that gather together resolved entities, i.e., entities that are denoting the same real-world object.
- In the spirit of previous work on tgds and egds, and fragments thereof, (e.g., [89, 39, 33, 46]), we provide the definition of *universal model* for a KER system, i.e., a model for which there is an homomorphism towards every other model of the KER system, and show that conjunctive query answering over a KER system reduces to query evaluation over a universal model.
- We show how to construct a universal model through a tailored chasing technique depending on TBox tgds and entity resolution rules. We remark that in our approach *the chase never fail*, since we can always merge equivalence classes interpreting entities each time an entity resolution rule infers that two entities have to be equated. At the same time, the chase is in general infinite, since in the framework we do not impose conditions on tgds and entity resolution rules to ensure termination. However, for the cases in which it is finite, it can be constructed in polynomial time with respect to the size of the ABox (data complexity).
- We then introduce functional dependencies in the ontology language, as those allowed in *DL-Lite_F*. Whereas functionalities on roles are assimilable to entity resolution rules, functionalities on attributes may enforce equalities on different values (e.g., different numbers or strings), which would cause a KER system to be inconsistent. To cope with this situation, we interpret functionalities on attributes as *matching dependencies* [23, 92], i.e., rules establishing conditions under which values have to be matched (rather than equated).

We resolve matching through a general function that takes the union of the values to be matched, in the spirit of the *union class of match and merge functions* considered in [18].

- We finally revise all main notions and constructions studied for the case of **KER** systems with tgds only (e.g., the universal model and the chase), show that also in the presence of functional attributes conjunctive query answering can be still reduced to query evaluation over the chase, and that when the chase is finite, it can be constructed in polynomial time in the size of the ABox.

We point out that, in our framework, answers to queries are tuples in which each component is either a set of entities or a value, when the TBox of the **KER** system is a set of tgds only, or tuples in which each component is a set of either entities or values, when the TBox of the **KER** system is a set of tgds and functionalities on attributes. Thus, the answers we get provide a compact report of all resolved entities and of all values associated to the same entity (or equivalent ones) by a functional attribute. This is a distinguishing feature of our approach, which we think is particularly useful from the practical point of view. Specifically, we are able to provide, in polynomial time, meaningful answers to conjunctive queries also in the presence of multiple values associated by functional attributes to the same entity, that is, in a situation in which standard ontology semantics would interpret the system as inconsistent (and thus query answering would become meaningless). We also point out that other declarative approaches to inconsistency management resort to reason over the repairs of the data instance [6, 22], which in the context of ontologies are typically defined as inclusion-maximal consistent subsets of the ABox [25]. In such approaches, conjunctive query answering is intractable, unless some approximate notion of repair is adopted [133], which however reduces the amount of consistent answers that can be obtained from an inconsistent ontology. Furthermore, the answers we return are different from both consistent answers (i.e., answers obtained through skeptical reasoning over all repairs and the TBox) and possible answers (i.e., answers inferred by any repair coupled with the TBox [25]). Instead, when an entity is associated to more than one value by a functional attribute, we pack together all such values (e.g., if a person p is aged both 35 and 38 we merge such conflicting values into the set $\{35, 38\}$ and say that such set is the age of p), and provide in this way the user with a more precise account of the inconsistencies that occur in the data (e.g., when asking for the age of p , she gets the answer $\{35, 38\}$, i.e., the set of all the alternatives present in the data).

The rest of the chapter is organized as follows:

- In Section 8.1 we give some technical preliminaries needed for this chapter;
- In Section 8.2 we introduce our framework, and provide syntax and semantics of **KER** systems, with TBoxes specified as sets of tgds only;

- In Section 8.3 we give the definition of Universal Model of a KER system;
- In Section 8.4 we define the notion of certain answers in our framework and show that computing the certain answers to a conjunctive query can be reduced to query evaluation over a Universal Model;
- In Section 8.4 we show how to construct a Universal Model through a (possibly infinite) sequence of chase steps tailored to the kind of rules and semantics adopted in our framework;
- In Section 8.6 we extend our treatment to the full setting, i.e., we add functionality assertions to the TBox language.

8.1 Preliminaries

In this section we provide some preliminary notions useful for our technical development.

8.1.1 Equivalence Classes

In the following we briefly review some mathematical basics on equivalence classes. For more details we refer the reader to [125].

Let be \mathcal{Z} a set of elements, a relation λ over \mathcal{Z} is a subset of the Cartesian product of \mathcal{Z} with itself, i.e., $\lambda \subseteq \mathcal{Z} \times \mathcal{Z}$. The relation λ is an *equivalence relation* if:

- λ is reflexive, i.e. $x \lambda x, \forall x \in \mathcal{Z}$,
- λ is symmetric, i.e. $x \lambda y \rightarrow y \lambda x, \forall x, y \in \mathcal{Z}$,
- λ is transitive, i.e. $x \lambda y \wedge y \lambda z \rightarrow x \lambda z, \forall x, y, z \in \mathcal{Z}$

Given an equivalence relation λ over a set \mathcal{Z} , and $x \in \mathcal{Z}$, $[x]^\lambda$ denotes the set $\{y \in \mathcal{Z} \mid y \lambda x\}$, called *equivalence class of x w.r.t. λ* . An equivalence class is always non-empty because the equivalence relation is reflexive. Obviously, $[x]^\lambda = [y]^\lambda$ for each pair of elements x, y such that $x \lambda y$, thus a non-empty set $E \subseteq \mathcal{Z}$ that is an equivalence class w.r.t. λ can be denoted with $[x]^\lambda$, where x is any element in E . If two equivalence classes w.r.t. λ have an element in common, they are equal. Since the equivalence relation is symmetric and transitive, equivalence classes without common elements are disjoint. Therefore, an equivalence relation over a set \mathcal{Z} partitions \mathcal{Z} . The *quotient set of λ on \mathcal{Z}* , denoted \mathcal{Z}/λ , is defined as the set of all equivalence classes over \mathcal{Z} w.r.t. λ , i.e. the set $\{[x]^\lambda \mid x \in \mathcal{Z}\}$. Obviously, given a quotient set \mathcal{Z}/λ , $\mathcal{Z} = \bigcup_{[x]^\lambda \in \mathcal{Z}/\lambda} [x]^\lambda$.

8.1.2 Ontologies with Concrete Domains

In the following we consider DLs equipped with concrete domains, which distinguish

- concepts from data types, the formers being abstractions for sets of objects, such as persons or cities, while the latter being value domains, such as strings or integers, and
- roles from attributes, where a role denotes a binary relation between objects, such as `father_of` or `lives_in`, and an attribute denotes a binary relation between objects and values, such as `name` or `height`.

Ontologies defined through these DL languages are as usual composed by a TBox and an ABox. In the following, TBoxes will be defined as sets of rules expressed in first-order syntax, rather than in the classical DL syntax. This choice allows us to have a more uniform treatment of TBox assertions and entity resolution rules, and to obtain a general framework that can be instantiated with various DL dialects. The rules we consider make use of atoms constructed on atomic concepts, atomic roles, and attributes, that is atoms of the form $C(x_1)$, $R(x_1, x_2)$, and $A(x_1, x_3)$, respectively, where x_1 , x_2 , and x_3 are variables. For these atoms, we call x_1 and x_2 *entity-variables*, as they occur as arguments of atoms whose predicate in an atomic concepts or an atomic roles, or as the first argument of an atom whose predicate is an attribute (these arguments are also called *entity positions*, since their meaningful binding is with entities). We then call x_3 a *value-variable* as it occurs as the second argument of an atom whose predicate is an attribute (these arguments are also called *value positions*, since their meaningful binding is with values). We will use this notion throughout the following sections.

The precise syntax of the TBox assertions we consider will be given in the following section. We now only anticipate that we will investigate two settings: (i) TBoxes expressed through tuple-generating dependencies, which capture so-called DL positive inclusions, as inclusions of the form $B_1 \sqsubseteq B_2$ and $Q_1 \sqsubseteq Q_2$ given in Section 5.2 to define the syntax of $DL-Lite_{\mathcal{R}}$ and $DL-Lite_{\mathcal{F}}$, and (ii) TBoxes expressed as sets of tuple-generating dependencies and functionality assertions, as those allowed in $DL-Lite_{\mathcal{F}}$ (which we here extend to attributes). We point out that separating the treatment in two parts is done only for the sake of presentation, since the latter setting completely captures the former one.

Hereinafter, we assume to have the following pairwise disjoint alphabets¹: N_E , N_V , N_C , N_R , N_A , i.e., the alphabet of *entities*, *values*, *atomic concepts*, *atomic roles*, and *attributes*, respectively. Besides the above alphabets we also consider N_F , i.e., the alphabet of *n-ary built-in predicates*, which will be used in entity-resolution rules. Obviously, N_F is disjoint from the other alphabets. We use the letters e and v , possibly with subscripts, to denote entities and values,

¹We consider one single concrete domain and do not distinguish between different data types.

respectively, whereas, we use the letters C , R , A , and F , possibly with subscripts, to denote atomic concepts, atomic roles, attributes, and built-in predicates, respectively.

8.2 KER systems

We now introduce KER systems (i.e., Knowledge bases enriched with Entity Resolution rules), which combine a DL ontology with a set of entity resolution rules, i.e., Horn rules, possibly using built-in predicates in their body, having a single equality atom in the head. In other terms, entity resolution rules are equality generating dependencies (of a special kind, as we will precisely define later).

A KER system \mathcal{K} is thus defined as a triple $\langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ where:

- \mathcal{T} is the TBox;
- \mathcal{A} is the ABox;
- \mathcal{E} is a set of entity resolution rules.

We will describe each component separately.

8.2.1 Terminological component of a KER system

We initially consider TBoxes constituted only by function-free Horn-rules, that is *tuple-generating dependencies* (tgds), whose predicates are in \mathbf{N}_C , \mathbf{N}_R , and \mathbf{N}_A .

A tgd is a formula of the form²:

$$\forall \vec{x}. \phi(\vec{x}) \rightarrow \exists \vec{y}. \psi(\vec{x}, \vec{y}) \quad (8.1)$$

In the above formula, both $\phi(\vec{x})$ and $\psi(\vec{x}, \vec{y})$ are conjunctions of atoms of the form $C(z)$, $R(z, z')$, or $A(z, w)$, where C , R , and A are an atomic concept, an atomic role, and an attribute, respectively, z and z' are entity-variables, and w is a value-variable. Each variable in \vec{x} or \vec{y} occurs either only in entity-positions or only in value-positions, that is, it is either an entity-variable or a value-variable. We call $\forall \vec{x}. \phi(\vec{x})$ the body of the tgd, and $\exists \vec{y}. \psi(\vec{x}, \vec{y})$ the head of the tgd.

In the following, we may write tdgs of the form (8.1) without the universal quantification over the variables \vec{x} (i.e., we may omit $\forall \vec{x}$ in the left-hand side of the tgd).

²As in [89], without loss of generality, \vec{x} denotes all variables occurring in the left-hand side of the tgd, not all necessarily occurring in the right-hand side of the tgd.

Example 8.1. Let \mathcal{T} be the TBox containing the following assertions:

$$\begin{aligned}\theta_1) & \textit{Professor}(x) \rightarrow \textit{Person}(x) \\ \theta_2) & \textit{Person}(x) \rightarrow \exists y.\textit{has_name}(x, y) \\ \theta_3) & \textit{teaches}(x, y) \rightarrow \textit{Professor}(x) \\ \theta_4) & \textit{lives_in}(x, y) \rightarrow \textit{City}(y)\end{aligned}$$

Such a TBox states that professors are persons (θ_1), every person has a name (θ_2), who teaches something is a professor (θ_3), who lives somewhere has to live in a City (θ_4). \square

We point out that reasoning over such dependencies is in general undecidable [14, 2], thus, to identify decidable (and possibly tractable) settings, some restrictions have to be imposed, e.g., some syntactic limitations on the form of the tgds and on their interaction with the entity resolution rules of a KER system. We however preferred to define a general framework, which is not limited to a specific ontological language, but which can be instantiated to different contexts. In particular, the tgds we consider capture several forms of *positive inclusion assertions* given in the classical DL syntax [9], used, e.g., to specify inclusions between concepts or roles, typings of the role domain/range, typings of the attribute domain, mandatory participation of a concept into a role or attribute, as allowed by positive inclusions in *DL-Lite* languages (but they rule out positive inclusions not expressible as horn-rules, as well as negative inclusions and cardinality restrictions different from minimum cardinality one). We further recall that tgds are also called existential rules [163] or Datalog+/- rules[36]. Thus, the TBoxes we consider can be also seen as specified in these languages, limitedly to the use of binary and unary predicates corresponding to concepts and their properties.

8.2.2 Assertional component of a KER system

The ABox \mathcal{A} in a KER system is formed by a finite set of membership assertions on atomic concepts, atomic roles, and attributes. Such assertions have the form:

$$C(\mathbf{e}) \quad R(\mathbf{e}_1, \mathbf{e}_2) \quad A(\mathbf{e}, v)$$

where $e, e_1, e_2 \in \mathbf{N}_E$ and $v \in \mathbf{N}_V$. Intuitively, $C(\mathbf{e})$ states that the entity e is an instance of the atomic concept C . $R(\mathbf{e}_1, \mathbf{e}_2)$ states that the pair constituted by the entities e_1 and e_2 is an instance of the atomic role R . $A(\mathbf{e}, v)$ states that the pair constituted by the entity e and the value v is an instance of the attribute A . For the rest of this chapter, in the examples, entities are written in bold and with a # prefix, whereas values are written in italic style.

Example 8.2. Let \mathcal{A} be the ABox containing the following assertions:

α_1) <i>Professor</i> (#Ullman)	α_9) <i>lives_in</i> (#Ullman , #Palo.Alto)
α_2) <i>Professor</i> (#A.Einstein)	α_{10}) <i>lives_in</i> (#J.Ullman , #Palo.Alto)
α_3) <i>Professor</i> (#Einstein)	α_{11}) <i>has_name</i> (#Ullman , Ullman)
α_4) <i>Person</i> (#J.Ullman)	α_{12}) <i>has_name</i> (#J.Ullman , J.Ullman)
α_5) <i>Person</i> (#Einstein.A)	α_{13}) <i>has_name</i> (#A.Einstein , A.Einstein)
α_6) <i>teaches</i> (#Ullman , #Database)	α_{14}) <i>has_name</i> (#Einstein , Einstein)
α_7) <i>lives_in</i> (#Einstein , #Zurich)	α_{15}) <i>has_name</i> (#Einstein.A , Einstein.A)
α_8) <i>lives_in</i> (#A.Einstein , #Zurich)	

This ABox states that **#Ullman**, **#A.Einstein**, **#Einstein** are professors (assertions α_1 , α_2 , and α_3), **#J.Ullman** and **#Einstein.A** are persons (α_4 , α_5), **#Ullman** teaches **#Database** (α_6), both **#Einstein** and **#A.Einstein** live in **#Zurich** (α_7 , α_8), both **#Ullman** and **#J.Ullman** live in **#Palo.Alto** (α_9 , α_{10}).

Finally the name of **#Ullman** is Ullman, the name of **#J.Ullman** is J.Ullman (α_{11} , α_{12}), the name of **#A.Einstein** is A.Einstein (α_{13}), the name of **#Einstein** is Einstein (α_{14}), and the name of **#Einstein** is Einstein.A (α_{15}). \square

8.2.3 Entity resolution component of a KER system

We now provide the definition of entity resolution rules. In rules of this kind we may also make use of n -ary built-in-predicates from \mathbb{N}_F , such as the *Jaccard similarity*, used to specify that two strings have a similarity above a certain threshold, where the similarity is measured as the size of the intersection of the two strings divided by the size of their union. We assume that they have the same semantics in every interpretation. Intuitively, we treat them as pre-interpreted special predicates, which do not occur in TBox and ABox assertions.

We first define atoms that may occur in entity resolution rules. Each such atom has the form $C(z)$, $R(z, z')$, $A(z, w)$, or $F(t_1, \dots, t_n)$, where C is an atomic concept, R is an atomic role, A is an attribute, F is a built-in predicate of arity n , z and z' are entity-variables, w is a value-variable, and each t_i is either a value-variable or a value from \mathbb{N}_V .

An *entity resolution rule* has the form:

$$\forall \vec{x}. \phi(\vec{x}) \rightarrow \sim(x_1, x_2) \quad (8.2)$$

where $\phi(\vec{x})$ is a conjunction of atoms of the form above, \vec{x} is the sequence of variables occurring in all such atoms, the symbol \sim denotes a special binary predicate, appearing only in the entity resolution rules, x_1 and x_2 are variables in \vec{x} , and the following conditions hold: (i) every variable

in \vec{x} occurs either only in entity positions or only in value positions; (ii) for each variable $w \in \vec{x}$ occurring in a built-in function there must be at least an atom of the form $A(z, w)$ in $\phi(\vec{x})$; (iii) x_1 and x_2 are entity-variables. We call $\forall \vec{x}.\phi(\vec{x})$ the body of the rule, and $x_1 = x_2$ the head of the rule.

As for tgds, in the following we may write rules of the form (8.2) without the universal quantification over the variables \vec{x} .

Example 8.3. Let \mathcal{E} be the set containing the following entity resolution rule ϵ_1 :

$$\begin{aligned} & Person(x) \wedge Person(y) \wedge lives_in(x, z) \wedge lives_in(y, z) \\ & \wedge has_name(x, w_1) \wedge has_name(y, w_2) \wedge jaccardsim(w_1, w_2, 0.9) \rightarrow \sim(x, y) \end{aligned}$$

In the above rule, x, y, z are entity-variables, whereas w_1, w_2 are value-variables. The rule ϵ_1 states that if two entities denoting persons live in the same place and the Jaccard similarity of their names is higher than 0.9, they are related to each other by the \sim relation (as we will see with the definition of the semantics of the entity resolution rules, this means that they have to be considered as the same person of the real-world). \square

8.2.4 Semantics of a KER system

We now provide the definition of interpretation of a KER system. Each such interpretation, differently from what is done in standard first-order logic, depends also by an equivalence relation, interprets entities in terms of equivalence classes, and interprets concepts, roles and attributes accordingly.

Definition 8.1 (interpretation). An interpretation \mathcal{I} of a KER system \mathcal{K} is triple $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \sim^{\mathcal{I}})$, where

- $\Delta^{\mathcal{I}} = \Delta_E^{\mathcal{I}} \cup \Delta_V^{\mathcal{I}}$, where $\Delta_E^{\mathcal{I}}$ and $\Delta_V^{\mathcal{I}}$ are two disjoint sets such that $N_E \subseteq \Delta_E^{\mathcal{I}}$ and $N_V \subseteq \Delta_V^{\mathcal{I}}$,
- $\cdot^{\mathcal{I}}$ is an interpretation function, detailed below,
- $\sim^{\mathcal{I}}$ is an equivalence relation over $\Delta_E^{\mathcal{I}}$.

Then, $\Delta_E^{\mathcal{I}}/\sim^{\mathcal{I}} = \{[x]^{\sim^{\mathcal{I}}} \mid x \in \Delta_E^{\mathcal{I}}\}$, i.e., the quotient set of $\sim^{\mathcal{I}}$ on $\Delta_E^{\mathcal{I}}$, is the interpretation domain for the entities, whereas $\Delta_V^{\mathcal{I}}$ is the interpretation domain for the values.

The interpretation function $\cdot^{\mathcal{I}}$ is defined as follows:

- $e^{\mathcal{I}} = [e]^{\sim^{\mathcal{I}}}$, for each entity $e \in N_E$;
- $v^{\mathcal{I}} = v$, for each value $v \in N_V$;

- $C^{\mathcal{I}} \subseteq \Delta_E^{\mathcal{I}}/\sim^{\mathcal{I}}$, for each atomic concept $C \in \mathbf{N}_C$;
- $R^{\mathcal{I}} \subseteq \Delta_E^{\mathcal{I}}/\sim^{\mathcal{I}} \times \Delta_E^{\mathcal{I}}/\sim^{\mathcal{I}}$, for each atomic role $R \in \mathbf{N}_R$;
- $A^{\mathcal{I}} \subseteq \Delta_E^{\mathcal{I}}/\sim^{\mathcal{I}} \times \Delta_V^{\mathcal{I}}$, for each atomic attribute $A \in \mathbf{N}_A$
- $F^{\mathcal{I}} \subseteq \times_{i=1}^n \mathbf{N}_V$, for each n -ary built-in predicate $F \in \mathbf{N}_F$.

We assume that the semantics of built-in predicates is the same in all interpretations of all KER systems, that is $F^{\mathcal{I}_1} = F^{\mathcal{I}_2}$ for any two interpretations \mathcal{I}_1 and \mathcal{I}_2 and each $F \in \mathbf{N}_F$. We thus in the following omit to specify how an interpretation function interprets such predicates.

To define the semantics of tgds and entity resolution rules we first introduce the notion of assignment. Since this definition will be used also when we will define the evaluation of a conjunctive query over an interpretation, and since a conjunctive query may contain constants (either entities or variables), we define here the notion of assignment for a conjunction of atoms in which also entities or values can occur (notice that constants are instead not allowed in tgds, and values can occur in entity resolution rules only in atoms constructed with built-in predicates).

Definition 8.2 (assignment). Let $\phi(\vec{x})$ be a conjunction of atoms of the form $C(t)$, $R(t, t')$, $A(t, t_1)$, or $F(t_1, \dots, t_n)$, where C, R, A, F are an atomic concept, an atomic role, an attribute, or a n -ary built-in predicate, respectively, t, t' are entity-variables in \vec{x} or entities, and t_1, \dots, t_n are value-variables in \vec{x} or values. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \sim^{\mathcal{I}})$ be an interpretation. An *assignment* from $\phi(\vec{x})$ to \mathcal{I} is a mapping μ from the variables \vec{x} to the set $\Delta_E^{\mathcal{I}}/\sim^{\mathcal{I}} \cup \Delta_V^{\mathcal{I}}$, such that $\langle \mu(x_1), \dots, \mu(x_k), c_1^{\mathcal{I}}, \dots, c_\ell^{\mathcal{I}} \rangle \in S^{\mathcal{I}}$, for every atom $S(x_1, \dots, x_k, c_1, \dots, c_\ell)$ of $\phi(\vec{x})$, where, for each $1 \leq i \leq \ell$, c_i is either an entity in \mathbf{N}_E or a value in \mathbf{N}_V ³. Let $\vec{x} = x_1, \dots, x_m$, the sequence $\mu(x_1), \dots, \mu(x_m)$ is called the *image* of μ .

Let $\phi(\vec{x})$ and $\psi(\vec{x}, \vec{y})$ be two conjunctions of atoms as in Definition 8.2, and μ be an assignment from $\phi(\vec{x})$ to an interpretation \mathcal{I} , an assignment μ' from $\phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y})$ to \mathcal{I} is an *extension* of μ if $\mu(x) = \mu'(x)$ for each x occurring in \vec{x} . Furthermore, $\mu(\phi(\vec{x}))$ is the *binding* of $\phi(\vec{x})$ through μ , i.e., the formula obtained by replacing each x belonging to \vec{x} with $\mu(x)$ in $\phi(\vec{x})$.

An Interpretation \mathcal{I} of a KER system \mathcal{K} satisfies:

- a tgd of the form (8.1) in \mathcal{K} , if each assignment μ from $\phi(\vec{x})$ to \mathcal{I} can be extended to an assignment μ' from $\phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y})$ to \mathcal{I} .
- an entity resolution rule of the form (8.2) in \mathcal{K} , if each assignment μ from $\phi(\vec{x})$ to \mathcal{I} is such that $\mu(x_1) = \mu(x_2)$.

³Without loss of generality we assume that entities and values, if any, occur always in the last ℓ positions of an atom. Notice also that $k \geq 0$, $\ell \geq 0$ and $k + \ell \geq 1$.

Interestingly, the above definition coincides with the standard one (modulo the adoption of our tailored notions of interpretation and assignment).

The interpretation \mathcal{I} satisfies an ABox assertion $C(e)$ (resp. $R(e_1, e_2)$, $A(e, v)$), if $e^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $\langle e_1^{\mathcal{I}}, e_2^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$, $\langle e^{\mathcal{I}}, v \rangle \in A^{\mathcal{I}}$).

Finally, we say that an Interpretation \mathcal{I} is a model of a KER system $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$, denoted as $\mathcal{I} \models \mathcal{K}$, if \mathcal{I} satisfies every tgd in \mathcal{T} , every assertion in \mathcal{A} , and every rule in \mathcal{E} . The set of all models of a KER system \mathcal{K} is denoted with $Mod(\mathcal{K})$, i.e., $Mod(\mathcal{K}) = \{\mathcal{I} \mid \mathcal{I} \models \mathcal{K}\}$.

