GENERAL TERMINOLOGY INDUCTION IN DESCRIPTION LOGICS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2017

By
Viachaslau Sazonau
School of Computer Science

Contents

Word count: 62121

List of Tables

List of Figures

Abstract

In computer science, an ontology is a machine-processable representation of knowledge about some domain. Ontologies are encoded in ontology languages, such as the Web Ontology Language (OWL) based on Description Logics (DLs). An ontology is a set of logical statements, called axioms. Some axioms make universal statements, e.g. all fathers are men, while others record data, i.e. facts about specific individuals, e.g. Bob is a father. A set of universal statements is called TBox, as it encodes terminology, i.e. schema-level conceptual relationships, and a set of facts is called ABox, as it encodes instance-level assertions.

Ontologies are extensively developed and widely used in domains such as biology and medicine. Manual engineering of a TBox is a difficult task that includes modelling conceptual relationships of the domain and encoding those relationships in the ontology language, e.g. OWL. Hence, it requires the knowledge of domain experts and skills of ontology engineers combined together. In order to assist engineering of TBoxes and potentially automate it, acquisition (or induction) of axioms from data has attracted research attention and is usually called Ontology Learning (OL).

This thesis investigates the problem of OL from general principles. We formulate it as General Terminology Induction that aims at acquiring general, expressive TBox axioms (called general terminology) from data. The thesis addresses and investigates in depth two main questions: how to rigorously evaluate the quality of general TBox axioms and how to efficiently construct them. We design an approach for General Terminology Induction and implement it in an algorithm called DL-Miner. We extensively evaluate DL-Miner, compare it with other approaches, and run case studies together with domain experts to gain insight into its potential applications.

The thesis should be of interest to ontology developers seeking automated

means to facilitate building or enriching ontologies. In addition, as our experiments show, DL-Miner can deliver valuable insights into the data, i.e. can be useful for data analysis and debugging.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank my supervisor Uli Sattler for invaluable help, prompt responses to my enquiries, and lengthy discussions of technical matters that made this thesis possible. I could not ask for a better supervisor. I would also like to thank Nicolas Matentzoglu for his help and advice on designing and running experiments. I am thankful to Bijan Parsia for his interest in my work and helpful advice. I thank Amanda Hicks and Michael Rutherford for providing data for a case study, participating in it, and giving precious feedback. I thank Oghenejokpeme Orhobor for his interest in my work, providing data for a case study, and interesting technical discussions.

Chapter 1

Introduction

This chapter provides an introduction to the subject of this thesis. We start from discussing philosophical roots of the problem and then briefly describe the topics covered in the thesis and contributions made.

1.1 From Scientific Explanation to Ontology Learning

The importance of scientific knowledge has been long recognised. It dates back to Antiquity when philosophers appreciated the value of scientific knowledge and started searching for it, e.g. Aristotle in *Posterior Analytics*. Already at that time, it was noticed that knowing *that* and knowing *why* are two essentially different types of knowledge [Sal06]. For example, knowing that every day the sun rises above the horizon is one thing and knowing why it does so is a completely different matter. While the first type of knowledge is a mere *observation*, or fact, the second one is an *explanation*[1] that may help to understand not only the observed phenomenon *per se* but also other, possibly unrelated phenomena.

For example, it is observed that every day the sun appears in the sky, moves across it, and disappears from it. A possible explanation for this observation is that the earth is static and the sun moves around it. It was the basis of the geocentric model that was prevailing for centuries. Nonetheless, this model could not explain some other observations, e.g. that the sun and the moon rise at different positions over the year, some planets do not rise for several months, and some

[1] Philosophers have identified different types of explanations depending on their purpose. We concentrate on scientific explanations.

planets even move in the direction opposite to the stars. Thus, an explanation of one particular observation can be unable to explain other observations that indicate the need for another, more general explanation.

A better explanation covering the observed motions of the sun, planets, and stars was given by the heliocentric model formulated by Nicolaus Copernicus in the 16th century. In the same way, this model was unable to explain other observations of planetary movements (orbits) and was refined by the laws of planetary motion formulated in *Astronomia nova* by Johannes Kepler in the 17th century. In this sense, science was developing as consecutive attempts to explain more observations, i.e. build a scientific *theory* that provides answers to all currently raised why questions. A scientific theory is a set of laws that are *universal*, i.e. applicable to all entities under consideration.

If there are multiple competing theories explaining all observations equally well, the choice of a theory can be guided by the principle called *Occam's razor*, named after William of Ockham who lived in the 14th century and made this principle well known. It states that, among two theories equal by all criteria but simplicity, the simpler one should be preferred. Various forms of it were proposed and accepted by many philosophers and scientists, including Aristotle, Kant, Galileo, Newton, etc.

While accepting Occam's razor, philosophers question what simplicity of a theory amounts to and why it should be relevant for describing the world [Sob15]. There are two types of simplicity that are commonly distinguished. The first one is syntactic simplicity which accounts for the number of laws a theory consists of and conciseness of those laws, e.g. to be measured by the syntactic length of a theory. Thus, shorter theories are syntactically simpler.

The second type of simplicity is ontological simplicity, also called semantic simplicity or parsimony, which captures the number and complexity of actual postulates of a theory (that can be represented in different syntactic forms via respective laws). Ontological simplicity can manifest itself in the assumptions that a theory makes in addition to explaining observations. The fewer assumptions are made beyond necessity by a theory, the semantically simpler the theory is. Consider Example 1.1.

Example 1.1. A hospital monitors patients suffering from cold and records their recovery times. According to the observations, patients taking vitamin C recover

from cold within one week, while others take longer to recover. Then, the observations can be explained by the theory $\mathcal{T}_1$ which postulates that "a patient taking vitamin C will recover from cold within one week". However, the observations can also be explained by another theory $\mathcal{T}_2$ stating that "a patient taking vitamin C will recover from cold or *flu* within one week". Since the hospital only monitors patients suffering from cold, i.e. there are no observations about patients suffering from flu, the theory $\mathcal{T}_2$ *assumes* that all patients taking vitamin C will recover within one week not only from cold but also from flu. Thus, $\mathcal{T}_2$ is syntactically and semantically more complex than $\mathcal{T}_1$. Therefore, according to Occam's razor, $\mathcal{T}_1$ should be preferred over $\mathcal{T}_2$.

In Example 1.1, the theory $\mathcal{T}_1$ captures a significant correlation evident from observations. However, this does not imply that the theory is actually explanatory, i.e. useful for explaining not only the given observations but the whole population as well, see Example 1.2.

Example 1.2. Suppose that, to confirm the theory $\mathcal{T}_1$, the hospital runs the second round of recordings. The new observations show that patients which do not take vitamin C recover from cold within one week, i.e. they do as well as patients taking vitamin C. Then, all observations recorded during the first and second round can be explained by a new theory $\mathcal{T}_3$ stating that "a patient will recover from cold within one week", i.e. regardless of taking vitamin C. Thus, in the light of the observations of the second round, the old theory $\mathcal{T}_1$ seems to be an *artefact* of the observations of the first round.

Example 1.2 shows the well-known fact that correlation does not imply causation (see the classic reference [HG93] and an interesting discussion in [Bar14]). Whether a correlation is meaningful[2] or not can only be tested in the presence of population-wide observations or confirmed by knowledge about the whole population.

Within Artificial Intelligence (AI), the process of seeking theories, or inferring general laws (of different types and shapes), that explain observations of particular instances is called *induction*.[3] It is usually opposed to *deduction*: while the result of deduction is certain, i.e. *provable*, the result of induction is only tentative, i.e. *probable*. Importantly, the goal of induction is usually a *generalisation*,

[2]See some examples of spurious correlations at `http://www.tylervigen.com`.

[3]Induction, or inductive reasoning, should not be confused with mathematical induction which is a type of deductive reasoning.

i.e. a theory explaining not only the given observations but the whole population, see Example 1.2. Clearly, the quality of generalisations varies depending on a number of observations that *support* them. The problem of induction is extensively studied in Machine Learning (ML) and Data Mining (DM), subfields of AI.

This thesis focuses on induction. In order to automate induction (as well as deduction), we need to formalise observations and theories in a way that makes them *machine-processable*. In other words, we need to encode theories and observations in a language that can be processed by a machine. Developing such languages is a concern of Knowledge Representation (KR), a subfield of AI.

A theory encoded in a machine-processable language is commonly called a *knowledge base*, or *ontology*. Many ontology languages are based on logic and encode an ontology as a set of logical statements. Being based on logic, such an ontology can be interpreted in a certain, unambiguous way that allows for deriving logical consequences from it, i.e. deductive reasoning. A popular ontology language is the Web Ontology Language [GHM+08] (OWL). OWL is based on Description Logics [BCM+10] (DLs) which are typically fragments of First Order Logic (FOL). OWL ontologies have been developed and extensively used in several domains, particularly in biology and medicine.[4] In the following, we only consider OWL ontologies.

Logical statements of an ontology are called *axioms*. Axioms can be of two different types: some represent universal statements, e.g. "all husbands are men who are married"; others are facts about particular individuals, e.g. "John is a husband", "John is married to Ann", "Ann is a wife", etc. A set of universal statements of the ontology is called *TBox* and represents schema-level conceptual relationships, or a *terminology*. A set of facts of the ontology is called *ABox* and represents instance-level assertions. Thus an ontology is generally the union of a TBox and ABox (though any of them can be empty).

For the purpose of induction, a theory can be encoded as a TBox and observations, which are henceforth called *data*, can be encoded as an ABox. The problem of inducing a TBox from an ABox is investigated in *Ontology Learning*[5] (OL). It is the main topic of this thesis.

The research interest in OL is caused by the fact that manual engineering

[4] See for example `http://bioportal.bioontology.org`.

[5] The term "ontology learning" means learning ontologies from data and should not be confused with studying or understanding or memorising ontologies.

of a TBox is a difficult task that includes modelling conceptual relationships of the domain and encoding those relationships in the ontology language, e.g. OWL. Hence, it requires the knowledge of domain experts and skills of ontology engineers combined together. Given some data, possibly from some database, one can attempt to acquire, or *learn*, generalisations from it. Such generalisations can then be used to assist ontology engineering and potentially automate constructing ontologies to a good extent. In the realms of constantly growing data, particularly (semi)structured data, OL could also help to bridge the "semantic gap" between knowledge and data [LV14].

Within the broad research area of AI, OL can be seen as a point of convergence of ML/DM and KR: the first is concerned with acquiring (learning, mining, or inducing) generalisations from data and the second investigates languages to make generalisations machine-processable. Generalisations can take different shapes that can capture some types of knowledge and cannot capture others. Shapes of generalisations are determined by a language which is chosen to encode generalisations. One language can be able to encode all shapes of another language and additionally encode shapes which that language is not able to encode. In other words, one language can be more *expressive* (powerful) than another. Clearly, more expressive languages allow for acquiring more generalisations, and possibly more useful ones, from data than less expressive languages. However, they also make searching for useful generalisations harder because the number of candidates grows. We discuss this trade-off throughout the thesis and consider some specific examples.

1.2 General Terminology Induction in Description Logics

Although OL has attracted considerable research interest, current approaches have evident limitations: they concentrate on learning restricted types of TBox axioms in DLs, require supervision and human intervention, or disregard the semantics of DLs. The *goal* of this thesis is to advance the state-of-the-art of OL and overcome some limitations of existing approaches by considering the problem of OL from general principles. Our *objectives* are as follows:

- We investigate *General Terminology Induction* (GTI): whether and how a

general TBox, called *general terminology*, can be induced from an ABox in DLs while respecting the standard semantics.

- We design, implement, and evaluate an approach for GTI.

A reasonable approach for GTI should be able to induce "good" generalisations. It can be formulated as search in the space of all possible generalisations and requires constructing generalisations and evaluating their *quality*. Such an approach is not straightforward to design as it faces several challenges in the realm of DLs.

Firstly, DLs provide expressive languages for encoding generalisations and, consequently, allow for acquiring rich knowledge. However, high expressivity leads to a vast space of generalisations that need to be checked. Thus, there is a *trade-off* between expressivity and computational feasibility.

Secondly, evaluating the quality of generalisations in DLs should work well with, or *respect*, the standard semantics. More specifically, DLs permit *incomplete* information about individuals in an ABox. For example, if the ontology does not state that John is a father nor that he is not a father, it does not imply that he is not a father. Instead, the information about John is assumed to be incomplete. This is commonly called the Open World Assumption (OWA). In addition, while inducing generalisations from the given ABox, we should also respect the given TBox, i.e. background knowledge. The latter means that we should treat explicit ABox facts in the same way as we treat implicit, entailed facts.

Thirdly, usefulness of a generalisation is *not* equivalent to its "correctness". In other words, not only correct generalisations are useful. More specifically, there might be *three types* of useful generalisations: those that enrich the TBox, those that reflect new interesting correlations, and those that indicate modelling flaws or data errors. Therefore, any induced generalisation should be viewed as a *hypothesis* whose usefulness is yet to be confirmed by a domain expert (see also Example 1.2).

1.3 Contributions of This Thesis

This thesis makes the following *contributions*.

- We formulate GTI as a multi-objective search problem in DLs.
- We propose a range of measures to evaluate quality of hypotheses in GTI. We distinguish three quality dimensions, i.e. readability, logical quality, and statistical quality, and define measures for each of them. The measures are designed for DLs, i.e. respect the standard semantics and background knowledge. We investigate general (universal) relationships between the measures.
- We design a data-driven algorithm, called DL-Apriori, that constructs only promising hypotheses (and avoids even considering almost all unpromising ones). We prove its correctness, termination, and completeness.
- We implement all proposed techniques, i.e. computing quality measures and constructing promising hypotheses, in one algorithm called DL-Miner.
- We empirically evaluate DL-Miner, i.e. design and run experiments, analyse their results and gain insights, as follows:
  - we select and analyse experimental data;
  - we investigate how the quality measures correlate with each other and compare their correlations with their general relationships;
  - we evaluate the computational performance of DL-Miner;
  - we compare DL-Miner with other OL approaches;
  - we design and execute two case studies to gain insight into usefulness of all three types of hypotheses of DL-Miner and its quality measures.

This thesis explicates and elaborates ideas originally published in [SSB15]. In order to explain some performance results, we refer to the results published in [SSB14]. The work has also been published and presented at several community workshops.

Chapter 2

Preliminaries

In this chapter, we will introduce basic notions which are required to understand this thesis. We will fix the terminology and explain the relevant concepts of Description Logics, Data Mining, and Machine Learning. We will also introduce a running example used in the following chapters. We assume the reader to be familiar with Data Mining, First Order Logic and, ideally, Description Logics.

2.1 Description Logics

As the basis of the Web Ontology Language [GHM+08] (OWL), Descriptions Logics [BCM+10] (DLs) provide means for encoding ontologies. The interested reader can consult the Description Logic Handbook [BCM+10] for additional details.

2.1.1 Brief Background

Descriptions Logics [BCM+10] (DLs) are a family of knowledge representation formalisms that are typically decidable fragments (subsets) of First Order Logic (FOL).[1] A DL knowledge base, called a DL *ontology*, is a set of logical formulae, called *axioms*.

Structurally, an ontology consists of terminological and assertional parts. The terminological part represents *concepts* (unary predicates) and *roles* (binary predicates) with their relationships. It is often separated into two parts: a *TBox*

[1] There are, however, DLs which are not decidable and not fragments of FOL

that encodes implications between concepts and an *RBox* that encodes implications between roles. For the sake of brevity, in this thesis we do not make this distinction and call the terminological part, including both concept and role implications, a TBox, or *terminology*. A TBox capture *background knowledge* as it is general concept-level knowledge about the domain. The assertional part of an ontology is called an *ABox* and encodes facts – *individuals* (constants) with their relationships. An ABox captures *data* as it contains factual, instance-level knowledge of the domain in the context of this thesis (but can be a means of modelling in general).

As FOL, DLs have formal, well-defined *semantics*. Therefore, the expressed knowledge is unambiguous, i.e. it is completely clear what it means based on the semantics. Then, if a suitable calculus exists, it becomes possible to automatically derive logical inferences of an ontology. This process is called *deductive reasoning*, or automated reasoning. For example, reasoning is used to check *consistency* of an ontology, i.e. to verify that it contains no contradictory axioms. In DLs logical inferences are called *entailments*. Entailments are implicit relationships turned to explicit ones by reasoning, i.e. they uncover knowledge which follows from an ontology, yet is not explicitly stated in it.

Let us formally define the aforementioned notions that will be explained by a running example later on. Let N_C, N_R, and N_I be pairwise disjoint sets of concept, role, and individual names, respectively. We call *term* any element in $N_C \cup N_R \cup N_I$. A concept A (role R) is called *atomic* if $A \in N_C$ ($R \in N_R$). Concepts and roles can be *complex*, see Definition 2.1.

Definition 2.1 (Complex concept, role)**.** A concept C (role R) is called *complex*[2] if $C \notin N_C$ ($R \notin N_R$) and C (R) is an expression built from $N_C \cup N_R \cup N_I$ (N_R) according to syntax rules, called *constructors*, of DLs.

Constructors of a DL determine its *expressivity*, i.e. which expressions it permits to use. For example, the DL $\mathcal{EL}$ provides no role constructors and the following concept constructors: top $\top$, conjunction $C \sqcap D$, existential restriction $\exists R.C$. Another notable DL $\mathcal{ALC}$ provides all the concept constructors of $\mathcal{EL}$ and, in addition, the following ones: bottom $\bot$, negation $\neg C$, disjunction $C \sqcup D$, universal restriction $\forall R.C$. Thus, all $\mathcal{EL}$ concepts can be expressed in $\mathcal{ALC}$ but not vice versa. We say that $\mathcal{ALC}$ is *more expressive* than $\mathcal{EL}$. Even more

[2]In the scope of this thesis, a complex role is any role which is not atomic.

expressive DL $\mathcal{SROIQ}$ provides additional concept and role constructors, e.g. inverse R^-, composition $R \circ S$, etc. In the following, we define the semantics for the aforementioned constructors. Complex concept and roles in a given DL are built by inductively applying its constructors, see Example 2.1.

Example 2.1. The following C and D are complex $\mathcal{EL}$ and $\mathcal{ALC}$ concepts, respectively:

$$C := A \sqcap \exists R.(B \sqcap \exists R.\top)$$
$$D := A \sqcup \exists R.(\neg B \sqcap \forall R.B)$$

A TBox encodes relations between concepts via *concept inclusions* and relations between roles via *role inclusions*, see Definition 2.2.

Definition 2.2 (TBox)**.** A *TBox* $\mathcal{T}$ is a finite set of *general concept inclusions* and *role inclusions*.

- A *general concept inclusion* (GCI), or *concept subsumption*, is an expression of the form $C \sqsubseteq D$, where C and D are (possibly complex) concepts. A concept C is said to be *subsumed* by a concept D. We call a GCI of the form $A \sqsubseteq B$, where $A, B \in N_C$, an *atomic concept subsumption*.
- A *role inclusion* (RI), or *role subsumption*, is an expression of the form $R \sqsubseteq S$, where R and S are (possibly complex) roles. A role R is said to be *subsumed* by a role S. We call a RI of the form $R \sqsubseteq S$, where $R, S \in N_R$, an *atomic role subsumption*.
- $C \equiv D$ ($R \equiv S$) is the abbreviation for $\{C \sqsubseteq D,\ D \sqsubseteq C\}$ ($\{R \sqsubseteq S,\ S \sqsubseteq R\}$). A concept C (role R) is said to be *equivalent* to a concept D (role S).
- A *concept definition* is a GCI of the form $A \equiv C$, where $A \in N_C$ (a concept name defined), C is a possibly complex concept (a concept description).

In contrast to FOL, DLs have a *variable-free* syntax since they do not explicitly use variables. DL axioms can usually be translated to FOL, see Example 2.2.

Example 2.2. The concepts C and D from Example 2.1 are straightforwardly translated to FOL formulae as follows:

$$C(x) := A(x) \wedge \exists y(R(x,y) \wedge B(y) \wedge \exists z R(y,z))$$
$$D(x) := A(x) \vee \exists y(R(x,y) \wedge \neg B(y) \wedge \forall z(R(y,z) \rightarrow B(z)))$$

The DL axiom $C \sqsubseteq D$ is translated to FOL as $\forall x(C(x) \rightarrow D(x))$.

An ABox encodes facts about individuals in the domain via concept and role *assertions*, see Definition 2.3.

Definition 2.3 (ABox)**.** An *ABox* $\mathcal{A}$ is a finite set of *concept assertions* and *role assertions*.

- A *concept assertion* (CA) is an expression of the form $C(a)$, where $a \in N_I$ and C is a (possibly complex) concept.
- A *role assertion* (RA) is an expression of the form $R(a, b)$, where $a, b \in N_I$ and R is a (possibly complex) role.

GCIs, RIs, CAs, and RAs are called *axioms*. A set of all axioms, i.e. the union of a TBox and ABox, is called a DL ontology, see Definition 2.4.

Definition 2.4 (DL Ontology)**.** Given a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, we call their union $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ a *DL ontology*.[3]

The vocabulary of an ontology is given by its *signature*, see Definition 2.5.