Example 8.4. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a KER system, where \mathcal{T} is the TBox in Example 8.1, \mathcal{A} is the ABox in Example 8.2 and \mathcal{E} is the set of entity resolution rules in Example 8.3. Let $\sim^{\mathcal{I}}$ be the following equivalence relation (we omit pairs that can be obtained by applying symmetry and transitivity to the given set and pairs associating each entity to itself):

$$\sim^{\mathcal{I}} = \{\langle \#Ullman, \#J.Ullman \rangle, \langle \#Einstein, \#A.Einstein \rangle, \langle \#A.Einstein, \#Einstein.A \rangle\}$$

The interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \sim^{\mathcal{I}} \rangle$ described below is a model for \mathcal{K} .

$$\Delta_V^{\mathcal{I}} = \{Ullman, J.Ullman, A.Einstein, Einstein, Einstein.A\}$$

$$\Delta_E^{\mathcal{I}} / \sim^{\mathcal{I}} = \{\{\#Ullman, \#J.Ullman\}, \{\#Einstein, \#A.Einstein, \#Einstein.A\}, \{\#Database\}, \{\#Zurich\}, \{\#Palo.Alto\}\}$$

$$Professor^{\mathcal{I}} = \{\{\#Ullman, \#J.Ullman\}, \{\#Einstein, \#A.Einstein, \#Einstein.A\}\}$$

$$Person^{\mathcal{I}} = \{\{\#Ullman, \#J.Ullman\}, \{\#Einstein, \#A.Einstein, \#Einstein.A\}\}$$

$$City^{\mathcal{I}} = \{\{\#Zurich\}, \{\#Palo.Alto\}\}$$

$$teaches^{\mathcal{I}} = \{\{\{\#Ullman, \#J.Ullman\}, \{\#Database\}\}\}$$

$$lives_in^{\mathcal{I}} = \{\{\{\#Ullman, \#J.Ullman\}, \{\#Palo.Alto\}\}, \{\{\#Einstein, \#A.Einstein, \#Einstein.A\}, \{\#Zurich\}\}\}$$

$$has_name^{\mathcal{I}} = \{\{\{\#Ullman, \#J.Ullman\}, Ullman\}, \{\{\#Ullman, \#J.Ullman\}, J.Ullman\},$$

$$\langle \{\#Einstein, \#A.Einstein, \#Einstein.A\}, A.Einstein \rangle,$$

$$\langle \{\#Einstein, \#A.Einstein, \#Einstein.A\}, Einstein \rangle, \langle \{\#Einstein, \#A.Einstein, \#Einstein.A\}, Einstein.A \rangle\}$$

□

8.3 Universal models

Among all possible models for a KER-system we focus on those having some special properties, called *universal models*. Before giving the definition of *universal model*, we introduce some preliminary notions.

Given an interpretation \mathcal{I} we define the set of entity-nulls $\Delta_{E_{\perp}}^{\mathcal{I}} = \Delta_E^{\mathcal{I}} \setminus N_E$ and the set of value-nulls $\Delta_{V_{\perp}}^{\mathcal{I}} = \Delta_V^{\mathcal{I}} \setminus N_V$. Then, we define homomorphisms between interpretations.

Definition 8.3 (homomorphism). Let \mathcal{K} be a KER system and let $\mathcal{I}_1 = \langle \Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1}, \sim^{\mathcal{I}_1} \rangle$ and

$\mathcal{I}_2 = \langle \Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2}, \sim^{\mathcal{I}_2} \rangle$ be two interpretations. An *homomorphism* $h : \mathcal{I}_1 \rightarrow \mathcal{I}_2$ is a mapping from the elements of $\Delta^{\mathcal{I}_1}$ to the elements of $\Delta^{\mathcal{I}_2}$ such that:

- $h(e) = e$, for each $e \in \mathbf{N}_E$;
- $h(v) = v$, for each $v \in \mathbf{N}_V$;
- $h(e_\perp) \in \Delta_E^{\mathcal{I}_2}$, for each $e_\perp \in \Delta_{E_\perp}^{\mathcal{I}_1}$;
- $h(v_\perp) \in \Delta_V^{\mathcal{I}_2}$, for each $v_\perp \in \Delta_{V_\perp}^{\mathcal{I}_1}$.
- if $e_1 \sim^{\mathcal{I}_1} e_2$, then $h(e_1) \sim^{\mathcal{I}_2} h(e_2)$, for each $e_1, e_2 \in \Delta_E^{\mathcal{I}_1}$;
- if $[e] \sim^{\mathcal{I}_1} \in C^{\mathcal{I}_1}$, then $[h(e)] \sim^{\mathcal{I}_2} \in C^{\mathcal{I}_2}$, for each atomic concept C in \mathcal{K} and each equivalence classes $[e] \sim^{\mathcal{I}_1} \in \Delta_E^{\mathcal{I}_1} / \sim^{\mathcal{I}_1}$;
- $\langle [e_1] \sim^{\mathcal{I}_1}, [e_2] \sim^{\mathcal{I}_1} \rangle \in R^{\mathcal{I}_1}$, then $\langle [h(e_1)] \sim^{\mathcal{I}_2}, [h(e_2)] \sim^{\mathcal{I}_2} \rangle \in R^{\mathcal{I}_2}$, for each atomic role R in \mathcal{K} and each $[e_1] \sim^{\mathcal{I}_1}, [e_2] \sim^{\mathcal{I}_1} \in \Delta_E^{\mathcal{I}_1} / \sim^{\mathcal{I}_1}$;
- if $\langle [e] \sim^{\mathcal{I}_1}, v \rangle \in A^{\mathcal{I}_1}$, then $\langle [h(e)] \sim^{\mathcal{I}_2}, h(v) \rangle \in A^{\mathcal{I}_2}$, for each atomic attribute A in \mathcal{K} , each $[e] \sim^{\mathcal{I}_1} \in \Delta_E^{\mathcal{I}_1} / \sim^{\mathcal{I}_1}$, and each $v \in \Delta_V^{\mathcal{I}_1}$.

An interpretation \mathcal{I}_1 is homomorphically equivalent to an interpretation \mathcal{I}_2 if there is an homomorphism $h : \mathcal{I}_1 \rightarrow \mathcal{I}_2$ and homomorphism $h' : \mathcal{I}_2 \rightarrow \mathcal{I}_1$.

We are now ready to provide our definition of universal model.

Definition 8.4 (universal model). Let \mathcal{K} be a KER system. A *universal model* for \mathcal{K} is a model $\mathcal{U} \in \text{Mod}(\mathcal{K})$ such that for every model $\mathcal{I}' \in \text{Mod}(\mathcal{K})$ there exists a homomorphism $h : \mathcal{U} \rightarrow \mathcal{I}'$.

The universal solutions are unique up to homomorphically equivalence.

8.4 Query answering

A CQ over a KER system has the same form of a CQ over an ontology (see Section 5.2.1). Since we are now using DL ontologies containing attributes, we prefer to repeat this definition and precisely describe the forms of CQ atoms. A CQ over a KER system is an open first-order logic formula of the form:

$$\exists \vec{y}. \text{conj}(\vec{x}, \vec{y}) \tag{8.3}$$

where \vec{x} are the free variables, \vec{y} are the existentially quantified variables, and $\text{conj}(\vec{x}, \vec{y})$ is a conjunction of atoms, each of the form $C(t)$ or $R(t, t')$ or $A(t, c)$, where C is an atomic concept, R is an atomic role, A is an attribute, t and t' are entity-variables occurring in either \vec{x} or \vec{y} , or

entities in N_E , c is a value-variable occurring in either \vec{x} or \vec{y} or values in N_V . As usual, each variable in \vec{x} or \vec{y} occur either always in entity-positions or always in value-positions. A usual, a query q of the form above can be also denoted as $q(\vec{x})$.

Given a query $q(x_1, \dots, x_n)$ of the form above and an interpretation \mathcal{I} , the set of answers to q with respect to \mathcal{I} , denoted with $q^{\mathcal{I}}$, is the set of tuples $\langle \mu(x_1), \dots, \mu(x_n) \rangle$ such that μ is an assignment from $\text{conj}(\vec{x}, \vec{y})$ to \mathcal{I} .

The following proposition follows from the definition of model and the definition of homomorphisms between interpretations (Definition 8.3)

Proposition 8.1. *Let \mathcal{K} be a KER system, q a query over \mathcal{K} , $\mathcal{I}_1 = \langle \Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1}, \sim^{\mathcal{I}_1} \rangle$ and $\mathcal{I}_2 = \langle \Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2}, \sim^{\mathcal{I}_2} \rangle$ two models of \mathcal{K} , and let $h : \mathcal{I}_1 \rightarrow \mathcal{I}_2$ be an homomorphism from \mathcal{I}_1 to \mathcal{I}_2 . If $\langle T_1, \dots, T_n \rangle \in q^{\mathcal{I}_1}$, then $\langle h(T_1), \dots, h(T_n) \rangle \in q^{\mathcal{I}_2}$, where for every $1 \leq i \leq n$: (i) $h(T_i) = [h(e)]^{\sim^{\mathcal{I}_2}}$, if $T_i = [e]^{\sim^{\mathcal{I}_1}}$, and (ii) $h(T_i) = h(T_i)$, if $T_i \in \Delta_V^{\mathcal{I}_1}$ (i.e., T_i is either a value or a value-null).*

In order to provide the definition of certain answers to a query over a KER system we give some preliminary notions.

Definition 8.5 (typed tuple). A *typed tuple* is a tuple of the form $\langle T_1, \dots, T_n \rangle$ where each T_i is either a non-empty set of entities or a value.

Two typed tuples $\vec{t} = \langle T_1, \dots, T_n \rangle$ and $\vec{t}' = \langle T'_1, \dots, T'_n \rangle$ are *equal*, denoted $\vec{t} = \vec{t}'$, if $T_i = T'_i$ for every $1 \leq i \leq n$. The *type* of a typed tuple $\langle T_1, \dots, T_n \rangle$ is a tuple $\langle \rho_1, \dots, \rho_n \rangle$ such that $\rho_i = \mathbf{e}$ if T_i is a set of entities or $\rho_i = \mathbf{v}$ if T_i is a value, for every $1 \leq i \leq n$. Two typed tuples with types $\langle \rho_1, \dots, \rho_n \rangle$ and $\langle \rho'_1, \dots, \rho'_n \rangle$, respectively, *have the same type* if $\rho_i = \rho'_i$ for every i with $1 \leq i \leq n$.

Let $\vec{t} = \langle T_1, \dots, T_n \rangle$ and $\vec{t}' = \langle T'_1, \dots, T'_n \rangle$ be two typed tuples of the same type. The tuple \vec{t}' is *equally or more informative* than \vec{t} , denoted with $\vec{t} \leq \vec{t}'$, if for every $1 \leq i \leq n$, it holds either (i) $T_i \subseteq T'_i$, if T_i and T'_i are sets of entities or (ii) $T_i = T'_i$ if T_i and T'_i are values. The tuple \vec{t}' is *more informative* than \vec{t} , denoted with $\vec{t} < \vec{t}'$, if $\vec{t} \leq \vec{t}'$ and $\vec{t} \neq \vec{t}'$.

When querying a KER system, as happens for query answering over DL ontologies, we are interested in querying a set of models rather than evaluating the query over a single interpretation. We thus adapted the classical notion of certain answers to our framework (note that in this case a certain answer is a typed tuple).

Definition 8.6 (certain answer). Let \mathcal{K} be a KER system and q a query of arity n over \mathcal{K} . A typed tuple $\vec{t} = \langle T_1, \dots, T_n \rangle$ is a *certain answer* to q w.r.t. \mathcal{K} if for each $\mathcal{I} \in \text{Mod}(\mathcal{K})$ there exists a tuple $\langle T'_1, \dots, T'_n \rangle \in q^{\mathcal{I}}$ such that for each $1 \leq i \leq n$:

- (i) if T_i is a set of entities, T'_i is an equivalence class in $\Delta_E^{\mathcal{I}} / \sim^{\mathcal{I}}$ and $T_i \subseteq T'_i$;

(ii) if T_i is a value, T'_i is a value in $\Delta_{\mathcal{V}}^{\mathcal{I}}$ and $T_i = T'_i$;

(iii) there is no typed tuple \vec{t}' that satisfies both (i) and (ii) such that $\vec{t} < \vec{t}'$.

Note that condition (iii) in the above definition says that a certain answer has to be a maximal typed tuple, with respect to the informativeness order we have defined on typed tuples, that satisfies conditions (i) and (ii).

We denote with $\text{cert}(\mathcal{K}, q)$ the set of certain answers to a query q w.r.t. a KER system \mathcal{K} .

Proposition 8.2. *Let \mathcal{K} be a KER system, q be a query over \mathcal{K} , and $\vec{t} = \langle T_1, \dots, T_n \rangle$ and $\vec{t}' = \langle T'_1, \dots, T'_n \rangle$ be two certain answers to q w.r.t. \mathcal{K} . If T_i and T'_j are sets of entities, then either $T_i = T'_j$ or $T_i \cap T'_j = \emptyset$, for any $1 \leq i, j \leq n$.*

Proof. Towards a contradiction, assume that T_i, T'_j are two non-empty sets of entities such that $T_i \neq T'_j$ and $T_i \cap T'_j = \emptyset$. Then $T_i \subsetneq T_i \cup T'_j$ or $T'_j \subsetneq T_i \cup T'_j$. Let's assume $T_i \subsetneq T_i \cup T'_j$. Since $\langle T_1, \dots, T_n \rangle$ and $\langle T'_1, \dots, T'_n \rangle$ are certain answers of q , for every model \mathcal{I} , there are tuples $\langle L_1, \dots, L_n \rangle, \langle L'_1, \dots, L'_n \rangle$ such that for every k we have $T_k \subseteq L_k$ if T_k is a set of entities, and $T_k = L_k$ if T_k is a value and $T'_k \subseteq L'_k$ if T'_k is a set of entities and $T'_k = L'_k$ if T'_k is a value. Thus we have that $T_i \subseteq L_i$ and $T'_j \subseteq L'_j$. Since $T_i \cap T'_j = \emptyset$, it follows that $L_i \cap L'_j = \emptyset$. However, T_i, T'_j are equivalence classes in the model \mathcal{I} , hence they must coincide, i.e. $L_i = T'_j$. It follows that $T_i \cup T'_j \subseteq L_i = T'_j$. Consider now the tuple $\langle T_1, \dots, T_{i-1}, T_i \cup T'_j, T_{i+1}, \dots, T_n \rangle$ which is obtained from $\langle T_1, \dots, T_n \rangle$ by replacing T_i by $T_i \cup T'_j$. Note that $\langle T_1, \dots, T_{i-1}, T_i, T_{i+1}, \dots, T_n \rangle < \langle T_1, \dots, T_{i-1}, T_i \cup T'_j, T_{i+1}, \dots, T_n \rangle$ because $T_i \subsetneq T_i \cup T'_j$. However for every model \mathcal{I} of \mathcal{K} , we have that $T_i \cup T'_j \subseteq L_i$. Moreover, for all $k \neq i$, we have that $T_k \subseteq L_k$ if T_k is a set of entities, or $T_k = L_k$ if T_k is a value. This, however, violates the more informativeness of $\langle T_1, \dots, T_n \rangle$, hence $\langle T_1, \dots, T_n \rangle$ is not a certain answer, which is a contradiction. \square

The following property easily follows from the above proposition.

Corollary 8.1. *Let \mathcal{K} be a KER system, q be a query over \mathcal{K} and $\vec{t} = \langle T_1, \dots, T_n \rangle$ be a typed tuple. If \vec{t} is a certain answer to q w.r.t. \mathcal{K} and if T_i and T_j , where $i \neq j$, are sets of entities, then either $T_i = T_j$ or $T_i \cap T_j = \emptyset$.*

Proof. The thesis follows from Proposition 8.2, when $\vec{t} = \vec{t}'$. \square

Proposition 8.2 and Corollary 8.1 tell that if q is a query, then the set of certain answers to q is a set if typed tuples such that if two sets of entities appear in the same or in different tuples, then those two sets of entities either are the same set or they are disjoint. In particular, we can identify the set of entities that appear in the set of certain answers with the equivalence classes of some equivalence relation.

We now explain the relation existing between universal models and certain answers. To this aim, we have to introduce an operator that suitably allows us to get rid of entity-nulls and value-nulls when evaluating a query over an model (analogously to what is done in the context of Data Exchange [89]). We apply this operator, denoted \downarrow , to a tuple \vec{t} whose components are sets of entities and entity nulls (that is, equivalence classes in an interpretation), values, or value-nulls. We first define how \downarrow applies to each component of the tuple, and then we extend it to the entire tuple, and to a set of tuples.

If T is a set of entities and entity-nulls, $T\downarrow$ returns the set of entities obtained from T by removing all entity-nulls in T . If T is a value, then $T\downarrow = T$. If $\langle T_1, \dots, T_n \rangle$ is a tuple such that each T_i is either a set of entities and entity-nulls or a value, then $\langle T_1, \dots, T_n \rangle\downarrow = \langle T_1\downarrow, \dots, T_n\downarrow \rangle$. Let Θ be a set of tuples of the form $\langle T_1, \dots, T_n \rangle$, where each T_i is a set of entities and entity-nulls, or a value, or a value-null, then $\Theta\downarrow$ is the set obtained from Θ by removing all tuples in Θ containing at least one value-null and all tuples containing a T_i such that $T_i\downarrow = \emptyset$, and replacing each other tuple $\langle T_1, \dots, T_n \rangle$ in Θ by the tuple $\langle T_1, \dots, T_n \rangle\downarrow$.

Theorem 8.1. *Let \mathcal{K} be a KER system, q be a query over \mathcal{K} , \vec{t} be a typed tuple, and \mathcal{U} be a universal model for \mathcal{K} . Then $\text{cert}(\mathcal{K}, q) = q^{\mathcal{U}}\downarrow$.*

Proof. For the sake of simplicity, but without real loss of generality, assume that q is a query whose head is $q(x_1, x_2, x_3)$, where x_1, x_3 range over entities and x_2 ranges over values.

Part I: $q^{\mathcal{U}}\downarrow \subseteq \text{cert}(\mathcal{K}, q)$. Let $(x_1, x_2, x_3) \in q^{\mathcal{U}}\downarrow$. Then x_1, x_3 are non-empty sets of entities and x_2 is a value, say $x_2 = v$. Moreover, by definition of $q^{\mathcal{U}}\downarrow$, there is a triple (y_1, y_2, y_3) in $q^{\mathcal{U}}$ such that $(x_1, x_2, x_3) = (y_1, y_2, y_3)\downarrow$. Hence, $x_1 = y_1\downarrow$, $x_2 = v = y_2$, $x_3 = y_3\downarrow$. This also means that y_1 and y_3 are equivalence classes of entities and/or entity-nulls in the universal model \mathcal{U} . Since x_1, x_3 are non-empty sets of entities, there must exist entities a_1 and a_3 such that $y_1 = [a_1]^{\sim^{\mathcal{U}}}$ and $y_3 = [a_3]^{\sim^{\mathcal{U}}}$. Thus $x_1 = [a_1]^{\sim^{\mathcal{U}}}\downarrow$, $x_3 = [a_3]^{\sim^{\mathcal{U}}}\downarrow$, where a_1 and a_3 are entities. Consider now a model \mathcal{I} of \mathcal{K} . Since \mathcal{U} is a universal model, there is a homomorphism $h : \mathcal{U} \rightarrow \mathcal{I}$. Since conjunctive query are preserved under homomorphism (by Proposition 8.1) and since $([a_1]^{\sim^{\mathcal{U}}}, v, [a_3]^{\sim^{\mathcal{U}}}) \in q^{\mathcal{U}}$, we have that $([h(a_1)]^{\sim^{\mathcal{I}}}, h(v), [h(a_3)]^{\sim^{\mathcal{I}}}) \in q^{\mathcal{I}}$. Since a_1, a_3 are entities and v is a value, we have that $h(a_1) = a_1$, $h(v) = v$, $h(a_3) = a_3$, hence $([a_1]^{\sim^{\mathcal{I}}}, v, [a_3]^{\sim^{\mathcal{I}}}) \in q^{\mathcal{I}}$. Since x_1, x_3 are set of entities, we have that for each entity c in $x_1 \cup x_3$, it holds that $h(c) = c$. Thus, since $x_1 \subseteq [a_1]^{\sim^{\mathcal{U}}}$ and $x_3 \subseteq [a_3]^{\sim^{\mathcal{U}}}$, we have that $x_1 \subseteq [a_1]^{\sim^{\mathcal{I}}}$, $x_3 \subseteq [a_3]^{\sim^{\mathcal{I}}}$. The preceding argument shows that the tuple (x_1, v, x_3) satisfies conditions (i) and (ii) in the definition of certain answer. It remains to show that (x_1, v, x_3) satisfies condition (iii) in the definition of certain answer, i.e., that it is maximal w.r.t. the properties (i) and (ii). The maximality of (x_1, v, x_3) with respect to typed tuples informativeness follows from the fact $x_1 = [a_1]^{\sim^{\mathcal{U}}}\downarrow$ and $x_3 = [a_3]^{\sim^{\mathcal{U}}}\downarrow$. Indeed, suppose that there was a triple (x'_1, v, x'_3) satisfying

properties (i) and (ii) and such that $x_1 \subseteq x'_1$, $x_3 \subseteq x'_3$. We will show that $x_1 = x'_1$ and $x_3 = x'_3$ which will imply the maximality of (x_1, v, x_3) . Since (x'_1, v, x'_3) satisfy properties (i) and (ii), we have that there are entities a'_1 and a'_3 such that $x'_1 \subseteq [a'_1]^{\sim\mu}$ and $x'_3 \subseteq [a'_3]^{\sim\mu}$. Therefore, $[a_1]^{\sim\mu} \cap [a'_1]^{\sim\mu} \neq \emptyset$ (both contain x_1) and $[a_3]^{\sim\mu} \cap [a'_3]^{\sim\mu} \neq \emptyset$ (both contain x_3). Therefore, $[a_1]^{\sim\mu} = [a'_1]^{\sim\mu}$ and $[a_3]^{\sim\mu} = [a'_3]^{\sim\mu}$. But $x_1 = [a_1]^{\sim\mu} \downarrow$ and $x_3 = [a_3]^{\sim\mu} \downarrow$ and $x_3 = [a_3]^{\sim\mu} \downarrow$ which means that x_1 is the set of all entities in $[a_1]^{\sim\mu}$ and x_3 is the set of all entities in $[a_3]^{\sim\mu}$. Since $x_1 \subseteq x'_1 \subseteq [a_1]^{\sim\mu} = [a'_1]^{\sim\mu}$, $x_3 \subseteq x'_3 \subseteq [a_3]^{\sim\mu} = [a'_3]^{\sim\mu}$ and since x'_1, x'_3 are sets of entities we must have that $x_1 = x'_1$ and $x_3 = x'_3$. This completes the proof that $q^\mu \downarrow \subseteq \text{cert}(\mathcal{K}, q)$.

Part II: $\text{cert}(\mathcal{K}, q) \subseteq q^\mu \downarrow$. Let (x_1, v, x_3) be a certain answer of q . Then there must exist entities a_1 and a_3 such that $x_1 \subseteq [a_1]^{\sim\mu}$ and $x_3 \subseteq [a_3]^{\sim\mu}$ and also $([a_1]^{\sim\mu}, v, [a_3]^{\sim\mu}) \in q^\mu$. By Part I, we have that $([a_1]^{\sim\mu} \downarrow, v, [a_3]^{\sim\mu} \downarrow)$ is a certain answer of q . Moreover, since x_1, x_3 are sets of entities, we have $x_1 \subseteq [a_1]^{\sim\mu} \downarrow$, $x_3 \subseteq [a_3]^{\sim\mu} \downarrow$. By the maximality of (x_1, v, x_3) with respect to typed tuples informativeness, we must have that $x_1 = [a_1]^{\sim\mu} \downarrow$ and $x_3 = [a_3]^{\sim\mu} \downarrow$. Hence $(x_1, v, x_3) \in q^\mu \downarrow$. \square

8.5 Computing a universal model

In the following we adapt the well-known notion of restricted chase [15, 116, 89, 46] to our framework. Intuitively, given a KER-system $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$, the chase of \mathcal{K} is a procedure that starts from a model “isomorphic” to the ABox \mathcal{A} , which we call the minimal model of \mathcal{A} , and incrementally constructs an interpretation of \mathcal{K} that satisfies the tgds of the TBox \mathcal{T} and the entity resolution rules in \mathcal{E} . This is obtained by iteratively applying two chase rules, as long as they are triggered in the (current) interpretation. Interestingly, the chase procedure we define never produces a failure, as, e.g., for the application equality-generating dependencies in the chase described in [89], since entities in an ABox have not to be considered hard-constants, which cannot be made equal, but rather are soft-constants that can be interpreted with the same equivalence class (which means that they are denoting the same object of the world). Thus, our chase procedure always produces a model of the KER system. However, because of the presence of cycles among inclusions of the TBox, the chase can be constituted by an infinite sequence of steps, and thus the model we obtain can be infinite. Regardless of this, the interpretation produced by the chase is always a universal model of the KER system, as we are going to show in this section.

We start with the notion of minimal model for an ABox.

Definition 8.7 (minimal model for an ABox). Let \mathcal{A} be an ABox, the *minimal model* for \mathcal{A} is the interpretation $\mathcal{I}^{\mathcal{A}} = \langle \Delta^{\mathcal{I}^{\mathcal{A}}}, \mathcal{I}^{\mathcal{A}}, \sim^{\mathcal{I}^{\mathcal{A}}} \rangle$, such that:

- $\Delta^{\mathcal{I}^{\mathcal{A}}} = \Delta^{\mathcal{I}^{\mathcal{A}}}_E \cup \Delta^{\mathcal{I}^{\mathcal{A}}}_V$, where $\Delta^{\mathcal{I}^{\mathcal{A}}}_E = N_E$ and $\Delta^{\mathcal{I}^{\mathcal{A}}}_V = N_V$;

- $\sim^{\mathcal{I}^A} = \{\langle e, e \rangle \mid e \in \Delta_E^{\mathcal{I}^A}\}$;
- $e^{\mathcal{I}^A} = [e]^{\sim^{\mathcal{I}^A}}$, for each entity $e \in \Delta_E^{\mathcal{I}^A}$;
- $v^{\mathcal{I}^A} = v$, for each value $v \in \Delta_V^{\mathcal{I}^A}$;
- $C^{\mathcal{I}^A} = \{[e]^{\sim^{\mathcal{I}^A}} \mid C(e) \in \mathcal{A}\}$, for each atomic concept $C \in \mathbf{N}_C$;
- $R^{\mathcal{I}^A} = \{\langle [e_1]^{\sim^{\mathcal{I}^A}}, [e_2]^{\sim^{\mathcal{I}^A}} \rangle \mid R(e_1, e_2) \in \mathcal{A}\}$, for each atomic role $R \in \mathbf{N}_R$;
- $A^{\mathcal{I}^A} = \{\langle [e]^{\sim^{\mathcal{I}^A}}, v \rangle \mid A(e, v) \in \mathcal{A}\}$, for each attribute $A \in \mathbf{N}_A$.

Note that the elements in $\Delta_E^{\mathcal{I}^A} / \sim^{\mathcal{I}^A}$ are singletons because $\sim^{\mathcal{I}^A}$ contains only pairs of entities equal only to themselves.

We next define two chase steps, one for the tgds of the TBox, and one for the entity resolution rules. To this aim, we assume to have an infinite ordered set of labeled entity-nulls, denoted Δ_{E_\perp} and an infinite ordered set of labeled value-nulls, denoted Δ_{V_\perp} , such that $\Delta_{E_\perp} \cap \mathbf{N}_E = \emptyset$, $\Delta_{V_\perp} \cap \mathbf{N}_V = \emptyset$, and $\Delta_{E_\perp} \cap \Delta_{V_\perp} = \emptyset$.

It is first convenient to define a total order on all the predicates, entities, values, labeled nulls, variables, and symbols used in tgds and entity resolution rules. From $\Delta_{E_\perp} = \{e_{1\perp}, e_{2\perp}, \dots\}$ and $\Delta_{V_\perp} = \{v_{1\perp}, v_{2\perp}, \dots\}$, we define Δ_{EV_\perp} as the set $\{e_{1\perp}, v_{1\perp}, e_{2\perp}, v_{2\perp}, \dots\}$, i.e., the set obtained by alternating the elements of Δ_{E_\perp} with those of Δ_{V_\perp} . We also assume to use only variables from an infinite ordered set of variables \mathbf{N}_V . We can now consider the set $\mathbf{N}_C \cup \mathbf{N}_R \cup \mathbf{N}_A \cup \mathbf{N}_F \cup \mathbf{N}_E \cup \mathbf{N}_V \cup \{\forall, \exists, \cdot, (,), \wedge, \cdot, \cdot\} \cup \mathbf{N}_V \cup \Delta_{EV_\perp}$, and assume that it is totally ordered and such that all the elements in Δ_{EV_\perp} follow, in the same order they have in Δ_{EV_\perp} , all the other elements.

We now define an order over assignments to an interpretation \mathcal{I} whose domain $\Delta^{\mathcal{I}} = \Delta_E^{\mathcal{I}} \cup \Delta_V^{\mathcal{I}}$ is such that $\Delta_E^{\mathcal{I}} \setminus \mathbf{N}_E \subseteq \Delta_{E_\perp}$ and $\Delta_V^{\mathcal{I}} \setminus \mathbf{N}_V \subseteq \Delta_{V_\perp}$. To this aim, we assume that every equivalence class E in the image of an assignment μ is represented in the bindings generated by μ as the string obtained by concatenating all the elements occurring in E , following their alphabetic order. Then, given an assignment μ from a conjunction of atoms ϕ to \mathcal{I} and an assignment μ' from a conjunction of atoms ψ to \mathcal{I} , we say that μ *precedes* μ' if $\mu(\phi)$ precedes $\mu'(\psi)$ in the lexicographic order. Obviously, we have always that either $\mu(\phi)$ precedes $\mu'(\psi)$ or $\mu'(\psi)$ precedes $\mu(\phi)$, unless $\phi = \psi$ and $\mu = \mu'$.

In what follows we also assume that the variables occurring in any rule (tgd or entity resolution rule) of a KER system \mathcal{K} are different from the variables used in all other rules of \mathcal{K} (which implies that an assignment applies only to the conjunction in the body of one single rule).

Definition 8.8 (Chase Step). Let $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1}, \sim^{\mathcal{I}_1})$, where $\Delta^{\mathcal{I}_1} = \Delta_E^{\mathcal{I}_1} \cup \Delta_V^{\mathcal{I}_1}$ and $\Delta_E^{\mathcal{I}_1} \cap \Delta_V^{\mathcal{I}_1} = \emptyset$, be an interpretation such that $\Delta_E^{\mathcal{I}_1} \setminus \mathbf{N}_E \subseteq \Delta_{E_\perp}$, and $\Delta_V^{\mathcal{I}_1} \setminus \mathbf{N}_V \subseteq \Delta_{V_\perp}$.

- (tuple generating dependency) let d be a tgd of the form (8.1). Let μ be an assignment from $\phi(\vec{x})$ to \mathcal{I}_1 such that there is no extension of μ to an assignment μ' from $\phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y})$ to \mathcal{I}_1 . We say that d is applicable to \mathcal{I}_1 with assignment μ (we also say that μ triggers d in \mathcal{I}_1).

Let \mathcal{I}_2 be the triple $(\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2}, \sim^{\mathcal{I}_2})$, defined as follows:

- extend μ to μ' such that each variable y_i in \vec{y} is assigned to either (i) a set containing only a fresh labeled entity-null from Δ_{E_\perp} that follows in the order all the entity-nulls and value-nulls in $\Delta^{\mathcal{I}_1}$, if y_i occurs in entity-position in d , (ii) a fresh labeled value-null from Δ_{V_\perp} that follows in the order all the entity-nulls and value-nulls in $\Delta^{\mathcal{I}_1}$, if y_i occurs in value-position in d ;
 - $\Delta^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_2} \cup \Delta_V^{\mathcal{I}_2}$, where $\Delta_E^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_1} \cup \Delta_{E_\perp}^{fresh}$ and $\Delta_V^{\mathcal{I}_2} = \Delta_V^{\mathcal{I}_1} \cup \Delta_{V_\perp}^{fresh}$, and $\Delta_{E_\perp}^{fresh}$ and $\Delta_{V_\perp}^{fresh}$ are the sets of fresh entity-nulls and fresh value-nulls in the image of μ' , respectively;
 - $\sim^{\mathcal{I}_2} = \sim^{\mathcal{I}_1} \cup \{\langle f, f \rangle \mid f \text{ is a fresh entity-null in the image of } \mu'\}$;
 - $e^{\mathcal{I}_2} = e^{\mathcal{I}_1}$ and $v^{\mathcal{I}_2} = v^{\mathcal{I}_1}$ for each entity or entity-null $e \in \Delta_E^{\mathcal{I}_1}$ and each value or value-null $v \in \Delta_V^{\mathcal{I}_1}$;
 - $f_e^{\mathcal{I}_2} = \{f_e\}$, for each fresh entity-null f_e in the image of μ' ;
 - $f_v^{\mathcal{I}_2} = f_v$, for each fresh value-null f_v in the image of μ' ;
 - $S^{\mathcal{I}_2} = S^{\mathcal{I}_1}$, for each predicate S in \mathcal{K} not occurring in $\psi(\vec{x}, \vec{y})$.
 - $C^{\mathcal{I}_2} = C^{\mathcal{I}_1} \cup \{\mu'(z)\}$, for each atomic concept C in \mathcal{K} such that $C(z) \in \psi(\vec{x}, \vec{y})$;
 - $R^{\mathcal{I}_2} = R^{\mathcal{I}_1} \cup \{\langle \mu'(z_1), \mu'(z_2) \rangle\}$, for each atomic role R in \mathcal{K} such that $R(z_1, z_2) \in \psi(\vec{x}, \vec{y})$;
 - $A^{\mathcal{I}_2} = A^{\mathcal{I}_1} \cup \{\langle \mu'(z_1), \mu'(z_2) \rangle\}$, for each attribute A in \mathcal{K} such that $A(z_1, z_2) \in \psi(\vec{x}, \vec{y})$.
- (entity resolution rule) let d be an entity resolution rule of the form (8.2). Let μ be an assignment from $\phi(\vec{x})$ to \mathcal{I}_1 such that $\mu(x_1) \neq \mu(x_2)$. We say that d is applicable to \mathcal{I}_1 with assignment μ (we also say that μ triggers d in \mathcal{I}_1).

Let \mathcal{I}_2 be the triple $(\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2}, \sim^{\mathcal{I}_2})$, defined as follows:

- $\Delta^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_2} \cup \Delta_V^{\mathcal{I}_2}$, where $\Delta_E^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_1}$ and $\Delta_V^{\mathcal{I}_2} = \Delta_V^{\mathcal{I}_1}$
- $\sim^{\mathcal{I}_2} = \sim^{\mathcal{I}_1} \cup \{\langle f, g \rangle \mid f, g \in \mu(x_1) \cup \mu(x_2)\}$;
- $\cdot^{\mathcal{I}_2}$ is obtained from $\cdot^{\mathcal{I}_1}$ by replacing each occurrence of $\mu(x_1)$ and $\mu(x_2)$ in the range of $\cdot^{\mathcal{I}_1}$ with $\mu(x_1) \cup \mu(x_2)$.

Let d be either a tgd or an entity resolution rule that can be applied to \mathcal{I}_1 with assignment μ , we say that \mathcal{I}_2 is the result of applying d to \mathcal{I}_1 with μ and we write $\mathcal{I}_1 \xrightarrow{d,\mu} \mathcal{I}_2$. We call $\mathcal{I}_1 \xrightarrow{d,\mu} \mathcal{I}_2$ a chase step.

From the above definition it is easy to see that $\Delta^{\mathcal{I}_1} \subseteq \Delta^{\mathcal{I}_2}$, $\sim^{\mathcal{I}_1} \subseteq \sim^{\mathcal{I}_2}$, and thus that $[e]^{\sim^{\mathcal{I}_1}} \subseteq [e]^{\sim^{\mathcal{I}_2}}$, for each entity or entity null $e \in \Delta_E^{\mathcal{I}_1}$.

We next prove that the triple \mathcal{I}_2 produced by a chase step from an interpretation \mathcal{I}_1 is in turn an interpretation.

Proposition 8.3. *Let \mathcal{I}_1 be an interpretation and $\mathcal{I}_2 = (\Delta^{\mathcal{I}_2}, \mathcal{I}_2, \sim^{\mathcal{I}_2})$ be such that $\mathcal{I}_1 \xrightarrow{d,\mu} \mathcal{I}_2$. Then, \mathcal{I}_2 is an interpretation.*

Proof. We first prove that $\Delta^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_2} \cup \Delta_V^{\mathcal{I}_2}$ has the form given in Definition 8.1, i.e., $\Delta_E^{\mathcal{I}_2} \cap \Delta_V^{\mathcal{I}_2} = \emptyset$, $N_E \subseteq \Delta_E^{\mathcal{I}_2}$, and $N_V \subseteq \Delta_V^{\mathcal{I}_2}$. If d is an entity resolution rule the property is trivially satisfied, since \mathcal{I}_1 is an interpretation and $\Delta^{\mathcal{I}_2} = \Delta^{\mathcal{I}_1}$. If d is a tgd, by definition of the tgd chase step we have that $\Delta^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_2} \cup \Delta_V^{\mathcal{I}_2}$, where $\Delta_E^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_1} \cup \Delta_{E_\perp}^{fresh}$, and $\Delta_V^{\mathcal{I}_2} = \Delta_V^{\mathcal{I}_1} \cup \Delta_{V_\perp}^{fresh}$. The thesis follows from the fact that $\Delta_{E_\perp}^{fresh} \cap \Delta_{V_\perp}^{fresh} = \emptyset$, \mathcal{I}_1 is an interpretation and thus $\Delta_E^{\mathcal{I}_1} \cap \Delta_V^{\mathcal{I}_1} = \emptyset$, $N_E \subseteq \Delta_E^{\mathcal{I}_1}$ and $N_V \subseteq \Delta_V^{\mathcal{I}_1}$.

We then prove that $\sim^{\mathcal{I}_2}$ is an equivalence relation over $\Delta_E^{\mathcal{I}_2}$. If d is an entity resolution rule, we have that $\Delta_E^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_1}$, and $\sim^{\mathcal{I}_2} = \sim^{\mathcal{I}_1} \cup \sim^e$, where $\sim^e = \{\langle f, g \rangle \mid f, g \in \mu(x_1) \cup \mu(x_2)\}$. Since $\mu(x_1) \cup \mu(x_2) \subseteq \Delta_E^{\mathcal{I}_1}$, then \sim^e is a relation over $\Delta_E^{\mathcal{I}_1}$ (the same set of entities of \mathcal{I}_1). It is also easy to see that \sim^e is an equivalence relation over $\mu(x_1) \cup \mu(x_2)$. Thus $\sim^{\mathcal{I}_2}$ is the union of two equivalence relations, which implies that reflexivity and symmetry are preserved in $\sim^{\mathcal{I}_2}$. To prove that $\sim^{\mathcal{I}_2}$ is also transitive, let us consider all possible cases. Let x, y, z be elements in $\Delta_E^{\mathcal{I}_2}$ (which in this case, as said is equal to $\Delta_E^{\mathcal{I}_1}$). The following situations are conceivable:

- $(x, y) \in \sim^{\mathcal{I}_1}$ and $(y, z) \in \sim^{\mathcal{I}_1}$, but since $\sim^{\mathcal{I}_1}$ is an equivalence relation then this implies that $(x, z) \in \sim^{\mathcal{I}_1}$ which in turn implies that $(x, z) \in \sim^{\mathcal{I}_2}$.
- $(x, y) \in \sim^e$ and $(y, z) \in \sim^e$, but since \sim^e is an equivalence relation then this implies that $(x, z) \in \sim^e$ which in turn implies that $(x, z) \in \sim^{\mathcal{I}_2}$.
- $(x, y) \in \sim^{\mathcal{I}_1}$ and $(y, z) \in \sim^e$. Since $(y, z) \in \sim^e$, then, by definition of chase step, $y, z \in \mu(x_1) \cup \mu(x_2)$, and $[y]^{\sim^{\mathcal{I}_1}} \subseteq \mu(x_1) \cup \mu(x_2)$. This last condition, together with the assumption that $(x, y) \in \sim^{\mathcal{I}_1}$, from which it follows that $x \in [y]^{\sim^{\mathcal{I}_1}}$, implies that also $x \in \mu(x_1) \cup \mu(x_2)$, and thus, by the definition of the chase step, also $(x, y) \in \sim^e$, which coincides with the previous case (for which we have already shown that transitivity holds).

If d is a tuple generating dependency, then $\sim^{\mathcal{I}_2} = \sim^{\mathcal{I}_1} \cup \{\langle f, f \rangle \mid f \text{ is a value-null or entity-null not occurring in } \Delta_E^{\mathcal{I}_1}\}$ (notice that the set of pairs of fresh nulls could even be empty). Thus $\sim^{\mathcal{I}_2}$ is clearly an equivalence relation.

We finally prove that $\cdot^{\mathcal{I}_2}$ is an interpretation function according to Definition 8.1. This however can be readily seen by the definition of both the tgdc and entity resolution chase steps (in particular, note that in the tgdc chase step the interpretation of a fresh entity-null $f^{\mathcal{I}_2}$ is the singleton $\{f\}$, which is the equivalence class of f w.r.t. $\sim^{\mathcal{I}_2}$, whereas in the entity resolution rule step $\mu(x_1) \cup \mu(x_2)$ is an equivalence class w.r.t. $\sim^{\mathcal{I}_2}$). \square

Definition 8.9 (Chase sequence). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a KER system, a *chase sequence* for \mathcal{K} , denoted $\sigma_{\mathcal{C}}(\mathcal{K})$, is a sequence (finite or infinite) $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2 \dots$, such that $\mathcal{I}_0 = \mathcal{I}^A$, and $\mathcal{I}_i \xrightarrow{d_i, \mu_i} \mathcal{I}_{i+1}$ is a chase step, for each pair $\mathcal{I}_i, \mathcal{I}_{i+1}$ in the sequence. We also say that a chase sequence is *fair* if in every chase step $\mathcal{I}_i \xrightarrow{d_i, \mu_i} \mathcal{I}_{i+1}$, μ_i is the assignment that precedes every other assignment that is triggering a tgdc or entity resolution rule in \mathcal{I}_i , and d_i is the tgdc or entity resolution rule triggered by μ_i in \mathcal{I}_i .⁴

It is easy to see that for a KER system \mathcal{K} there is exactly one fair chase sequence, thus we refer to it as *the* fair chase sequence for \mathcal{K} .

We define the length of a chase sequence $\sigma_{\mathcal{C}}(\mathcal{K})$, denoted $\text{len}(\sigma_{\mathcal{C}}(\mathcal{K}))$, has the number of interpretations occurring in $\sigma_{\mathcal{C}}(\mathcal{K})$. When the sequence is infinite, $\text{len}(\sigma_{\mathcal{C}}(\mathcal{K})) = \infty$.

It is worthwhile to note also that, let \mathcal{I}_i and \mathcal{I}_j two elements in $\sigma_{\mathcal{C}}(\mathcal{K})$ such that $j > i$, $\Delta^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_j}$, $\sim^{\mathcal{I}_i} \subseteq \sim^{\mathcal{I}_j}$, and thus that $[e]^{\sim^{\mathcal{I}_i}} \subseteq [e]^{\sim^{\mathcal{I}_j}}$, for each entity or entity null $e \in \Delta_E^{\mathcal{I}_i}$.

We now prove that a (possibly infinite) fair chase sequence is well defined, which intuitively means that if an assignment is triggering a tgdc or entity resolution rule d at some chase step, d will eventually be triggered by such an assignment at a next chase step, or alternatively, the assignment will become non-triggering, due to the execution of other chase steps.

To precisely formalize the above property we need to introduce the additional definition of propagation of an assignment. Let \mathcal{I}_i and \mathcal{I}_j be two interpretations in the chase sequence such that $j > i$, let d be either a tgdc or an entity resolution rule, and let μ be an assignment from the body of d to \mathcal{I}_i . The *propagation of μ to \mathcal{I}_j* , denoted $\mu^{\rightarrow j}$, is an assignment from the body of d to \mathcal{I}_j such that $\mu^{\rightarrow j}(x) = [e]^{\sim^{\mathcal{I}_j}}$, for every variable x in the body of d such that $\mu(x) = [e]^{\sim^{\mathcal{I}_i}}$. Note that since $[e]^{\sim^{\mathcal{I}_i}} \subseteq [e]^{\sim^{\mathcal{I}_j}}$ for each \mathcal{I}_i and \mathcal{I}_j belonging to the chase sequence such that $j > i$ and each $e \in \Delta_E^{\mathcal{I}_i}$, $\mu^{\rightarrow j}$ correctly associates each variable x to exactly one equivalence class in $\Delta_E^{\mathcal{I}_j} / \sim^{\mathcal{I}_j}$.

Proposition 8.4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a KER system, and d be either a tgdc in \mathcal{T} or an entity resolution rule in \mathcal{E} , and $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2 \dots$ be the fair chase sequence for \mathcal{K} . If there is an assignment μ that triggers d in an interpretation \mathcal{I}_i in the sequence, then there is an interpretation \mathcal{I}_j in the sequence such that $j > i$ and $\mu^{\rightarrow j}$ doesn't trigger d in \mathcal{I}_j .*

⁴Since we have assumed that different rules use different variables, the rule d_i is unique.

Proof. When the fair chase sequence is finite, the property is trivially verified. Let us thus consider the case of an infinite chase sequence. Assume by contradiction that for every $j > i$ there exists at least an assignment μ' different from μ that triggers either a tgdt or an entity resolution rule d' in \mathcal{I}_j , such that μ' precedes μ (and thus μ' is used in the chase step from \mathcal{I}_j to \mathcal{I}_{j+1}). This implies that there are infinitely many such assignments. Since \mathcal{A} , \mathcal{T} and \mathcal{E} in \mathcal{K} are finite sets, since $S^{\mathcal{I}}$ is a finite set for each predicate S occurring in \mathcal{A} or \mathcal{T} and each interpretation \mathcal{I} in the chase sequence, and since every variable occurring in a built-in predicate of an entity resolution rule occurs also in a non-built-in predicate, from every tgdt or entity resolution rule there is always a finite number of assignments to an interpretation in the chase sequence. Therefore, the infinitely many assignments that precede μ have to be generated through the (infinitely many) steps that follow \mathcal{I}_i in the chase sequence. Furthermore, since the ABox \mathcal{A} contains a finite number of entities from \mathbf{N}_E and values from \mathbf{N}_V , the images of the infinitely many assignments that precede μ must involve entity-nulls and/or value-nulls. Without loss of generality, let us assume that each chase step generates one such assignment, and that μ' triggering d' in \mathcal{I}_j has such form and is generated through the chase step $\mathcal{I}_{j-1} \xrightarrow{d_{j-1}, \mu_{j-1}} \mathcal{I}_j$. This means that μ' does not trigger d' in \mathcal{I}_{j-1} , that d_{j-1} is a tgdt, and that μ_{j-1} is extended by means of fresh value-nulls or fresh entity-nulls that make μ' triggering d' in \mathcal{I}_j (remember also that μ' precedes μ). However, by definition of chase step, fresh value-nulls and fresh entity-nulls used to extend μ_{j-1} must follow alphabetically all entities, values, entity-nulls and value-nulls occurring in the domain of \mathcal{I}_{j-1} , which leads to a contradiction. \square

In the following we only consider fair chase sequences, and thus we may simply call one such sequence the chase sequence.

We are now ready to provide the definition of chase.