Definition 2.5 (Signature)**.** The *signature* $\widetilde{\alpha}$ of an axiom α is a set of all terms from N_C, N_R, and N_I occurring in α. Then, the *signature* $\widetilde{\mathcal{O}}$ of an ontology $\mathcal{O}$ is defined as follows: $\widetilde{\mathcal{O}} := \bigcup\{\widetilde{\alpha} \mid \alpha \in \mathcal{O}\}$. The sets of all *individual names* $in(\mathcal{O})$, *concept names* $cn(\mathcal{O})$, *role names* $rn(\mathcal{O})$ of an ontology $\mathcal{O}$ are defined, respectively, as follows: $in(\mathcal{O}) := \widetilde{\mathcal{O}} \cap N_I$, $cn(\mathcal{O}) := \widetilde{\mathcal{O}} \cap N_C$, $rn(\mathcal{O}) := \widetilde{\mathcal{O}} \cap N_R$. Let $crn(\mathcal{O}) := cn(\mathcal{O}) \cup rn(\mathcal{O})$.

An ontology specifies a set of constraints that determine what is possible and what is not possible in the domain. In other words, it acts like a "filter" to disallow certain situations. Formally, the semantics of DLs is based on *interpretations*.

Definition 2.6 (Interpretation)**.** An *interpretation* $\mathcal{I}$ is a structure $\mathcal{I} := (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set (called *domain*) and $\cdot^{\mathcal{I}}$ is a function (called *interpretation function*) which maps every concept name $A \in N_C$ to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, every role name $R \in N_R$ to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, every individual name $a \in N_I$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is inductively extended to complex concepts and roles, see Table 2.1 (see [BCM+10] for the semantics of other DL constructors).

DL	Syntax	Semantics	Name
$\mathcal{EL}$	$\top$	$\Delta^{\mathcal{I}}$	top
	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	conjunction
	$\exists R.C$	$\{a \mid \exists b : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$	existential restriction
$\mathcal{ALC}$	$\bot$	$\emptyset$	bottom
	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$	negation
	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	disjunction
	$\forall R.C$	$\{a \mid \forall b : (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$	universal restriction
$\mathcal{SROIQ}$	$\geq (\leq)\ nR.C$	$\{a \mid \lvert\{b : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \geq (\leq)\ n\}$	atleast(atmost) restriction
	R^-	$\{(b,a) \mid (a,b) \in R^{\mathcal{I}}\}$	role inverse
	$R \circ S$	$\{(a,c) \mid \exists b : (a,b) \in R^{\mathcal{I}} \wedge (b,c) \in S^{\mathcal{I}}\}$	role composition
	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	universal role
	$\ldots$		

Table 2.1: Syntax and semantics of DL constructors

An interpretation that satisfies the constraints given by axioms of the ontology is called its *model*, see Definition 2.7.

Definition 2.7 (Model)**.** An interpretation $\mathcal{I}$ *satisfies*

- a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$;
- a RI $R \sqsubseteq S$ if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$;
- a CA $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$;
- a RA $R(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$;
- an ontology $\mathcal{O}$ if $\mathcal{I}$ satisfies each axiom in $\mathcal{O}$.

An interpretation $\mathcal{I}$ that satisfies an ontology $\mathcal{O}$ is called a *model* of $\mathcal{O}$.

Intuitively, a model describes a "world" which is allowed to exist according to the knowledge expressed by the ontology. For example, given the ontology $\mathcal{O} := \{A \sqsubseteq B\}$ which encodes that "all A's are also B's", the interpretation $\mathcal{I}_1$, where $\Delta^{\mathcal{I}_1} = \{a\}$, $A^{\mathcal{I}_1} = \{a\}$, $B^{\mathcal{I}_1} = \{a\}$, is a model of $\mathcal{O}$, while the interpretation $\mathcal{I}_2$, where $\Delta^{\mathcal{I}_2} = \{a\}$, $A^{\mathcal{I}_2} = \{a\}$, $B^{\mathcal{I}_2} = \emptyset$, is not. The semantics of interpretations determines the *entailment relation* denoted as $\models$, see Definition 2.8.

[3]Throughout the thesis, we use $\mathcal{O}$ to denote ontologies.

Definition 2.8 (Entailment)**.** The *entailment relation* $\models$ is defined as follows.

- An ontology $\mathcal{O}$ *entails* another ontology $\mathcal{O}'$, written as $\mathcal{O} \models \mathcal{O}'$, if all models of $\mathcal{O}$ are models of $\mathcal{O}'$.
- $\mathcal{O} \equiv \mathcal{O}'$ is the abbreviation for $\mathcal{O} \models \mathcal{O}'$ and $\mathcal{O}' \models \mathcal{O}$. Such ontologies $\mathcal{O}$ and $\mathcal{O}'$ are said to be *equivalent*.
- $\mathcal{O} \models \alpha$ is the abbreviation for $\mathcal{O} \models \{\alpha\}$. Such an axiom α is called an *entailment* of $\mathcal{O}$, or $\mathcal{O}$ is said to *entail* α.
- $\alpha \models \alpha'$ is the abbreviation for $\{\alpha\} \models \{\alpha'\}$, $\alpha \equiv \alpha'$ is the abbreviation for $\{\alpha\} \equiv \{\alpha'\}$.
- An ontology $\mathcal{O}$ is called *inconsistent* if $\mathcal{O}$ has no models, i.e. $\mathcal{O} \models \top \sqsubseteq \bot$. Otherwise, $\mathcal{O}$ is called *consistent*.
- Any entailment α of the empty ontology is called a *tautology*, i.e. $\emptyset \models \alpha$, abbreviated as $\models \alpha$ or simply α.
- An individual a is called an *instance* of a concept C if $\mathcal{O} \models C(a)$.

An entailment can be an axiom of any type, i.e. a GCI, RI, CA, or RA. Trivially, any axiom in the ontology is entailed by that ontology. Since a tautology follows from the empty ontology, it follows from any ontology.

A GCI stating that the extensions of two concepts are disjoint in all models is called a *disjointness axiom*, see Definition 2.9.

Definition 2.9 (Disjointness axiom)**.** Let C and D be concepts, α a GCI. If $\alpha \models C \sqsubseteq \neg D$, then α is called a *disjointness axiom* for C and D. A concept C is said to be *disjoint* from a concept D.

In other words, a disjointness axiom enforces that there is no model where the extensions of the given concepts have an element in common. Please note that a disjointness axiom can take many syntactically different (but semantically equivalent) shapes: $C \sqsubseteq \neg D$, $D \sqsubseteq \neg C$, $C \sqcap D \sqsubseteq \bot$, etc.

Checking, or deciding, the entailment relation $\models$ is called *reasoning*. There are several reasoning problems that are commonly distinguished in DLs, see Definition 2.10.

Definition 2.10 (Reasoning problems)**.** Let $\mathcal{O}$ be an ontology.

- The reasoning problem of testing whether $\mathcal{O}$ is consistent is called *consistency checking*.
- Given concepts C and D, the reasoning problem of testing $\mathcal{O} \models C \sqsubseteq D$ is called *subsumption checking*.
- Given a concept C, the reasoning problem of testing $\mathcal{O} \models C \sqsubseteq \bot$ is called *satisfiability checking*. If $\mathcal{O} \models C \sqsubseteq \bot$, C is called *unsatisfiable*. Otherwise, C is called *satisfiable*.
- The *concept hierarchy* of $\mathcal{O}$ is defined as follows: $ch(\mathcal{O}) := \{A \sqsubseteq B \mid \mathcal{O} \models A \sqsubseteq B \ \wedge \ A, B \in cn(\mathcal{O}) \cup \{\top, \bot\}\}$. The reasoning problem of computing $ch(\mathcal{O})$ is called *ontology classification*.
- Given a concept C and an individual $a \in in(\mathcal{O})$, the reasoning problem of testing $\mathcal{O} \models C(a)$ is called *instance checking*. If $\mathcal{O} \models C(a)$, a is called an *instance* of C in $\mathcal{O}$.
- Given a concept C, the reasoning problem of retrieving all instances of C in $\mathcal{O}$ is called *instance retrieval*.

The concept hierarchy of an ontology can be computed by checking subsumptions for all pairs of concept names occurring in the ontology (including $\top$ and $\bot$). Hence, it can be reduced to subsumption checking.

Instance retrieval is essentially a type of query answering in DLs, where a query is given by a concept C and potential answers are all individuals $in(\mathcal{O})$ occurring in the ontology $\mathcal{O}$. It can be computed by checking for each individual whether it is an instance of C in $\mathcal{O}$. Hence, instance retrieval can be reduced to instance checking.

Algorithms that provide decision procedures for reasoning problems in DLs are called *reasoning services*. Tools that implement reasoning services for DLs are called *reasoners*. Many highly-optimised reasoners have been developed, including popular HermiT [SMH08], Pellet [SPG+07], FaCT++ [TH06], ELK [KKS11].

There is a variety of DLs that differ in their expressivity and computational complexity of reasoning. Higher expressivity generally, but not necessarily, implies higher computational complexity. Thus, one can choose a DL according to the required expressivity and affordable complexity.

2.1.2 A Running Example

In order to explain the basics of DLs, let us introduce a running example, see Example 2.3, which will also be used throughout the thesis.

Example 2.3. (Kinship) The example presents ontology Kinship $:= \mathcal{T} \cup \mathcal{A}$ that models family relations. It was adopted from the UCI Machine Learning Repository.[4] The TBox $\mathcal{T}$ captures the following background knowledge. All men and women are humans, all mothers are women, all fathers are men. Men are disjoint from women, i.e. no individual can be a man and a woman simultaneously. Marriage is a symmetric relation, having a parent is the inverse of having a child.

$$\begin{aligned}\mathcal{T} = \{&Man \sqsubseteq Human,\ Woman \sqsubseteq Human,\\ &Father \sqsubseteq Man,\ Mother \sqsubseteq Woman,\\ &Man \sqsubseteq \neg Woman,\ marriedTo \sqsubseteq marriedTo^{-},\\ &hasParent \sqsubseteq hasChild^{-},\ hasChild \sqsubseteq hasParent^{-}\}.\end{aligned}$$

The ABox $\mathcal{A}$ encodes the following data. $Chris$ is married to $Penelope$ and they have two children, $Victoria$ and $Arthur$. $James$ is married to $Victoria$ and they have a daughter, $Charlotte$. $Arthur$ is married to $Margaret$.

$$\begin{aligned}\mathcal{A} = \{&Man(Arthur),\ Father(Chris),\ Father(James),\\ &Woman(Charlotte),\ Woman(Margaret),\ Mother(Penelope),\\ &Mother(Victoria),\\ &hasParent(Charlotte, James),\ hasParent(Charlotte, Victoria),\\ &hasParent(Victoria, Chris), hasParent(Victoria, Penelope)\\ &hasParent(Arthur, Penelope),\ hasParent(Arthur, Chris),\\ &marriedTo(Chris, Penelope),\ marriedTo(James, Victoria)\\ &marriedTo(Arthur, Margaret)\}.\end{aligned}$$

The TBox in Example 2.3 has the signature that consists of 5 concept names: $Human$, Man, $Woman$, $Father$, $Mother$; and 3 role names: $marriedTo$, $hasParent$, $hasChild$. The concept $\neg Woman$ is a complex concept, the roles $marriedTo^{-}$ and $hasChild^{-}$ are complex roles.

The TBox consists of 8 axioms: 5 GCIs and 3 RIs. For example, the GCI

[4] `https://archive.ics.uci.edu/ml/datasets/Kinship`

$Mother \sqsubseteq Woman$ means that all mothers are women. The ABox consists of 16 axioms: 7 CAs and 9 RAs. For example, the CA $Man(Arthur)$ means that the individual $Arthur$ is an instance of the concept Man and the RA $hasParent(Arthur, Penelope)$ means that $Arthur$ has a parent named $Penelope$.

An ABox is essentially a graph where nodes are individuals labelled with concepts and edges represent role assertions between them. The ABox in Example 2.3 is illustrated as a graph in Figure 2.1, where p stands for $hasParent$, m for $marriedTo$.

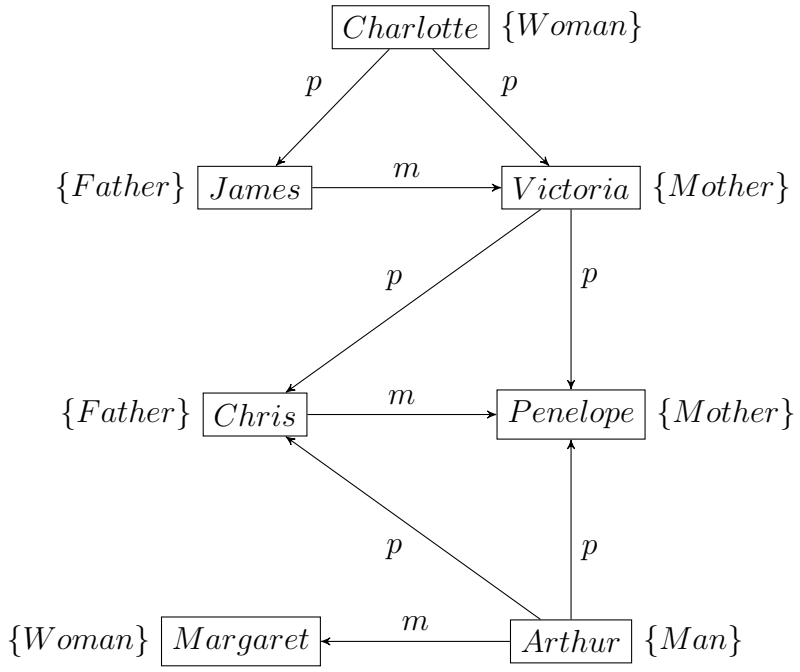


Figure 2.1: ABox graph of Kinship

The ontology Kinship is consistent, i.e. there are no contradictions. The axiom $Man \sqsubseteq \neg Woman$, which can also be written as $Man \sqcap Woman \sqsubseteq \bot$, means that men are disjoint from women. Therefore, if one adds the CA $Woman(Arthur)$ to the ABox, then the ontology becomes inconsistent.

Since Kinship contains axioms $Mother \sqsubseteq Woman$ and $Woman \sqsubseteq Human$, it entails that $Mother \sqsubseteq Human$, written as Kinship $\models Mother \sqsubseteq Human$. Another example of entailment is $Father \sqsubseteq \neg Mother$, i.e. mothers are disjoint from fathers. The entailments $Mother \sqsubseteq \top$ and $Mother \sqcap Woman \sqsubseteq Woman$ are tautologies, i.e. they follow from any ontology.

An ontology can also entail instance-level facts. For example, since Kinship states that $Chris$ is a $Father$, it entails that $Chris$ is also a Man, written as

Kinship $\models Man(Chris)$. Since the ontology states that having a parent is the inverse of having a child and $Victoria$ has parent $Chris$, it entails that $Chris$ has child $Victoria$, or Kinship $\models hasChild(Chris, Victoria)$.

As an example of instance retrieval, let us query for all instances of the concept $Woman \sqcap \exists marriedTo.Man$, i.e. all individuals who are women and married to some man. The answer set is $\{Victoria, Penelope, Margaret\}$.

2.1.3 Resource Description Framework

The *Resource Description Framework*[5] (RDF) is a specification, standardised by the World Wide Web Consortium (W3C), for modelling web resources. RDF is essentially a data model that describes resources via *subject-predicate-object triples*, where a subject and object are resources, a predicate defines a (directed) relationship between them. RDF has several serialisation formats, e.g. Turtle, N-Triples, RDF/XML. Within this thesis it is sufficient to understand RDF as another syntax for DL ABoxes, see Example 2.4.

Example 2.4. All ABox axioms of Kinship, see Example 2.3, can be written as RDF triples in the Turtle format as follows (some axioms are skipped for brevity):
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix kin: <http://owl.cs.manchester.ac.uk/kinship.owl#> .
kin:Arthur rdf:type kin:Man .
kin:Chris rdf:type kin:Father .
...
kin:Charlotte kin:hasParent kin:James .
kin:Charlotte kin:hasParent kin:Victoria .
...

Nonetheless, not every DL ABox can be written using RDF. In particular, RDF is unable to encode assertions of complex concepts, i.e. $C(a)$, where C is a complex concept. RDF is domain-independent and users define their own vocabulary in a schema language, called *RDF Schema*.[6] In addition, RDF Schema allows specifying concept and role hierarchies, i.e. expressing some TBox axioms.

[5] `https://www.w3.org/RDF`
[6] `https://www.w3.org/TR/rdf-schema`

2.1.4 Web Ontology Language

Despite the great volumes of data on the Web, it is largely machine-processable, yet not machine-understandable. The *Semantic Web* is a term referring to the idea of machine-understandable data on the Web. It is an initiative to provide semantics to the data, or bridge the so-called "semantic gap". Ontologies are considered as a key technology to achieve this goal since they provide common vocabularies where terms can be mapped to the content of web documents.

The Web Ontology Language [HPVH03, GHM+08] (OWL) is one of the most popular ontology languages nowadays. OWL is standardized by the World Wide Web Consortium (W3C).[7]

OWL is based on DLs. The current version of OWL, OWL 2 [GHM+08], corresponds to the expressive DL $\mathcal{SROIQ}(\mathcal{D})$.[8] As reasoning in $\mathcal{SROIQ}$ is computationally complex [HKS06], OWL[9] provides 3 sub-languages such that one can select a required ratio of language expressivity and reasoning complexity. Like RDF, OWL has several serialisation formats, e.g. OWL2 XML, RDF/XML, Manchester.

Since OWL can express all (TBox and ABox) axioms of $\mathcal{SROIQ}$, OWL is just another syntax for it, see Example 2.5.

Example 2.5. The concepts C and D from Example 2.1 can be written in OWL in the Manchester syntax as follows:

$$C := A \textit{ and } R \textit{ some } (B \textit{ and } R \textit{ some owl:Thing})$$
$$D := A \textit{ or } R \textit{ some } (\textit{not } B \textit{ and } R \textit{ only } B)$$

where *owl:Thing* corresponds to $\top$ in DLs. The DL axiom $C \sqsubseteq D$ is written as "*C SubClassOf D*".

A significant number of OWL ontologies have been developed for academia and industry. In particular, this is evident in bioinformatics, healthcare, and life sciences where ontologies are found to be very useful for managing large terminologies. Prominent examples of such ontologies are NCI Thesaurus, GALEN, SNOMED CT, International Classification of Diseases (ICD). The NCBO BioPortal repository[10] contains a large collection of biomedical ontologies. Ontologies

[7] `https://www.w3.org/OWL`
[8] $\mathcal{SROIQ}(\mathcal{D})$ is more expressive than $\mathcal{SROIQ}$.
[9] In the following, by OWL we mean OWL 2.
[10] `http://bioportal.bioontology.org/`

have also been used in chemistry, astronomy, defence and other areas.

Ontologies, particularly their TBoxes, are often built manually by ontology engineers.[11] Tools used for ontology engineering are called *ontology editors*. One of popular ontology editors is Protégé.[12]

Manual ontology engineering requires knowledge representation skills and domain expertise. Not surprisingly, the process can consume significant human efforts. As an example, the National Cancer Institute Thesaurus[13] (NCIt) is a reference terminology and biomedical ontology that provides definitions, synonyms, and other information on more than 10,000 cancers and related diseases. NCIt has been developed for more than a decade and is currently updated monthly.

2.1.5 Open World Assumption

The semantics of DLs permits encoding incomplete information. This permission is commonly called the *Open World Assumption* (OWA). It is the opposite of the *Closed World Assumption* (CWA), or the "negation as failure" semantics, conventionally used in databases and logic programming. In contrast to the CWA, information that is missed in the knowledge base under the OWA is considered being just *incomplete*, but not false.

For a DL ontology, the OWA means that an axiom which is not entailed by the ontology (and hence not contained in it either), can hold in some (but not all) models of it. In other words, there are worlds where the axiom is true and worlds where it is false. Such an axiom can be either a TBox axiom or an ABox assertion. In practice, the OWA is the reason why an axiom which is not entailed by the ontology is not necessarily false: the ontology may simply be incomplete. As OWL is based on DLs, it also permits the OWA.

Within this thesis, it is important to note that the OWA in DLs implies that it is possible that $\mathcal{O} \not\models C(a)$ and $\mathcal{O} \not\models \neg C(a)$. In other words, it can be unknown whether an individual a is an instance of a concept C or its negation $\neg C$. For example, the lack of information in Kinship does *not* imply that $Arthur$ is not a $Father$ and $Margaret$ is not a $Mother$: these facts may simply be not encoded by the ontology.

[11] Some OWL ontologies are the result of translations from other formats, e.g. OBO.
[12] `http://protege.stanford.edu`
[13] `http://www.cancer.gov/research/resources/terminology`

2.1.6 Unique Name Assumption

The *Unique Name Assumption* (UNA) is a simplifying assumption commonly used in AI including KR. The UNA states that different names always refer to different entities. Consequently, the absence of the UNA implies that different names may refer to the same entity.

Some DLs, particularly DL-lite [CDL+05, ACKZ09], use the UNA to make reasoning easier. Indeed, the absence of the UNA in a DL introduces additional choices for reasoning: different individual names may refer to the same entity (the same element in the domain) in some models and may refer to different entities in other models. For example, the absence of the UNA in Kinship means that there are models where $Charlotte$ and $Margaret$ refer to the same person. Please note that, however, there are no models where $Charlotte$ and $Chris$ are the same person because of the restriction that all men are disjoint from women, i.e. $Man \sqsubseteq \neg Woman$.

Although the UNA seems a reasonable assumption to make for Kinship, it is not always the case. Sometimes, different names do refer to the same object, e.g. different spelling variants, alternative forms, etc. For example, $Margaret$ has a spelling variant, $Margareth$, which would imply to be a different person in Kinship under the UNA. A prominent example where this matters is UniProt[14] which is a heterogeneous knowledge base that contains multiple identifiers (names) for the same protein.