Definition 8.10 (Chase). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a KER system, and let $\sigma_{\mathcal{C}}(\mathcal{K}) = \mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2 \dots$ be the chase sequence for \mathcal{K} . The *chase of \mathcal{K}* , denoted $chase(\mathcal{K})$, is the triple $\langle \Delta^{c^{\mathcal{K}}}, \cdot^{c^{\mathcal{K}}}, \sim^{c^{\mathcal{K}}} \rangle$, such that:

- $\Delta^{c^{\mathcal{K}}} = \Delta_E^{c^{\mathcal{K}}} \cup \Delta_V^{c^{\mathcal{K}}}$, where:
 - $\Delta_E^{c^{\mathcal{K}}} = \{e \mid \text{there exists } \mathcal{I}_i \in \sigma_{\mathcal{C}}(\mathcal{K}) \text{ such that } e \in \Delta_E^{\mathcal{I}_i}\}$.
 - $\Delta_V^{c^{\mathcal{K}}} = \{v \mid \text{there exists } \mathcal{I}_i \in \sigma_{\mathcal{C}}(\mathcal{K}) \text{ such that } v \in \Delta_V^{\mathcal{I}_i}\}$.
- $\sim^{c^{\mathcal{K}}} = \{\langle e_1, e_2 \rangle \mid \text{there exists } \mathcal{I}_i \in \sigma_{\mathcal{C}}(\mathcal{K}) \text{ such that } \langle e_1, e_2 \rangle \in \sim^{\mathcal{I}_i}\}$;
- $e^{c^{\mathcal{K}}} = [e]_{\sim^{c^{\mathcal{K}}}}$, for each entity or entity-null $e \in \Delta_E^{c^{\mathcal{K}}}$;
- $v^{c^{\mathcal{K}}} = v$, for each value or value-null $v \in \Delta_V^{c^{\mathcal{K}}}$;

- $C^{c^{\mathcal{K}}} = \{[e] \sim^{c^{\mathcal{K}}} \mid \text{there exists } \mathcal{I}_i \in \sigma_{\mathcal{C}}(\mathcal{K}) \text{ such that } [e] \sim^{\mathcal{I}_i} \in C^{\mathcal{I}_i}\}$, for each atomic concept $C \in \mathbf{N}_{\mathcal{C}}$;
- $R^{c^{\mathcal{K}}} = \{([e_1] \sim^{c^{\mathcal{K}}}, [e_2] \sim^{c^{\mathcal{K}}}) \mid \text{there exists } \mathcal{I}_i \in \sigma_{\mathcal{C}}(\mathcal{K}) \text{ such that } ([e_1] \sim^{\mathcal{I}_i}, [e_2] \sim^{\mathcal{I}_i}) \in R^{\mathcal{I}_i}\}$, for each atomic role $R \in \mathbf{N}_{\mathcal{R}}$;
- $A^{c^{\mathcal{K}}} = \{([e] \sim^{c^{\mathcal{K}}}, v) \mid \text{there exists } \mathcal{I}_i \in \sigma_{\mathcal{C}}(\mathcal{K}) \text{ such that } ([e] \sim^{\mathcal{I}_i}, v) \in A^{\mathcal{I}_i}\}$, for each attribute $A \in \mathbf{N}_{\mathcal{A}}$.

In the remaining part of this section we prove that $\text{chase}(\mathcal{K})$ is a universal model of a KER system \mathcal{K} . To this aim, we first show that $\text{chase}(\mathcal{K})$ is an interpretation, then that it is a model, and finally we prove that it is also universal.

Lemma 8.1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a KER system. Then $\text{chase}(\mathcal{K}) = \langle \Delta^{c^{\mathcal{K}}}, \cdot^{c^{\mathcal{K}}}, \sim^{c^{\mathcal{K}}} \rangle$ is an interpretation.*

Proof. Let $\sigma_{\mathcal{C}}(\mathcal{K}) = \mathcal{I}_0, \mathcal{I}_1, \dots$ be the chase sequence of \mathcal{K} . We soon recall that each \mathcal{I}_i is an interpretation, by the virtue of Proposition 8.3.

In order to prove that $\text{chase}(\mathcal{K})$ is an interpretation, we first prove that $\Delta^{c^{\mathcal{K}}}$ has the form given in Definition 8.1. From Definition 8.10, we have that $\Delta^{c^{\mathcal{K}}} = \Delta_E^{c^{\mathcal{K}}} \cup \Delta_V^{c^{\mathcal{K}}}$, and it is easy to see that $\Delta_E^{c^{\mathcal{K}}} = \bigcup_{i=0}^{\text{len}(\sigma_{\mathcal{C}}(\mathcal{K}))} \Delta_E^{\mathcal{I}_i}$ and $\Delta_V^{c^{\mathcal{K}}} = \bigcup_{i=0}^{\text{len}(\sigma_{\mathcal{C}}(\mathcal{K}))} \Delta_V^{\mathcal{I}_i}$, thus $\Delta_E^{\mathcal{I}_0} \subseteq \Delta_E^{c^{\mathcal{K}}}$ and $\Delta_V^{\mathcal{I}_0} \subseteq \Delta_V^{c^{\mathcal{K}}}$, and since $\mathcal{I}_0 = \mathcal{I}_{\mathcal{A}}$, the thesis easily follows from Definition 8.7.

We then prove that $\sim^{c^{\mathcal{K}}}$ is an equivalence relation. From Definition 8.8 and Definition 8.9, it is easy to see that $\sim^{\mathcal{I}_i} \subseteq \sim^{\mathcal{I}_{i+1}}$ for $0 \leq i < \text{len}(\text{chase}(\mathcal{K})) - 1$, and thus $\sim^{c^{\mathcal{K}}} = \bigcup_{i=0}^{\text{len}(\text{chase}(\mathcal{K})) - 1} \sim^{\mathcal{I}_i}$. Then, the thesis easily follows from Proposition 8.3, which in particular says that $\sim^{\mathcal{I}_i}$ for each \mathcal{I}_i in $\sigma_{\mathcal{C}}(\mathcal{K})$ is an equivalence relation.

Finally we prove that $\cdot^{c^{\mathcal{K}}}$ is an interpretation function as given in Definition 8.1. This however follows straightforwardly from Definition 8.10 and the fact that, as proved above, $\sim^{c^{\mathcal{K}}}$ is an equivalence relation and $\Delta^{c^{\mathcal{K}}}$ has the form given in Definition 8.1. □

We now prove that the chase is also a model of the KER system.

Lemma 8.2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a KER system. Then $\text{chase}(\mathcal{K}) = \langle \Delta^{c^{\mathcal{K}}}, \cdot^{c^{\mathcal{K}}}, \sim^{c^{\mathcal{K}}} \rangle$ is a model of \mathcal{K} .*

Proof. From Lemma 8.1, it follows that $\text{chase}(\mathcal{K})$ is an interpretation. We need to prove that $\text{chase}(\mathcal{K})$ satisfies all membership assertions in \mathcal{A} , all tgds in \mathcal{T} and all entity resolution rules in \mathcal{E} . We show each point separately:

- (i) $\text{chase}(\mathcal{K})$ satisfies all membership assertions in \mathcal{A} since $\mathcal{I}_0 = \mathcal{I}^{\mathcal{A}}$ and $\mathcal{I}^{\mathcal{A}}$ satisfies \mathcal{A} . More precisely, by Definition 8.7, $[e]^{\sim_{\mathcal{I}_0}} \in C^{\mathcal{I}_0}$ for each membership assertion $C(e) \in \mathcal{A}$, and thus, according to Definition 8.10, $[e]^{\sim_{e^{\mathcal{K}}}} \in C^{e^{\mathcal{K}}}$, which means that $\text{chase}(\mathcal{K})$ satisfies $C(e)$. Satisfaction of the other forms of membership assertions can be proved analogously.
- (ii) To prove that $\text{chase}(\mathcal{K})$ satisfies all tgds in \mathcal{T} , let us assume by contradiction that there is a tgd $d = \forall \vec{x}. \phi(\vec{x}) \rightarrow \exists \vec{y}. \psi(\vec{x}, \vec{y})$ that is not satisfied by $\text{chase}(\mathcal{K})$. This means that there is an assignment μ from $\phi(\vec{x})$ to $\text{chase}(\mathcal{K})$ such that there does not exist an extension μ' of μ from $\phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y})$ to $\text{chase}(\mathcal{K})$. But thus d is triggered by μ in an interpretation \mathcal{I}_i belonging to $\sigma_{\mathcal{C}}(\mathcal{K})$, and from Proposition 8.4 we get a contradiction.
- (iii) The fact that $\text{chase}(\mathcal{K})$ satisfies all the entity resolution rules in \mathcal{E} can be proved as done in (ii) for satisfaction of tgds in \mathcal{T} . □

The following Lemma will be exploited to prove that $\text{chase}(\mathcal{K})$ is not simply a model, but it is also universal. A similar Lemma has been used in [89] to prove an analogous result in the context of Data Exchange.

Lemma 8.3. *Let $\mathcal{I}_1 \xrightarrow{d, \mu} \mathcal{I}_2$ be a chase step. Let \mathcal{I} be an interpretation such that: (i) \mathcal{I} satisfies d and (ii) there exists a homomorphism $h_1 : \mathcal{I}_1 \rightarrow \mathcal{I}$. Then there exists a homomorphism $h_2 : \mathcal{I}_2 \rightarrow \mathcal{I}$.*

Proof. The proof follows the line of the proof of Lemma 3.4 in [89], and exploits the fact that the composition of an assignment with an homomorphism yields a new assignment. However, since our structures are based on equivalence classes, the various steps of the proof are more involved.

Case of tgd chase step: Let d be the tgd $\phi(\vec{x}) \rightarrow \exists \vec{y}. \psi(\vec{x}, \vec{y})$. By the definition of the chase step, μ is an assignment from $\phi(\vec{x})$ to \mathcal{I}_1 . We define the composition of homomorphism h_1 with assignment μ , denoted $h_1 \circ \mu$, as follows: (i) $h_1 \circ \mu(x) = h_1(\mu(x)) = [h_1(e)]^{\sim_{\mathcal{I}}}$ if $\mu(x) = [e]^{\sim_{\mathcal{I}_1}}$ and $e \in \Delta_E^{\mathcal{I}_1}$, and (ii) $h_1 \circ \mu(x) = h_1(\mu(x)) = h_1(v)$ if $\mu(x) = v$ and $v \in \Delta_V^{\mathcal{I}_1}$. We notice that point (i) of the above definition is independent from the representative element chosen for the equivalence class $[e]^{\sim_{\mathcal{I}_1}}$. This follows from the following property, which is straightforward from the definition of homomorphism (Definition 8.3): let \mathcal{J}_1 and \mathcal{J}_2 be a pair of interpretations, $E \in \Delta_E^{\mathcal{J}_1} / \sim^{\mathcal{J}_1}$ be an equivalence class, h be an homomorphism from \mathcal{J}_1 to \mathcal{J}_2 , and let e_1, e_2 be two elements of E , then $h(e_1)$ and $h(e_2)$ belong to the same equivalence class in $\Delta_E^{\mathcal{J}_2} / \sim^{\mathcal{J}_2}$.

It is easy to see that $h_1 \circ \mu$ is an assignment from $\phi(\vec{x})$ to \mathcal{I} .

Since \mathcal{I} satisfies d , there is an assignment μ' from $\phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y})$ to \mathcal{I} , such that μ' is an extension of $h_1 \circ \mu$, that is, $\mu'(x) = h_1(\mu(x))$ for each x in \vec{x} . For each variable y in \vec{y} , we denote

by f_y each labeled fresh null introduced in $\Delta^{\mathcal{I}_2}$ by the chase step to extend the assignment μ , i.e., f_y is either a fresh entity-null such that y is assigned to the singleton containing f_y , or a fresh value-null such that y is assigned to f_y . Let us now define the mapping h_2 from elements in $\Delta^{\mathcal{I}_2}$ to elements in $\Delta^{\mathcal{I}}$ as follows. For every $z \in \Delta^{\mathcal{I}_2} \cap \Delta^{\mathcal{I}_1}$, $h_2(z) = h_1(z)$. Then, for every $f_y \in \Delta^{\mathcal{I}_2}$ (by construction $f_y \notin \Delta^{\mathcal{I}_1}$), $h_2(f_y) = \mu'(y)$ if y is assigned by μ' to a value-null in $\Delta_V^{\mathcal{I}}$, and $h_2(f_y) = e$ if y is assigned by μ' to an equivalence class $E \in \Delta_E^{\mathcal{I}} / \sim^{\mathcal{I}}$ and e is any element in E . Notice that in this way h_2 is defined for all the elements in $\Delta^{\mathcal{I}_2}$, since $\Delta^{\mathcal{I}_2}$ is given by the union of $\Delta^{\mathcal{I}_1}$ and the fresh nulls introduced by the chase step.

We now show that h_2 is an homomorphism from \mathcal{I}_2 to \mathcal{I} . The first four properties of homomorphisms given in Definition 8.3 are satisfied by construction of h_2 . We have then to prove that if $e_1 \sim^{\mathcal{I}_2} e_2$ then $h_2(e_1) \sim^{\mathcal{I}} h_2(e_2)$ for each $e_1, e_2 \in \Delta_E^{\mathcal{I}_2}$. In the case in which $e_1, e_2 \in \Delta_E^{\mathcal{I}_2} \cap \Delta_E^{\mathcal{I}_1}$, then the property can be easily shown, since $e_1 \sim^{\mathcal{I}_2} e_2$ implies $e_1 \sim^{\mathcal{I}_1} e_2$, by definition of the chase step, $h_2(e_1) = h_1(e_1)$ and $h_2(e_2) = h_1(e_2)$, by definition of h_2 , and h_1 is an homomorphism from \mathcal{I}_1 to \mathcal{I} . If $e_1, e_2 \in \Delta_E^{\mathcal{I}_2} \setminus \Delta_E^{\mathcal{I}_1}$, they are either different fresh nulls, and thus they are not equivalent in $\sim^{\mathcal{I}_2}$, or the same fresh null f , and we have only to verify whether $f \sim^{\mathcal{I}_2} f$ implies $h_2(f) \sim^{\mathcal{I}} h_2(f)$, which is obviously true, since $\sim^{\mathcal{I}}$ is reflexive. It remains to prove that if $[e']^{\sim^{\mathcal{I}_2}} \in A^{\mathcal{I}_2}$ then $[h_2(e')]^{\sim^{\mathcal{I}}} \in A^{\mathcal{I}}$, for each atomic concept in the KER system, and analogously for atomic roles and attributes. We show the case of atomic concepts. The other cases can be proved similarly. When $e' \in \Delta_E^{\mathcal{I}_2} \cap \Delta_E^{\mathcal{I}_1}$, then, by definition of chase step, $[e']^{\sim^{\mathcal{I}_2}} \in C^{\mathcal{I}_2}$ implies $[e']^{\sim^{\mathcal{I}_1}} \in C^{\mathcal{I}_1}$, and since h_1 is an homomorphism from \mathcal{I}_1 to \mathcal{I} , then $[e']^{\mathcal{I}_1} \in A^{\mathcal{I}_1}$ implies $[h_1(e')]^{\mathcal{I}} \in A^{\mathcal{I}}$. Since in this case $h_2(e') = h_1(e')$, the property is verified. When $e' \in \Delta_E^{\mathcal{I}_2} \setminus \Delta_E^{\mathcal{I}_1}$, then $e' = f_y$, for some variable y occurring in \vec{y} in entity position. By definition of h_2 , in this case $h_2(f_y) = e$, where e is any element in the equivalence class $\mu'(y)$. Since f_y is a fresh entity-null, and $[f_y]^{\sim^{\mathcal{I}_2}} \in C^{\mathcal{I}_2}$, then the atom $C(y)$ must occur in $\psi(\vec{x}, \vec{y})$, and since μ' in an assignment from $\phi(\vec{x}) \wedge \psi(\vec{x}, \vec{y})$ to \mathcal{I} (remember that \mathcal{I} satisfies d), then $\mu'(y) \in C^{\mathcal{I}}$, that is $[h_2(f_y)]^{\sim^{\mathcal{I}}} \in C^{\mathcal{I}}$.

Case of entity resolution rule chase step: Let d be the entity resolution rule $\phi(\vec{x}) \rightarrow \sim(x_1, x_2)$. We take h_2 to be h_1 (notice that $\Delta^{\mathcal{I}_2} = \Delta^{\mathcal{I}_1}$ when d is an entity resolution rule). We have to show that h_1 is still a homomorphism when considered from \mathcal{I}_2 to \mathcal{I} . Obviously, h_1 satisfies the first four conditions given in Definition 8.3. We have then to prove that $e_1 \sim^{\mathcal{I}_2} e_2$ implies $h_1(e_1) \sim^{\mathcal{I}} h_1(e_2)$. We recall that, by definition of chase step $\sim^{\mathcal{I}_1} \subseteq \sim^{\mathcal{I}_2}$. When e_1 and e_2 are such that also $e_1 \sim^{\mathcal{I}_1} e_2$, then obviously the property is verified, since h_1 is an homomorphism from \mathcal{I}_1 to \mathcal{I} . The other possible case is that $e_1 \not\sim^{\mathcal{I}_1} e_2$. This means that $e_1 \sim^{\mathcal{I}_2} e_2$ is introduced by the chase step, and that $e_1 \in \mu(x_1)$ and $e_2 \in \mu(x_2)$. Since, as shown above for the tgcd chase step, $h_1 \circ \mu$ is an assignment from $\phi(\vec{x})$ to \mathcal{I} , and \mathcal{I} satisfies d , it necessarily holds that $h_1(\mu(x_1)) = h_1(\mu(x_2))$, which implies that $e_1 \sim^{\mathcal{I}} e_2$. We now prove that if $[e]^{\sim^{\mathcal{I}_2}} \in A^{\mathcal{I}_2}$ then

$[h_1(e)]^{\sim \mathcal{I}} \in C^{\mathcal{I}}$, for each atomic concept C in the **KER** system and entity $e \in \Delta_{\mathcal{E}}^{\mathcal{I}_2}$ (the case of atomic role and of attribute are analogous). Since $[e]^{\sim \mathcal{I}_1} \subseteq [e]^{\sim \mathcal{I}_2}$, and $[e]^{\sim \mathcal{I}_1}$ is not empty, there is always $\hat{e} \in [e]^{\sim \mathcal{I}_1} \cap [e]^{\sim \mathcal{I}_2}$. Since h_1 is an homomorphism from \mathcal{I}_1 to \mathcal{I} then $[h_1(\hat{e})]^{\sim \mathcal{I}} \in C^{\mathcal{I}}$. The thesis easily follows from the fact that $[h_1(\hat{e})]^{\sim \mathcal{I}} = [h_1(e)]^{\sim \mathcal{I}}$, for every $e \in [e]^{\sim \mathcal{I}_2}$. \square

Theorem 8.2. *Let \mathcal{K} be a **KER** system, $\text{chase}(\mathcal{K})$ is a universal model of \mathcal{K} .*

Proof. From Lemma 8.1 it follows that $\text{chase}(\mathcal{K})$ is an interpretation, whereas from Lemma 8.2 it follows that $\text{chase}(\mathcal{K})$ is a model for \mathcal{K} . It remains to prove that it is universal, i.e., that there exists an homomorphism from $\text{chase}(\mathcal{K})$ to every model of \mathcal{K} . To this aim we need an additional property

Lemma 8.4. *Let \mathcal{J} be a model for \mathcal{K} and let \mathcal{I}_0 be the first interpretation in $\sigma_{\mathcal{C}}(\mathcal{K})$. The identity function $\text{id} : \Delta^{\mathcal{I}_0} \rightarrow \Delta^{\mathcal{J}}$, i.e., such that $\text{id}(x) = x$ for each element $x \in \Delta^{\mathcal{I}_0}$, is an homomorphism from \mathcal{I}_0 to \mathcal{J} .*

Proof. We have to prove that the function id is compliant with Definition 8.3 (homomorphism). First thing, we recall that $\mathcal{I}_0 = \mathcal{I}^A$, and note that $\Delta^{\mathcal{I}^A} \subseteq \Delta^{\mathcal{J}}$, since $\Delta^{\mathcal{I}^A} = \Delta_{\mathcal{E}}^{\mathcal{I}^A} \cup \Delta_{\mathcal{V}}^{\mathcal{I}^A}$, where $\Delta_{\mathcal{E}}^{\mathcal{I}^A} = \mathbf{N}_{\mathcal{E}}$ and $\Delta_{\mathcal{V}}^{\mathcal{I}^A} = \mathbf{N}_{\mathcal{V}}$ (see Definition 8.7), and $\mathbf{N}_{\mathcal{E}} \cup \mathbf{N}_{\mathcal{V}} \subseteq \Delta^{\mathcal{J}}$ by definition of interpretation (see Definition 8.1). Thus id is also a function from $\Delta^{\mathcal{I}_0}$ to $\Delta^{\mathcal{J}}$. It is then easy to see that id satisfies the conditions that an homomorphism has to respect when applied to entities, values, entity-nulls and value-nulls (the last two cases are trivially satisfied since \mathcal{I}^A does not contain null values). Then we have to show that if $\langle e_1, e_2 \rangle \in \sim^{\mathcal{I}^A}$, then $\langle h(e_1), h(e_2) \rangle \in \sim^{\mathcal{J}}$, for each $e_1, e_2 \in \Delta_{\mathcal{E}}^{\mathcal{I}^A}$, but this also is straightforward, since $\sim^{\mathcal{I}^A}$ contains only pairs of the form $\langle e, e \rangle$, which are mapped to themselves by the id function and obviously belong to $\sim^{\mathcal{J}}$ because of the reflexivity of the equality relation. Let C be an atomic concept, we have to show that if $[e]^{\sim \mathcal{I}^A} \in C^{\mathcal{I}^A}$, then $[\text{id}(e)]^{\sim \mathcal{J}} \in C^{\mathcal{J}}$. This follows from the fact that $C^{\mathcal{I}^A}$ contains all and only the classes $[e]^{\mathcal{I}^A}$ such that $C(e) \in \mathcal{A}$, and from the fact that \mathcal{J} satisfies \mathcal{A} and $[\text{id}(e)]^{\sim \mathcal{J}} = [e]^{\sim \mathcal{J}}$. The remaining conditions on satisfaction of atomic roles and attributes can be proved analogously. \square

The thesis can be thus proved by applying Lemma 8.3 to each $\mathcal{I}_i, \mathcal{I}_{i+1}$ in $\sigma_{\mathcal{C}}(\mathcal{K})$, starting from $\mathcal{I}_0, \mathcal{I}_1$ with, $h_1 = \text{id}$ homomorphism from \mathcal{I}_0 to a model \mathcal{J} of \mathcal{K} . \square

We conclude this section with an interesting result about the data complexity of chase computation, for the cases in which the chase terminates.

Theorem 8.3. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a **KER** system. If $\text{chase}(\mathcal{K})$ terminates, then it can be computed in polynomial time in data complexity.*

Proof. (Sketch) When the chase terminates, there is a polynomial in the size of \mathcal{A} that bounds the number of new facts that can be generated through the chase steps. Indeed, the number of possible assignments triggering a rule at a certain chase step is expressible through a polynomial in the size of \mathcal{A} of degree b , where b is the maximum number of atoms in the body of a rule. Then, the number of atoms that can be generated at each chase step is bounded by the maximum number of atoms in the head of a tgd. Based on the above arguments, it is possible to show that the degree of the polynomial we are looking for is $k \cdot b$, where $k = \text{len}(\sigma_{\mathcal{C}}(\mathcal{K}))$, which is a positive integer by hypothesis. Since each fact is never generated more than once in a chase sequence, then the thesis follows. \square

The next result is an immediate consequence of Theorem 8.3 and Theorem 8.1, since a *DL-Lite_{RDFS}* TBox can be expressed as a set of full tgds (i.e., tgds without existentially quantified variables in their heads), for which the chase always terminates [89].

Corollary 8.2. *Answering conjunctive queries in KER systems with a DL-Lite_{RDFS} TBox is polynomial in data complexity.*

We notice that besides the *DL-Lite_{RDFS}* case mentioned above, it is possible to show that query answering is polynomial in data complexity for some settings of tgds enjoying conditions that ensure termination of the (classical) restricted chase, such as, e.g., for weakly acyclic tgds [89] or for sets of tgds respecting the acyclicity conditions described in ???. We point out that termination of the chase is not the only condition ensuring that conjunctive query answering can be solved in our framework. Cases in which the chase is infinite are definitely worthwhile to be investigated, as e.g., KER systems with a *DL-Lite_R* TBox. In these cases, however, as done in the literature of ontology-mediated query answering [26], approaches different from simple chase computation have to be obviously considered, such as investigating whether the possibly infinite chase can be represented using finitary means, or whether query answering can be solved through query rewriting, or even combining (partial) chase computation with query rewriting (in standard ontology-mediated query answering, examples of the previous approaches are [37, 38, 155, 50]).

8.6 Adding functionalities

Some popular ontology languages tailored to data management, as, e.g., *DL-Lite_F* [46], are equipped with functionality assertions (or simply functionalities). Whereas in *DL-Lite_F* functionalities can be asserted only on roles (see Section 5.2), DLs allowing for attributes (as *DL-Lite_A* [176]) also consider functionalities on attributes. In both cases, such assertions state that an entity cannot have more than one filler with respect to a certain role or attribute, that

is, given an entity e and a functional role R (resp. attribute A), there is at most an entity e' (resp. value v), i.e., at most one filler, that R (resp. A) associates to e . Such assertions are similar to restrictions imposing maximum cardinality 1 on relationships or attributes in Entity-Relationship diagrams (or in UML class diagrams).

We soon notice that at the semantic level functionalities on roles act as entity resolution rules. Indeed, different fillers of a functional role for a certain entity can be interpreted with the same real-world object, which in our framework means with the same equivalence class. This correspondence holds also at the syntactic level, since a functionality on a role can be expressed through an entity resolution rule of the form (8.2). For example, the assertion stating that the role *lives_in* is functional (expressed, e.g., as $\text{funct}(\textit{lives_in})$ in *DL-Lite_F*) can be specified as the rule $\forall x, y, z. \textit{lives_in}(x, y) \wedge \textit{lives_in}(x, z) \rightarrow \sim(y, z)$ (note that y and z are entity-variables).

On the other hand, functionalities on attributes are not expressible as entity resolution rules since they enforce an equality on value-variables, which might not be satisfied by any interpretation, possibly resulting in an inconsistent ontology. The above situation often arises in contexts such as data integration or information extraction, where data coming from different databases or documents may result contradictory with respect to the rules of the ontology (e.g., a person having two dates of birth, an attribute which is of course usually asserted as functional). The number of conflicts of the above kind is typically increased by the presence of entity resolution rules, which lead to identify different entities, possibly coming from different data sources and associated to different values by the same functional attribute.

Typical approaches to cope with such inconsistencies are procedural data cleaning [113, 181], which aims at solving the inconsistency through ad-hoc cleansing algorithms, or consistent query answering [22, 25, 21, 133], which “tolerate” the inconsistency through the definition of tailored semantics based on repairs, i.e., consistent data instances that are as close as possible to the original instance (typically are its inclusion- or cardinality-maximal consistent subsets), and compute answers to queries by reasoning over such repairs and the schema of the data or knowledge base.

Our approach to manage violations of functional attributes follows a different direction. We interpret functionalities on attributes as matching dependencies [23, 92], which are rules specifying conditions under which certain values of attributes have to be matched. Thus, instead of trying to equate the values as imposed by a functional attribute assertion, we combine them through a merging function, so that we can still have a structure satisfying the functionality. We adopt a general and common merging function that takes the union of the values filling functional attributes for a single entity. This function actually belongs to the *Union class of match and merge functions* introduced in [18] and also analyzed in [23].

In the rest of this section we investigate query answering over KER systems whose TBox

is as a set of tgds plus functionality assertions. As said, we see role functionalities as (special) entity resolution rules, and thus hereinafter we do not explicitly consider them. We instead give a novel definition of the interpretation of functional attributes, to see them as matching dependencies enforcing merging of values through a union function. This requires to revise the notion of interpretation of a **KER** system. Consequently, we need to modify the notions of homomorphism between **KER** interpretations, universal model, certain answers, and the chase. We also show that the chase and query answering properties proved in the previous sections continue to hold under the new semantics, and in particular Theorem 8.1, Theorem 8.2 and Theorem 8.3 are still valid for **KER** systems equipped with functionality assertions on attributes.

8.6.1 Functionalities on attributes as matching dependencies

Given an attribute A , a *functionality assertion* on A is defined as an equality generating dependency of the following form:

$$\forall x, y, z. A(x, y) \wedge A(x, z) \rightarrow y = z \quad (8.4)$$

Notice that this kind of assertion is not captured by the syntax of entity resolution rules (see equation (8.2)), since y and z are value-variables (thus they are not entity-variables as required in (8.2)). Notice also that Equation (8.4) expresses as a first-order rule a property that in DL syntax is typically specified as $\text{funct}(A)$ (see also Section 5.2).

As already said, in the following we revise the semantics of a **KER** system in order to properly interpret functional assertions of the form (8.4) in the spirit of matching dependencies. In particular, we will consider interpretations that assign ranges of attributes (i.e., the second component of attribute predicates) with *sets of values*, rather than a single value, as done so far. This will allow us to cope with situations in which a standard interpretation of functional attributes would lead to an inconsistency. At the same time this gives us the opportunity to consider new forms of selections and joins over ranges of attributes. To this aim we introduce a new type of atom to be used in the body of tgds and entity resolution rules, or in conjunctive queries. More precisely, we call *matching atom* an expression the form $t_1 \approx_m t_2$, where t_1 and t_2 are variables or values in \mathbf{N}_V , and at least one of them is a variable⁵.

Intuitively, the intended meaning of a matching atom of the form $y \approx_m v$, where y is a variable and v is a value, is that the v has to belong to the set of values binding the variable y . Similarly, an atom of the form $y_1 \approx_m y_2$, where y_1 and y_2 are variables, expresses that the intersection of the sets of values binding the variables y_1 and y_2 has to be non-empty.

⁵Our results apply straightforwardly to the more general case where t_1 and t_2 can be variables or *sets* of values, but we do not explicitly consider this option for ease of exposition.

Remark. Hereinafter, we consider tgds and entity resolution rules of the form given in (8.1) and (8.2), respectively, where the conjunction $\phi(\vec{x})$ (rule body) may contain matching atoms involving value-variables (that is, variables that occur in at least a value-position in some other non-matching atom in $\phi(\vec{x})$).⁶

We notice that standard selections and joins between ranges of attributes are still allowed in $\phi(\vec{x})$. As usual, they are expressed through atoms of the form $A(x, v)$, where A is an attribute, x is a variable, and v is a value, or through multiple occurrences of the same value-variable in different non-matching atoms, respectively. Thus, we may express in $\phi(\vec{x})$ both exact and approximate selections or joins on ranges of attributes. For instance, the conjunction $A_1(x, y_1) \wedge A_2(z, y_2) \wedge A_3(w, y_2) \wedge y_1 \approx_m y_2$ is satisfied by assigning the same set of values Ω to the ranges of A_2 and A_3 (variable y_2) and a set of value Ω' to the range of A_1 such that $\Omega \cap \Omega' \neq \emptyset$.

In the following, we provide a formal characterization of the semantics of rules allowing for matching atoms. We first revise the notion of interpretation given in Definition 8.1, and then the notion of assignment given in Definition 8.2.

Definition 8.11 (interpretation (revisited)). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be KER system, such that \mathcal{T} is a set of tgds of the form (8.1) and functionalities of the form (8.4). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \sim^{\mathcal{I}})$ for \mathcal{K} is defined as in Definition 8.1, except for the interpretation of attributes and built-in predicates, for which the following conditions hold⁷

- $A^{\mathcal{I}} \subseteq \Delta_E^{\mathcal{I}} / \sim^{\mathcal{I}} \times 2^{\Delta_V^{\mathcal{I}}}$, for each attribute $A \in \mathbf{N}_A$;
- $F^{\mathcal{I}} \subseteq \times_{i=1}^n 2^{\mathbf{N}_V}$, for each n -ary built-in predicate $F \in \mathbf{N}_F$.

Based on the above definition, we have also to modify the notion of satisfaction of ABox assertions of the form $A(e, v)$, where A is an attribute in \mathbf{N}_A , e is an entity in \mathbf{N}_E , and v is a value in \mathbf{N}_V . Namely, we say that an interpretation \mathcal{I} satisfies the ABox assertion $A(e, v)$ if there exists a set $\Omega \in 2^{\Delta_V^{\mathcal{I}}}$ such that $v^{\mathcal{I}} \in \Omega$ and $\langle [e]^{\sim^{\mathcal{I}}}, \Omega \rangle \in A^{\mathcal{I}}$ (remember also that $v^{\mathcal{I}} = v$).

We are now ready to redefine the notion of assignment.

Definition 8.12 (assignment (revisited)). Let $\phi(\vec{x})$ be a conjunction of atoms of the form $C(t)$, $R(t, t')$, $A(t, t_1)$, $F(t_1, \dots, t_n)$, $y \approx_m v$, $v \approx_m y$, or $y \approx_m z$, where C , R , A , F are an atomic concept, an atomic role, an attribute, or a n -ary built-in predicate, respectively, t, t' are entity-variables in \vec{x} or entities, t_1, \dots, t_n are value-variables in \vec{x} or values, y, z are value-variables, and v is a value. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \sim^{\mathcal{I}})$ be an interpretation. An *assignment from $\phi(\vec{x})$ to \mathcal{I}* is a mapping μ from the variables \vec{x} to the set $\Delta_E^{\mathcal{I}} / \sim^{\mathcal{I}} \cup 2^{\Delta_V^{\mathcal{I}}}$, such that

⁶Note that we do not allow for matching atoms occurring in the right-hand side of tgds.

⁷Given a set Γ , with 2^Γ we denote the power set of Γ

1. for each atom in $\phi(\vec{x})$ of the form $S(x_1, \dots, x_k, c_1, \dots, c_\ell)$, where x_1, \dots, x_k are variables, and, for each $1 \leq i \leq \ell$, c_i is either an entity in \mathbf{N}_E or a value in \mathbf{N}_V ⁸, the tuple $\langle \mu(x_1), \dots, \mu(x_k), c_1^{\mathcal{I}}, \dots, c_\ell^{\mathcal{I}} \rangle \in S^{\mathcal{I}}$; ⁹
2. for each atom in $\phi(\vec{x})$ of the form $y \approx_m v$ or $v \approx_m y$, $v^{\mathcal{I}} \in \mu(y)$
3. for each atom in $\phi(\vec{x})$ of the form $y \approx_m z$, $\mu(y_1) \cap \mu(y_2) \neq \emptyset$

As in Definition 8.2, let $\vec{x} = x_1, \dots, x_m$, the sequence $\mu(x_1), \dots, \mu(x_m)$ is called the *image of μ* .

The notion of *extension* of an assignment is as defined in Section 8.2.4.

We notice that point 2 and point 3 above formalize the approximate forms of selections and joins mentioned before, in the spirit of the relaxed relation algebra considered in [23].

Given a tgd or an entity resolution rule ρ (possibly containing matching atoms in its body) and an interpretation \mathcal{I} of the form of Definition 8.11, the satisfaction of ρ by \mathcal{I} is as defined in Section 8.2.4, by using an assignment as defined in Definition 8.12. It remains to provide the semantics of functionalities. We say that \mathcal{I} satisfies a functionality assertion of the form (8.4) if for each assignment μ from $A(x, y) \wedge A(x, z)$ to \mathcal{I} , $\mu(y) = \mu(z)$ (notice that this requires that the two sets assigned to y and z have to be equal). We remark that the above definition coincides with the standard one for functionality assertions, modulo the use of our tailored definitions of interpretation and assignment. Furthermore, such definition allows us to treat functionalities as matching dependencies, for which we adopt a merge function returning the union of the values associated to the same entity (or set of equivalent entities) by a functional attribute (cf. the union class of match and merge functions of [18]).

Example 8.5. Consider the KER system $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$, where \mathcal{T} contains only the functional dependency $age(x, y) \wedge age(x, z) \rightarrow y = z$, $\mathcal{A} = \{age(\#Ullman, 78), age(\#Ullman, 80)\}$, and $\mathcal{E} = \emptyset$. The interpretation \mathcal{I} such that $\langle [\#Ullman]^{\sim \mathcal{I}}, \{78, 80\} \rangle \in age^{\mathcal{I}}$ satisfies the functionality assertions in \mathcal{T} .

Obviously, a *model* for a KER system is an interpretation satisfying all tgds and functionalities in \mathcal{T} , all membership assertions in \mathcal{A} , and all entity resolution rules in \mathcal{E} . Then, the notion of entailment is the usual one.

From the practical perspective, our semantics allow us to obtain a model even for KER systems that would be considered inconsistent under standard first-order semantics. Our models do not clean the inconsistency, nor repair it (e.g., by considering maximal subsets of the ABox that are consistent with the TBox and the entity resolution rules, as in repair-based approaches).

⁸Without loss of generality we assume that entities and values, if any, occur as the last ℓ arguments of the atom with predicate S . Notice also that $k \geq 0$, $\ell \geq 0$, and $k + \ell \geq 1$.

⁹With a slight abuse of notation, if $c_i \in \mathbf{N}_V$, here we consider $c_i^{\mathcal{I}} = \{c_i\}$.

Rather, we gather together in a set all possible values that a functional attribute A associates to an entity e , and treat such a union as the only filler for e with respect to A .

8.6.2 Universal Models

The universal model of a KER system with a TBox expressed through tgds and functionalities is as established in Definition 8.4, provided that the following revised notion of homomorphism is adopted.

Definition 8.13 (homomorphism (revisited)). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be KER system, such that \mathcal{T} is a set of tgds of the form (8.1) and functionalities of the form (8.4). Let $\mathcal{I}_1 = \langle \Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1}, \sim^{\mathcal{I}_1} \rangle$ and $\mathcal{I}_2 = \langle \Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2}, \sim^{\mathcal{I}_2} \rangle$ be two interpretations. An homomorphism $h : \mathcal{I}_1 \rightarrow \mathcal{I}_2$ is defined as in Definition 8.3, except for the last condition on interpretation of attributes, which is replaced by the following one:

- if $\langle [e]^{\sim^{\mathcal{I}_1}}, \Omega \rangle \in A^{\mathcal{I}_1}$, then $\langle [h(e)]^{\sim^{\mathcal{I}_2}}, \Omega' \rangle \in A^{\mathcal{I}_2}$, such that $h(\Omega) \subseteq \Omega'$, where $h(\Omega) = \{h(v) \mid v \in \Omega\}$, for each attribute A in \mathcal{K} , each $[e]^{\sim^{\mathcal{I}_1}} \in \Delta_E^{\mathcal{I}_1} / \sim^{\mathcal{I}_1}$, and each $\Omega \in 2^{\Delta_V^{\mathcal{I}_1}}$.

8.6.3 Query answering in the presence of functional attributes

We consider conjunctive queries of the form (8.3), extended to the presence of matching atoms.¹⁰ The notion of evaluation of a CQ over an interpretation remains unchanged (cf. Section 8.4), modulo the use of the new notions of interpretation and assignment. However, we note that now a tuple in the evaluation of a query never contains single values, but may contain sets of values from the interpretation domain (of course, such sets may be singletons). That is, given a query q of arity n and an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}, \sim^{\mathcal{I}} \rangle$, $q^{\mathcal{I}}$ is a (possibly empty) set of tuples of the form $\langle T_1, \dots, T_n \rangle$, where each T_i is either a non-empty set of entities in $\Delta_E^{\mathcal{I}}$ or a non-empty set of values in $\Delta_V^{\mathcal{I}}$.

We point out that the property given in Proposition 8.1 is still valid, provided that we account for the above mentioned difference. This is formally stated in the proposition below, whose proof follows from the definition of query evaluation and from Definition 8.13 (homomorphism revised).

Proposition 8.5. *Let \mathcal{K} be a KER system, q a query over \mathcal{K} , $\mathcal{I}_1 = \langle \Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1}, \sim^{\mathcal{I}_1} \rangle$ and $\mathcal{I}_2 = \langle \Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2}, \sim^{\mathcal{I}_2} \rangle$ two models of \mathcal{K} , and let $h : \mathcal{I}_1 \rightarrow \mathcal{I}_2$ be an homomorphism from \mathcal{I}_1 to \mathcal{I}_2 .*

¹⁰Our approach applies straightforwardly to CQs in which sets of values from \mathbb{N}_V occur in exactly the same positions where single values from \mathbb{N}_V may occur (as, e.g., in the query $\exists x, y. \text{age}(x, \{78, 80\})$, which asks to verify the existence of an individual that is both 78 and 80 years old, or in the query $\exists x. \text{age}(x, y) \wedge y \approx_m \{78, 80\}$, asking for the existence of some one that is either 78 or 80, or both). This would require to slightly modify Definition 8.12. For the sake of presentation we do not consider this case (notice also that CQs containing sets of values can always be rewritten into equivalent unions of CQs with values only).

If $\langle T_1, \dots, T_n \rangle \in q^{\mathcal{I}_1}$, then $\langle X_1, \dots, X_n \rangle \in q^{\mathcal{I}_2}$, where for every $1 \leq i \leq n$: (i) $X_i = h(T_i) = [h(e)]^{\sim \mathcal{I}_2}$, if $T_i = [e]^{\sim \mathcal{I}_1}$, and (ii) $h(T_i) \subseteq X_i \in 2^{\Delta_V^{\mathcal{I}_2}}$, where $h(T_i) = \{h(v) \mid v \in T_i\}$, if $T_i \in 2^{\Delta_V^{\mathcal{I}_1}}$.

We then note that a *typed tuple (revisited)* is now a sequence $\langle T_1, \dots, T_n \rangle$, such that each T_i is either a non-empty set of entities in \mathbf{N}_E or a non-empty set of values in \mathbf{N}_V . The notions of equality between typed tuples, type, and informativeness of a typed tuple naturally apply to this revised notion (cf. Section 8.4). In particular, let $\vec{t} = \langle T_1, \dots, T_n \rangle$ and $\vec{t}' = \langle T'_1, \dots, T'_n \rangle$ be two typed tuples of the same type, we say that \vec{t}' is *equally or more informative* than \vec{t} , denoted with $\vec{t} \leq \vec{t}'$, if $T_i \subseteq T'_i$ for every $1 \leq i \leq n$. Similarly, we can revise the definition of certain answer as follows.

Definition 8.14 (certain answer (revisited)). Let \mathcal{K} be a KER system and q a query of arity n over \mathcal{K} . A typed tuple $\vec{t} = \langle T_1, \dots, T_n \rangle$ is a *certain answer* to q w.r.t. \mathcal{K} if for each $\mathcal{I} \in \text{Mod}(\mathcal{K})$ there exists a tuple $\langle T'_1, \dots, T'_n \rangle \in q^{\mathcal{I}}$ such that:

(i) $T_i \subseteq T'_i$, for each $1 \leq i \leq n$;

(ii) there is no typed tuple \vec{t}' that satisfies (i) such that $\vec{t} < \vec{t}'$.

It is easy to see that Proposition 8.2 and Corollary 8.1 continue to hold.

In order to (re-)establish the relation between universal models and certain answers, we first need to extend the operator \downarrow in such a way that it applies also to sets of values, from which it eliminates value-nulls they may contain, exactly as it does for sets of entities. Then, given a set Θ of tuples of the form $\langle T_1, \dots, T_n \rangle$, where each T_i is either a set of entities and entity-nulls, or a set of values or value-nulls, $\Theta \downarrow$ is the set obtained from Θ by removing all tuples in Θ containing a T_i such that $T_i \downarrow = \emptyset$, and replacing each other tuple $\langle T_1, \dots, T_n \rangle$ in Θ by the tuple $\langle T_1, \dots, T_n \rangle \downarrow$.

We notice that all the revisited definitions we have given above are indeed smoother than the analogous ones given in Section 8.4, since now both the value and the entity components of a typed tuple are sets, which allows us to provide a more uniform treatment.

We conclude this section by remarking that Theorem 8.1 continues to hold also under the revisited notions of universal model, query evaluation, certain answer, and \downarrow operator discussed so far. The proof is essentially as the one we have given in Section 8.4, where however values have to be substituted by sets of values. For instance, in Part I of the proof, x_2 has to be equal to $\{v_1, \dots, v_k\} \downarrow$ such that $([a_1]^{\sim \mathcal{U}}, \{v_1, \dots, v_k\}, [a_3]^{\sim \mathcal{U}}) \in q^{\mathcal{U}}$ and $([a_1]^{\sim \mathcal{I}}, \Omega, [a_3]^{\sim \mathcal{I}}) \in q^{\mathcal{I}}$, where $\{v_1, \dots, v_k\} \subseteq \Omega$ (then satisfaction of properties (i) and (ii) of Definition 8.14 is proved exactly as done in Theorem 8.1 for properties (i) and (iii) of Definition 8.6).

8.6.4 Revisiting the Chase

We now revise the definition of chase step. The revision accounts for the presence of functionality assertions in the TBox of a KER system, which requires to slightly modify the *tg*d chase step and to introduce the additional functionality assertion step (whereas the entity resolution step remains the same as in Definition 8.8). As for the *tg*d step, the only change we need is in the extension of μ to μ' . Indeed, in this case μ' assigns existential value-variables in the head of a *tg*d with a singleton containing a fresh value-null.

We recall that, in the chase step definition, $\Delta_{E_{\perp}}$ and $\Delta_{V_{\perp}}$ are infinite ordered sets of labeled entity-nulls and value-nulls, respectively, such that $\Delta_{E_{\perp}} \cap \mathbf{N}_{\mathbf{E}} = \emptyset$, $\Delta_{V_{\perp}} \cap \mathbf{N}_{\mathbf{V}} = \emptyset$, and $\Delta_{E_{\perp}} \cap \Delta_{V_{\perp}} = \emptyset$. We also recall that we have defined an order on assignments to an interpretation \mathcal{I} whose domain $\Delta^{\mathcal{I}} = \Delta_E^{\mathcal{I}} \cup \Delta_V^{\mathcal{I}}$ is such that $\Delta_E^{\mathcal{I}} \setminus \mathbf{N}_{\mathbf{E}} \subseteq \Delta_{E_{\perp}}$ and $\Delta_V^{\mathcal{I}} \setminus \mathbf{N}_{\mathbf{V}} \subseteq \Delta_{V_{\perp}}$. Such order also holds on assignments as given in Definition 8.12, provided that every set Ω of values and value-nulls occurring in the image of an assignment μ is represented in the bindings obtained through μ as the string constructed by concatenating all the elements occurring in Ω , following the alphabetic order (as done for equivalence classes). Obviously, also assignments from the conjunction in the body of functionality assertions of the form (8.4) to \mathcal{I} are ordered as above.

We are now ready to give the revised definition of chase step.

Definition 8.15 (Chase Step (revisited)). Let $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1}, \sim^{\mathcal{I}_1})$, where $\Delta^{\mathcal{I}_1} = \Delta_E^{\mathcal{I}_1} \cup \Delta_V^{\mathcal{I}_1}$ and $\Delta_E^{\mathcal{I}_1} \cap \Delta_V^{\mathcal{I}_1} = \emptyset$, be an interpretation such that $\Delta_E^{\mathcal{I}_1} \setminus \mathbf{N}_{\mathbf{E}} \subseteq \Delta_{E_{\perp}}$, and $\Delta_V^{\mathcal{I}_1} \setminus \mathbf{N}_{\mathbf{V}} \subseteq \Delta_{V_{\perp}}$.

- (tuple generating dependency) this step is as in Definition 8.8, except for the extension of μ to μ' , which for the case where the variable y_i in \vec{y} is a value-variable (first bullet, point (ii)), is now as follows: [the variable y_i is assigned to] (ii) a set containing only a fresh labeled value-null from $\Delta_{V_{\perp}}$ that follows in the order all the entity-nulls and value-nulls in $\Delta^{\mathcal{I}_1}$;
- (entity resolution rule) this step is as in Definition 8.8;
- (functionality assertion) let d be a functionality assertion of the form (8.4). Let μ be an assignment from $A(x, y) \wedge A(x, z)$ to \mathcal{I}_1 such that $\mu(y) \neq \mu(z)$ (i.e., $\mu(y)$ and $\mu(z)$ are different sets of values from $\Delta_V^{\mathcal{I}_1}$). We say that d is applicable to \mathcal{I}_1 with assignment μ (we also say that μ triggers d in \mathcal{I}_1). Let \mathcal{I}_2 be the triple $(\Delta^{\mathcal{I}_2}, \cdot^{\mathcal{I}_2}, \sim^{\mathcal{I}_2})$, defined as follows:

- $\Delta^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_2} \cup \Delta_V^{\mathcal{I}_2}$, where $\Delta_E^{\mathcal{I}_2} = \Delta_E^{\mathcal{I}_1}$ and $\Delta_V^{\mathcal{I}_2} = \Delta_V^{\mathcal{I}_1}$;
- $\sim^{\mathcal{I}_2} = \sim^{\mathcal{I}_1}$;
- $\cdot^{\mathcal{I}_2}$ is as $\cdot^{\mathcal{I}_1}$ except for the interpretation of A , for which we have $A^{\mathcal{I}_2} = A^{\mathcal{I}_1} \setminus \{\langle \mu(x), \mu(y) \rangle, \langle \mu(x), \mu(z) \rangle\} \cup \{\langle \mu(x), \mu(y) \cup \mu(z) \rangle\}$.