Modern DLs do not normally use the UNA, excluding those that maintain tractability of reasoning, e.g. DL-lite. In order to express that two individuals a and b are equal (different), DLs support assertions of the form $a \approx b$ ($a \not\approx b$). Similarly, OWL does not use the UNA but provides the language constructs *owl:sameAs* (*owl:differentFrom*) to encode the equality (inequality).

2.1.7 Logic Programming

Logic Programming [Llo87] (LP) is another KR formalism. It is important for this thesis to clarify the relationship between LP and DLs. LP is based on the *Horn* fragment (subset) of FOL. A knowledge base in LP is called a *logic program*. A logic program consists of Horn clauses. A clause in LP is typically written as an implication of the form $A \leftarrow L_1, L_2, \ldots, L_n$, where A is an atom called *head*

[14] `http://www.uniprot.org/`

and $L_1, L_2, \ldots, L_n$ is a conjunction of literals called *body*.

Thus, as most DLs, LP is a fragment of FOL. However, LP has different from DLs expressivity. In other words, some knowledge can be expressed in DLs, but cannot be expressed in LP (the reverse is also true as LP permits predicates of more than two variables). Most importantly, the head A of a clause $A \leftarrow L_1, L_2, \ldots, L_n$ can only be an atom. This means that LP is unable to express GCIs $C \sqsubseteq D$, where D is a complex concept. For example, it is possible to state in DLs that "having a father implies having a mother" via the axiom $\exists hasParent.Father \sqsubseteq \exists hasParent.Mother$ but not possible in LP because existentially quantified variables are not allowed in the head. Another example is "being married to a mother implies being not a mother" which can be expressed in DLs via the axiom $marriedTo.Mother \sqsubseteq \neg Mother$ but cannot be expressed in LP because negation is not allowed in the head (though it is possible to express disjointness axioms, e.g. "mothers are disjoint from fathers"). Thus, LP clauses are similar to DL concept definitions, where a head A is defined by its body $L_1, L_2, \ldots, L_n$.

In addition, LP has different from DLs semantics. For example, a DL axiom $B \sqsubseteq A$ entails $\neg A \sqsubseteq \neg B$, while the respective LP clause $A \leftarrow B$ does not. In contrast to DLs, LP normally makes the CWA, i.e. assumes that what is not known to be true is false. LP normally assumes that the interpretation domain is closed for individuals of the ABox (constants), while DLs permit domain elements not mapped to any ABox individual (generally infinite number of them).

2.2 Machine Learning and Data Mining

Methods of Machine Learning (ML) and Data Mining (DM) are concerned with tasks of automated data analysis, processing, manipulation, and organisation. The interested reader can consult ML/DM handbooks, e.g. [Bis06, TSK13], for additional details.

2.2.1 Brief Background

In the literature, ML is often conflated with DM. The difference relevant for this thesis is that the first commonly aims at *predictive* data analysis, while the second focuses on *exploratory* data analysis. In other words, ML methods usually have a target variable (target variables) whose value they attempt to predict given values

of other variables. To achieve that, they automatically build, or *learn*, a model[15] of the data given training examples and then use that model for predictions. On the other hand, DM methods focus on searching for, or *mining*, patterns in the given data in order to uncover its structure and discover interesting dependencies. Thus, DM methods do not pursue prediction as their main goal (though it can be one of their goals), as ML methods do. In spirit, this thesis is closer to DM than ML, however, in order to comply with the terminology used in the related literature, the words "learn" and "mine" are used interchangeably.

In ML the process of learning is often viewed as a *search*, i.e. a learning algorithm (implicitly or explicitly) searches the space of possible models. A model is usually a function from the space of known variables to the space of target variables, e.g. probability distribution, decision tree, set of rules, etc. Some models are more powerful, or *expressive*, than others, i.e. able to encode more data patterns (in the same way as more expressive ontologies are able to encode more knowledge).

The quality of a model is determined by the data. Hence, any model is a *hypothesis*. A seemingly good model can turn out to be bad. This can happen due to various reasons, e.g. incorrect, noisy, or biased data. Another, not so obvious reason is that a model, while being powerful, can *overfit* the data, i.e. reproduce too many details of it. This can be caused by the mistakenly chosen learning algorithm (that has built the model). A simple example is a model that trivially memorises all the data.

2.2.2 How Data is Viewed

In ML, data is commonly given as a collection of objects of the same type, e.g. images, text documents, patients, customers, products, etc. An object is described by its *features*, also called *attributes* or *variables*, e.g. pixel values, word counts, age, gender, etc. Features are usually decided for the collection by human experts and their values are specified for each object. Thus, data is normally viewed as a *table*, or matrix, where rows are labelled by objects and columns are labelled by features.

ML data is usually *complete* with respect to features, i.e. all feature values are known for every given object.[16] In other words, there is no object with a

[15] A ML model should be confused with a DL model, see Section 2.1.1.

[16] There are approaches that investigate handling incomplete data, e.g. [STP07, SCB14].

missing value for one of the features. Please note that data completeness is not assumed with respect to objects, i.e. there may be objects that are not recorded by the data. Objects are normally assumed to be *independent and identically distributed* which is the so-called *i.i.d* assumption.

If data is incomplete, we first have to add all missing information. To do that in practice, we may attempt to gather missing information from relevant resources. If it is hard or impossible to do, we may assume that what is missing is false, or apply the CWA. In fact, this is what ML and DM approaches tend to do (and often justifiably). Example 2.6 illustrates how data is viewed in ML.

Example 2.6. We adopt Example 2.3. Each object, a person from Example 2.3, is described by its features of different types: categorical (*Gender*), Boolean (*Parent*, *Married*), numerical (*Children*). Please notice that information about relations between objects is missing. We apply the CWA, e.g. we assume that Arthur is not a father and Margaret is not a mother. As a result, we obtain Table 2.2. Please note that this is different from the way the data is viewed in DLs, see Figure 2.1.

	Gender	*Parent*	*Married*	*Children*
Charlotte	Female	No	No	0
James	Male	Yes	Yes	1
Victoria	Female	Yes	Yes	1
Chris	Male	Yes	Yes	2
Penelope	Female	Yes	Yes	2
Margaret	Female	No	Yes	0
Arthur	Male	No	Yes	0

Table 2.2: Kinship data in ML

2.2.3 Supervised and Unsupervised Learning

In the ML literature, there is a conventional distinction of methods to *supervised* and *unsupervised* learning. As suggested by the name, supervised learning is the process of learning a model of the data under some supervision which is provided by human experts. Such supervision may take different forms and often comes as a set of desired outputs, or *training examples*. There is usually a *learning goal*, i.e. a target variable whose value a model is aimed to predict based on values of other variables.

A model is aimed to *generalise* from the data, i.e. describe well not only seen but also unseen inputs. Supervision is used to evaluate how well a model does. In order to reasonably estimate its prediction error, a model is usually trained on the training data and tested on the test data which is disjoint from the training data. Standard evaluation techniques, such as *k-fold cross-validation*, ensure that the data is used as efficiently as possible. Thus, coupled with the data, supervision gives a clear signal when a model is right and wrong and allows to steer the search when necessary.

In contrast to supervised learning, unsupervised learning builds a model of the data without any supervision. It is generally harder than supervised learning since there is no clear signal indicating how well a model performs. Consequently, it is more difficult to evaluate the progress and guide the search accordingly. On the other hand, supervision can be costly (or even infeasible for certain tasks) to provide since it requires human expertise.

Let us now explain the basic notions by Example 2.7, see also [Bis06].

Example 2.7. A set of images of handwritten digits is given where all digits occur. The task is to build a model that recognises all handwritten digits. Figure 2.2 shows images of handwritten digits from 0 to 9 (first line) which are correctly recognised by the trained model (second line).

Figure 2.2: Recognised handwritten digits for the NIST database (IBM Research)

It is clear that a model for Example 2.7 could be constructed by manually encoding rules describing shapes of digits on images. However, due to the high variability of handwriting, that would be a hard task and the resulting model would be unsuitable in most cases.

ML methods can be employed to accomplish the task of Example 2.7. If supervision is given, e.g. images are manually labelled with corresponding digits, a model can be learned that *classifies* images based on their features and labels into distinct categories, or classes. The learned model can then be used to classify

unlabelled images of digits. This task is called *classification*.[17]

If supervision is not available, a model can be learned that groups, or *clusters*, images based on their features, i.e. similar images are placed in the same cluster. As in the supervised learning scenario, the resulting model can be used to recognise digits for unseen images, i.e. for classification. However, its quality can be worse because similarity alone may be insufficient for recognition. This task is called *clustering* which is a common type of unsupervised learning.

Thus, Example 2.7 can be approached by both supervised and unsupervised learning. Although supervised learning may produce a better model, it is more demanding as it requires appropriate training examples and sufficient quantities of them for each digit.

2.2.4 Inductive Logic Programming

Inductive Logic Programming [Qui90, Mug91] (ILP) is a type of *symbolic* ML that uses Logic Programming (LP), see Section 2.1.7, as a language for representing models, called *hypotheses*. Thus, it aims at human-readable, symbolic models based on logic. The distinctive feature of ILP is that it can naturally use background knowledge which is represented in the same language as hypotheses.

ILP is a type of *supervised* learning, i.e. it generalises from training examples. The most common setting of ILP is learning logical *definitions of predicates* where examples are tuples that belong or do not belong to a target predicate. The task is to learn a definition that satisfies all positive examples and no negative examples, see Definition 2.11.

Definition 2.11 (ILP learning problem)**.** Let $\mathcal{D}$ be a knowledge base (logic program), A be a *target* predicate, $\mathcal{L}$ be a language bias, $E := E^+ \cup E^-$ be a set of *examples* for A, where E^+ are *positive* and E^- are *negative* examples. Then, an *ILP learning problem* is to find a definition H for A such that H conforms to $\mathcal{L}$ and

$$\text{for all } e \in E^+ \quad \mathcal{D} \cup \{H\} \models e \text{ and for all } e \in E^- \quad \mathcal{D} \cup \{H\} \not\models e.$$

Any definition is called a *hypothesis*. A hypothesis is normally a Horn clause (or a set of). Importantly, the head of a hypothesis clause is fixed to be A, i.e. it consists only of the target predicate. In other words, the task is to find a

[17] ML classification should not be confused with ontology classification in DLs and OWL, see Section 2.1.1.

hypothesis *body* that "fits" training examples. A hypothesis H must conform to a language bias $\mathcal{L}$ that determines a *hypothesis space* $\mathbb{H}$, i.e. a set of all hypotheses under consideration. For instance, a language bias can limit the number of literals in the body of a hypothesis clause, thus making a hypothesis space finite.

Definition 2.11 specifies the requirement for a hypothesis to be a *solution*: it must satisfy *all* positive and *no* negative examples. This requirement is often relaxed to the requirement of satisfying *almost* all positive and *almost* no negative examples. This relaxation allows for considering hypotheses of varying *quality* instead of solutions and non-solutions. The quality of a hypothesis is calculated based on the number of satisfied training examples like in traditional ML. Example 2.8 illustrates the basics of ILP.

Example 2.8. We adopt the Kinship ontology, see Example 2.3, and rewrite DL axioms as LP clauses. Irrelevant information is skipped for brevity.

$$\begin{aligned}\mathcal{D} = \{&Woman(x) \leftarrow \ not\ Man(x),\ hasChild(x,y) \leftarrow hasParent(y,x),\\ &Man(Arthur),\ Man(Chris),\ Man(James),\\ &Woman(Penelope),\ Woman(Victoria),\\ &Woman(Charlotte),\ Woman(Margaret),\\ &hasParent(Charlotte, James),\ hasParent(Charlotte, Victoria),\\ &hasParent(Victoria, Chris), hasParent(Victoria, Penelope)\\ &hasParent(Arthur, Penelope),\ hasParent(Arthur, Chris)\}.\end{aligned}$$

The task is to learn a definition of the predicate $Mother$, given the following positive and negative examples:

$$\begin{aligned}E^+ &= \{Mother(Penelope),\ Mother(Victoria)\},\\ E^- &= \{Mother(Charlotte),\ Mother(Margaret),\\ &\qquad Mother(Arthur),\ Mother(Chris),\ Mother(James)\}.\end{aligned}$$

Thus, we assume that Charlotte and Victoria are *not* mothers. The following clause is a possible solution since it satisfies all positive and no negative examples:

$$Mother(x) \leftarrow Woman(x),\ hasChild(x,y).$$

It is translated to FOL as follows: $\forall x \exists y (Woman(x) \wedge hasChild(x,y) \rightarrow Mother(x))$.

An ILP algorithm methodically searches the hypothesis space. As the search space is usually vast, the following two steps are crucial:

(i) the space is structured such that it can be unfolded systematically;

(ii) suitable heuristics navigate the search such that promising hypotheses are considered earlier than others.

The step (i) is usually done via ordering the space of all body clauses by their *generality*. Informally, a clause is more general (more specific) than another clause if it imposes less (more) restrictions. For example, the clause $Woman(x)$ is more general than the clause $Woman(x),\ hasChild(x, y)$ since the last one has the additional requirement "to have a child". A more general (specific) clause is called a *generalisation* (*specialisation*). In practice, the syntactic notion of generality is normally used in order to maintain computational feasibility. The search starts from the most general clause. A clause is systematically refined, i.e. made more specific. Refinements are specified by so-called *refinement operators* that determine a set of specialisations (refinements) of a clause. A refinement operator normally defines a set of minimal (most general) specialisations, e.g. extensions with one additional literal in their body.

The step (ii) is done by choosing suitable heuristics to explore the search space ordered by generality. Structurally, the ordering is a directed graph where nodes are clauses and edges connect them to their respective refinements. While expanding the refinement graph, there is a choice which node to refine next. This choice is made by heuristics which are usually based on the number of satisfied (covered) positive and negative examples. As the search space is usually large, often only promising nodes are further refined using greedy or beam search. One of first ILP algorithms making the aforementioned steps was FOIL [Qui90]. It has inspired many other algorithms and their implementations, e.g. PROGOL [Mug95] and ALEPH.[18]

2.2.5 Association Rule Mining

Association Rule Mining [TSK13] (ARM) is a popular method of DM. It is a type of *unsupervised* learning. One of the main applications of ARM is market basket analysis that we will use to explain its basics.

[18] `http://www.cs.ox.ac.uk/activities/machlearn/Aleph/`

Market basket analysis is used by retailers in order to understand the purchasing behaviour of customers. Retailers normally record sales transactions in databases. Given a sequence of transactions, retailers can use analytics to inspect recorded transactions and identify items frequently bought together. Interesting associations, called *rules*, between items can be mined. This information is used to boost sales via cross-selling, promotions, discounts, loyalty programs, inventory planning, etc.

A *transaction* is usually defined as a set of items purchased by a customer in a single order. Let T be a transaction and $\mathcal{D}$ be a set of transactions, i.e. $T \in \mathcal{D}$. Let M be a set of all items available for purchasing, i.e. any transaction $T \subseteq M$. Any set $X \subseteq M$ of items is called *itemset*. A set of transactions can be viewed as a table, where rows are labelled with transactions $T \in \mathcal{D}$ and columns are labelled with items $m \in M$. Items purchased in a transaction are marked in the respective row.

Definition 2.12 (Association rule)**.** An *association rule* is an implication of the form $X \Rightarrow Y$, where $X \subseteq M$, $Y \subseteq M$.

Informally, an association rule $X \Rightarrow Y$ means that the presence of an itemset X implies the presence of an itemset Y in the same transaction. The left-hand side (LHS) of a rule, X, is called *antecedent* and the right-hand side (RHS), Y, is called *consequent*.

There are different measures used to evaluate the quality (or interestingness) of an association rule. The common quality measures are given by Definition 2.13.

Definition 2.13 (Association rule quality measures)**.** Let $\mathcal{D}$ be a set of transactions. The *frequency* of an itemset X in $\mathcal{D}$ is defined as follows:

$$count(X, \mathcal{D}) := |\{T \in \mathcal{D} \mid X \subseteq T\}|.$$

Let $X \Rightarrow Y$ be an association rule, where X, Y are itemsets in $\mathcal{D}$. The *coverage, support, confidence* of $X \Rightarrow Y$ in $\mathcal{D}$ are defined, respectively, as follows:

$$\begin{aligned} cov(X \Rightarrow Y, \mathcal{D}) &:= count(X, \mathcal{D}) \\ sup(X \Rightarrow Y, \mathcal{D}) &:= count(X \cap Y, \mathcal{D}) \\ conf(X \Rightarrow Y, \mathcal{D}) &:= \frac{count(X \cap Y, \mathcal{D})}{count(X, \mathcal{D})}. \end{aligned}$$

The coverage counts the number of transactions containing all items from the antecedent. The support of a rule shows how many transactions contain all items from both the antecedent and consequent. The confidence counts how many transactions contain all items from both the antecedent and consequent out of those transactions that contain all items from the antecedent. The confidence is in the range $[0, 1]$. Please note that support is a *symmetric* measure, i.e. $sup(X \Rightarrow Y, \mathcal{D}) = sup(Y \Rightarrow X, \mathcal{D})$, while coverage and confidence are *asymmetric*. Association rules with high support and high confidence are commonly assumed to be of high quality.

The measures in Definition 2.13 can be given probabilistic meanings via replacing $count(X, \mathcal{D})$ with $\mathbf{P}_{\mathcal{D}}(X) := count(X, \mathcal{D})/|\mathcal{D}|$, i.e. the probability of an itemset X to occur in $\mathcal{D}$. The latter makes coverage and support normalised, i.e. to be in the range $[0, 1]$, and does not affect the confidence. The measures are written via probabilities as follows:

$$\begin{aligned} cov_{[0,1]}(X \Rightarrow Y, \mathcal{D}) &:= \mathbf{P}_{\mathcal{D}}(X) \\ sup_{[0,1]}(X \Rightarrow Y, \mathcal{D}) &:= \mathbf{P}_{\mathcal{D}}(X, Y) \\ conf(X \Rightarrow Y, \mathcal{D}) &:= \frac{\mathbf{P}_{\mathcal{D}}(X, Y)}{\mathbf{P}_{\mathcal{D}}(X)}. \end{aligned}$$

Besides the measures given in Definition 2.13, other measures exist that highlight different quality characteristics [GH06]. Amongst popular measures are lift and conviction defined, respectively, as follows:

$$\begin{aligned} lift(X \Rightarrow Y, \mathcal{D}) &:= \frac{\mathbf{P}_{\mathcal{D}}(X, Y)}{\mathbf{P}_{\mathcal{D}}(X) \cdot \mathbf{P}_{\mathcal{D}}(Y)} \\ conv(X \Rightarrow Y, \mathcal{D}) &:= \frac{\mathbf{P}_{\mathcal{D}}(X) \cdot \mathbf{P}_{\mathcal{D}}(\neg Y)}{\mathbf{P}_{\mathcal{D}}(X, \neg Y)}. \end{aligned}$$

Lift measures how frequently X and Y occur together in comparison to the case when X and Y are statistically independent. Conviction measures how frequently X occurs with $\neg Y$ (i.e. X occurs without Y) if X and $\neg Y$ are independent in comparison to the actual occurrence of X and $\neg Y$. In contrast to support and confidence, lift and conviction can be infinite. Please note that lift is a symmetric measure, while conviction is not. High values of lift and conviction usually indicate an unexpected and potentially interesting rule.

The *goal* of ARM is to find association rules of high quality for a given set

of transactions. A user normally defines thresholds for minimal support and confidence. If a rule has a high support and confidence, then it is considered to be interesting. Please note that ARM, as it is usual in DM, requires *complete* data, see Section 2.2.2. Example 2.9 illustrates the basic notions of ARM.

Example 2.9. The set of transactions $\mathcal{D} := \{T_1, T_2, T_3, T_4, T_5\}$ is given by Table 2.3, where items are $M := \{bread, milk, butter, beer, snacks\}$, "X" means that an item is included in a transaction and the blank space means that it is not. The user-specified minimal thresholds are $sup_{min} = 3$ and $conf_{min} = 0.7$. Then, the following are examples of association rules that can be mined from $\mathcal{D}$:

$$\begin{aligned} \{milk, butter\} &\Rightarrow \{bread\} && (sup = 3,\ conf = 1.0) \\ \{beer\} &\Rightarrow \{snacks\} && (sup = 3,\ conf = 1.0) \\ \{bread\} &\Rightarrow \{snacks\} && (sup = 3,\ conf = 0.75). \end{aligned}$$

	bread	*milk*	*butter*	*beer*	*snacks*
T_1	X	X	X		X
T_2				X	X
T_3	X			X	X
T_4	X	X	X		
T_5	X	X	X	X	X

Table 2.3: Sales transactions

An ARM algorithm usually performs two main steps:

(i) find all itemsets whose frequency is above a minimal support, called *frequent* itemsets;

(ii) frequent itemsets are used to find all association rules whose confidence is above a minimal confidence.

Step (ii) is rather straightforward. The challenge of step (i) is that the brute-force search is usually infeasible since there are exponentially many itemsets to check. Therefore, most approaches focus on step (i). One of the first algorithms which used an intelligent search was Apriori [AS94]. It is the basis for many state-of-the-art approaches. Apriori greatly reduces the search space by exploiting the so-called *anti-monotone* property of the frequency of an itemset, see Lemma 2.1.

Lemma 2.1 (Anti-monotone property of itemsets)**.** *Let $\mathcal{D}$ be a set of transactions, X, Y itemsets. Then, $X \subseteq Y$ implies $count(X, \mathcal{D}) \geq count(Y, \mathcal{D})$.*

Lemma 2.1 means that if an itemset is not frequent, then all itemsets that contain it are also not frequent and, hence, can be pruned from the search *a priori*.