Let d be a tgd, an entity resolution rule or a functionality assertion that can be applied to \mathcal{I}_1 with assignment μ , we say that \mathcal{I}_2 is the result of applying d to \mathcal{I}_1 with μ and we write $\mathcal{I}_1 \xrightarrow{d, \mu} \mathcal{I}_2$. We call $\mathcal{I}_1 \xrightarrow{d, \mu} \mathcal{I}_2$ a chase step.

It is easy to see that *the structure \mathcal{I}_2 that is the result of applying d to \mathcal{I}_1 is an interpretation* according to Definition 8.11. The proof is analogous to the proof of Proposition 8.3. In particular, for the case in which d is a functionality assertion, $\Delta^{\mathcal{I}_2} = \Delta^{\mathcal{I}_1}$, $\sim^{\mathcal{I}_2} = \sim^{\mathcal{I}_1}$, and the fact that \mathcal{I}_2 is an interpretation function can be readily seen by the definition of functionality assertion chase step. Then \mathcal{I}_2 is an interpretation since \mathcal{I}_1 is an interpretation.

A (fair) *chase sequence* is defined as in Definition 8.9. Furthermore, Proposition 8.4 (showing that the fair chase sequence is well defined) continues to hold even in the case in which the TBox of the KER system contains also functionality assertions.

We remark also that, given a KER system \mathcal{K} , and let \mathcal{I}_i and \mathcal{I}_j two elements in the chase sequence $\sigma_{\mathcal{C}}(\mathcal{K})$ such that $j > i$, we have that $\Delta^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_j}$, $\sim^{\mathcal{I}_i} \subseteq \sim^{\mathcal{I}_j}$, which implies that $[e]^{\sim^{\mathcal{I}_i}} \subseteq [e]^{\sim^{\mathcal{I}_j}}$ for each entity or entity-null $e \in \Delta_E^{\mathcal{I}_i}$, and also that for each attribute A and each pair $\langle [e]^{\sim^{\mathcal{I}_i}}, \Omega \rangle \in A^{\mathcal{I}_i}$, there exists a pair $\langle [e]^{\sim^{\mathcal{I}_j}}, \Omega' \rangle \in A^{\mathcal{I}_j}$ such that $\Omega \subseteq \Omega'$.

We are finally able to provide a revised definition for the Chase.

Definition 8.16 (Chase (revisited)). Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}, \mathcal{E} \rangle$ be a KER system, and let $\sigma_{\mathcal{C}}(\mathcal{K}) = \mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2 \dots$ be the chase sequence for \mathcal{K} . The *chase of \mathcal{K}* , denoted $chase(\mathcal{K})$, is the triple $\langle \Delta^{c^{\mathcal{K}}}, c^{\mathcal{K}}, \sim^{c^{\mathcal{K}}} \rangle$ defined as in Definition 8.10, except for the interpretation of attributes, which is as follows:

- $A^{c^{\mathcal{K}}} = \Theta_A \setminus \Theta_A^-$, for each attribute $A \in \mathbf{N}_A$, where
 - $\Theta_A = \{ \langle [e]^{\sim^{c^{\mathcal{K}}}}, \Omega \rangle \mid \text{there exists } \mathcal{I}_i \in \sigma_{\mathcal{C}}(\mathcal{K}) \text{ such that } \langle [e]^{\sim^{\mathcal{I}_i}}, \Omega \rangle \in A^{\mathcal{I}_i} \}$, and
 - $\Theta_A^- = \{ \langle [e]^{\sim^{c^{\mathcal{K}}}}, \Omega \rangle \mid \text{there exist } \mathcal{I}_i, \mathcal{I}_j \in \sigma_{\mathcal{C}}(\mathcal{K}), \text{ where } j > i, \text{ such that } \langle [e]^{\sim^{\mathcal{I}_i}}, \Omega \rangle \in A^{\mathcal{I}_i} \text{ and } \langle [e]^{\sim^{\mathcal{I}_j}}, \Omega' \rangle \in A^{\mathcal{I}_j} \text{ and } \Omega \subset \Omega' \}$.

Intuitively, among all pairs $\langle [e]^{\sim^{\mathcal{I}_i}}, \Omega \rangle$ in the interpretation of A with respect to any \mathcal{I}_i occurring in $\sigma_{\mathcal{C}}(\mathcal{K})$, $chase(\mathcal{K})$ takes only the one where Ω is the maximal with respect to set inclusion.

Given a KER system \mathcal{K} , whose TBox contains both tgds and functionality assertions, it is possible to show that $chase(\mathcal{K})$ is a universal model of \mathcal{K} , following the line of the proof of Theorem 8.2. It is indeed easy to see that $chase(\mathcal{K})$ is an interpretation according to Definition 8.11 (the proof essentially coincides with the proof of Lemma 8.1), and that such interpretation is a model. The proof for this last property is the same used for Lemma 8.2, with the addition of a case showing that the chase satisfies all functionality assertions in the TBox, which can be proved by contradiction, as we have done for satisfaction of tgds and

entity resolution rules. It is then possible to show that Lemma 8.3 holds also in the presence of functionalities in the TBox. From the above properties it immediately follows that $chase(\mathcal{K})$ is a universal model, as shown in the proof of Theorem 8.2. It is finally worth noting that also Theorem 8.3 is still valid in the general framework allowing for functional dependencies. Thus, also in this case, termination of the chase implies that conjunctive query answering is polynomial in data complexity.

8.7 Final Remarks

In this chapter we have dealt with entity resolution over DL ontologies. We have provided a general theoretical framework for ontologies expressed as sets of tgds and functionalities on attributes, coupled with entity resolution rules, which are expressed as equality generating dependencies (possibly using built-in predicates in their body). We have thus defined KER systems, composed by a TBox, an ABox, and a set of entity resolution rules.

In our framework, we have revised the traditional way of interpreting rules in first-order logic in two different ways. First, we have based the semantics of KER systems on interpretations depending on an equivalence relation used to group entities in equivalence classes (according to the conditions specified by entity resolution rules). Second, we have interpreted functionalities on attributes as matching dependencies specifying conditions under which values have to be merged, which in our framework is done through a union merge function.

We have shown that universal models of a KER system can be used for answering conjunctive queries (similar to what happens in data exchange, Datalog+/- programs, existential rule systems, or *DL-Lite* ontologies), and have provided a chase-based procedure to construct one such model. Whereas in general the chase is infinite, for the cases in which it terminates we have shown that it can be constructed in polynomial time in the size of the ABox.

The material presented in this chapter reports on our initial investigation on entity resolution, and our main contribution is the framework, with its nice semantic properties. We believe that it allows to smoothly combine powerful entity resolution rules with classical ontology languages for data management. Our computational results are still preliminary, and concern with the “simple” cases in which the chase terminates. An interesting line of research is to investigate query answering for settings in which termination of the chase is not always guaranteed, as for all DLs of the *DL-Lite* family. Another crucial point is how to deal with negative inclusions typically allowed in ontology languages: resolving entities in the presence of negative inclusions may indeed introduce forms of inconsistencies that cannot be dealt with as we have done for inconsistency caused by functionalities. A possible solution could be to interpret entity resolution rules as soft rules, so that the merge of equivalence classes they imply is done only if this does not lead to

violations of negative inclusions. This problem, however, deserves additional investigation.

Chapter 9

Related Work

In this section we discuss relevant research literature linked to the themes and topics treated in of this thesis. This chapter is organized as follows:

- In Section 9.1 we discuss the work related to OBDA i.e. the semantic data integration approach introduced in Chapter 5, which inspired the definition of OBDS systems.
- In Section 9.2 we review some studies on rule-based information extraction that share with OBDS the declarative nature of the approach.
- In Section 9.3 we specifically deal with work on ontology-based information extraction, mentioning both statistical and rule-based methods.
- In Section 9.4 we focus on the literature on entity resolution, related to the KER system formalization proposed in Chapter 8.

9.1 OBDA

As stated previously, the OBDS approach can also be seen as a form of OBDA over unstructured data sources. The main differences between the two frameworks is that in OBDS the link between ontology and data is no longer expressed through mappings but extraction assertions.

OBDA takes as a starting point the extensive work done in the field of Information Integration. Particularly relevant is the work on virtual data integration [147, 77], which aims to provide centralized access to data via a global schema (typically relational) connected through mappings to a set of possibly heterogeneous data sources. In this context various techniques for query answering, i.e., processing queries posed over the global schema, have been proposed, mainly based on query rewriting [1, 126, 179, 150, 83], possibly in presence of integrity constraints specified over the global schema [32, 40].

Differently from virtual data integration, where data are not moved from the sources, Data Exchange [90, 5] studies the problem of materializing an instance of the global schema (in this context called target schema). Here, query answering is typically solved through completion of the target instance based on the classical chase procedure [116, 90, 34, 16], which adds to the original instance new facts implied by the source-to-target mappings, expressed as tuple-generating dependences (tgds) and essentially corresponding to so-called GLAV mappings studied in virtual data integration. Also in the data exchange context integrity constraints are considered on the target schema, expressed as tgds and equality-generating dependencies (egds).

The more recent OBDA approach enriches the traditional virtual data integration framework by substituting the “classical” global schema with an ontology [177], thus making it possible to have a rich conceptual representation of the domain of interest, which allows a system designer to effectively abstract away from the logical and physical aspects of the source databases. As mentioned above, in this context query answering is still the main reasoning service, faced via query rewriting [177, 58, 37, 206, 174], or via chase-based algorithms [35], or even through combined approaches [128, 165].

For the above reason, and starting from the seminal works in [151, 49, 111, 45], the interest of the knowledge representation community for the problem of query answering over ontologies has grown considerably. This problem is typically referred to as Ontology-Mediated Query Answering (OMQA)[168, 169], to properly distinguish it from the OBDA framework, characterized by the presence of mappings towards data sources. In OMQA, instead, the focus is on a single ontology, and query answering amounts to reason over the ontology to return complete answers. In OMQA, various combinations of ontology and query languages have been analyzed in order to find the best setting that guarantees data tractability [84, 169]. For data-intensive OMQA applications, the tractable settings mainly studied are those in which the query language is that of conjunctive queries, and the ontology is expressed in one of the logics of the *DL-Lite* family [45] or of the \mathcal{EL} family [7], or in one of the more expressive Horn Description Logics (DLs) [84], a fragment of Horn first-order logic. We also notice that the distinction between OBDA and OMQA is a bit blurry in the literature, and some of the papers on OBDA we have mentioned do not really consider mappings. In other terms, in such papers OBDA acts as a motivation to study query answering, and, at the same time, query answering techniques they provide can be easily extended to OBDA when mappings are GAV (i.e., roughly, are forms of full tgds). For more detail on OMQA we refer the reader to the survey [27].

Other valuable reasoning services proposed on OBDA systems in the presence of relational data sources are those related to the exposure of data services [60], for open-data publishing [59] and for data quality purposes [64].

The success of OBDA is not only limited to theoretical research, but is also conveyed in a

series of enterprise tools, such as MASTRO [42], Ontop [41], Stardog¹ and Ultrawap [196], largely used for industrial applications [75, 122, 187].

However, the OBDA studies proposed in the literature so far have mainly focused on relational data sources, with few exceptions, like [29], [54], and [142] (the last paper presents the framework we have discussed in Chapter 6). In [29] the authors propose an OBDA approach over JSON documents managed by MongoDB, a document-based DBMS and currently one of the most popular NoSQL DBMSs. In that paper, the authors mainly focus on query answering via a virtual approach, proposing algorithms for query rewriting into queries processable by MongoDB. The proposed solutions take into account the features in MongoDB and have been tested in practice through an ad-hoc implementation of Ontop. In [54], the specific case of tabular datasets represented as CSV files has been considered, and a framework, called Morph-CSV, for querying tabular data has been proposed. The ultimate goal of Morph-CSV is to exploit information from OBDA inputs (e.g., mappings, queries) to enforce constraints that can be used together with any SPARQL-to-SQL OBDA engine in order to increase query answering performances (with respect to an approach that simply stores tables in a relational DBMS without integrity constraints).

Except for the mentioned papers, OBDA in the presence of non-relational databases has not been so far the subject of specific investigations, but it is typically faced through wrapping [65, 12, 185], which allows to transform some non-relational data formats into the relational one [77]. However, wrappings introduce a further level of data virtualization, which can drastically downgrade performances and, in general, cannot manage unstructured sources (such as raw text).

9.2 Declarative Information Extraction

As stated previously, OBDS systems express in a declarative way the link between the portion of the text identified by AQL extractors and the predicates of the ontology.

This section discusses the most successful IE approaches in the literature, which share with the OBDS framework the characteristics of being declarative and formally well-defined from a theoretical point of view.

Rule-based declarative approaches for IE have been studied in the last two decades [207, 80, 154, 12]. Initially, the approaches were still presented some procedural flavor and generally based on Common Specification Pattern Language, a rule language on which JAPE [71] and TOKENSREGEX [156] are also based on. Unlike Document Spanners, these languages do not have formal declarative semantics underlying them [86]. After realizing the difficulty of maintaining

¹<https://www.stardog.com>

applications based on rules with an operational semantics, the IE community moved to approaches inspired by declarative languages based on database principles [114, 124]. For a foundational and theoretical study on declarative IE over free text, we refer to the work on Document Spanners [86]. In the context of Document Spanners, several extensions have also been studied that are capable of dealing with inconsistencies [87], with recursion [175], to annotate documents in CSV format. Some studies on Document Spanners have also analyzed the computational complexity [94, 3, 97] and logic underlying them [96].

Lixto In addition to IE over raw text carried out by systems like SYSTEMT [57], which has already been presented in Chapter 7, one of the most successful applications of declarative IE resides in the extraction of information from the web pages [127]. Currently, most of the information on the web is available in the form of HTML documents. Due to its structure, this type of document is not suitable for automatic processing and database-like queries. This problem has been addressed by a large bulk of work on so-called Web wrappers [154, 12], rule-based programs that extract the relevant information from HTML documents and translate it into a more machine-friendly format, such as XML, which can be easily queried and further processed. *Lixto* proposed in [12] by Baumgartner et.al, is a system for generating HTML wrappers through a language called ELog (“**E**xtraction by **data**log”). Elog has a solid and well-understood theoretical basis. Properly, it is syntactically and semantically defined by its underlying logic, which corresponds to monadic datalog, a function-free datalog fragment [102, 103].

Theoretical studies have shown that Elog has some interesting computational properties. Namely, it is proved that the evaluation of monadic datalog programs on tree structures (e.g., an HTML document) can be done in linear time in data complexity, making then Elog suitable for data-intensive tasks. Within *Lixto* there is also the possibility of using some dialects less expressive than Elog. These sublanguages allow to express wrappers through visual iterations, thus simplifying their creation. The main drawback of these wrapper-based information extraction approaches is that, although they are particularly effective in the presence of well-structured HTML documents, they do not take into account the information present in free text, differently from the OBDS approach presented in this dissertation.

XLog Another declarative IE language based on datalog is XLog, proposed in [198]. The syntax and semantics of this language match those of datalog with controlled use of built-in predicates and functions. In their paper, the authors show that the main advantages of XLog lie in the evaluation performance, also by virtue of the several optimizations proposed to reduce the execution time. Among these, techniques based on the order of execution of the rules using a suited query plan are particularly interesting.

9.3 Ontology-Based Information Extraction

Ontology-Based Information Extraction (OBIE) systems can be classified into two broad sets, those that are tasked with populating a given ontology, such as the methods proposed in this thesis, and those that also attempt to extract the intensional layer of an ontology along with the extensional one [186, 214, 178, 199]. In addition, depending on the approach chosen, we can identify OBIE systems based on rules [141], statistical methods [51], or combined techniques [199]. In the following, we report on some of the best-known systems in the literature, also resulting in industrial products, and that make use of some of the technologies used in this thesis, shown in Chapters 3 and Chapter 4. We point out however that all proposed solutions in OBIE simply look at the ontology as a conceptual model, and do not exploit reasoning services to support the IE process, as we aim to do in this thesis.

Text2Onto In [61], Cimiano et. al propose a framework to extract ontologies from text, relying on both statistical and rule-based algorithms. Text2Onto is based on GATE and uses its main components, including JAPE, to generate from the text both the intentional and extensional levels of the ontology. It extracts concepts, subclass relations, mereological (part-of) relations, general relations, attributes, and instances of these, based on a probabilistic structure. The system currently seems to be discontinued, as it has not been updated since 2008. In the literature, similar approaches are those proposed by Velardi et al. with Ontoloeam [205], which, however, has the main aim of creating taxonomies from free text with the purpose of disambiguation. This system shares with the pipeline proposed in Chapter 3 the technologies used, i.e., GATE, but also presents the issues we have previously mentioned about GATE and the JAPE language.

NELL Tom Mitchel et al. proposed in [51] an architecture inspired by human learning, based on never-stop learning techniques for populating and enriching an initial general-purpose ontology. This architecture, namely, Never-Ending Language Learner (NELL), learns new facts, new concepts, and new relationships from the web through text mining and rule-based deductive knowledge completion steps, properly using inductive-learned and hand-written horn rules. NELL is currently supported and has been active since 2010 [24]. The current output of the system consists of an immense well-curated knowledge graph created semi-automatically and refined through human-in-the-loop techniques. In fact, before inserting a fact into the knowledge graph, the system requires that there is another trustworthy step, typically endorsed by external human intervention. Concerning the work proposed in this thesis, although we do not address the approach through learning techniques, the Horn rules defined for NELL can be seen as TBox assertions. We note that the context of NELL, similar to that of a large RDFS graph, does not provide incomplete information, unlike the notions studied in this dissertation that go over this

limitation.

DeepDive Christopher Re et al. [199] proposed a system called DeepDive that employs popular techniques, such as distant supervision and Markov logic language, in order to populate a Knowledge Base. DeepDive, whose foundations are those of Stanford’s CoreNLP, also proposes a declarative datalog-like language to manage the complexity of extractions. Unlike the previous systems, it allows for integrating knowledge bases present on the web, such as YAGO, DBpedia, Freebase, and Wikidata, through entity linking mechanisms. DeepDive first converts several input data (e.g., raw corpora and ontologies) into relational features, to populate the knowledge base using standard NLP tools and custom code. These features are then used to train statistical models representing the correlations between linguistic patterns and target relations. Finally, DeepDive combines the trained statistical models with additional knowledge (e.g., domain knowledge) into a Markov logic program, which is then used to transform the relational features (e.g., candidate entity mentions and linguistic patterns) into a knowledge base with entities, relationships, and their provenance. Related to work, we may say that the role of reasoning and how entity resolution is dealt with in DeepDive are not well identified.

9.4 Entity Resolution

Entity Resolution (ER) [85] is a task that has been attracting growing attention to address the influx of structured and semi-structured data from a multitude of heterogeneous sources. As stated previously, Entity Resolution is the task of identifying different entities that describe the same real-world object [129]. It is a core task for Data Integration [85], applying to any kind of data, from the structured entities [91] of relational databases [129, 20], to the semistructured entities in Semantic Web [166, 201], and the unstructured entities that are automatically extracted from free text [142]. Generally, ER consists of two parts: *(i)* the candidate selection step, which determines the entities worth comparing, and *(ii)* the candidate matching step, or simply Matching, which compares the selected entities to determine whether they represent the same real-world object. There are two main ways to define the candidate selection step. The *schema-aware* method needs to use a-priori schema information and selects some specific attribute values or a combination of these attribute values as parameters for the matching [121]. In this context, the selected attributes are usually considered suitable for matching because they are discriminative or contain less noisy data. Otherwise, *schema-agnostic* methods are used to process data in order to resolve the ER task without considering schema information [28]. It is possible to classify further ER methodologies based on their algorithmic foundations, which can be learning-based, both supervised [183, 120, 180] and unsupervised [118], or rule-based [121]. Finally, we want to remark that the ER community, in the last decades, focused its attention also

on the problem of reducing the number of comparisons between records of data that are needed to resolve entities, which is an extremely crucial task in the presence of a massive amount of data. This task is mainly faced using Blocking techniques [158, 208] or through identifying special settings for the ER tools [19, 17]. In the following, we focus on the schema-aware rule-based ER methods characterized by a solid theoretical background, as they represent the work closest to the **KER** framework we propose in Chapter 8.

Swoosh In [19] Benjelloun et al. introduced Swoosh, a generic conceptual framework for entity resolution. The article defines the notions of matching function and merging function. Intuitively, the former is used to identify database tuples that refer to the same entity, and is usually built through similarity functions that perform a careful comparison of the attributes of the table containing the tuples to be matched. The merging function, on the other hand, indicates what to do when a match happens. For example, it is possible to combine somehow the values appearing in the match, or select only one of them, based on a sort of more informativeness principle. Relying upon the so-called ICAR properties, Benjelloun et al. show that, if such properties are respected by the match and merge functions, the ER process leads to higher efficiency. Differently from **KER** systems, Swoosh considers the setting of a single relational table, and therefore does not deal with a rich data schema, as the one represented through an ontology TBox. Moreover, Swoosh works at the record (tuple) level, that is, it merges pairs of tuples of values, and does not consider entities explicitly, whereas **KER** systems work with entities, their relationships, and their attributes. A consequence of this is that the merging in Swoosh remains local to a tuple, even because there is no formal link between different tuples, whereas in **KER** systems every merging of equivalence classes of entities is done globally in the entire model (cf. Definition 8.8, entity resolution rule step)² Note also that, at the end of the Swoosh process, some tuples may be discarded (those “dominated” by others in the instance, if this is applicable according to the merging function adopted), which is something that does not happen in the never-failing chase procedure defined for **KER** systems. It is however worth noticing that Swoosh considers general match and merge functions, whereas in **KER** systems we make a specific choice. In particular, match functions are realized through entity resolution rules and functionalities on attributes, whereas the merge is realized through union (for both entities and values).

Matching Dependencies In [91], Wenfei Fan proposed a new class of dependencies, called Matching dependencies (MDs). Formally studied in [93], MDs are a form of semantic constraints for data cleaning, useful to increase data quality over databases. More in-depth, MDs specify that a pair of attribute values in two database tuples are to be matched, i.e., made equal, if similarities hold between other pairs of values in the same tuples. The framework leaves the

²Note instead that, in **KER** systems, the merging of values enforced by functionalities on attributes is local to the entity which the attribute refers to.

implementation details of the data cleaning process with MDs completely unspecified, and implicitly demands them to the application on hand. In [20] Bertossi et al. propose an extension of the MDs framework, to overcome some of their limitations. In particular, in [20], matching functions inducing a lattice framework on the attribute domains are considered. Another interesting contribution given by that paper is the study of query answering in the presence of MDs, which was not addressed in the Fan's seminal paper. Compared to our *KER* systems, the work of Bertossi et al. focuses on database tuples and it does not distinguish semantically between entities and values. It also shares with Swoosh a local view of the data cleaning process, i.e., when two tuples have to be merged, the merging does not affect the whole database, unlike what we have proposed in this thesis. It is also worth noting that the semantics of matching atoms, we propose in Section 8.6, allows us to define an approximated form of selection, which is also enabled in the relaxed relational algebra proposed in [20].

Chapter 10

Conclusion

This chapter concludes the thesis with a brief discussion and possible directions for future work.

10.1 Discussion

One of the main contributions of this thesis is the formal study of rule-based information extraction mediated by an ontology. To the best of our knowledge, this is the first formal comprehensive study that semantically connect ontology languages with information extraction rules, so that ontologies can be exploited as means for reasoning over the knowledge representation they provide, and not simply treated as static conceptual models.

Motivated by the issues encountered in our initial implementation experiences based on the two most popular open-source information extraction technologies, GATE and CoreNLP, we have introduced the theoretical framework of Ontology-Based Document Spanning (OBDS) systems. In designing this framework we leveraged the recent formal study on document spanners for information extraction and the well-known properties of Ontology-Based Data Access (OBDA) systems. In this respect, our framework can be seen as an extension of OBDA to access unstructured data, as those contained in text documents.

Within our framework, we have conducted an accurate analysis of conjunctive query answering over OBDS systems characterized by ontologies specified in the Description Logics of the *DL-Lite* family, and extraction rules equipped expressive spanners (i.e., belonging to the class $\llbracket \text{RGX}^{\{\cup, \pi, \bowtie, \zeta =\}} \rrbracket$) of spanners, which are able to capture the core of AQL, the extraction language used by IBM SYSTEMT). Interestingly, in the practical and frequent cases in which mappings are GAV (but also for GLAV mappings and *DL-Lite_R* ontologies), answering CQs in the above settings can be reduced to the evaluation of core spanners over the underlying documents, through a query rewriting style technique.