2.2.6 Formal Concept Analysis

Formal Concept Analysis [GW99] (FCA) is a mathematical theory of concepts and concept hierarchies emerged from lattice theory. It also refers to a set of techniques for conceptual clustering and knowledge representation based on *formal contexts*. FCA has been used in DM, ML, knowledge management, text mining, biology, etc. Its central notion is *formal context*, see Definition 2.14.

Definition 2.14 (Formal context)**.** A *formal context* is a structure $\mathcal{K} := (G, M, I)$ where G is a set of *objects*, M is a set of *attributes*, and I is an *incidence relation* $I \subseteq G \times M$. An object $g \in G$ *possesses* an attribute $m \in M$ if $(g, m) \in I$, written as gIm.

Thus, an incidence relation I specifies dependencies between objects G and attributes M. Importantly, information in a formal context is *complete*, i.e. an object either possesses an attribute or it does not. An *implication* on attributes in a formal context is defined as follows, see Definition 2.15.

Definition 2.15 (Implication)**.** Let g^I be the set of all attributes that apply to an object $g \in G$, or formally $g^I := \{m \in M \mid gIm\}$. Given an attribute set M, an *implication*, written as $X \to Y$ with $X, Y \subseteq M$, *holds* in a formal context $\mathcal{K} := (G, M, I)$ if $X \subseteq g^I$ implies $Y \subseteq g^I$ for all objects $g \in G$.

Informally, an implication $X \to Y$ holds in a formal context if every object which has all attributes in X also has all attributes in Y. An implication either holds in a formal context or it does not.

One can notice that FCA has many similarities with ARM. Transactions can be viewed as objects and items as attributes. An incidence relation is simply a function that maps transactions to their items. Implications are similar to association rules. As it follows from Definition 2.15, implications that hold in the formal context are association rules of the confidence 1.0.

The data in FCA is viewed exactly the same as in ARM, in particular, it is complete. Therefore, we can use Table 2.3 to illustrate FCA, see Example 2.10.

Example 2.10. A formal context is given by Table 2.4 which is the same as Table 2.3. Objects are transactions $G := \{T_1, T_2, T_3, T_4, T_5\}$ and attributes are items $M := \{bread, milk, butter, beer, snacks\}$. "X" means that an object possesses an attribute and the blank space means that it does not. The following are examples of implications:

$$\{milk, butter\} \to \{bread\} \quad \text{(holds)}$$
$$\{beer\} \to \{snacks\} \quad \text{(holds)}$$
$$\{bread\} \to \{snacks\} \quad \text{(does not hold)}$$
$$\{butter, snacks\} \to \{bread, milk\} \quad \text{(holds)}$$

Please note that the first three implications correspond to the rules in Example 2.9. The third implication does not hold because of T_4 (the respective rule has the confidence lower than 1.0).

	bread	*milk*	*butter*	*beer*	*snacks*
T_1	X	X	X		X
T_2				X	X
T_3	X			X	X
T_4	X	X	X		
T_5	X	X	X	X	X

Table 2.4: Formal context for sales transactions

A set of all implications that hold in the context is called *implication theory*. An implication theory can be large and can contain implications redundant with respect to others. Therefore, one is interested in its minimal subset which is sufficient to derive all other implications. Such subset is called *minimal implication base* [GW99]. Finding a minimal implication base for the given context is a common goal of FCA.

2.2.7 Probabilistic Graphical Models

Probabilistic Graphical Models (PGMs) are ML models where probabilistic dependencies between variables (features) are represented by a graph. There is a variety of PGMs in the ML literature. Here we briefly describe one of the

most popular PGMs, *Bayesian networks* [Pea88] (BNs). Some approaches try to combine structural/logical knowledge with PGMs, e.g. Relational Bayesian Networks [Jae97], Markov Logic Networks [RD06], Probabilistic Relation Models [GT07], Multi-Entity Bayesian Networks [Las08].

A BN $(\mathcal{G}, \mathcal{P})$ is a *generative* model, i.e. it represents a joint probability distribution $P(x_1, \ldots, x_n)$ of a set of variables. The *structure* $\mathcal{G} := (V, E)$ of a BN is a directed acyclic graph (DAG) where each node is a variable $x_i \in V$ and an edge represents a *direct* probabilistic dependence between variables. If two nodes are not directly connected, they are *conditionally independent*. Nodes $parents(x_i)$ which are connected to a given node x_i via incoming edges are called its *parents*, i.e. $parents(x_i) := \{x_j \mid \langle x_j, x_i \rangle \in E\}$.

Each node x_i in a DAG is assigned a conditional probability table (CPT) that determines the conditional probability of x_i given its parents, i.e. $P(x_i \mid par(x_i))$. CPTs are called *parameters* $\mathcal{P}$ of a BN. Figure 2.3 illustrates a BN for the kinship data from Table 2.2.

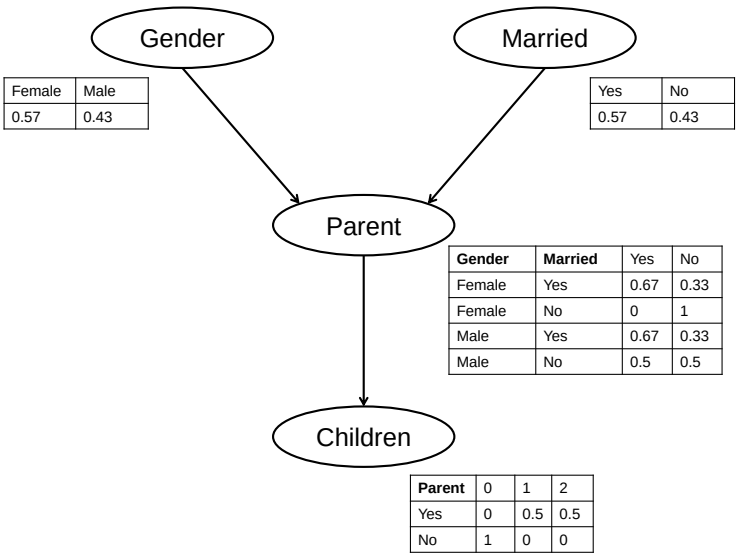


Figure 2.3: Bayesian network for Kinship

The BN in Figure 2.3 encodes that the variables *Married* and *Gender* directly influence the variable *Parent* which, in turn, directly influences the variable *Children*. As there are no links from *Married* and *Gender* to *Children*, the latter is *conditionally* independent from the former given *Parent*. In other words, *Married* and *Gender* can only influence *Children* through *Parent*.

The same probability distribution can be represented by different structures

(DAGs). Thus, the DAG of a BN should not be confused with a causal graph. Since the DAG of a BN encodes independence assumptions for (usually) many pairs of variables, it greatly simplifies computing conditional probabilities over variables. The joint probability distribution can be effectively decomposed as follows:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid par(x_i)).$$

Both parameters and structure of a BN can be learned from data. There are supervised and unsupervised approaches that vary in computational efficiency and quality of their results.

2.3 Summary

We have introduced the standard notions that we will use throughout the thesis. They should help to understand basic principles of an approach that we develop in this thesis.

In summary, DLs provide expressive syntax for encoding ontologies and form the basis of the commonly used OWL. Their formal semantics makes represented knowledge unambiguous and defines the entailment relation used for reasoning, i.e. deriving inferences (entailments). Importantly, the semantics of DLs is based on the OWA which permits incomplete information, see Section 2.1.5. Standard reasoning services are provided by optimised implementations called reasoners.

ML and DM provide methods for learning and mining patterns from data. In contrast to DLs, they usually assume that data is complete, see Section 2.2.2. The brief overview of standard ML/DM approaches, i.e. ILP, ARM, FCA, PGMs, should help to understand state-of-the-art approaches in Ontology Learning that we discuss in Chapter 3.

Chapter 3

Related Work in Ontology Learning

The term *"ontology learning"* was coined in 2001 by Mädche and Staab [MS01] for an emerging field of research concerned with the problem of automated generation of ontologies from data. While the task was already understood to be hard at the moment, it has turned out to be even more challenging and diverse.

Nowadays, Ontology Learning (OL) is commonly defined as a field that comprises techniques for automated acquisition of *ontological knowledge* from data. Thus, the paradigm has shifted such that many approaches do not aim at generating a full-fledged, gold-standard ontology from data anymore, but rather focus on acquiring axioms of certain shapes, e.g. concept definitions, atomic subsumptions, disjointness axioms, etc.

In this chapter we describe OL approaches of *primary interest for this thesis* and methodology they are based on. The interested reader is referred to the surveys in [VÖ9] and [LV14].

3.1 Ontology Learning Dimensions

OL is now a diverse and interdisciplinary field of research. Its diversity springs from the diversity of data sources and variety of ontological knowledge. Its interdisciplinarity emerges from applications of ontologies in multiple fields. Thus, one can observe the presence of *dimensions* that OL approaches can be located in. We distinguish the following OL dimensions, see also Figure 3.1.

Input data Ontologies are being learned from both unstructured data, e.g. text, and structured data, e.g. RDF datasets, DL ABoxes.

Target knowledge Ontological knowledge being learned ranges from taxonomies to concept definitions and simple forms of GCIs.

Semantics Approaches can use own semantics during learning which can differ from the standard DL semantics, e.g. apply the Closed World Assumption (CWA). Some of them ignore available background knowledge, i.e. the TBox, others consider it only to a limited extent.

Supervision The degree of human involvement in the learning process varies across approaches. Some approaches are supervised, i.e. require training examples, or interactive, i.e. ask questions to a domain expert during learning; others are unsupervised.

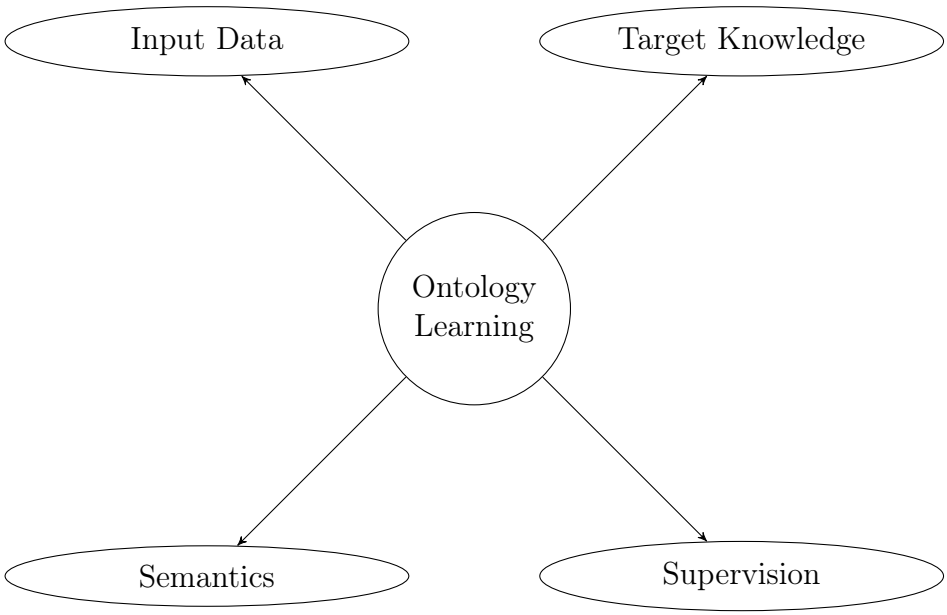


Figure 3.1: OL dimensions

The OL dimensions are important because they help to understand differences and commonalities between approaches. Firstly, they specify what *computational problem* an OL approach aims to solve, i.e. what data is used for learning, what knowledge is being learned. Secondly, they specify what *design choices* an OL approach makes, i.e. what semantics is used, what supervision (if any) is required. A problem coupled with design choices suggests what *methods* an approach can use to hit its targets and vice versa.

There is a common distinction of OL approaches by input data, i.e. structured and unstructured data. Learning ontologies from unstructured data, in particular

from texts and linguistic resources, was historically the first type of OL, called *lexical* OL [LV14].

The main limitation of lexical OL is inadequate quality of learned ontologies which is commonly seen as a result of inherent ambiguity of natural language. Therefore, there is growing interest in learning ontologies from structured data, called *logical* OL [LV14]. This thesis focuses on logical OL and the word "logical" is omitted in the following. In particular, we investigate the problem of acquiring a TBox from an ABox and hence review OL approaches concerned with this problem.

3.2 Related Ontology Learning Approaches

Most OL approaches try to adapt standard methods of ML and DM to complete their tasks, see Section 2.2. This is not surprising since OL is mainly the problem of learning from data. Therefore, ML and DM are reasonably expected to provide the basis for OL approaches. In the following, we describe main OL approaches relevant for this thesis and locate them in the OL dimensions. We cover Concept Description Learning, Knowledge Base Completion, Statistical Schema Induction, and BelNet, see Figure 3.2.

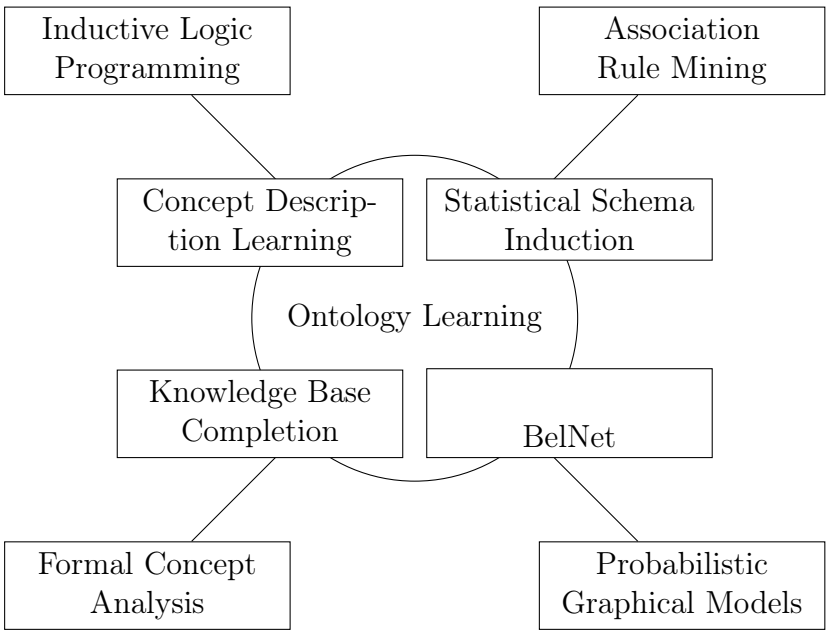


Figure 3.2: OL approaches

3.2.1 Concept Description Learning

Concept Description Learning [CH94, BST07, IPF07, FDE08, LH10] (CDL) is a popular approach in OL. The goal of CDL is to find a *description*, or *definition*, of a target concept name, given a DL ontology. A description is a concept expression (complex concept) which is induced from a set of examples which are instances (and non-instances) of a target concept name. Hence, CDL is essentially a type of *supervised* learning.

Most CDL approaches were inspired by inductive logic programming (ILP), see Section 2.2.4, and have successfully adapted ILP techniques for DLs starting from the DL CLASSIC [CH94]. The CDL problem is commonly defined as follows.

Definition 3.1 (CDL problem)**.** Let $\mathcal{O}$ be a DL ontology, $A \in N_C$ be a *target* concept name, $\mathcal{L}$ be a language bias, $E := E^+ \cup E^- \subseteq in(\mathcal{O})$ be a set of *examples* for A, where E^+ are *positive* and E^- are *negative* examples. Then, a *CDL learning problem* is to find a concept expression C, a *description* of A, such that C conforms to a language bias $\mathcal{L}$ and

$$\text{for all } e \in E^+ \;\; \mathcal{O} \models C(e) \text{ and for all } e \in E^- \;\; \mathcal{O} \not\models C(e).$$

Definition 3.1 is the ILP learning problem, see Definition 2.11, rephrased for DLs. Informally, the goal of CDL is to complete the axiom $A \equiv C$ (concept definition), where A is a fixed concept name and C is a variable concept expression, such that it satisfies all positive and no negative examples. As in ILP, see Section 2.2.4, this requirement can be relaxed to satisfying *almost* all positive and *almost* no negative examples. A hypothesis C must also conform to a language bias $\mathcal{L}$. For instance, a language bias $\mathcal{L}$ can be all $\mathcal{ALC}$ concepts of a maximal length ℓ, denoted as $\mathcal{ALC}(\ell)$. Thus, a language bias $\mathcal{L}$ determines a hypothesis space $\mathbb{H}$.

A CDL algorithm is driven by the same mechanics as an ILP one. A hypothesis space is traversed by *refinement operators*. Those, however, have to be designed specifically for DLs which is not straightforward [BNC00, FDE08, LH10]. A refinement operator can be *downward* and *upward*. A downward (upward) operator specifies a set of specialisations (generalisations) of a concept.

Definition 3.2 (Refinement operator)**.** A *quasi-ordering* is a reflexive and transitive relation. Let $(S, \preceq)$ be a quasi-ordered space, where S is a set of concepts

in $\mathcal{DL}$, $\preceq$ is a quasi-ordering. A *downward (upward) refinement operator* ρ is a mapping from S to 2^S, such that for all concepts $C \in S$ a concept $C' \in \rho(C)$ implies $C' \preceq C$ ($C \preceq C'$). A concept C' is called a *specialisation (generalisation)* of C, or *more specific (more general)* than C.

The DL subsumption $\sqsubseteq$ is normally used as a quasi-ordering. Thus, a concept C' is a specialisation (generalisation) of a concept C if C' is subsumed by (subsumes) C, i.e. $C' \sqsubseteq C$ ($C \sqsubseteq C'$).[1] Computational performance of a CDL algorithm critically depends on instance retrieval because it is done for a usually large number of potential solutions (concept descriptions). In order to illustrate CDL, we alter Example 2.8 for DLs, see Example 3.1.

Example 3.1. Consider the ontology $\mathcal{O}$ which is a subset of the Kinship ontology, see Example 2.3.

$$\begin{aligned}\mathcal{O} = \{&Man \sqsubseteq \neg Woman,\ hasParent \sqsubseteq hasChild^{-},\\ &Man(Arthur),\ Man(Chris),\ Man(James),\\ &Woman(Penelope),\ Woman(Victoria),\\ &Woman(Charlotte),\ Woman(Margaret),\\ &hasParent(Charlotte, James),\ hasParent(Charlotte, Victoria),\\ &hasParent(Victoria, Chris), hasParent(Victoria, Penelope)\\ &hasParent(Arthur, Penelope),\ hasParent(Arthur, Chris)\}.\end{aligned}$$

Assume the task is to learn a definition of the concept name $Mother$. In order to obtain negative examples automatically, CDL commonly makes the CWA. In this example we are forced to assume that all women which are unknown to be mothers, i.e. Charlotte and Margaret, are *not* mothers. The resulting sets of positive E^+ and negative E^- examples are as follows:

$$\begin{aligned}E^+ &= \{Penelope,\ Victoria\},\\ E^- &= \{Charlotte,\ Margaret,\ Arthur,\ Chris,\ James\}.\end{aligned}$$

Given the downward refinement operator ρ for $\mathcal{EL}$, a concept space is traversed as follows:

$\rho(\top) = \{Man,\ Woman,\ \exists hasChild.\top,\ \exists hasParent.\top\}$

[1] The statement $C' \sqsubseteq C$ is the abbreviation of $\models C' \sqsubseteq C$.

$\rho(Man) = \{Man \sqcap Woman,\ Man \sqcap \exists hasChild.\top,\ Man \sqcap \exists hasParent.\top\}$

$\rho(Woman) = \{Woman \sqcap Man,\ Woman \sqcap \exists hasChild.\top,$
$Woman \sqcap \exists hasParent.\top\}$

$\rho(\exists hasChild.\top) = \{Man \sqcap \exists hasChild.\top,\ Woman \sqcap \exists hasChild.\top,$
$\exists hasParent.\top \sqcap \exists hasChild.\top,\ \exists hasChild.Man,\ \exists hasChild.Woman\}$

...

If we consider the TBox, the concept $Man \sqcap Woman$ and all its further refinements can be skipped from the search since $\mathcal{O} \models Man \sqsubseteq \neg Woman$. Thus, the TBox can be used to optimise the search. The following concept is a possible solution because it satisfies all positive and no negative examples:

$$Woman \sqcap \exists hasChild.\top.$$

CDL approaches can learn concept descriptions of expressivity up to $\mathcal{ALC}$ (with some support for number restrictions and concrete domains, see [LH10] for details). One of commonly used (and cited) implementations is DL-Learner[2] [LH07, LABT11, BLW16].

ILP has been applied to learning Datalog [AHV95] rules which are added to a DL ontology thus creating a hybrid knowledge base [Lis08]. Yet, ILP is not the only method for CDL. One of early CDL approaches was based on a different method – computing the *least common subsumer* [BK98, BST07]. The method, however, is only applicable to weakly expressive DLs of the $\mathcal{EL}$ family. There is also an application of *terminological decision trees* [FdE10] to CDL where one of the challenges is the OWA of DLs. Although CDL approaches mainly focus on learning from DL ABoxes, there are also attempts to learn from Linked Data, i.e. RDF triples [CRH10, BL12].

Among limitations of CDL is that it requires suitable learning examples that can be hard and costly to prepare, as this requires domain expertise. In order to obtain training examples automatically, i.e. operate in the *unsupervised mode*, CDL applies a form of the CWA [LABT11]. More specifically, given an ontology $\mathcal{O}$ and a target concept name A, its positive examples are obtained from its instances, i.e. $e \in E^+$ if $\mathcal{O} \models A(e)$, and its negative examples are obtained from other individuals, i.e. $e \in E^-$ if $\mathcal{O} \not\models A(e)$. Usually $E^+ := \{e \in in(\mathcal{O}) \mid \mathcal{O} \models A(e)\}$ and $E^- := in(\mathcal{O}) \backslash E^+$. Clearly, such assumption

[2] `http://dl-learner.org`

is not always correct because some negative examples can be unknown instances of A (as in Example 2.3 we have assumed that Charlotte and Margaret are not mothers).

Another limitation is that CDL by definition is unable to learn general TBox axioms, i.e. complex GCIs $C \sqsubseteq D$, where C and D are both complex concepts ($C \notin N_C$ and $D \notin N_C$). A TBox is used, yet to a limited extent, to optimise the search. For example, DL-Learner checks the concept hierarchy and disjointness while refining a concept.

3.2.2 Statistical Schema Induction

Statistical Schema Induction [VN11] (SSI) is an OL approach based on Association Rules Mining (ARM), see Section 2.2.5. Hence, SSI is *unsupervised*. It focuses on learning from Linked Data, i.e. RDF triples. Similar approaches are described in [JT06, NL10, GTHS13].

As a reminder, association rules are mined from a set of transactions which can be represented as a table. Since RDF data is structurally a labelled graph, in order to apply ARM, it has to be transformed into a table. The general idea of such transformation is that RDF resources that correspond to data-level *objects* can be viewed as transactions, while RDF resources that correspond to schema-level *types* can be viewed as items. Although there is no distinction (neither syntactic nor semantic) between data and schema levels in RDF, under certain assumptions such distinction can be made for some datasets [VN11].

Extracted types are treated as DL concept (role) names. The set of types is further extended with complex concepts (and roles) of interest, e.g. with $\exists R.A$, where $A \in N_C$, $R \in N_R$. Objects which belong to a type, i.e. its instances, are identified via querying the RDF dataset. Importantly, as ARM requires complete data, the CWA is made. Consequently, if a query is not able to determine whether an object belongs to a type, then the object does not belong to that type. Once a transaction table is constructed, association rules are mined from it in a standard way, i.e. minimal thresholds for support and confidence are specified and an ARM algorithm returns a set of rules satisfying those.

The basic assumption (and idea) of SSI is that mined association rules straightforwardly correspond to DL axioms, see Assumption 3.1.

Assumption 3.1. Let $\mathbb{C}$ be a set of concepts of interest. An association rule $X \Rightarrow Y$, where $X, Y \subseteq \mathbb{C}$, is an axiom $C \sqsubseteq D$, where $C := \bigsqcap_{C' \in X} C'$, $D := \bigsqcap_{D' \in Y} D'$.

The validity of Assumption 3.1 clearly depends on minimal support and confidence thresholds imposed, see Section 2.2.5.

Different types of DL axioms lead to constructing different transaction tables. In [VN11] 6 transactions tables are constructed for 8 types of axioms, that all belong to $\mathcal{EL}^{++}$. All axioms that SSI can generate are as follows:

- $\bigsqcap_{C' \in X} C' \sqsubseteq \bigsqcap_{D' \in Y} D'$, where $X, Y \subseteq \mathbb{C}$ and $\mathbb{C} := \{A,\ \exists R.A,\ \exists R^{-}.A \mid A \in N_C\ \wedge\ R \in N_R\}$ (axioms with plain conjunctions of concepts of interest);

- $R \sqsubseteq S,\quad R \circ R \sqsubseteq R$, where $R, S \in N_R$ (atomic role subsumptions, role transitivity axioms).

There are also approaches, e.g. [VVSH07, FV11], that focus on learning disjointness axioms. Some use a standard ML classifier which is trained on manually tagged examples [VVSH07].

The ideas of SSI can be applied to learning ontologies from DL ABoxes. In order to transform an ABox to a transaction table, we view individuals as transactions and concepts of interest as items. Example 3.2 demonstrates the adaptation of SSI for learning from ABoxes.

Example 3.2. We use the Kinship ontology, see Example 2.3. Suppose we are interested in mining all associations involving concepts

$$\mathbb{C} := \{Man,\ Woman,\ Father,\ Mother,\ \exists hasChild.\top,\ \exists marriedTo.\top\}.$$

The constructed transaction table is shown in Table 3.1. Suppose that a rule is trustworthy if its quality exceeds thresholds $sup_{min} = 2$ and $conf_{min} = 0.6$. Then, the following association rules can be mined:

$$\begin{aligned}
\{Woman, \exists hasChild.\top\} &\Rightarrow \{Mother\} && (sup = 2,\ conf = 1.0)\\
\{Man, \exists hasChild.\top\} &\Rightarrow \{Father\} && (sup = 2,\ conf = 1.0)\\
\{\exists hasChild.\top\} &\Rightarrow \{\exists marriedTo.\top\} && (sup = 4,\ conf = 1.0)\\
\{Man\} &\Rightarrow \{\exists marriedTo.\top\} && (sup = 3,\ conf = 1.0)\\
\{\exists marriedTo.\top\} &\Rightarrow \{\exists hasChild.\top\} && (sup = 4,\ conf = 0.67).
\end{aligned}$$

These association rules are straightforwardly translated to DL axioms as follows:

$$Woman \sqcap \exists hasChild.\top \sqsubseteq Mother \quad (1)$$

$$Man \sqcap \exists hasChild.\top \sqsubseteq Father \quad (2)$$

$$\exists hasChild.\top \sqsubseteq \exists marriedTo.\top \quad (3)$$

$$Man \sqsubseteq \exists marriedTo.\top \quad (4)$$

$$\exists marriedTo.\top \sqsubseteq \exists hasChild.\top \quad (5)$$

Please note that the description of concept *Mother* is learned unintentionally (Axiom 1), as well as *Father* (Axiom 2). While these two axioms seem credible, others do not. This additionally shows that any acquired knowledge should be treated as a hypothesis.

	Man	$Woman$	$Father$	$Mother$	$\exists hasChild.\top$	$\exists marriedTo.\top$
$Charlotte$		X				
$James$	X		X		X	X
$Victoria$		X		X	X	X
$Chris$	X		X		X	X
$Penelope$		X		X	X	X
$Margaret$		X				X
$Arthur$	X					X

Table 3.1: Transaction table for Kinship

Among limitations of SSI and other ARM-based approaches is that they are forced to apply the CWA, i.e. for each individual a and concept C, if $\mathcal{O} \not\models C(a)$ and $\mathcal{O} \not\models \neg C(a)$, then it is assumed that $\mathcal{O} \models \neg C(a)$. Some approaches force some form of the CWA, e.g. [GTHS13] makes the partial completeness assumption which states that if one R-successor is known for an individual a, then all R-successors are known for a. Another limitation of ARM-based approaches is that they ignore the TBox while mining association rules. In other words, once transaction tables are constructed, association rules are mined from them disregarding the TBox. As a result, rules can contradict it, be superfluous with respect to it, etc.

Moreover, it is usually hard to come up with a good set of concepts of interest,

as this requires domain expertise. The brute-force generation of those can result in too many concepts. Due to the mechanics of ARM, resulting axioms can only contain conjunctions of initial concepts. In other words, new negations, disjunctions, and quantifiers are not generated. As a result, ARM-based approaches have so far been limited to produce axioms of weak expressivity.

3.2.3 Knowledge Base Completion

Knowledge Base Completion [Rud04, BGSS07, Rud08, Ser09, Dis10] (KBC) is an interactive OL approach based on Formal Concept Analysis, FCA, see Section 2.2.6. The goal is to complete an ontology in a certain, well-defined sense via interactions with a human expert who possesses domain knowledge. More specifically, an ontology is systematically updated by asking questions to an expert in order to formalise her knowledge. FCA is used to ensure that the number of questions that a domain expert is asked is kept minimal.

In order to apply FCA, we first need to define a formal context for an ontology. Intuitively, objects correspond to individuals, attributes correspond to concepts, and an incidence relation specifies which individuals are instances of which concepts. However, it is not as straightforward as that.

As discussed in Section 2.2.6, a formal context does not permit incomplete information. On the other hand, the semantics of DLs uses the OWA, see Section 2.1.5, i.e. it is possible that $\mathcal{O} \not\models C(a)$ and $\mathcal{O} \not\models \neg C(a)$. This problem was approached by defining so-called *partial contexts* [BGSS07] which permit incomplete information. More specifically, the incidence relation specifies attributes that an object certainly does *not* have in addition to attributes that it certainly has. The possession of remaining attributes is interpreted to be unknown.

In contrast to a formal context in FCA, it can be unknown whether an implication holds in a partial context or it does not. Such an implication is called *undecided* (and otherwise *decided*). In the case when an implication certainly does not hold in a partial context, it is said to be *refuted* by it. In other words, there is an object that certainly does not have an attribute forced by the implication.

The aforementioned notions are straightforwardly transferred to DLs. A partial context for an ontology $\mathcal{O}$ is a structure $\mathcal{K}_\mathcal{O} := (G, M, I)$, where objects are individuals, i.e. $G = in(\mathcal{O})$, attributes are concepts $\mathbb{C}$ of interest, i.e. $M = \mathbb{C}$, and the incidence relation I specifies which attributes an object certainly has and which ones it certainly does not have. Similarly to ARM, implications correspond

to axioms as follows, see Assumption 3.2.

Assumption 3.2. Let $\mathbb{C}$ be a set of concepts of interest. An implication $X \to Y$, where $X, Y \subseteq \mathbb{C}$, is an axiom $C \sqsubseteq D$, where $C := \bigsqcap_{C' \in X} C'$, $D := \bigsqcap_{D' \in Y} D'$.

Given an ontology $\mathcal{O}$, an individual $a \in in(\mathcal{O})$ is called a *counterexample* for an axiom $C \sqsubseteq D$ if $\mathcal{O} \models C(a)$ and $\mathcal{O} \models \neg D(a)$. If an axiom has a counterexample, then the corresponding implication is refuted.

A KBC algorithm operates as follows. For each undecided implication $X \to Y$ a domain expert is asked a question: "Is $X \to Y$ refuted?" If the answer is "no", then the implication is added to the computed implication base and the corresponding axiom is added to the ontology. If the answer is "yes", then the expert should extend the ontology such that the implication is refuted, i.e. provide a counterexample, e.g. add the $\neg D(a)$ if $\mathcal{O} \models C(a)$. Hence, not only the TBox but also the ABox is extended. Once all implications are decided, an ontology is said to be *complete*, see Definition 3.3, and the algorithm terminates.

Definition 3.3. Let $\mathcal{K}_\mathcal{O}$ be a partial context of an ontology $\mathcal{O}$. An ontology $\mathcal{O}$ is $\mathbb{C}$-*complete* if for any $X, Y \subseteq \mathbb{C}$ either

- $X \to Y$ holds in $\mathcal{K}_\mathcal{O}$ or
- $X \to Y$ is refuted by $\mathcal{K}_\mathcal{O}$.

Informally, Definition 3.3 means that an ontology is complete if any implication is either holds in its context or refuted by it. It is possible to rewrite Definition 3.3 in a more convenient form that uses axioms of the ontology instead of implications of its context, see Lemma 3.1.

Lemma 3.1. *An ontology $\mathcal{O}$ is* $\mathbb{C}$-complete *if for any $X, Y \subseteq \mathbb{C}$ either*

- $\mathcal{O} \models C \sqsubseteq D$ *or*
- $C \sqsubseteq D$ *has a counterexample in $\mathcal{O}$,*

where $C := \bigsqcap_{C' \in X} C'$, $D := \bigsqcap_{D' \in Y} D'$.

Lemma 3.1 means that an ontology is complete when any axiom involving conjunctions of concepts of interest either follows from the ontology or has a counterexample in it. Thus, all axioms that KBC can generate are as follows:

- $\bigsqcap_{C' \in X} C' \sqsubseteq \bigsqcap_{D' \in Y} D'$, where $X, Y \subseteq \mathbb{C}$.

Example 3.3 illustrates the basic notions of KBC.

Example 3.3. We use the Kinship ontology, see Example 2.3. Suppose the task is to complete the ontology with respect to the following concepts:

$$\mathbb{C} := \{Man,\ Woman,\ Father,\ Mother,\ \exists hasChild.\top,\ \exists marriedTo.\top\}.$$

The partial context is shown by Table 3.2. Please notice that it is incomplete, i.e. there are question marks "?" when it is unknown whether an individual is an instance of a concept (indicated by "X") or it is not (indicated by a blank space).

The following axioms are examples of those whose implications are undecided in the context:

$$Woman \sqcap \exists hasChild.\top \sqsubseteq Mother \quad (1)$$

$$Man \sqcap \exists hasChild.\top \sqsubseteq Father \quad (2)$$

$$Woman \sqsubseteq \exists marriedTo.\top \quad (3)$$

$$Man \sqsubseteq \exists marriedTo.\top \quad (4)$$

Suppose the implications of these are suggested to the domain expert in the order of appearance. She accepts Axiom 1 and then Axiom 2 which are added to the ontology in turn. After that, she rejects Axiom 3 and is asked to supply a counterexample. She does this via adding axiom $(\neg\exists marriedTo.\top)(Charlotte)$, i.e. the question mark is turned into the blank space (negated class assertion). Then, the domain expert rejects Axiom 4 and again is asked to provide a counterexample: she introduces a new individual a and adds the axioms $Man(a)$ and $(\neg\exists marriedTo.\top)(a)$.

Although KBC ensures a minimal number of questions by the means of FCA, a domain expert can still be overwhelmed because that minimal number can be large. To be more specific, every question asks to judge some implication. If an implication is refuted, an expert must provide a counterexample for it. Such interactive process can be tedious and burdensome, particularly if the input ontology is complex.

Another difficulty is that specifying relevant concepts $\mathbb{C}$ of interest is hard and requires domain knowledge. The approach in [Rud04, Rud08] approaches this problem via *relational exploration*. It iteratively generates additional concepts (attributes) when they are required by the KBC algorithm. All generated

	Man	*Woman*	*Father*	*Mother*	$\exists hasChild.\top$	$\exists marriedTo.\top$
Charlotte		X		?	?	?
James	X		X		X	X
Victoria		X		X	X	X
Chris	X		X		X	X
Penelope		X		X	X	X
Margaret		X		?	?	X
Arthur	X		?		?	X

Table 3.2: Partial context for Kinship

concepts are from $\mathcal{FLE}$[3] with a maximal role depth δ, denoted as $\mathcal{FLE}(\delta)$. Due to the mechanics of FCA, GCIs are only constructed from those concepts using conjunctions. Thus, resulting axioms are all from $\mathcal{FLE}(\delta)$, i.e. weakly expressive.

There are also approaches to KBC that are not based on FCA. The approach in [KLOW14] is based on Angluin's framework of exact learning via queries [Ang88]. An ontology is learned not from data, but from an oracle, e.g. a human expert, which possesses domain knowledge. An oracle has the "target TBox in mind" and is able to answer queries in order to formalise its knowledge. Queries are of two types: "Is an axiom entailed by the target TBox?" and "Is a TBox equivalent to the target TBox?". The approach considers weakly expressive DLs, i.e. $\mathcal{EL}$ and DL-lite.

3.2.4 BelNet

Bayesian Description Logic Network, or BelNet [ZGP+13, ZGP+15], is an OL approach based on Bayesian Networks (BNs), see Section 2.2.7. BelNet combines DLs and BNs. The basic assumption (and idea) of BelNet is that a TBox can be transformed to the structure (DAG) of a BN and vice versa. More specifically, BelNet is defined as follows, see Definition 3.4.

Definition 3.4 (BelNet)**.** Let $\mathcal{O}$ be an ontology and $\mathbb{C}$ a set of concepts of

[3] $\mathcal{FLE}$ only allows for $\sqcap$, $\exists$, $\forall$.

interest. Let $eq(C, \mathcal{O}) := \{C' \in \mathbb{C} \mid \mathcal{O} \models C \equiv C'\}$. Let

$$parents(C, \mathcal{O}, \mathbb{C}) := \{C' \in \mathbb{C} \mid \mathcal{O} \models C' \sqsubseteq C \ \wedge \ \nexists C'' \in \mathbb{C} \ : \\ \mathcal{O} \models C' \sqsubseteq C'' \ \wedge \ \mathcal{O} \models C'' \sqsubseteq C\}.$$

Then, the BelNet of $\mathcal{O}$ is a BN $(\mathcal{G}, \mathcal{P})$, where structure $\mathcal{G} := (V, E)$ is such that

$$V := \{eq(C, \mathcal{O}) \mid C \in \mathbb{C}\},$$
$$E := \{\langle eq(C', \mathcal{O}), eq(C, \mathcal{O})\rangle \mid C' \in parents(C, \mathcal{O}, \mathbb{C})\}.$$

Parameters $\mathcal{P}$ are CPTs over variables V.

Informally, the correspondence between an ontology and a DAG is based on the following assumptions:

- each equivalence-representative concept corresponds to a binary variable;
- subsumptions between concepts correspond to probabilistic dependencies between their variables;
- equivalent concepts are represented by a single node.

BelNet is learned from the ABox. Since BNs use the standard ML view of data, see Section 2.2.2, so does BelNet. All individuals are assumed to be independent and different from each other. The ABox is viewed under the CWA, i.e. for a concept C its probability in an ontology $\mathcal{O}$ is estimated as follows:

$$P(C = true) \ := \ \frac{|\{a \in in(\mathcal{O}) \mid \mathcal{O} \models C(a)\}|}{|in(\mathcal{O})|},$$
$$P(C = false) \ := \ 1 - P(C = true).$$

Once BelNet is learned, it is translated to a set of DL axioms as follows. For each link from concept C to concept D in the DAG the axiom $C \sqsubseteq D$ is added. Additionally, disjointness axioms are extracted by means of probabilistic inference in the underlying BN. More specifically, for any two concepts $C, D \in \mathbb{C}$ if $P(C = true, \ D = true) < t_{disj}$, i.e. their joint probability does not exceed a user-defined threshold, then the axiom $C \sqcap D \sqsubseteq \bot$ is added. Thus, all axioms that BelNet generates are as follows:

- $C \sqsubseteq D$, $C \sqcap D \sqsubseteq \bot$, where $C, D \in \mathbb{C}$ and $\mathbb{C} := \{A,\ \exists R.\top \mid A \in N_C \ \wedge\ R \in N_R\}$.

Please note that additional axioms involving conjunctions of concepts from $\mathbb{C}$ can be extracted using probabilistic inference similarly to disjointness axioms. However, this requires running many probabilistic queries and specifying additional thresholds that can be difficult to choose. Example 3.4 illustrates BelNet.

Example 3.4. We use the Kinship ontology, see Example 2.3. Suppose we are interested in finding all dependences involving concepts

$$\mathbb{C} := \{Man,\ Woman,\ Father,\ Mother,\ \exists hasChild.\top,\ \exists marriedTo.\top\}.$$

The data table is identical to Table 3.1. A possible BelNet learned from Table 3.1 is shown by Figure 3.3. Its structure represents the following DL axioms:

$$Father \sqsubseteq Man \tag{1}$$
$$Mother \sqsubseteq Woman \tag{2}$$
$$Father \sqsubseteq \exists hasChild.\top \tag{3}$$
$$Mother \sqsubseteq \exists hasChild.\top \tag{4}$$
$$\exists hasChild.\top \sqsubseteq \exists marriedTo.\top \tag{5}$$

Please notice that Axiom 1 and 2 are already in the TBox, hence, they are uninformative. Additionally, for any $t_{disj} > 0$ the following disjointness axioms are extracted: $Man \sqcap Woman \sqsubseteq \bot$, $Father \sqcap Mother \sqsubseteq \bot$, $Man \sqcap Mother \sqsubseteq \bot$, $Woman \sqcap Father \sqsubseteq \bot$. Please notice that all of these are entailed by the TBox and, therefore, uninformative.

Among the limitations of BelNet is that its core assumptions seem to be too strong. Firstly, the correspondence of probabilistic dependencies $C \to D$ to subsumption axioms $C \sqsubseteq D$ is not obvious. For example, the link $C \to D$ can also correspond to the disjointness axiom $C \sqsubseteq \neg D$ (links for disjointness are, in fact, introduced in [ZGP$^+$15]). Secondly, the ABox is viewed under the CWA. Since only subsumption and disjointness axioms for concepts of interest are learned, resulting ontologies are of weak expressivity.

The aforementioned assumptions are coupled with some shortcomings in the results of learning. For example, equivalence axioms $C \equiv D$ cannot be learned,

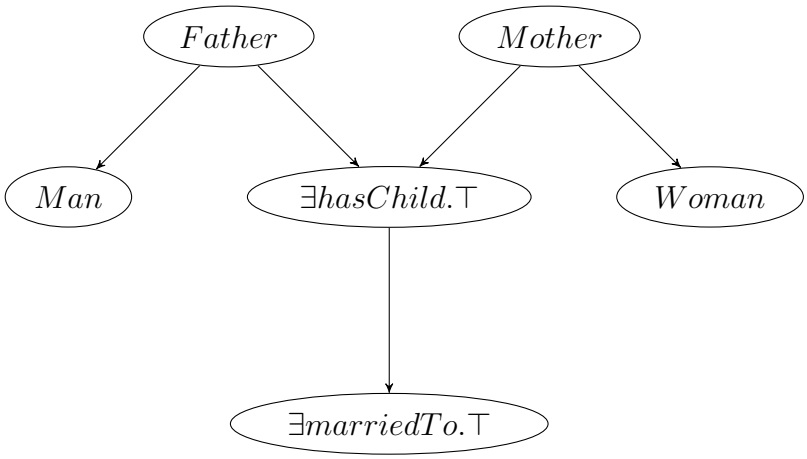


Figure 3.3: BelNet for Kinship

despite the fact that the formalism itself can represent equivalences via representative nodes. In addition, there is a little appreciation of the TBox (only consistency is checked) which results in many uninformative axioms. Finally, a BN learned from the ABox represents a single, "optimal" solution which may not be true considering the problem of learning from data. To be more specific, since there is no guarantee that the data faithfully represents the domain knowledge, any part of the learned ontology can simply be the result of data idiosyncrasies, e.g. noise or bias, see Example 3.2. Another example of applying probabilistic graphical models to OL is [DLK$^+$08].