Through the development of MASTRO SYSTEM-T, we have then shown that the notions

introduced and the technical results presented in the context of OBDS systems are not only theoretically interesting but have also a concrete practical counterpart, which we extensively tested in challenging financial scenarios, like the EDGAR or the CONSOB ones.

We have also formally studied the problem of entity resolution, which is a major recurring issue in information extraction, as shown by our GATE experience in the OSINT context, and which is also crucial in data integration applications in general. Properly, we have introduced the notion of the KER system, i.e., an extension of the standard two-level representation provided by ontologies with a third component composed by a set of horn rules, useful to solve the problem of entity resolution. We have formally defined the semantics of KER systems characterized by ontologies given in terms of very general tuple-generating dependencies, studied their properties, and provided a novel chase-based technique for conjunctive query answering. Then, we have extended our results to ontologies allowing for functionalities on properties, and we have provided a novel semantics to interpret them as matching dependencies, thus suitably coping with possible inconsistencies due to violations of the functionality.

We point out that the thesis left open some interesting and challenging problems, which are detailed in the following list.

- From the theoretical perspective, it would be obviously interesting to close the case of $DL-Lite_{\mathcal{F}}$ OBDS systems with GLAV extraction assertions, by providing a tailored rewriting technique for this setting (as said in Chapter 6, we expect that query answering in this case can be reduced to the execution of a recursive spanner [202]).
- It would be interesting to enrich the expressiveness of spanners used in extraction assertions of OBDS systems, and study query answering in this case. Natural candidates for this extension are spanners allowing for recursion [202], and spanners able to manage incomplete information [157].
- In our studies, we reported results related to data complexity of query answering, as it is essential in data-intensive applications such as IE. To devise a complete picture of the complexity of the problem, it would be however interesting to find also the respective combined complexity results.
- We have not covered the case of KER systems with a TBox allowing for disjointness. A nice extension of our work would be to study how they act combined with entity resolution rules (see also discussion in Section 8.7).
- Query answering in KER systems with TBox expressed in $DL-Lite_{\mathcal{R}}$ or $DL-Lite_{\mathcal{F}}$ remains open. Besides the issue with disjointness, mentioned above, a crucial aspect is that for these languages the chase may not terminate. As said in Section 8.7, techniques based on

query rewriting or combined approaches, in the spirit of [155], are the best candidates to solve the problem in these settings.

- An implementation of the **KER** framework would provide insights into whether the theoretical results we achieved actually have an impact in practice as well.

10.2 Future works

In addition to tackling the open problems mentioned in the foregoing list, we see many other interesting avenues for future research, including:

- Studying OBDS systems where the TBox is expressed in other DLs for which standard query answering over ontologies is polynomial in data complexity, e.g. \mathcal{EL} [8], or *Horn* DLs [112, 167].
- Exploiting reasoning in the OBDS framework to identify anomalies in the specification of extraction rules (e.g., intensional inconsistencies), in the spirit of the work on mapping analysis in OBDA [137].
- Enriching extraction assertions in OBDS systems by allowing their left-hand side to contain both spanners and atoms referring to ontology predicates. Through this enrichment it should be possible to recall in an extraction assertions some extraction specifications already associated to an ontology predicate. We have the intuition that, under suitable conditions, we can reduce the enriched specification to a “standard” one, by compiling the TBox of the ontology into the extraction assertions through query rewriting mechanisms.
- Devising optimization techniques for OBDS systems, similar to OBDA [132] but tailored to ocument spanners.
- Examining cases where the language for expressing queries over OBDS systems goes beyond UCQs.
- Enriching the form of entity resolution rules in **KER** systems, by, e.g., considering negation, possibly interpreted under epistemic operators [44].
- Introducing blocking and filtering based optimization algorithms [171], very common in entity resolution, in the chase-based algorithm we proposed for query answering over **KER** systems.
- Developing an all-in one inclusive User Interface for **MASTRO SYSTEM-T**, offering also functionalities to build the AQL extractors directly inside the environment.

We believe that each of the above issues is an interesting research problem that deserves to be investigated.

Bibliography

- [1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In A. O. Mendelzon and J. Paredaens, editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, pages 254–263. ACM Press, 1998.
- [2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.
- [3] A. Amarilli, P. Bourhis, S. Mengel, and M. Niewerth. Constant-delay enumeration for nondeterministic document spanners. *ACM SIGMOD Record*, 49(1):25–32, 2020.
- [4] A. P. Apro시오 and G. Moretti. Italy goes to stanford: a collection of corenlp modules for italian. *arXiv preprint arXiv:1609.06204*, 2016.
- [5] M. Arenas, P. Barceló, L. Libkin, and F. Murlak. *Foundations of Data Exchange*. Cambridge University Press, 2014.
- [6] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, pages 68–79, 1999.
- [7] F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 364–369, 2005.
- [8] F. Baader, S. Brandt, and C. Lutz. Pushing the \mathcal{EL} envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 364–369, 2005.
- [9] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, 2007.

-
- [10] F. Baader, I. Horrocks, and U. Sattler. Description logics. In *Handbook on ontologies*, pages 3–28. Springer, 2004.
- [11] R. Baldoni and R. De Nicola. The white book on cyber-security, 2015.
- [12] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. 2001.
- [13] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, et al. Owl web ontology language reference. *W3C recommendation*, 10(02), 2004.
- [14] C. Beeri and M. Y. Vardi. The implication problem for data dependencies. In *Proc. of the 8th Coll. on Automata, Languages and Programming (ICALP)*, volume 115 of *Lecture Notes in Computer Science*, pages 73–85. Springer, 1981.
- [15] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *J. of the ACM*, 31(4):718–741, 1984.
- [16] M. Benedikt, G. Konstantinidis, G. Mecca, B. Motik, P. Papotti, D. Santoro, and E. Tsamoura. Benchmarking the chase. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 37–52, 2017.
- [17] O. Benjelloun, H. Garcia-Molina, H. Gong, H. Kawai, T. E. Larson, D. Menestrina, and S. Thavisomboon. D-swoosh: A family of algorithms for generic, distributed entity resolution. In *27th International Conference on Distributed Computing Systems (ICDCS'07)*, pages 37–37. IEEE, 2007.
- [18] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. 18(1):255–276, 2009.
- [19] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal*, 18(1):255–276, 2009.
- [20] L. Bertossi, S. Kolahi, and L. V. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. *Theory of Computing Systems*, 52(3):441–482, 2013.
- [21] L. E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

- [22] L. E. Bertossi. Database repairs and consistent query answering: Origins and further developments. pages 48–58, 2019.
- [23] L. E. Bertossi, S. Kolahi, and L. V. S. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. *Theoretical Comput. Sci.*, 52(3):441–482, 2013.
- [24] J. Betteridge, A. Carlson, S. A. Hong, E. R. Hruschka Jr, E. L. Law, T. M. Mitchell, and S. H. Wang. Toward never ending language learning. In *AAAI spring symposium: Learning by reading and learning to read*, pages 1–2, 2009.
- [25] M. Bienvenu and C. Bourgaux. Inconsistency-tolerant querying of description logic knowledge bases. In J. Z. Pan, D. Calvanese, T. Eiter, I. Horrocks, M. Kifer, F. Lin, and Y. Zhao, editors, *Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering - 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures*, volume 9885, pages 156–202, 2016.
- [26] M. Bienvenu and M. Ortiz. Ontology-mediated query answering with data-tractable description logics. In W. Faber and A. Paschke, editors, *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures*, volume 9203 of *Lecture Notes in Computer Science*, pages 218–307. Springer, 2015.
- [27] M. Bienvenu and M. Ortiz. Ontology-mediated query answering with data-tractable description logics. In *Reasoning Web International Summer School*, pages 218–307. Springer, 2015.
- [28] C. Böhm, G. De Melo, F. Naumann, and G. Weikum. Linda: distributed web-of-data-scale entity matching. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2104–2108, 2012.
- [29] E. Botoeva, D. Calvanese, B. Cogrel, J. Corman, and G. Xiao. Ontology-based data access—beyond relational sources. *Intelligenza Artificiale*, 13(1):21–36, 2019.
- [30] E. Botoeva, D. Calvanese, B. Cogrel, M. Rezk, and G. Xiao. OBDA beyond relational DBs: A study for MongoDB. In *Proc. of the 29th Int. Workshop on Description Logic (DL)*, 2016.
- [31] J. Broekstra and A. Kampman. Serql: An rdf query and transformation language draft. 2004.

- [32] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Inf. Syst.*, 29(2):147–163, 2004.
- [33] A. Cali, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. 48:115–174, 2013.
- [34] A. Cali, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.
- [35] A. Cali, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.
- [36] A. Cali, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. 14:57–83, 2012.
- [37] A. Cali, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *Logic in Computer Science (LICS), 2010 25th Annual IEEE Symposium on*, pages 228–242. IEEE, 2010.
- [38] A. Cali, G. Gottlob, and A. Pieris. New expressive languages for ontological query answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011.
- [39] A. Cali, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, pages 260–271, 2003.
- [40] A. Cali, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 16–21, 2003.
- [41] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering sparql queries over relational databases. *Semantic Web*, 8(3):471–487, 2017.
- [42] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The mastro system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.

- [43] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The Mastro system for ontology-based data access. *Semantic Web J.*, 2(1):43–53, 2011.
- [44] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Eql-lite: Effective first-order query processing in description logics. In *IJCAI*, volume 7, pages 274–279, 2007.
- [45] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.
- [46] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
- [47] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. *Artificial Intelligence*, 195:335–360, 2013.
- [48] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Ontology-based data access and integration. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems, Second Edition*. 2018.
- [49] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 149–158, 1998.
- [50] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Actions and programs over description logic ontologies. In *Proc. of the 20th Int. Workshop on Description Logic (DL)*, volume 250 of *CEUR Electronic Workshop Proceedings*, <http://ceur-ws.org/>, pages 29–40, 2007.
- [51] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3. Atlanta, 2010.
- [52] S. Castano, A. Ferrara, S. Montanelli, and D. Lorusso. Instance matching for ontology population. In *SEBD*, pages 121–132, 2008.
- [53] A. X. Chang and C. D. Manning. Tokensregex: Defining cascaded regular expressions over tokens. *Stanford University Computer Science Technical Reports. CSTR*, 2:2014, 2014.

- [54] D. Chaves-Fraga, E. Ruckhaus, F. Priyatna, M.-E. Vidal, and O. Corcho. Enhancing virtual ontology based access over tabular data with morph-csv. *Semantic Web*, (Preprint):1–34, 2021.
- [55] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, and S. Vaithyanathan. SystemT: an algebraic approach to declarative information extraction. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 128–137, 2010.
- [56] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan. Systemt: an algebraic approach to declarative information extraction. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 128–137. Association for Computational Linguistics, 2010.
- [57] L. Chiticariu, Y. Li, S. Raghavan, and F. R. Reiss. Enterprise information extraction: recent developments and open challenges. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1257–1258, 2010.
- [58] A. Chortaras, D. Trivela, and G. B. Stamou. Optimized query rewriting for owl 2 ql. In *CADE*, volume 11, pages 192–206. Springer, 2011.
- [59] G. Cima. Preliminary results on ontology-based open data publishing. *arXiv preprint arXiv:1705.10480*, 2017.
- [60] G. Cima, M. Lenzerini, and A. Poggi. Semantic characterization of data services through ontologies. In *IJCAI*, pages 1647–1653, 2019.
- [61] P. Cimiano and J. Völker. text2onto. In *International conference on application of natural language to information systems*, pages 227–238. Springer, 2005.
- [62] C. Civili, M. Console, G. De Giacomo, D. Lembo, M. Lenzerini, L. Lepore, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, et al. Mastro studio: managing ontology-based data access applications. *Proceedings of the VLDB Endowment*, 6(12):1314–1317, 2013.
- [63] M. Console and M. Lenzerini. Data quality in ontology-based data access: The case of consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.
- [64] M. Console and M. Lenzerini. Data quality in ontology-based data access: The case of consistency. In C. E. Brodley and P. Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1020–1026. AAAI Press, 2014.

- [65] V. Crescenzi, G. Mecca, P. Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118, 2001.
- [66] F. Croce, G. Cima, M. Lenzerini, and T. Catarci. Ontology-based explanation of classifiers. In *EDBT/ICDT Workshops*, 2020.
- [67] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *J. of Web Semantics*, 6(4):309–322, 2008.
- [68] H. Cunningham. GATE, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223—254, 2002.
- [69] H. Cunningham. *Developing Language Processing Components with GATE Version 8*. University of Sheffield Department of Computer Science, 2014.
- [70] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proc. of ACL'02*, 2002.
- [71] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damjanovic, et al. Developing language processing components with gate version 6 (a user guide). *University of Sheffield, Department of Computer Science*, 2011.
- [72] H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva. Getting more out of biomedical documents with gate’s full lifecycle open source text analytics. *PLoS Computational Biology*, 9(2).
- [73] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. W3C Recommendation, W3C, Sept. 2012. Available at <http://www.w3.org/TR/r2rml/>.
- [74] G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Using ontologies for semantic data integration. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years.*, pages 187–202. 2018.
- [75] G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, M. Ruzzi, and D. F. Savo. Mastro: A reasoner for effective ontology-based data access. In *Proc. of the OWL Reasoner Evaluation Workshop (ORE 2012)*, volume 858, 2012.
- [76] G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, M. Ruzzi, and D. F. Savo. MASTRO: A reasoner for effective Ontology-Based Data Access. In *Proc. of the 1st Int. Workshop on OWL Reasoner Evaluation (ORE)*, 2012.

- [77] A. Doan, A. Halevy, and Z. Ives. *Principles of data integration*. Elsevier, 2012.
- [78] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.
- [79] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th international conference on World Wide Web*, pages 662–673, 2002.
- [80] A. Doan, J. F. Naughton, R. Ramakrishnan, A. Baid, X. Chai, F. Chen, T. Chen, E. Chu, P. DeRose, B. Gao, et al. Information extraction challenges in managing unstructured data. *ACM SIGMOD Record*, 37(4):14–20, 2009.
- [81] J. Doleschal, N. Bratman, B. Kimelfeld, and W. Martens. The complexity of aggregates over extractions by regular expressions. *arXiv preprint arXiv:2002.08828*, 2020.
- [82] O. M. Duschka, M. R. Genesereth, and A. Y. Levy. Recursive query plans for data integration. *J. of Logic Programming*, 43(1):49–73, 2000.
- [83] O. M. Duschka, M. R. Genesereth, and A. Y. Levy. Recursive query plans for data integration. *The Journal of Logic Programming*, 43(1):49–73, 2000.
- [84] T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao. Query rewriting for horn-shiq plus rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.
- [85] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16, 2006.
- [86] R. Fagin, B. Kimelfeld, F. Reiss, and S. Vansummeren. Document spanners: A formal approach to information extraction. *Journal of the ACM*, 62(2):1–51, 2015.
- [87] R. Fagin, B. Kimelfeld, F. Reiss, and S. Vansummeren. Declarative cleaning of inconsistencies in information extraction. *ACM Trans. on Database Systems*, 41(1):6:1–6:44, 2016.
- [88] R. Fagin, P. G. Kolaitis, D. Lembo, L. Popa, and F. Scafoglieri. A Framework for Combining Entity Resolution and Query Answering in Knowledge Bases. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, pages 229–239, 8 2023.
- [89] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.

- [90] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, 2005.
- [91] W. Fan. Dependencies revisited for improving data quality. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 159–170, 2008.
- [92] W. Fan. Dependencies revisited for improving data quality. pages 159–170, 2008.
- [93] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *Proceedings of the VLDB Endowment*, 2(1):407–418, 2009.
- [94] F. Florenzano, C. Riveros, M. Ugarte, S. Vansummeren, and D. Vrgoc. Constant delay algorithms for regular document spanners. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 165–177, 2018.
- [95] D. Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2/3):169–202, 2000.
- [96] D. D. Freydenberger. A logic for document spanners. *Theory of Computing Systems*, 63(7):1679–1754, 2019.
- [97] D. D. Freydenberger and M. Holldack. Document spanners: From expressive power to decision problems. *Theory of Computing Systems*, 62(4):854–898, 2018.
- [98] G. Ganino, D. Lembo, M. Mecella, and F. Scafoglieri. Ontology population for open-source intelligence: A gate-based solution. *Softw. Pract. Exp.*, 48(12):2302–2330, 2018.
- [99] G. Ganino, D. Lembo, M. Mecella, and F. Scafoglieri. Ontology population for open-source intelligence (discussion paper). In S. Bergamaschi, T. D. Noia, and A. Maurino, editors, *Proceedings of the 26th Italian Symposium on Advanced Database Systems, Castellaneta Marina (Taranto), Italy, June 24-27, 2018*, volume 2161 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.
- [100] G. Ganino, D. Lembo, and F. Scafoglieri. Ontology population from raw text corpus for open-source intelligence. In I. Garrigós and M. Wimmer, editors, *Current Trends in Web Engineering - ICWE 2017 International Workshops, Liquid Multi-Device Software and EnWoT, practi-O-web, NLPIT, SoWeMine, Rome, Italy, June 5-8, 2017, Revised Selected Papers*, volume 10544 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2017.

- [101] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. Hermit: An owl 2 reasoner. *J. of Automated Reasoning*, 53(3):245–269, 2014.
- [102] G. Gottlob and C. Koch. Logic-based web information extraction. *ACM SIGMOD Record*, 33(2):87–94, 2004.
- [103] G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *Journal of the ACM (JACM)*, 51(1):74–113, 2004.
- [104] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. Owl 2: The next step for owl. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008.
- [105] R. Grishman and B. Sundheim. Message understanding conference- 6: A brief history. In *Proc. of the 16th Int. Conf. on Computational Linguistics (COLING)*, pages 466–471, 1996.
- [106] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993.
- [107] N. Guarino, D. Oberle, and S. Staab. What is an ontology? In *Handbook on Ontologies*, pages 1–17. Springer, 2009.
- [108] S. Harris and A. Seaborne. SPARQL 1.1 query language. W3C Recommendation, W3C, Mar. 2013. Available at <http://www.w3.org/TR/sparql11-query>.
- [109] R. Hoffmann, C. Zhang, X. Ling, L. S. Zettlemoyer, and D. S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. pages 541–550, 2011.
- [110] I. Horrocks. Ontologies and the Semantic Web. *Communications of the ACM*, 51(12):58–67, 2008.
- [111] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.
- [112] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 466–471, 2005.
- [113] I. F. Ilyas and X. Chu. *Data Cleaning*. 2019.

- [114] A. Jain, A. Doan, and L. Gravano. Sql queries over unstructured text databases. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1255–1257. IEEE, 2007.
- [115] S. Jean, Y. Aït-Ameur, and G. Pierra. Querying ontology based database using ontoql (an ontology query language). In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 704–721. Springer, 2006.
- [116] D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, 28(1):167–189, 1984.
- [117] D. Jurafsky and J. H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall, Pearson Education International, 2009.
- [118] A. Jurek, J. Hong, Y. Chi, and W. Liu. A novel ensemble learning approach to unsupervised record linkage. *Information Systems*, 71:40–54, 2017.
- [119] H. Karanikas, C. Tjortjis, and B. Theodoulidis. An approach to text mining using information extraction. In *Proc. Workshop Knowledge Management Theory Applications (KMTA 00)*. Citeseer, 2000.
- [120] J. Kasai, K. Qian, S. Gurajada, Y. Li, and L. Popa. Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042*, 2019.
- [121] A. R. Khan and H. Garcia-Molina. Attribute-based crowd entity resolution. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 549–558, 2016.
- [122] E. Kharlamov, D. Hovland, M. G. Skjæveland, D. Bilidas, E. Jiménez-Ruiz, G. Xiao, A. Soyulu, D. Lanti, M. Rezk, D. Zheleznyakov, et al. Ontology based data access in statoil. *Journal of Web Semantics*, 44:3–36, 2017.
- [123] B. Kimelfeld. Database principles in information extraction. In *Proc. of the 33rd ACM SIGACT SIGMOD SIGAI Symp. on Principles of Database Systems (PODS)*, pages 156–163, 2014.
- [124] B. Kimelfeld. Database principles in information extraction. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 156–163, 2014.
- [125] B. Kolman, R. C. Busby, and S. C. Ross. *Discrete Mathematical Structures*. Prentice-Hall, Inc., USA, 5 edition, 2003.

- [126] G. Konstantinidis and J. L. Ambite. Scalable query rewriting: a graph-based approach. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 97–108, 2011.
- [127] N. Konstantinou, E. Abel, L. Bellomarini, A. Bogatu, C. Civili, E. Irfanie, M. Koehler, L. Mazilu, E. Sallinger, A. A. Fernandes, et al. Vada: an architecture for end user informed data preparation. *Journal of Big Data*, 6(1):1–32, 2019.
- [128] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyashev. The combined approach to ontology-based data access. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2656–2661, 2011.
- [129] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493, 2010.
- [130] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. 3(1):484–493, 2010.
- [131] R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, and H. Zhu. Systemt: a system for declarative information extraction. *ACM SIGMOD Record*, 37(4):7–13, 2009.
- [132] D. Lanti, G. Xiao, and D. Calvanese. Cost-driven ontology-based data access. In *International Semantic Web Conference*, pages 452–470. Springer, 2017.
- [133] D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant query answering in ontology-based data access. 33:3–29, 2015.
- [134] D. Lembo, Y. Li, L. Popa, K. Qian, and F. Scafoglieri. Ontology mediated information extraction with MASTRO SYSTEM-T. In K. L. Taylor, R. S. Gonçalves, F. Lécué, and J. Yan, editors, *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC)*, volume 2721 of *CEUR Workshop Proceedings*, pages 256–261. CEUR-WS.org, 2020.
- [135] D. Lembo, Y. Li, L. Popa, and F. M. Scafoglieri. Ontology mediated information extraction in financial domain with mastro system-t. In D. Burdick and J. Pujara, editors, *Proceedings of the Sixth International Workshop on Data Science for Macro-Modeling, DSMM 2020, In conjunction with the ACM SIGMOD/PODS Conference, Portland, OR, USA, June 14, 2020*, pages 3:1–3:6. ACM, 2020.

- [136] D. Lembo, A. Limosani, F. Medda, A. Monaco, and F. M. Scafoglieri. Information extraction through AI techniques: The kids use case at CONSOB. *CoRR*, abs/2202.01178, 2022.
- [137] D. Lembo, J. Mora, R. Rosati, D. F. Savo, and E. Thorstensen. Mapping analysis in ontology-based data access: Algorithms and complexity. In *Proc. of the 14th Int. Semantic Web Conf. (ISWC)*, pages 217–234, 2015.
- [138] D. Lembo, D. Pantaleone, V. Santarelli, and D. F. Savo. Easy OWL drawing with the graphol visual ontology language. In C. Baral, J. P. Delgrande, and F. Wolter, editors, *Proc. of the 15th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2016)*, pages 573–576. AAAI Press, 2016.
- [139] D. Lembo, D. Pantaleone, V. Santarelli, and D. F. Savo. Drawing OWL 2 ontologies with eddy the editor. *AI Commun.*, 31(1):97–113, 2018.
- [140] D. Lembo and F. M. Scafoglieri. Coupling ontologies with document spanners. In M. Simkus and G. E. Weddell, editors, *Proceedings of the 32nd International Workshop on Description Logics, Oslo, Norway, June 18-21, 2019*, volume 2373 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.
- [141] D. Lembo and F. M. Scafoglieri. Coupling ontologies with document spanners. In *Description Logics*, 2019.
- [142] D. Lembo and F. M. Scafoglieri. A formal framework for coupling document spanners with ontologies. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 155–162. IEEE, 2019.
- [143] D. Lembo and F. M. Scafoglieri. Ontology-based document spanning systems for information extraction. *Int. J. Semantic Comput.*, 14(1):3–26, 2020.
- [144] D. Lembo and F. M. Scafoglieri. Ontology-based document spanning systems for information extraction. *International Journal of Semantic Computing*, 14(01):3–26, 2020.
- [145] D. Lembo and F. M. Scafoglieri. Comparing state of the art rule-based tools for information extraction. In A. Fensel, A. Ozaki, D. Roman, and A. Soylu, editors, *Rules and Reasoning - 7th International Joint Conference, RuleML+RR 2023, Oslo, Norway, September 18-20, 2023, Proceedings*, volume 14244 of *Lecture Notes in Computer Science*, pages 157–165. Springer, 2023.