3.3 Summary

We have briefly reviewed related work in OL focusing on learning from structured data (logical OL). To summarise, we place each approach in the OL dimensions, see Table 3.3.

What have we learned? All approaches try to employ standard methods from ML and DM, i.e. ILP, ARM, FCA, and PGMs. However, adapting ML and DM methods also results in limitations of OL approaches. This happens because of significant differences between ML/DM and DLs, see Chapter 2. Firstly, they differ in data representation: ML and DM use the tabular view under the CWA, while DLs use the labelled graph under the OWA. Secondly, they differ in knowledge representation: ML and DM use rules and networks, while DLs use axioms. Although LP, a language for hypotheses in ILP, uses logical formula, it differs

	Input Data	**Target Knowledge**	**Semantics**	**Supervision**
CDL	ABox, TBox, target A	$A \equiv C$, where $A \in N_C$, C is a concept from $\mathcal{ALC}(\ell)$	OWA, partial consideration of TBox	supervised: positive and negative examples; unsupervised: CWA
SSI	RDF	$\bigsqcap_{C' \in X} C' \sqsubseteq \bigsqcap_{D' \in Y} D'$, where $X, Y \subseteq \mathbb{C}$ and $\mathbb{C} := \{A,\ \exists R.A,\ \exists R^-.A \mid A \in N_C \wedge R \in N_R\}$; $R \sqsubseteq S$, $R \circ R \sqsubseteq R$, where $R, S \in N_R$	CWA, no TBox	unsupervised
KBC	ABox, TBox	$\bigsqcap_{C' \in X} C' \sqsubseteq \bigsqcap_{D' \in Y} D'$, where X, Y are sets of concepts from $\mathcal{FLE}(\delta)$	OWA, full consideration of TBox	supervised: interactive learning where a domain expert verifies axioms or provides counterexamples
BelNet	ABox, TBox	$C \sqsubseteq D$, $C \sqcap D \sqsubseteq \bot$, where $C, D \in \mathbb{C}$ and $\mathbb{C} := \{A,\ \exists R.\top \mid A \in N_C \wedge R \in N_R\}$	CWA, mostly disregards TBox	unsupervised

Table 3.3: OL approaches in OL dimensions

from DLs in expressivity and semantics, see Section 2.1.7. As a consequence, OL approaches have the following common limitations.

(i) **Weak expressivity of acquired knowledge** OL approaches can only learn axioms of restricted shapes, i.e. concept definitions or axioms with plain conjunctions of concepts of interest. Good concepts of interest are hard to specify, as this requires domain expertise. Automated generation of concepts of interest is essentially brute-force and, hence, can result in too many concepts. As a consequence, approaches can only produce weakly expressive axioms.

(ii) **Disregard for the standard semantics** An ABox is represented in the form of a binary table and viewed under the CWA (or some form of the CWA is used). A TBox is either respected to a limited extent or ignored. Any logical interaction between learned axioms is usually ignored.

(iii) **High degree of human involvement** Supervised OL approaches require specifying training examples. Interactive OL approaches require the intense involvement of a domain expert that answers questions and provides counterexamples.

This thesis addresses all of these challenges. In contrast to other OL approaches, we do not attempt to adapt a method from ML or DM for OL, but design an approach specifically for DLs to overcome the aforementioned limitations. While the approach is guided by basic principles of ML and DM, it is aimed at learning in the realm of DLs. This calls for introducing a general computational problem for OL and making suitable design choices.

Chapter 4

General Terminology Induction

In this chapter, we discuss the problem of Ontology Learning (OL) from general principles. In particular, we discuss what is an output of OL, how much should we trust it, and how to evaluate its quality. As a result of that discussion, we introduce a new computational problem for OL, called *General Terminology Induction*. We discuss how to design a reasonable approach and outline the architecture of its implementation.

4.1 A New Look at Ontology Learning

This thesis investigates the problem of acquiring TBox axioms (generalisations) from an ABox (data). TBoxes can be of different shape and expressivity. For example, a TBox can be a simple concept hierarchy. Yet, it can also be a sophisticated terminology with complex conceptual relationships encoded in expressive DLs, such as $\mathcal{SROIQ}$. Then, the following questions arise:

- How should an acquired TBox look like in general?
- How can we know whether it is of good or bad quality?

These questions are discussed below.

4.1.1 Thinking in Terms of Hypotheses

Suppose we can evaluate the *quality* of an acquired TBox via some automated means, e.g. measuring how well it is inductively supported by the data (as OL approaches normally do). Then, given that an acquired TBox has the best possible

quality (perfect score), can we take it as an ultimate solution and stop searching?

We argue that the answer is clearly "no". The reason is that OL is essentially a problem of learning from data and data does not guarantee to be domain-representative in general. Therefore, an acquired output, even with a perfect score, can simply be the result of *data idiosyncrasies*, e.g. noise or bias, see Example 3.2. Those idiosyncrasies can lead away from the true conceptualisation of the domain which is the primary purpose of any ontology (and its TBox).

Therefore, we treat any result of OL as a *hypothesis* regardless of its measured quality. As discussed in Chapter 3, OL approaches learn axioms of restricted shapes, see Table 3.3: concept definitions (CDL), axioms with plain conjunctions (SSI and KBC), simple subsumption and disjointness axioms (BelNet). However, in the most general sense, a hypothesis is a TBox,[1] i.e. a set of general concept inclusions (GCIs) and role inclusions (RIs), see Definition 4.1.

Definition 4.1 (Hypothesis)**.** A set H of axioms is called a *hypothesis* if H is a TBox.

In contrast to other OL approaches, see Section 3.2, we argue that it is impossible to decide automatically whether a hypothesis is a solution or it is not. All we can do is to order the set of all hypotheses, or *hypothesis space*, by their measured quality and hope that such ordering resembles their true quality ordering. The verdict whether a hypothesis is valid, interesting, or useful should be left for human experts. An expert can use the ordering of hypotheses to navigate through them.

4.1.2 Hypothesis Quality Dimensions

Let us discuss which quality we can use to compare hypotheses in a reasonable way. Clearly, quality should evaluate how well the data supports a hypothesis. We call it *statistical quality*. Is this quality sufficient to compare hypotheses? While it may be sufficient for standard ML and DM problems, we think it is *not* sufficient for OL in DLs. As discussed in Section 2.1, DLs have a formal semantics. Hence, hypotheses in DLs can have different *logical quality*, e.g. they can be consistent or inconsistent with the ontology, informative or uninformative to it. In addition, a hypothesis should be readable since it is intended to be examined by humans. Hence, the third, human-centric quality is *readability* which evaluates how easily

[1] There are approaches that learn an ABox from text resources.

a hypothesis can be read by humans, e.g. how long it is. Thus, we can talk about different *quality dimensions*:[2]

- statistical quality
- logical quality
- readability

While we think these are essential quality dimensions of a hypothesis in OL, we do not claim that the list is exhaustive. There may be other dimensions, e.g. application-specific ones. In the context of this thesis, quality dimensions are rather abstract notions that define criteria which hypotheses can be compared by. The actual comparison requires measuring quality along its dimensions in some way, i.e. *quality measures* are required. Each quality dimension can be evaluated by multiple quality measures.

4.1.3 Ontology Learning as a Multi-Objective Search

In ML and DM the process of learning is often considered as a search in the hypothesis space. A learning algorithm normally searches for a hypothesis that maximises a single objective. Thus, the result of a single-objective search is usually one, "best" hypothesis (unless multiple hypotheses share the best value).

Since OL seems to require evaluating a hypothesis along multiple quality dimensions, we consider the process of learning as a *multi-objective search*. In contrast to a single-objective search, the result of a multi-objective search is usually many "best" hypotheses because one hypothesis can be maximal on one objective, while another hypothesis can be maximal on another objective.

Moreover, quality measures only estimate the actual quality of a hypothesis. Therefore, instead of returning only best hypotheses and ignoring all others, we suggest ordering all hypotheses by their quality values. Such ordering allows for navigating through hypotheses starting from most promising ones. Of course, hypotheses of high quality can still be selected using their quality values.

In the light of the above considerations, we introduce a new computational problem for OL, called *General Terminology Induction* (GTI), see Definition 4.2.

[2]Hypothesis quality dimensions should not be confused with OL dimensions, see Section 3.1.

Definition 4.2 (General Terminology Induction)**.** Given an ontology $\mathcal{O}$, a language bias $\mathcal{L}$, and a set of quality measures Q, *General Terminology Induction* is the problem of

- constructing hypotheses $\mathbb{H}$ for $\mathcal{O}$ that conform to $\mathcal{L}$;
- evaluating $\mathbb{H}$ by Q; and
- computing an ordering $(\mathbb{H}, \prec^{\mathcal{O},Q})$, where $\prec^{\mathcal{O},Q}$ is a binary relation on $\mathbb{H}$.

Thus, the goal of GTI is to construct, evaluate, and order (rank) hypotheses for an ontology $\mathcal{O}$. In other words, GTI consists of three sub-problems: hypothesis construction, hypothesis evaluation, and hypothesis ranking. Please note that GTI is rather underspecified since only an input ontology $\mathcal{O}$ is completely clear. A language bias $\mathcal{L}$ and quality measures Q are rather abstract at that point. In the following, we will make these notions precise and suggest how they can be specified.

Let us discuss relationships between GTI and other OL approaches described in Chapter 3. In fact, if we consider each approach as a computational problem, each of those problems can be seen as a version of GTI as follows. The language bias for each problem is given by the column "Target Knowledge" in Table 3.3. The set of quality measures consists of a single, binary measure q such that $q = 1$ if a hypothesis satisfies the requirements, e.g. minimal thresholds, and $q = 0$ otherwise. The quality measure q can be based on supervision, e.g. positive and negative examples. Then, the goal of each computational problem is to rank all hypotheses by q and select top ones which are treated as solutions.

Thus, GTI is more general than existing computational problems in OL in two aspects. Firstly, GTI permits a hypothesis to be a general TBox, i.e. unrestricted. Secondly, GTI considers multiple competing quality measures. We argue that each OL approach can, in principle, consider multiple quality measures (including measures defined in this thesis) and improve its results since multiple quality measures provide additional information about hypothesis quality. Such information can be used to better compare hypotheses and identify most promising ones.

4.2 Designing DL-Miner

The generality of GTI suggests that there are many ways to approach the problem. An approach can make various design choices and assumptions. In this section, we design an approach for GTI. We discuss its general assumptions and outline the architecture of its implementation.

4.2.1 Design Choices

In Section 3.3, we have discussed common limitations of OL approaches. We argue that these limitations can be (at least partly) overcome by a carefully designed approach. We call its implementation DL-Miner, as it aims at mining hypotheses in DLs. We make the following *design choices*:

(i) A language bias $\mathcal{L}$ permits GCIs $C \sqsubseteq D$ and RIs $R \sqsubseteq S$, where $C, D \in \mathbb{C}$, $R, S \in \mathbb{R}$, $\mathbb{C}$ and $\mathbb{R}$ are finite sets of possibly complex concepts and roles, respectively.

(ii) The standard DL semantics is respected, i.e. the OWA is permitted and the TBox is fully taken into account during learning.

(iii) Learning is unsupervised in the sense that human experts only intervene at the final stage to examine the results of learning (though they may need to specify input parameters). Preparing training examples or providing counterexamples is not required.

Thus, DL-Miner aims at acquiring more expressive hypotheses, respecting the standard semantics of DLs, and unsupervised learning, simultaneously. This is a challenging design which has never been pursued by any other OL approach. Firstly, the hypothesis space becomes significantly larger, and, hence, good hypotheses are harder to find. Secondly, the search is not guided by supervision. Finally, the problem is not simplified by disregarding the semantics and hypothesis evaluation is likely to require reasoning that can be costly.

4.2.2 General Assumptions

In addition to the above design choices, we make the following *assumptions*.

(i) The input ontology $\mathcal{O}$ belongs to a DL for which the problems of subsumption checking and instance checking are decidable, i.e. any popular DL such as $\mathcal{EL}$, $\mathcal{ALC}$, $\mathcal{SHIQ}$, $\mathcal{SROIQ}$.

(ii) The input ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ contains a non-empty ABox $\mathcal{A}$, i.e. $\mathcal{A} \neq \emptyset$.

(iii) The input ontology $\mathcal{O}$ is consistent.

The assumption (i) ensures that we can use reasoning since reasoning services are guaranteed to terminate and implemented by reasoners. As we focus on learning from an ABox, the assumption (ii) guarantees that it is non-empty, i.e. there is some data to learn from. As we aim at respecting the semantics and, hence, using reasoning, the assumption (iii) guarantees that reasoning is sensible. If an ontology is inconsistent, reasoning is useless because everything follows from the ontology. While the assumptions (i) and (ii) are firm in the scope of this thesis, the assumption (iii) is not: we show how to cope with inconsistent inputs in the following.

4.2.3 Architecture of DL-Miner

Having made the design choices and general assumptions, we now outline the high-level architecture of DL-Miner. First, let us describe its input parameters. There is one mandatory input parameter, i.e. an ontology $\mathcal{O}$, and three optional parameters, i.e. a language bias $\mathcal{L}$, a signature Σ of interest (a set of focus terms), and a set Q of quality measures. The architecture of DL-Miner consists of the following functional blocks, see Figure 4.1.

- **Ontology Cleaner** repairs an input ontology $\mathcal{O}$ from inconsistency. The result is a consistent ontology $\mathcal{O}'$.
- **Hypothesis Constructor**, given an optional language bias $\mathcal{L}$ and signature Σ of interest, constructs suitable concepts $\mathbb{C}$ and roles $\mathbb{R}$ from Σ conforming to $\mathcal{L}$. Then, hypotheses $\mathbb{H}$ are generated from $\mathbb{C}$ and $\mathbb{R}$.
- **Hypothesis Evaluator**, given an optional set Q of quality measures and a consistent ontology $\mathcal{O}'$, evaluates hypotheses $\mathbb{H}$. The result is the quality function $qf(H, q)$ that returns the quality value for a hypothesis $H \in \mathbb{H}$ given a quality measure $q \in Q$.

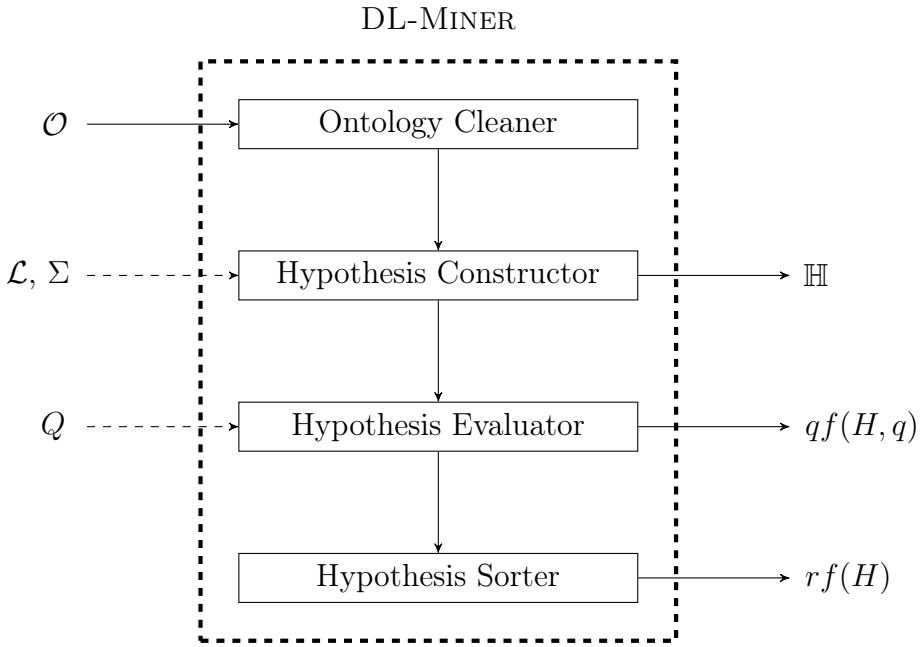


Figure 4.1: Architecture of DL-Miner

- **Hypothesis Sorter**, given the quality function $qf(\cdot)$,[3] orders hypotheses $\mathbb{H}$ according to the binary relation $\prec$.[4] The result is the ranking function $rf(H)$ that returns the quality rank of a hypothesis $H \in \mathbb{H}$.

The output of DL-Miner is a set $\mathbb{H}$ of hypotheses, quality function $qf(\cdot)$, and ranking function $rf(\cdot)$. Domain experts and ontology engineers are supposed to navigate through the hypotheses using the quality and ranking functions. Thus, all hypotheses can be methodically examined. Clearly, it is possible to select only best hypotheses if necessary. As the reader will find in the following, hypotheses of DL-Miner can, in fact, be used for various purposes and in different scenarios.

In the following, we clarify the parameters and unfold the functionality of each block. Hypothesis Evaluator is covered in Chapter 5, where we define quality measures that can be used in Q, and Chapter 6, where we develop techniques to compute those measures. Hypothesis Constructor is explained in Chapter 7 where we show how to construct suitable concepts $\mathbb{C}$ (roles $\mathbb{R}$) given a language bias $\mathcal{L}$ and generate hypotheses $\mathbb{H}$ from $\mathbb{C}$ ($\mathbb{R}$). Ontology Cleaner and Hypothesis Sorter are both covered in Chapter 8 where we also integrate all techniques in DL-Miner. Finally, we empirically evaluate DL-Miner in Chapter 9.

[3]The symbol "$\cdot$" stands for the arguments of the function if they are clear or irrelevant.
[4]When $\mathcal{O}$ and Q are clear from the context, we denote the binary relation $\prec^{\mathcal{O},Q}$ by $\prec$.

Chapter 5

Defining Hypothesis Quality Measures

As discussed in Chapter 4, Ontology Learning (OL) can be seen as a search in the space of hypotheses. While enumerating hypotheses, we need to test their quality. In the context of DLs, OL seemingly requires evaluating multiple quality dimensions of a hypothesis. We have introduced three quality dimensions: readability, logical quality, and statistical quality.

In this chapter, we address the question how to evaluate hypothesis quality in a reasonable way. We introduce several quality measures for each quality dimension. Those quality measures can be used as input parameters for DL-Miner, see Figure 4.1. We investigate formal properties of quality measures and establish relationships between them.

5.1 Readability of a Hypothesis

DLs can encode complex knowledge. Not only can axioms involve multiple terms plainly connected by logical operators, but they can also include multi-level nestings due to role quantifiers, thus forming tree-like structures. This affects computational complexity, i.e. performance of reasoning algorithms, and cognitive complexity, i.e. understandability of ontologies by humans. In addition to syntactic complications, humans can also experience semantic complications, e.g. when comprehending entailments of the ontology. Some questions of cognitive complexity of DLs are discussed in [HBPS11, NS13, WMCM14].

As a hypothesis in OL consists of axioms, we can talk about its cognitive complexity in the standard sense. In this thesis, we are concerned with a particular aspect of cognitive complexity – readability. *Readability* of a hypothesis is the ease with which it can be *read* by a human. The process of reading is considered to be tightly connected with parsing. Thus, in the scope of this thesis, readability is concerned with the syntactic side of cognitive complexity.

5.1.1 Syntactic Length

A possible measure of readability of a hypothesis is its *syntactic length*, Informally, it counts how many terms and logical operators are used to write a hypothesis. We first define the syntactic length of an axiom. This is the common measure, e.g. used in [HPS08, NS13], which we straightforwardly extend with additional types of axioms, e.g. role chains $S \circ P \sqsubseteq R$, see Definition 5.1.

Definition 5.1 (Syntactic Length of an Axiom)**.** Let $A \in N_C$ be a concept name, C, D complex concepts, $R \in N_R$ a role name, S, P complex roles, $o \in N_I$ an individual name. The *syntactic length of a role* is defined as follows:

$$\ell(R) := 1,\ \ell(S^-) := 1 + \ell(S),$$
$$\ell(S \circ P) := 1 + \ell(S) + \ell(P).$$

The *syntactic length of a concept* is defined as follows:

$$\ell(\top) = \ell(\bot) = \ell(A) = \ell(\{o\}) := 1,$$
$$\ell(\neg C) := 1 + \ell(C),$$
$$\ell(C \sqcap D) = \ell(C \sqcup D) := 1 + \ell(C) + \ell(D),$$
$$\ell(\exists S.C) = \ell(\forall S.C) = \ell(\geq nS.C) = \ell(\leq nS.C) := \ell(S) + \ell(C).$$

The *syntactic length of an axiom* is defined as follows: $\ell(C \sqsubseteq D) := \ell(C) + \ell(D)$, $\ell(S \sqsubseteq P) := \ell(S) + \ell(P)$, $\ell(C(a)) := \ell(C)$, $\ell(R(a, b)) := \ell(R)$, $\ell(a \approx b) = \ell(a \not\approx b) := 1$.

The syntactic length of a hypothesis is simply the summary length of its axioms, see Definition 5.2.

Definition 5.2 (Syntactic Length of a Hypothesis)**.** The *syntactic length* of a hypothesis H is defined as follows:

$$\ell(H) := \sum_{\alpha \in H} \ell(\alpha).$$

We sometimes omit "syntactic" for the sake of brevity when we talk about syntactic length. Example 5.1 shows how length is calculated.