- [146] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS)*, pages 233–246, 2002.
- [147] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, 2002.
- [148] M. Lenzerini, L. Lepore, and A. Poggi. Metamodeling and metaquerying in OWL 2 QL. *Artif. Intell.*, 292:103432, 2021.
- [149] L. Lepore, M. Namici, G. Ronconi, M. Ruzzi, and V. Santarelli. The mastro ecosystem: Ontology-based data management from theory to practice. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 101–102. IEEE, 2019.
- [150] A. Y. Levy, A. O. Mendelzon, and Y. Sagiv. Answering queries using views. In *Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 95–104, 1995.
- [151] A. Y. Levy and M.-C. Rousset. Combining horn rules and description logics in carin. *Artificial intelligence*, 104(1-2):165–209, 1998.
- [152] H. R. Lewis and C. H. Papadimitriou. Elements of the theory of computation. *ACM SIGACT News*, 29(3):62–78, 1998.
- [153] L. Libkin. Certain answers as objects and knowledge. *Artificial Intelligence*, 232:1–19, 2016.
- [154] L. Liu, C. Pu, and W. Han. Xwrap: An xml-enabled wrapper construction system for web information sources. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*, pages 611–621. IEEE, 2000.
- [155] C. Lutz, I. Seylan, D. Toman, and F. Wolter. The combined approach to obda: Taming role hierarchies using filters. In *International semantic web conference*, pages 314–330. Springer, 2013.
- [156] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

- [157] F. Maturana, C. Riveros, and D. Vrgoc. Document spanners for extracting incomplete information: Expressiveness and complexity. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 125–136, 2018.
- [158] N. McNeill, H. Kardes, and A. Borthwick. Dynamic record blocking: efficient linking of massive databases in mapreduce. In *Proceedings of the 10th international workshop on quality in databases (QDB)*. Citeseer, 2012.
- [159] B. Motik, A. Fokoue, I. Horrocks, Z. Wu, C. Lutz, and B. Cuenca Grau. OWL Web Ontology Language profiles. W3C Recommendation, W3C, Oct. 2009. Available at <http://www.w3.org/TR/owl-profiles/>.
- [160] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, et al. Owl 2 web ontology language profiles. *W3C recommendation*, 27:61, 2009.
- [161] B. Motik, B. Parsia, and P. F. Patel-Schneider. OWL 2 Web Ontology Language structural specification and functional-style syntax (second edition). W3C Recommendation, W3C, Dec. 2012. Available at <http://www.w3.org/TR/owl2-syntax/>.
- [162] M. Mugnier. Data access with horn ontologies: Where description logics meet existential rules. *Künstliche Intell.*, 34(4):475–489, 2020.
- [163] M. Mugnier and M. Thomazo. An introduction to ontology-based query answering with existential rules. volume 8714, pages 245–278, 2014.
- [164] M. Namici and G. D. Giacomo. Comparing query answering in OBDA tools over w3c-compliant specifications. In M. Ortiz and T. Schneider, editors, *Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October 27th - to - 29th, 2018*, volume 2211 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018.
- [165] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu, and J. Banerjee. Rdfx: A highly-scalable RDF store. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, pages 3–20, 2015.
- [166] M. Nentwig, M. Hartung, A.-C. Ngonga Ngomo, and E. Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.

- [167] M. Ortiz, S. Rudolph, and M. Simkus. Query answering in the Horn fragments of the description logics *SHOIQ* and *SROIQ*. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1039–1044, 2011.
- [168] M. Ortiz and M. Šimkus. Reasoning and query answering in description logics. In *Reasoning Web International Summer School*, pages 1–53. Springer, 2012.
- [169] M. Ortiz and M. Šimkus. Revisiting the hardness of query answering in expressive description logics. In *International Conference on Web Reasoning and Rule Systems*, pages 216–223. Springer, 2014.
- [170] G. Papadakis, E. Ioannou, E. Thanos, and T. Palpanas. *The Four Generations of Entity Resolution*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2021.
- [171] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas. A survey of blocking and filtering techniques for entity resolution. *arXiv preprint arXiv:1905.06167*, 2019.
- [172] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.
- [173] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.
- [174] H. Pérez-Urbina, B. Motik, and I. Horrocks. Tractable query answering and rewriting under description logic constraints. *J. Applied Logic*, 8(2):186–209, 2010.
- [175] L. Peterfreund, B. t. Cate, R. Fagin, and B. Kimelfeld. Recursive programs for document spanners. *arXiv preprint arXiv:1712.08198*, 2017.
- [176] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
- [177] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. In *Journal on data semantics X*, pages 133–173. Springer, 2008.
- [178] B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, and A. Kirilov. Kim-a semantic platform for information extraction and retrieval. *Natural language engineering*, 10(3-4):375, 2004.
- [179] R. Pottinger and A. Levy. A scalable algorithm for answering queries using views. In *VLDB*, pages 484–495, 2000.

- [180] K. Qian, L. Popa, and P. Sen. Active learning for large-scale entity resolution. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1379–1388, 2017.
- [181] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 23(4):3–13, 2000.
- [182] O. Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):1–24, 2008.
- [183] O. F. Reyes-Galaviz, W. Pedrycz, Z. He, and N. J. Pizzi. A supervised gradient-based learning algorithm for optimized entity resolution. *Data & Knowledge Engineering*, 112:106–129, 2017.
- [184] R. Rosati and A. Almatelli. Improving query answering over dl-lite ontologies. *KR*, 10:51–53, 2010.
- [185] M. T. Roth, M. Arya, L. Haas, M. Carey, W. Cody, R. Fagin, P. Schwarz, J. Thomas, and E. Wimmers. The garlic project. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, page 557, 1996.
- [186] H. Saggion, A. Funk, D. Maynard, and K. Bontcheva. Ontology-based information extraction for business intelligence. In *Proc. of the 6th Int. Semantic Web Conf. and the, 2nd Asian Semantic Web Conf. (ISWC + ASWC)*, pages 843–856, 2007.
- [187] V. Santarelli, D. Lembo, M. Ruzzi, G. Ronconi, P. Bouquet, A. Molinari, F. Pompermaier, D. Caltabiano, E. Catoni, A. Fabrizi, et al. Semantic technologies for the production and publication of open data in aci-automobile club d’italia. In *ISWC Satellites*, pages 307–308, 2019.
- [188] V. Santarelli, D. Lembo, M. Ruzzi, G. Ronconi, P. Bouquet, A. Molinari, F. Pompermaier, D. Caltabiano, E. Catoni, A. Fabrizi, M. Minenna, M. Punchina, and F. Scafoglieri. Semantic technologies for the production and publication of open data in ACI - automobile club d’italia. In M. C. Suárez-Figueroa, G. Cheng, A. L. Gentile, C. Guéret, C. M. Keet, and A. Bernstein, editors, *Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019*, volume 2456 of *CEUR Workshop Proceedings*, pages 307–308. CEUR-WS.org, 2019.
- [189] S. Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

- [190] F. Scafoglieri, D. Lembo, A. Limosani, F. Medda, and M. Lenzerini. Boosting information extraction through semantic technologies: The kids use case at CONSOB. In O. Seneviratne, C. Pesquita, J. Sequeda, and L. Etcheverry, editors, *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021*, volume 2980 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.
- [191] F. M. Scafoglieri and D. Lembo. A formal framework for coupling document spanners with ontologies. In *2nd IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2019, Sardinia, Italy, June 3-5, 2019*, pages 155–162. IEEE, 2019.
- [192] F. M. Scafoglieri, A. Monaco, G. Neccia, D. Lembo, A. Limosani, and F. Medda. Automatic information extraction from investment product documents. In G. Amato, V. Bartalesi, D. Bianchini, C. Gennaro, and R. Torlone, editors, *Proceedings of the 30th Italian Symposium on Advanced Database Systems, SEBD 2022, Tirrenia (PI), Italy, June 19-22, 2022*, volume 3194 of *CEUR Workshop Proceedings*, pages 77–84. CEUR-WS.org, 2022.
- [193] M. Scannapieco, G. Barcaroli, D. Summa, and M. Scarnò. Using internet as a data source for official statistics: a comparative analysis of web scraping technologies. In *Proc. of NTTS'15*, 2015.
- [194] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proc. of the Int. Conf. on New Methods in Language Processing*, pages 44–49, 1994.
- [195] A. Seaborne. Rdfql—a query language for rdf (member submission). *Hewlett-Packard, January*, 2004.
- [196] J. F. Sequeda and D. P. Miranker. Ultrawrap: Sparql execution on relational data. *Journal of Web Semantics*, 22:19–39, 2013.
- [197] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proc. of the 33rd Int. Conf. on Very Large Data Bases (VLDB)*, pages 1033–1044, 2007.
- [198] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*, volume 7, pages 1033–1044, 2007.
- [199] J. Shin, S. Wu, F. Wang, C. De Sa, C. Zhang, and C. Ré. Incremental knowledge base construction using deepdive. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, volume 8, page 1310. NIH Public Access, 2015.

- [200] S. Staab and R. Studer, editors. *Handbook on Ontologies*, International Handbooks on Information Systems. Springer, 2009.
- [201] K. Stefanidis, V. Efthymiou, M. Herschel, V. Christophides, et al. Entity resolution in the web of data. In *WWW (Companion Volume)*, pages 203–204, 2014.
- [202] B. ten Cate, B. Kimelfeld, L. Peterfreund, and R. Fagin. Recursive programs for document spanners. 2019.
- [203] M. A. Valenzuela-Escárcega, G. Hahn-Powell, and D. Bell. Odinson: A fast rule-based information extraction framework. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2183–2191, 2020.
- [204] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC)*, pages 137–146, 1982.
- [205] P. Velardi, S. Faralli, and R. Navigli. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707, 2013.
- [206] T. Venetis, G. Stoilos, and G. B. Stamou. Query extensions and incremental query rewriting for OWL 2 QL ontologies. *J. Data Semantics*, 3(1):1–23, 2014.
- [207] D. Z. Wang, M. J. Franklin, M. Garofalakis, J. M. Hellerstein, and M. L. Wick. Hybrid in-database inference for declarative information extraction. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 517–528, 2011.
- [208] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 219–232, 2009.
- [209] R. Witte. Multi-lingual noun phrase extractor(munpex).
- [210] R. Witte. OwlExporter guide for users and developers, 2014.
- [211] R. Witte, N. Khamis, and J. Rilling. Flexible ontology population from text: The OwlExporter. In *Proc. of LREC’10*, may 2010.
- [212] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyashev. Ontology-based data access: A survey. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 5511–5519, 2018.

-
- [213] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zharkaryashev. Ontology-based data access: A survey. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 5511–5519, 2018.
- [214] B. Yildiz and S. Miksch. onttox-a method for ontology-driven information extraction. In *International conference on computational science and its applications*, pages 660–673. Springer, 2007.

Appendix A

Consob Appendix

Documento informativo

Intento

Il presente documento contiene informazioni chiave relative a questo prodotto d'investimento. Non si tratta di un documento promozionale. Le informazioni, prescritte per legge, hanno lo scopo di aiutarvi a capire le caratteristiche, i rischi, i costi, i guadagni e le perdite potenziali di questo prodotto e di aiutarvi a fare un raffronto con altri prodotti d'investimento.

Prodotto

Nome del prodotto / ISIN:	100% ProNote con Partecipazione in USD su Thomson Reuters GI. Resource Prot. Select Index, ISIN: CH0524993752 (il prodotto)
Ideatore del prodotto:	Credit Suisse AG , il nostro sito web: www.credit-suisse.com/derivatives , per ulteriori informazioni chiamare il numero +41 (0)44 335 76 00 .
Emittente:	Credit Suisse AG, Zurigo, tramite la sua succursale di Londra, UK
Autorità competente:	L'autorità di controllo competente

Il presente documento è stato creato il febbraio 27, 2020, 06:51 CET.

State per acquistare un prodotto che non è semplice e può essere di difficile comprensione.

Cos'è questo prodotto?

Tipo: Diritti valori (Wertrechte) disciplinati dal diritto svizzero.

Obiettivi: Il prodotto è uno strumento finanziario complesso collegato ad un sottostante (Thomson Reuters GI. Resource Prot. Select Index (Indice azionario), il **sottostante**, si veda la tabella seguente). Investendo nel prodotto, è possibile partecipare ad una percentuale dell'eventuale performance positiva del sottostante. Nella data di rimborso finale l'investitore riceverà l'importo di rimborso finale, pari al 100% del taglio. Tale importo è denominato importo di rimborso minimo e non dipende dalla performance del sottostante. Inoltre, nella data di rimborso finale è possibile ricevere un importo di pagamento in caso di performance del sottostante favorevole per l'investitore. In caso di performance del sottostante sfavorevole per l'investitore, l'importo di pagamento può essere pari a zero.

Nel dettaglio:

- In caso di livello finale **più alto** al prezzo di esercizio, riceverete un importo di payout che aumenterà in funzione della performance positiva del **sottostante**. L'importo di pagamento sarà pari al taglio moltiplicato per il prodotto tra (i) la partecipazione e (ii) la differenza tra il livello finale e 100%. A causa della partecipazione, gli investitori parteciperanno a leva all'eventuale performance positiva del sottostante.
- In caso di livello finale **pari o più basso** del prezzo di esercizio, l'importo di payout sarà pari a zero e riceverete esclusivamente l'importo di rimborso minimo alla data di rimborso finale.

Il prodotto non produce interessi o altra remunerazione periodica nel corso della sua durata. Il profilo rischio/rendimento del succitato prodotto sarà differente qualora il prodotto sia venduto prima della data di rimborso finale.

Dati del prodotto

Taglio	USD 1'000	Lotto minimo di negoziazione	USD 1'000
Prezzo d'emissione	100% del taglio (USD 1'000)	Partecipazione	120%
Periodo di fixing iniziale	23.03.2020 - 25.03.2020	Fixing finale	22.09.2023
Livello iniziale	100% della media dei livelli di chiusura del sottostante nel periodo di fixing iniziale.	Livello finale	100% del livello di chiusura del sottostante alla data Fixing finale.
Data d'emissione	30.03.2020	Data di rimborso finale	29.09.2023
Ultima data di negoziazione	21.09.2023	Prezzo di Esercizio	100% del livello iniziale del sottostante
Importo di rimborso minimo	100% del taglio	Valuta del prodotto	dollaro statunitense (USD)

Dati del sottostante

Sottostante	Bloomberg Ticker	Tipo di indice	Livello iniziale
Thomson Reuters GI. Resource Prot. Select Index (Indice azionario)	TRGRPSE INDEX		128.26

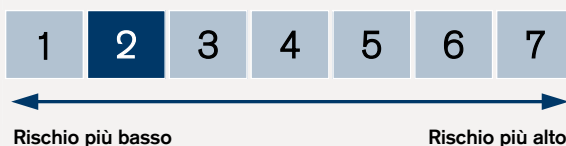
Investitore non professionale a cui si intende commercializzare il prodotto: Il prodotto è destinato agli investitori non professionali, dotati di conoscenze ed esperienza sufficienti in materia di prodotti strutturati: Prodotti a protezione del capitale e assimilati, con una capacità ridotta di sostenere perdite d'investimento e un orizzonte d'investimento a medio termine.

Termine: La data di rimborso finale del prodotto è il settembre 29, 2023. La durata del prodotto termina alla data di rimborso finale. Il prodotto prevede che qualora si verificino alcuni eventi straordinari, l'emittente possa cessarlo anticipatamente. Questi eventi riguardano principalmente il prodotto, l'emittente e il sottostante. L'importo che riceverete in caso di tale cessazione anticipata straordinaria differirà e potrebbe essere più basso dell'importo investito.

Eventuali pagamenti di dividendi effettuati sul sottostante, che rappresenta una quota o altro titolo di capitale, non saranno distribuiti agli investitori.

Quali sono i rischi e qual è il potenziale rendimento?

Indicatore sintetico di rischio



L'indicatore di rischio presuppone che il prodotto sia mantenuto fino alla data di scadenza precisa. Il rischio effettivo può variare in misura significativa in caso di disinvestimento in una fase iniziale e la somma rimborsata potrebbe essere minore. Potrebbe non essere possibile vendere facilmente il prodotto o potrebbe essere possibile vendere soltanto ad un prezzo che incide significativamente sull'importo incassato.

L'indicatore sintetico di rischio è un'indicazione orientativa del livello di rischio di questo prodotto rispetto ad altri prodotti. Esso esprime la probabilità che il prodotto subisca perdite monetarie a causa di movimenti sul mercato o a causa della nostra incapacità di pagarvi quanto dovuto. Abbiamo classificato questo prodotto al livello 2 su 7, che corrisponde alla classe di rischio bassa. Ciò significa che le perdite potenziali dovute alla performance futura del prodotto sono classificate nel livello basso e che è molto improbabile che le cattive condizioni di mercato influenzino la nostra capacità di pagarvi quanto dovuto. **Attenzione al rischio di cambio** se la valuta di riferimento è diversa dalla valuta del prodotto. Potreste ricevere pagamenti in una valuta straniera, quindi il rendimento finale che otterrete dipenderà dal tasso di cambio tra le due valute. Questo rischio non è contemplato nell'indicatore sopra riportato. Avete diritto alla restituzione di almeno 100% del vostro capitale. Qualsiasi importo più alto a quello indicato e qualsiasi rendimento aggiuntivo dipendono dalla performance futura del mercato e sono incerti. Se (noi) non (siamo) in grado di pagarvi quanto dovuto, potreste perdere il vostro intero investimento.

Scenari di performance

Non è possibile prevedere con precisione l'andamento futuro dei mercati. Gli scenari illustrati sono solo un'indicazione di alcuni degli sbocchi possibili in base ai rendimenti recenti. I rendimenti effettivi potrebbero essere inferiori a quanto indicato negli scenari di performance.

Investimento USD 10'000

Scenari		1 anno	2 anni	3 anni 6 mesi (periodo di detenzione raccomandato)
Scenario di stress	Quanto potreste essere rimborsati al netto dei costi	USD 9'688.29	USD 9'791.32	USD 10'000.00
	Rendimento medio annuale	-3.12%	-1.05%	0.00%
uno scenario sfavorevole	Quanto potreste essere rimborsati al netto dei costi	USD 10'036.06	USD 9'872.82	USD 10'000.00
	Rendimento medio annuale	0.36%	-0.64%	0.00%
uno scenario moderato	Quanto potreste essere rimborsati al netto dei costi	USD 10'786.73	USD 10'738.88	USD 10'522.15
	Rendimento medio annuale	7.87%	3.63%	1.46%
uno scenario favorevole	Quanto potreste essere rimborsati al netto dei costi	USD 12'201.33	USD 13'092.51	USD 14'358.81
	Rendimento medio annuale	22.01%	14.42%	10.89%

Questa tabella mostra gli importi dei possibili rimborsi nei prossimi 3 anni 6 mesi, in scenari diversi, ipotizzando un investimento di USD 10'000. Gli scenari presentati mostrano la possibile performance dell'investimento. Possono essere confrontati con gli scenari di altri prodotti. Gli scenari presentati sono una stima della performance futura sulla base di prove relative alle variazioni passate del valore di questo investimento e non sono un indicatore esatto. Gli importi dei rimborsi varieranno a seconda della performance del mercato e del periodo di tempo per cui è mantenuto l'investimento/il prodotto. Lo scenario di stress indica quale potrebbe essere l'importo rimborsato in circostanze di mercato estreme e non tiene conto della situazione in cui non siamo in grado di pagarvi. Le cifre riportate comprendono tutti i costi del prodotto in quanto tale, ma possono non comprendere tutti i costi da voi pagati al consulente o al distributore. Le cifre non tengono conto della vostra situazione fiscale personale, che può anch'essa incidere sull'importo del rimborso.

Cosa accade se Credit Suisse AG non è in grado di corrispondere quanto dovuto?

Se Credit Suisse AG diventa insolvente, gli investitori devono essere disposti, nel peggiore dei casi, a sostenere la perdita totale del loro investimento. Il prodotto non è coperto da sistemi di garanzia dei depositi istituzionali o di altro tipo. Se l'emittente e/o il garante sono soggetti a eventuali misure di risoluzione (ad es. bail-in), i vostri diritti possono essere pari a zero, convertiti in azioni o la data di scadenza può essere modificata.

Quali sono i costi?

La diminuzione del rendimento (Reduction in Yield - RIY) esprime l'impatto dei costi totali sostenuti sul possibile rendimento dell'investimento. I costi totali tengono conto dei costi una tantum, correnti e accessori. Gli importi qui riportati corrispondono ai costi cumulativi del prodotto in tre periodi di detenzione differenti, comprendono le potenziali penali per uscita anticipata. Questi importi si basano sull'ipotesi che siano investiti USD 10'000. Gli importi sono stimati e potrebbero cambiare in futuro.

Andamento dei costi nel tempo

La persona che vende questo prodotto o fornisce consulenza riguardo ad esso potrebbe addebitare altri costi, nel qual caso deve fornire informazioni su tali costi e illustrare l'impatto di tutti i costi sull'investimento nel corso del tempo.

Investimento USD 10'000

Scenari	In caso di disinvestimento dopo 1 anno	In caso di disinvestimento dopo 2 anni	In caso di disinvestimento alla fine del periodo di detenzione raccomandato
Costi totali	USD 432.92	USD 415.02	USD 405.27
Diminuzione del rendimento (RIY) per anno	4.33%	2.05%	1.14%

Composizione dei costi

La seguente tabella presenta:

- l'impatto, per ciascun anno, dei differenti tipi di costi sul possibile rendimento dell'investimento alla fine del periodo di detenzione raccomandato;
- il significato delle differenti categorie di costi.

La presente tabella mostra l'impatto sul rendimento per anno

Costi una tantum	Costi di ingresso	1.14%	Impatto dei costi da sostenere al momento della sottoscrizione dell'investimento. Questo è l'importo massimo che si paga; si potrebbe pagare di meno.
	Costi di uscita	n/a	Impatto dei costi di uscita dall'investimento alla scadenza.
Costi correnti	Costi delle operazioni di portafoglio, altri costi correnti	n/a	Non sono applicati costi correnti per questo prodotto.
Oneri accessori	Commissioni di performance, commissioni di overperformance	n/a	Non sono applicati oneri accessori per questo prodotto.

Per quanto tempo devo detenerlo? Posso ritirare il capitale prematuramente?

Periodo di detenzione raccomandato: 3 anni 6 mesi (ossia fino alla data di rimborso finale)

La durata del prodotto è di 3 anni 6 mesi. Non è previsto il diritto di recesso per l'investitore. Di conseguenza, gli investitori devono essere disposti a mantenere l'investimento per la durata del prodotto. L'unica possibilità per disinvestire anticipatamente è la vendita del prodotto attraverso la borsa su cui è quotato o all'ideatore/emittente del prodotto al di fuori di tale borsa. Il prodotto sarà quotato su Borsa Italiana, Electronic Bond Market. In condizioni di mercato normali l'ideatore del prodotto cercherà di fornire i prezzi denaro/lettera del prodotto in ogni giorno lavorativo, ma non è legalmente obbligato a farlo. In particolare, la vendita del prodotto potrebbe non essere possibile in situazioni di mercato eccezionali o in caso di guasti tecnici. L'investitore che venda il prodotto nel corso della durata potrebbe incassare un ricavato della vendita più basso al prezzo d'emissione del prodotto.

Come presentare reclami?

Qualsiasi reclamo relativo alla persona che ha consigliato o venduto il prodotto può essere inviato direttamente alla persona in questione.

Qualsiasi reclamo in merito al prodotto (termini), al presente documento o alla condotta dell'ideatore del prodotto può essere presentato per iscritto a Credit Suisse AG Cross Asset Derivatives Sales PO Box CH-8070 Zurich, o via e-mail a structured.products@credit-suisse.com, o visitando il nostro sito web www.credit-suisse.com/kid.

Altre informazioni rilevanti

Il presente documento informativo non contiene tutte le informazioni relative al prodotto. Per le condizioni generali giuridicamente vincolanti del prodotto nonché per una descrizione dettagliata dei rischi e i benefici connessi al prodotto, consultare il prospetto. Il prospetto è disponibile su www.credit-suisse.com/derivatives ed è possibile richiedere una copia cartacea gratuita a Credit Suisse AG, Transaction Advisory Group, Uetlibergstrasse 231, 8070 Zurigo, Svizzera. Le informazioni contenute nel presente documento informativo non costituiscono una raccomandazione per l'acquisto o la vendita del prodotto e non sostituiscono la consulenza personalizzata della banca o del consulente dell'investitore. Eventuali versioni aggiornate del presente documento informativo saranno pubblicate su: www.credit-suisse.com/kid.