Example 5.1. Consider the axiom $\alpha := A \sqcap \exists R.B \sqsubseteq C \sqcup \forall R.(\neg D \sqcap \exists S.B)$, where A, B, C, D are concept names. Its length is calculated as follows:

$$\begin{aligned}
\ell(\alpha) &= \ell(A \sqcap \exists R.B) + \ell(C \sqcup \forall R.(\neg D \sqcap \exists S.B)) \\
&= \ell(A) + 1 + \ell(\exists R.B) + \ell(C) + 1 + \ell(\forall R.(\neg D \sqcap \exists S.B)) \\
&= \ell(A) + 1 + \ell(B) + 1 + \ell(C) + 1 + 1 + \ell(\neg D \sqcap \exists S.B)) \\
&= \ell(A) + 1 + \ell(B) + 1 + \ell(C) + 1 + 1 + \ell(\neg D) + 1 + \ell(\exists S.B) \\
&= \ell(A) + 1 + \ell(B) + 1 + \ell(C) + 1 + 1 + 1 + \ell(D) + 1 + 1 + \ell(B) \\
&= 12.
\end{aligned}$$

There is some evidence, e.g. in [HPS08, HBPS11, NS13], that shorter axioms are easier to understand for humans. This supports (yet not fully justifies) the intuition that we should prefer shorter hypotheses over longer ones. We will justify this intuition via experiments in Section 9.1.1.

5.1.2 Role Depth

The length of a hypothesis is not the only readability measure. In particular, axioms can have a plain structure, i.e. propositional shape. On the other hand, axioms can involve nestings due to role quantifiers. Therefore, a readability measure that quantifies the complexity of nestings would be helpful. It is the standard measure called *role depth* [BST07]. We first define the role depth of an axiom, see Definition 5.3.

Definition 5.3 (Role Depth of an Axiom)**.** Let $A \in N_C$ be a concept name, C, D complex concepts, $R \in N_R$ a role name, S, P complex roles, $o \in N_I$ an individual

name. The *role depth of a role* is defined as follows:

$$\delta(R) := 1,\ \delta(S^-) := \delta(S),$$
$$\delta(S \circ P) := \delta(S) + \delta(P).$$

The *role depth of a concept* is defined as follows:

$$\delta(\top) = \delta(\bot) = \delta(A) = \delta(\{o\}) := 0,$$
$$\delta(\neg C) := \delta(C),$$
$$\delta(C \sqcap D) = \delta(C \sqcup D) := max\{\delta(C), \delta(D)\},$$
$$\delta(\exists S.C) = \delta(\forall S.C) = \delta(\geq nS.C) = \delta(\leq nS.C) := \delta(S) + \delta(C).$$

The *role depth of a GCI and RI* is $\delta(C \sqsubseteq D) := max\{\delta(C), \delta(D)\}$ and $\delta(S \sqsubseteq P) := max\{\delta(S), \delta(P)\}$, respectively.

Informally, the role depth of a concept is the depth of its parsing tree, i.e. the length of the longest path from the root to leaves. The role depth of a GCI (RI) is the maximal role depth of its left-hand side (LHS) and right-hand side (RHS). The role depth of a hypothesis is simply the maximal role depth of its axioms, see Definition 5.4.

Definition 5.4 (Role Depth of a Hypothesis)**.** The *role depth* of a hypothesis H is defined as follows:

$$\delta(H) := max\{\delta(\alpha) \mid \alpha \in H\}.$$

Example 5.2 shows how role depth is calculated.

Example 5.2. Consider the axiom $\alpha := A \sqcap \exists R.B \sqsubseteq C \sqcup \forall R.(\neg D \sqcap \exists S.B)$ from Example 5.1. Its role depth is calculated as follows:

$$\begin{aligned}
\delta(\alpha) &= max\{\delta(A \sqcap \exists R.B),\ \delta(C \sqcup \forall R.(\neg D \sqcap \exists S.B))\} \\
&= max\{\delta(A),\ \delta(\exists R.B),\ \delta(C),\ \delta(\forall R.(\neg D \sqcap \exists S.B))\} \\
&= max\{\delta(A),\ 1 + \delta(B),\ \delta(C),\ 1 + \delta(\neg D \sqcap \exists S.B))\} \\
&= max\{\delta(A),\ 1 + \delta(B),\ \delta(C),\ 1 + \delta(\neg D),\ 1 + \delta(\exists S.B)\} \\
&= max\{\delta(A),\ 1 + \delta(B),\ \delta(C),\ 1 + \delta(D),\ 1 + 1 + \delta(B)\} \\
&= 2.
\end{aligned}$$

The role depth is just one of possible characteristics of axiom's parsing tree. One can think of alternative characteristics, such as the summary length of all paths from the root to leaves. Intuitively, axioms of higher role depth seem to be harder to read. As for the syntactic length, we will give experimental evidence for this intuition in Section 9.1.1.

5.2 Logical Quality of a Hypothesis

Since a hypothesis is generally a DL TBox, i.e. a logical theory, it is worthwhile to consider its logical quality. We observe several aspects of logical quality which can be measured. A hypothesis can be more or less general than another hypothesis, i.e. have different logical strength. It can also be redundant or non-redundant. Since a hypothesis can interact with the ontology, its logical quality should also quantify that interaction. In particular, a hypothesis can be either consistent or inconsistent with the ontology. A hypothesis can carry new information with regard to the ontology or just repeat its knowledge. A hypothesis can be more or less complex and surprising with respect to the ontology. In the following, we introduce measures for all mentioned aspects of logical quality and investigate their properties.

5.2.1 Consistency

An evident measure of logical quality of a hypothesis is its consistency with the ontology. Informally, consistency checks whether a hypothesis contradicts the ontology or not, see Definition 5.5. Consistent hypotheses should be preferred over inconsistent ones. Nevertheless, inconsistency is not a verdict for the hypothesis to be rejected. The reason is that an ontology can contain erroneous data (even if the ontology is consistent).

Definition 5.5 (Consistency)**.** A hypothesis H of an ontology $\mathcal{O}$ is called *consistent* with $\mathcal{O}$ if $\mathcal{O} \cup H$ is consistent. Otherwise, H is called *inconsistent*.

Thus, a hypothesis is consistent with the ontology if the ontology augmented by it is consistent. Intuitively, if a hypothesis is inconsistent with the ontology, this may indicate that the hypothesis is incorrect, see Example 5.3.

Example 5.3. Consider the ontology $\mathcal{O} :=$ Kinship, see Example 2.3. The hypothesis $H := \{Father \sqsubseteq Mother\}$ is inconsistent with $\mathcal{O}$.

As a consequence of Definition 5.5, if a hypothesis is inconsistent with $\mathcal{O}$, then all equivalent hypotheses are inconsistent with $\mathcal{O}$, see Lemma 5.1.

Lemma 5.1. *Let $\mathcal{O}$ be an ontology, H and H' hypotheses. If H is inconsistent with $\mathcal{O}$ and $H \equiv H'$, then H' is inconsistent with $\mathcal{O}$.*

Proof. Follows from Definition 5.5 since $H \equiv H'$ implies $\mathcal{O} \cup H \equiv \mathcal{O} \cup H'$. □

5.2.2 Informativeness

A hypothesis can be informative or uninformative with respect to the ontology. We call a hypothesis *informative* if it does not repeat what is already known from the ontology, see Definition 5.6. Informative hypotheses should be preferred over uninformative ones.

Definition 5.6 (Informativeness)**.** Given an ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$, an axiom α is called *informative* for $\mathcal{T}$ if $\mathcal{T} \not\models \alpha$. A hypothesis H is called *informative* for $\mathcal{T}$ if $H \neq \emptyset$ and every axiom $\alpha \in H$ is informative for $\mathcal{T}$. Otherwise, H is called *uninformative* for $\mathcal{T}$.

Thus, a hypothesis is informative for the TBox if the hypothesis is not empty and does not contain entailments of the TBox. If a hypothesis contains at least one axiom which follows from the TBox, then it is uninformative, see Example 5.4.

Example 5.4. Consider the TBox $\mathcal{T}$ of Kinship, see Example 2.3. The hypothesis $H_1 := \{Man \sqcap \exists hasChild.\top \sqsubseteq Father\}$ is informative for $\mathcal{T}$, while $H_2 := \{Father \sqsubseteq Human\}$ is not.

An uninformative hypothesis can trivially be turned into informative one via detecting and removing uninformative axioms unless it contains only uninformative axioms. In the latter case a hypothesis is entailed by the TBox, i.e. $\mathcal{T} \models H$, and removing all uninformative axioms gives the empty hypothesis which is uninformative by Definition 5.6. Please note that, unlike consistency, informativeness is not the same for all equivalent hypotheses, see Example 5.5.

Example 5.5. Consider the TBox $\mathcal{T} := \{A \sqsubseteq C\}$ and two hypotheses $H_1 := \{A \sqsubseteq B,\ B \sqsubseteq C\}$ and $H_2 := \{A \sqsubseteq B,\ B \sqsubseteq C,\ A \sqsubseteq C\}$. Although $H_1 \equiv H_2$, H_1 is informative for $\mathcal{T}$ and H_2 is not.

5.2.3 Redundancy

A hypothesis can contain axioms which are superfluous, or *redundant*, within the hypothesis. Axioms in a hypothesis can also have *redundant parts*, see [HPS08, Hor11]. We call a hypothesis *redundant* if it contains redundant axioms or axioms with redundant parts, see Definition 5.7. Clearly, non-redundant hypotheses should be preferred over redundant ones.

Definition 5.7 (Redundancy)**.** Let H be a hypothesis.

- An axiom $\alpha \in H$ is called *redundant* in H if $H \backslash \{\alpha\} \equiv H$.
- An axiom $\alpha \in H$ is said to have *redundant parts* in H if there exists an axiom $\alpha' \in pi(H) \backslash H$ such that $H \cup \{\alpha'\} \backslash \{\alpha\} \equiv H$, where $pi(H)$ returns all weaker and shorter forms of axioms in H (see Definition 19 at page 177 in [Hor11]).
- A hypothesis H is called *redundant* if there is an axiom $\alpha \in H$ which is redundant or has redundant parts. Otherwise, H is called *non-redundant*.

Redundancy can be detected and eliminated, i.e. a redundant hypothesis can be turned into a non-redundant one, as we describe in detail in Chapter 6. Example 5.6 illustrates redundant hypotheses.

Example 5.6. The following hypotheses are given:

$$H_1 := \{A \sqsubseteq B,\ B \sqsubseteq C,\ A \sqsubseteq C\}$$
$$H_2 := \{A \sqsubseteq B,\ \neg B \sqsubseteq \neg A\}$$
$$H_3 := \{A \sqsubseteq B \sqcap D,\ A \sqsubseteq C \sqcap D\}$$

The axiom $A \sqsubseteq C$ is redundant in H_1, the axiom $\neg B \sqsubseteq \neg A$ is redundant in H_2. While H_3 does not contain redundant axioms, its axiom $A \sqsubseteq C \sqcap D$ contains the redundant part D. The following hypotheses are equivalent to the listed above, respectively, and non-redundant:

$$H_1' := \{A \sqsubseteq B,\ B \sqsubseteq C\}$$
$$H_2' := \{A \sqsubseteq B\}$$
$$H_3' := \{A \sqsubseteq B \sqcap D,\ A \sqsubseteq C\}$$

As it follows from Definition 5.7, if a hypothesis is redundant, then there is an equivalent hypothesis of a smaller length, see Lemma 5.2.

Lemma 5.2. *If a hypothesis H is redundant, then there is a hypothesis H' such that $H' \equiv H$ and $\ell(H') < \ell(H)$.*

Proof. By Definition 5.7, if H is redundant, then there is $\alpha \in H$ such that α is (i) redundant or (ii) has redundant parts. In the case (i) $H' = H \backslash \{\alpha\}$ and $\ell(H') < \ell(H)$. In the case (ii) $H' = H \cup \{\alpha'\} \backslash \{\alpha\}$ and $\ell(H') < \ell(H)$ since $\ell(\alpha') < \ell(\alpha)$. □

The reverse direction in Lemma 5.2 does not hold, see Example 5.7.

Example 5.7. Consider the following hypotheses:

$$H_1 := \{A \sqsubseteq B,\ B \sqsubseteq C\}$$
$$H_2 := \{\top \sqsubseteq (\neg A \sqcup B) \sqcap (\neg B \sqcup C)\}$$

They are equivalent, i.e. $H_1 \equiv H_2$, both of them are non-redundant by Definition 5.7, and $\ell(H_1) = 4 < 10 = \ell(H_2)$ by Definition 5.1. Thus, the existence of an equivalent hypothesis of a smaller length does not imply that the hypothesis is redundant.

Please note that Definition 5.7 only defines redundancy which is caused by excessive length. However, there are other causes for redundancy in DLs, see Example 5.8.

Example 5.8. Consider the hypothesis $H := \{\exists R.A \sqcap \forall R.A \sqsubseteq B\}$. It is non-redundant by Definition 5.7. However, the restriction $\exists R.A$ can be simplified given $\forall R.A$. In other words, there is $H' := \{\exists R.\top \sqcap \forall R.A \sqsubseteq B\}$ such that $H' \equiv H$, $\ell(H') = \ell(H)$, and H' expresses less restrictions than H.

Definition 5.7 does not consider an ontology $\mathcal{O}$, i.e. a hypothesis is redundant or non-redundant regardless of its ontology $\mathcal{O}$. The reason is that an axiom $\alpha \in H$, which is not redundant in H but redundant in $\mathcal{O} \cup H$ can be interesting. Such an axiom can reveal yet only implicit (and possibly unknown) relations between classes. In addition, checking redundancy would require checking entailments of $\mathcal{O} \cup H$ instead of H. The latter could be much more costly for computationally hard ontologies.

Hypotheses can be redundant not only on their own but with respect to other hypotheses as well. In particular, one hypothesis can be equivalent to another

hypothesis. We call such hypotheses *syntactic variations* since they capture the same knowledge but differ syntactically, see Definition 5.8.

Definition 5.8 (Syntactic variation)**.** An axiom α' is called a *syntactic variation* of an axiom α if $\alpha' \neq \alpha$ and $\alpha' \equiv \alpha$. A hypothesis H' is called a *syntactic variation* of a hypothesis H if $H' \neq H$ and $H' \equiv H$.

In general, a hypothesis can have infinitely many syntactic variations. Clearly, if we intend to explore a set $\mathbb{H}$ of hypotheses, we would like to avoid syntactic variations of already explored hypotheses because the former simply repeat knowledge already encoded by the latter.

5.2.4 Logical Strength

A hypothesis can be logically *stronger* or *weaker* than another hypothesis. Intuitively, a stronger hypothesis is more constrained, or more *specific*, and a weaker hypothesis is less constrained, or more *general*, see Definition 5.9. Logical strength allows for ordering hypotheses and navigating through them in a methodical way.

Definition 5.9 (Logical strength)**.** An ontology $\mathcal{O}$ is said to be *weaker* than another ontology $\mathcal{O}'$ (or $\mathcal{O}'$ is said to be *stronger* than $\mathcal{O}$), written as $\mathcal{O}' \rhd \mathcal{O}$, if $\mathcal{O}' \models \mathcal{O}$ and $\mathcal{O} \not\models \mathcal{O}'$. An axiom α is said to be *weaker* than another axiom α' if $\{\alpha'\} \rhd \{\alpha\}$, abbreviated as $\alpha' \rhd \alpha$. A hypothesis H is said to be *weaker* than another hypothesis H' if $H' \rhd H$.

There is a similar notion of logical strength in CDL, see Section 3.2.1, called *generality*, but defined for concepts. Similarly, we call a weaker hypothesis *more general* and a stronger hypothesis *more specific*. Informally, a hypothesis H is weaker than a hypothesis H' if H follows from H' and they are not equivalent, see Example 5.9.

Example 5.9. The following hypotheses are given:

$$H_1 := \{A \sqsubseteq C\}$$
$$H_2 := \{A \sqcap B \sqsubseteq C\}$$
$$H_3 := \{A \sqsubseteq B \sqcup C\}$$

Then, H_2 is weaker than H_1, since $H_1 \models H_2$, and H_3 is weaker than H_1, since $H_1 \models H_3$. Hypotheses H_2 and H_3 are incomparable with respect to their logical strength since none of them follows from another.

Please note that, unlike the aforementioned quality measures, logical strength is a binary relation, i.e. it can only be evaluated with regard to another hypothesis. Another difference is that hypotheses can be incomparable with respect to logical strength. Hence, logical strength imposes a partial order $(\mathbb{H}, \rhd)$ on hypotheses $\mathbb{H}$. Logical strength has the following properties, see Lemma 5.3.

Lemma 5.3. *Let $\mathcal{O}$ be an ontology, H, H_1, H_2 hypotheses. Then*

(i) *$H_1 \subset H_2$ and $H_1 \not\equiv H_2$ implies $H_2 \rhd H_1$;*

(ii) *$H_1 \rhd H$ and $H_1 \equiv H_2$ implies $H_2 \rhd H$;*

(iii) *H_1 is inconsistent with $\mathcal{O}$ and $H_2 \rhd H_1$ implies H_2 is inconsistent with $\mathcal{O}$;*

(iv) *$H_2 \rhd H_1$ implies either $\mathcal{O} \cup H_2 \rhd \mathcal{O} \cup H_1$ or $\mathcal{O} \cup H_2 \equiv \mathcal{O} \cup H_1$.*

Proof. (i) Since DLs are monotonic,[1] $H_1 \subset H_2$ implies $H_2 \models H_1$ and $H_1 \not\equiv H_2$ implies $H_1 \not\models H_2$. Hence $H_2 \rhd H_1$ by Definition 5.9.

(ii) follows from Definition 5.9 since $H_1 \models H$ and $H \not\models H_1$ and $H_1 \equiv H_2$ implies $H_2 \models H$ and $H \not\models H_2$.

(iii) follows from Definition 5.9 since $H_2 \models H_1$ implies $\mathcal{O} \cup H_2 \models \mathcal{O} \cup H_1$, i.e. $\mathcal{O} \cup H_2$ has no models because $\mathcal{O} \cup H_1$ has no models.

(iv) follows from Definition 5.9 since $H_2 \models H_1$ implies $\mathcal{O} \cup H_2 \models \mathcal{O} \cup H_1$ and $H_1 \not\models H_2$ implies either $\mathcal{O} \cup H_1 \not\models \mathcal{O} \cup H_2$ or $\mathcal{O} \cup H_1 \models \mathcal{O} \cup H_2$.

□

Informally, Lemma 5.3 states the following properties: (i) equivalent hypotheses have the same logical strength; (ii) adding axioms to a hypothesis never produces a weaker hypothesis; (iii) if a hypothesis is inconsistent with the ontology, then all stronger hypotheses are inconsistent with it; (iv) adding the same ontology to a stronger and weaker hypotheses either preserves their ordering or makes them equivalent.

[1] A logic is monotonic if adding a formula to a theory never removes its consequences.

5.2.5 Dissimilarity

For a single axiom, one can measure how "dissimilar" its LHS and RHS are with respect to the TBox. Intuitively, the more dissimilar the LHS and RHS are, the more "surprising" the axiom is for the TBox. This is a useful notion because a domain expert might wish to view the most surprising hypotheses first. In principle, any concept similarity measure, e.g. [APS14, EPT15], can be adapted for measuring the dissimilarity of a GCI. We adapt the concept similarity measure from [APS14], see Definition 5.10.

Definition 5.10 (Dissimilarity)**.** Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, $\mathbb{C}$ a finite set of concepts. Let the set of all subsumers of a concept C in $\mathbb{C}$ given $\mathcal{T}$ be

$$subs(C, \mathcal{T}, \mathbb{C}) \ := \ \{C' \in \mathbb{C} \cup \{C\} \mid \mathcal{T} \models C \sqsubseteq C'\}.$$

Then, the dissimilarity of a GCI $C \sqsubseteq D$ is defined as follows:

$$dsim(C \sqsubseteq D, \ \mathcal{T}, \mathbb{C}) \ := \ 1 - \frac{|subs(C, \mathcal{T}, \mathbb{C}) \cap subs(D, \mathcal{T}, \mathbb{C})|}{|subs(C, \mathcal{T}, \mathbb{C}) \cup subs(D, \mathcal{T}, \mathbb{C})|}.$$

Informally, the dissimilarity of $C \sqsubseteq D$ measures how many subsumers in $\mathbb{C}$ the concepts C and D have in common given the TBox $\mathcal{T}$. The dissimilarity of a RI is defined analogously and omitted for the sake of brevity. The quality measure is in the range $[0, 1]$. The minimal value implies that all subsumers are the same, i.e. the LHS and RHS are equivalent, and the maximal value implies that all subsumers are different. Example 5.10 illustrates calculating dissimilarity.

Example 5.10. Consider the following TBox:

$$\begin{aligned} \mathcal{T} := \{&C_1 \sqsubseteq B_1, \ B_1 \sqsubseteq A_1, \ A_1 \sqsubseteq A, \\ &C_2 \sqsubseteq B_2, \ B_2 \sqsubseteq A_2, \ A_2 \sqsubseteq A\}. \end{aligned}$$

Given $\mathbb{C} := \widetilde{\mathcal{T}}$, the dissimilarity of the axiom $\alpha_1 := C_1 \sqsubseteq C_2$ is calculated as follows:

$$subs(C_1, \mathcal{T}, \mathbb{C}) = \{A, A_1, B_1, C_1\}, \ subs(C_2, \mathcal{T}, \mathbb{C}) = \{A, A_2, B_2, C_2\}$$

$$dsim(\alpha_1, \mathcal{T}, \mathbb{C}) = 1 - \frac{|\{A\}|}{|\{A, A_1, B_1, C_1, A_2, B_2, C_2\}|} = \frac{6}{7}.$$

To compare, the dissimilarity of the axiom $\alpha_2 = A_1 \sqsubseteq C_2$ (please notice that A_1

is more general than C_1 according to $\mathcal{T}$) is as follows:

$$dsim(\alpha_2, \mathcal{T}, \mathbb{C}) = 1 - \frac{|\{A\}|}{|\{A, A_1, A_2, B_2, C_2\}|} = \frac{4}{5}.$$

Thus, $dsim(\alpha_1, \mathcal{T}, \mathbb{C}) > dsim(\alpha_2, \mathcal{T}, \mathbb{C})$.

Please note that dissimilarity is a *symmetric* measure, i.e. it does not change if the LHS and RHS are swapped. In principle, the dissimilarity of a multi-axiom hypothesis H could be defined as the sum of the dissimilarities of its axioms: $dsim(H, \mathcal{T}, \mathbb{C}) := \sum_{\alpha \in H} dsim(\alpha, \mathcal{T}, \mathbb{C})$. However, that definition would disregard the semantics of DLs. As a result, the measure could overestimate the quality value, see Example 5.11.

Example 5.11. Consider the hypothesis $H_3 := \{C_1 \sqsubseteq C_2,\ C_1 \sqsubseteq B_2,\ C_1 \sqsubseteq A_2\}$ in Example 5.10. Although H_3 is equivalent to the hypothesis $H_1 := \{C_1 \sqsubseteq C_2\}$ given $\mathcal{T}$, i.e. $H_3 \cup \mathcal{T} \equiv H_1 \cup \mathcal{T}$, H_3 would have a considerably higher aggregated dissimilarity than H_1, i.e. $dsim(H_3, \mathcal{T}, \mathbb{C}) > dsim(H_1, \mathcal{T}, \mathbb{C})$.

5.2.6 Complexity

While uninformative hypotheses do not bring new knowledge to the ontology, informative ones do. Yet, some informative hypotheses may bring *more new knowledge* than others do. We aim at quantifying how much new knowledge a hypothesis contains for the given ontology via the measure called *complexity*. More complex hypotheses presumably carry more knowledge for the TBox. For example, the simplest hypotheses are ones entailed by the TBox, i.e. they are uninformative. Such hypotheses carry no new knowledge for the TBox and hence are useless. On the other hand, a hypothesis can be *overcomplicated*. Occam's razor, see Section 1.1, suggests that a hypothesis should not be more complex than necessary. The challenge is to determine when it is exactly as complex as necessary. Complexity is intended to show which hypotheses are more complex than others and quantify by how much, thus potentially identifying overcomplicated ones.

In the context of the philosophy of science, a scientific theory explains some observations, see Section 1.1. Besides explaining observations, a theory can have implications which reflect its complexity such that a theory having more implications is more complex. Given an ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ and a hypothesis H, we can

consider the TBox $\mathcal{T}$ as an existing theory and $\mathcal{T} \cup H$ as a new theory. Then, we can compare the complexity of the new theory $\mathcal{T} \cup H$ with the complexity of the old theory $\mathcal{T}$ by quantifying how many *new entailments* (implications) the new theory has. In the same way, given another hypothesis H', we can compare the complexity of $\mathcal{T} \cup H'$ with the complexity of $\mathcal{T}$. After that, we can compare the quantified complexities of $\mathcal{T} \cup H$ and $\mathcal{T} \cup H'$ to determine which hypothesis, H or H', is more complex with respect to $\mathcal{T}$. As the set of new entailments is infinite in general, we only consider a certain finite subset of them, see Definition 5.11.

Definition 5.11 (Complexity)**.** Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, $\mathbb{C}$ and $\mathbb{R}$ finite sets of concepts and roles, $\eta := C_1 \sqsubseteq C_2$ or $\eta := R_1 \sqsubseteq R_2$, where $C_1, C_2 \in \mathbb{C}$, $R_1, R_2 \in \mathbb{R}$. The *complexity* of a hypothesis H is defined as follows:

$$com(H, \mathcal{T}, \mathbb{C}, \mathbb{R}) \ := \ |\{\eta \mid \mathcal{T} \cup H \models \eta \ \wedge \ \mathcal{T} \not\models \eta\}|.$$

Thus, we only count new entailments that are subsumptions between concepts from a fixed set $\mathbb{C}$ or between roles from $\mathbb{R}$. Complexity has the following properties, see Lemma 5.4.

Lemma 5.4. *Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, H and H' hypotheses, $\mathbb{C}$ and $\mathbb{R}$ finite sets of concepts and roles. Then*

(i) *$H' \equiv H$ implies $com(H', \mathcal{T}, \mathbb{C}, \mathbb{R}) = com(H, \mathcal{T}, \mathbb{C}, \mathbb{R})$;*

(ii) *$H' \rhd H$ implies $com(H', \mathcal{T}, \mathbb{C}, \mathbb{R}) \geq com(H, \mathcal{T}, \mathbb{C}, \mathbb{R})$;*

(iii) *$\mathcal{T} \models H$ if and only if $com(H, \mathcal{T}, \mathbb{C}, \mathbb{R}) = 0$;*

(iv) *$com(\emptyset, \mathcal{T}, \mathbb{C}, \mathbb{R}) = 0$.*

Proof. (i) follows from Definition 5.11 because $H' \equiv H$ implies $\mathcal{T} \cup H' \equiv \mathcal{T} \cup H$, i.e. $\mathcal{T} \cup H'$ and $\mathcal{T} \cup H$ have the same entailments.

(ii) follows from Definition 5.11 because $H' \rhd H$ implies $H' \models H$ and, hence, $\mathcal{T} \cup H' \models \mathcal{T} \cup H$, i.e. all entailments of $\mathcal{T} \cup H$ are also entailments of $\mathcal{T} \cup H'$, but $\mathcal{T} \cup H'$ can have more entailments than $\mathcal{T} \cup H$.

(iii) follows from Definition 5.11 because $\mathcal{T} \models H$ if and only if the set of new entailments is empty.

(iv) follows from (iii).

□

Informally, Lemma 5.4 states the following properties: (i) equivalent hypotheses have equal complexity; (ii) a stronger hypothesis cannot have lower complexity; (iii) the complexity equals zero if and only if a hypothesis is entailed (uninformative); (iv) the complexity of the empty hypothesis equals zero;

At first, complexity resembles dissimilarity: both show how surprising a hypothesis is with respect to the TBox. However, complexity is applicable to a set of axioms, while dissimilarity is only to a single axiom. In contrast to dissimilarity, complexity is *asymmetric*. In addition, the properties of Lemma 5.4 do not hold for dissimilarity. As a result, they are rather independent measures, see Example 5.12.

Example 5.12. Let us calculate the complexity of the hypotheses $H_1 := \{\alpha_1\}$, where $\alpha_1 := C_1 \sqsubseteq C_2$, and $H_2 := \{\alpha_2\}$, where $\alpha_2 := A_1 \sqsubseteq C_2$, from Example 5.10 ($\mathbb{C}$ is the same, $\mathbb{R} := \emptyset$) and compare it with the calculated dissimilarity:

$$\begin{aligned} com(H_1, \mathcal{T}, \mathbb{C}, \mathbb{R}) &= |\{C_1 \sqsubseteq C_2, C_1 \sqsubseteq B_2, C_1 \sqsubseteq A_2\}| = 3, \\ com(H_2, \mathcal{T}, \mathbb{C}, \mathbb{R}) &= |\{C_1 \sqsubseteq C_2, C_1 \sqsubseteq B_2, C_1 \sqsubseteq A_2, \\ &\qquad B_1 \sqsubseteq C_2, B_1 \sqsubseteq B_2, B_1 \sqsubseteq A_2, \\ &\qquad A_1 \sqsubseteq C_2, A_1 \sqsubseteq B_2, A_1 \sqsubseteq A_2\}| = 9. \end{aligned}$$

Thus, H_1 has lower complexity than H_2 but higher dissimilarity. In addition, consider the hypothesis $H_3 := \{\alpha_3\}$, where $\alpha_3 := B_1 \sqcap C_2 \sqsubseteq A_1$: $com(H_3, \mathcal{T}, \mathbb{C}, \mathbb{R}) = 0$ since $\mathcal{T} \models \alpha_3$ but

$$dsim(\alpha_3, \mathcal{T}, \mathbb{C}) = 1 - \frac{|\{A, A_1\}|}{|\{A, B_1, A_1, C_2, B_2, A_2\}|} = \frac{2}{3}.$$

5.3 Statistical Quality of a Hypothesis

Given the inductive nature of the OL problem, statistical quality of a hypothesis appears to be crucial. However, it is challenging to measure in DLs. One of the challenges is the standard OWA which permits incomplete information, see Section 2.1.5. As discussed in Chapter 3, existing OL approaches tend to disregard the OWA and make the CWA. Although the CWA greatly simplifies measuring

statistical quality, it can be misleading. In the following, we propose statistical quality measures that respect the standard DL semantics and its OWA. We separate measures which are designed for a single axiom, i.e. *axiom measures*, from those which are designed for an axiom set (arbitrary TBox), i.e. *axiom set measures*.

5.3.1 Axiom Measures

We first introduce basic axiom measures that evaluate different aspects of statistical quality of a TBox axiom which is either a GCI or RI. After that, we introduce main axiom measures that respect the semantics of DLs better than basic axiom measures. We define most axiom measures based on the standard measures used in Association Rule Mining (ARM), see Section 2.2.5, but adjust them specifically for DLs respecting the semantics and the OWA.

5.3.1.1 Preliminary Definitions

Definitions of statistical quality measures use some auxiliary notions that we define below. According to the standard OWA of OWL, for each individual $a \in in(\mathcal{O})$ the following cases are possible:

(i) a is an instance of C, i.e. $\mathcal{O} \models C(a)$;

(ii) a is an instance of $\neg C$, i.e. $\mathcal{O} \models \neg C(a)$;

(iii) a is neither an instance of C nor $\neg C$, i.e. $\mathcal{O} \not\models C(a)$ and $\mathcal{O} \not\models \neg C(a)$.

Within this thesis, it is important to note the difference between the case (ii) and (iii). While the case (ii) states that an individual is a *known* instance of $\neg C$, the case (iii) states that an individual is neither a known instance of C nor $\neg C$, i.e. information is incomplete. Please note that the OWA separates the cases (ii) and (iii), while the CWA treats the case (iii) as (ii), i.e. what is not known to be true is assumed to be false, see Section 2.1.5. If quality measures are aimed at respecting the OWA, they should also separate the cases (ii) and (iii).

In the following, we use the notion of *instance function*, see Definition 5.12.

Definition 5.12 (Instance function)**.** Let $\mathcal{O}$ be an ontology, C a (possibly complex) concept, $\mathring{C} \in \{C, {}^{?}C\}$ a *concept variable*. Then, the *instance function* is

defined as follows:

$$inst(\mathring{C}, \mathcal{O}) := \begin{cases} \{a \in in(\mathcal{O}) \mid \mathcal{O} \models C(a)\} & \text{if } \mathring{C} = C \\ \{a \in in(\mathcal{O}) \mid \mathcal{O} \not\models C(a) \ \wedge \ \mathcal{O} \not\models \neg C(a)\} & \text{if } \mathring{C} = ?C \end{cases}$$

According to Definition 5.12, $inst(C, \mathcal{O})$ is the set of instances of C, called *positive instances*; $inst(\neg C, \mathcal{O})$ is the set of instances of $\neg C$, called *negative instances*; and $inst(?C, \mathcal{O})$ is the set of individuals which are instances of neither C nor $\neg C$, called *unknown instances*. Thus, the instance function performs the instance retrieval[2] of either C, $\neg C$, or $?C$.

As it follows from Definition 5.12, the union of instances of C, $\neg C$, and $?C$ gives the set of all individuals, see Lemma 5.5. In other words, the instances of $?C$ are all individuals which are instances of neither C nor $\neg C$.

Lemma 5.5. $inst(C, \mathcal{O}) \cup inst(\neg C, \mathcal{O}) \cup inst(?C, \mathcal{O}) = in(\mathcal{O})$.

Proof. Immediately follows from Definition 5.12. □

The instance function of some complex concepts can be represented via instance functions of simpler concepts, see Lemma 5.6.

Lemma 5.6. *Let $\mathcal{O}$ be an ontology, C and D are concepts. Then*

(i) $inst(C \sqcap D, \ \mathcal{O}) = inst(C, \mathcal{O}) \cap inst(D, \mathcal{O})$

(ii) $inst(C \sqcup D, \ \mathcal{O}) = inst(C, \mathcal{O}) \cup inst(D, \mathcal{O})$

(iii) $inst(?(\neg C), \ \mathcal{O}) = inst(?C, \mathcal{O})$

(iv) $inst(?(C \sqcup D), \ \mathcal{O}) = inst(?C, \mathcal{O}) \cup inst(?D, \mathcal{O}) \backslash (inst(C, \mathcal{O}) \cup inst(D, \mathcal{O}))$

(v) $inst(?(C \sqcap D), \ \mathcal{O}) = inst(?C, \mathcal{O}) \cup inst(?D, \mathcal{O}) \backslash (inst(\neg C, \mathcal{O}) \cup inst(\neg D, \mathcal{O}))$

Proof. (i) $inst(C \sqcap D, \ \mathcal{O}) = \{a \in in(\mathcal{O}) \mid \mathcal{O} \models (C \sqcap D)(a)\}$
(by Definition 5.12)
$= \{a \in in(\mathcal{O}) \mid \mathcal{O} \models C(a) \ \wedge \ \mathcal{O} \models D(a)\}$
(since $\mathcal{O} \models (C \sqcap D)(a)$ if and only if $\mathcal{O} \models C(a)$ and $\mathcal{O} \models D(a)$)
$= \{a \in in(\mathcal{O}) \mid \mathcal{O} \models C(a)\} \cap \{a \in in(\mathcal{O}) \mid \mathcal{O} \models D(a)\}$
$= inst(C, \mathcal{O}) \cap inst(D, \mathcal{O})$ (by Definition 5.12).

[2]Instance retrieval is a type of query answering in DLs where a query is specified by a concept, see Section 2.1.1.

(ii) is analogous to (i).

(iii) By Definition 5.12 $inst(?(\neg C),\ \mathcal{O}) = \{a \in in(\mathcal{O}) \mid \mathcal{O} \not\models \neg C(a) \wedge \mathcal{O} \not\models (\neg\neg C)(a)\} = inst(?C,\ \mathcal{O})$.

(iv) $inst(?(C \sqcup D),\ \mathcal{O}) = \{a \in in(\mathcal{O}) \mid \mathcal{O} \not\models (C \sqcup D)(a) \wedge \mathcal{O} \not\models (\neg(C \sqcup D))(a)\}$ (by Definition 5.12)

$= \{a \in in(\mathcal{O}) \mid (\mathcal{O} \not\models C(a) \wedge \mathcal{O} \not\models D(a)) \wedge \mathcal{O} \not\models (\neg C \sqcap \neg D)(a)\}$
(since $\mathcal{O} \not\models (C \sqcup D)(a)$ if and only if $\mathcal{O} \not\models C(a)$ and $\mathcal{O} \not\models D(a)$)

$= \{a \in in(\mathcal{O}) \mid (\mathcal{O} \not\models C(a) \wedge \mathcal{O} \not\models D(a)) \wedge (\mathcal{O} \not\models \neg C(a) \vee \mathcal{O} \not\models \neg D)(a))\}$
(since $\mathcal{O} \not\models (\neg C \sqcap \neg D)(a)$ if and only if $\mathcal{O} \not\models \neg C(a)$ or $\mathcal{O} \not\models \neg D(a)$)

$= \{a \in in(\mathcal{O}) \mid (\mathcal{O} \not\models C(a) \wedge \mathcal{O} \not\models D(a) \wedge \mathcal{O} \not\models \neg C(a)) \vee (\mathcal{O} \not\models C(a) \wedge \mathcal{O} \not\models D(a) \wedge \mathcal{O} \not\models \neg D)(a))\}$ (by propositional distribution)

$= \{a \in in(\mathcal{O}) \mid \mathcal{O} \not\models C(a) \wedge \mathcal{O} \not\models \neg C(a) \wedge \mathcal{O} \not\models D(a)\} \cup \{a \in in(\mathcal{O}) \mid \mathcal{O} \not\models D(a) \wedge \mathcal{O} \not\models \neg D)(a) \wedge \mathcal{O} \not\models C(a)\}$

$= \{a \in in(\mathcal{O}) \mid \mathcal{O} \not\models C(a) \wedge \mathcal{O} \not\models \neg C(a)\} \cup \{a \in in(\mathcal{O}) \mid \mathcal{O} \not\models D(a) \wedge \mathcal{O} \not\models \neg D)(a)\} \setminus (\{a \in in(\mathcal{O}) \mid \mathcal{O} \models C(a)\} \cup \{a \in in(\mathcal{O}) \mid \mathcal{O} \models D(a)\})$

$= inst(?C, \mathcal{O}) \cup inst(?D, \mathcal{O}) \setminus (inst(C, \mathcal{O}) \cup inst(D, \mathcal{O}))$ (by Definition 5.12).

(v) is analogous to (iv).

□

In the following, we use the notion of *joint probability distribution of concept variables* which is the relative size of the intersection of the instances of the given concept variables, see Definition 5.13.

Definition 5.13 (Joint probability distribution of concept variables)**.** Given an ontology $\mathcal{O}$, the *joint probability distribution of concept variables* $\mathring{C}_1, \ldots, \mathring{C}_n$ is defined as follows:

$$\mathbf{P}_{\mathcal{O}}(\mathring{C}_1, \ldots, \mathring{C}_n) := \frac{1}{|in(\mathcal{O})|} |\bigcap_{i=1}^{n} inst(\mathring{C}_i, \mathcal{O})|.$$

The value of the distribution for given values of $\mathring{C}_1, \ldots, \mathring{C}_n$ is called *probability.* As a consequence of Definition 5.13, the following properties hold for any concept, see Lemma 5.7.

Lemma 5.7. *Let C be a concept. The following properties hold for C:*

(i) $\mathbf{P}_{\mathcal{O}}(C) + \mathbf{P}_{\mathcal{O}}(\neg C) + \mathbf{P}_{\mathcal{O}}(\text{?}C) = 1$,

(ii) $\mathbf{P}_{\mathcal{O}}(\top) = 1$, *(iii)* $\mathbf{P}_{\mathcal{O}}(\bot) = 0$.

Proof. The property (i) follows from Definition 5.13 and Lemma 5.5, (ii) and (iii) from Definition 5.13 and Definition 5.12. □

The probability of conjunctions and disjunctions can be written via probabilities of their operands, see Lemma 5.8.

Lemma 5.8. *Let $\mathcal{O}$ be an ontology, C and D are concepts. Then*

$$\text{(i) } \mathbf{P}_{\mathcal{O}}(C \sqcap D) = \mathbf{P}_{\mathcal{O}}(C, D)$$
$$\text{(ii) } \mathbf{P}_{\mathcal{O}}(C \sqcup D) = \mathbf{P}_{\mathcal{O}}(C) + \mathbf{P}_{\mathcal{O}}(D) - \mathbf{P}_{\mathcal{O}}(C, D)$$

Proof. (i) $\mathbf{P}_{\mathcal{O}}(C \sqcap D) = \frac{|inst(C \sqcap D, \mathcal{O})|}{|in(\mathcal{O})|}$ (by Definition 5.13)
$= \frac{|inst(C,\mathcal{O}) \cap inst(D,\mathcal{O})|}{|in(\mathcal{O})|}$ (by Lemma 5.6)
$= \mathbf{P}_{\mathcal{O}}(C, D)$ (by Definition 5.13)

(ii) $\mathbf{P}_{\mathcal{O}}(C \sqcup D) = \frac{|inst(C \sqcup D, \mathcal{O})|}{|in(\mathcal{O})|}$ (by Definition 5.13)
$= \frac{|inst(C,\mathcal{O}) \cup inst(D,\mathcal{O})|}{|in(\mathcal{O})|}$ (by Lemma 5.6)
$= \frac{|inst(C,\mathcal{O})| + |inst(D,\mathcal{O})| - |inst(C,\mathcal{O}) \cap inst(D,\mathcal{O})|}{|in(\mathcal{O})|}$
(since $|S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2|$ for any sets S_1, S_2)
$= \mathbf{P}_{\mathcal{O}}(C) + \mathbf{P}_{\mathcal{O}}(D) - \mathbf{P}_{\mathcal{O}}(C, D)$ (by Definition 5.13)

□

5.3.1.2 Basic Measures

Let us consider a GCI $C \sqsubseteq D$. The axiom states that all instances of C are also instances of D. Given an ontology $\mathcal{O}$, we can check *how well* its data (ABox) supports this statement. We define 4 basic quality measures, namely *basic coverage, support, contradiction, assumption*, using simple counting of relevant instances, see Definition 5.14. The basic measures are intended to capture the following notions.

- *Coverage* shows the area of applicability of an axiom, i.e. the number of individuals directly involved in the implication.

- *Support* shows how many individuals support an axiom. Axioms with higher support have more evidence for their validity.

- *Contradiction* shows how many individuals contradict an axiom, i.e. how many counterexamples it has. As a reminder, given an ontology $\mathcal{O}$, an individual $a \in in(\mathcal{O})$ is called a counterexample for a GCI $C \sqsubseteq D$ if $\mathcal{O} \models (C \sqcap \neg D)(a)$, see Section 3.2.3. Axioms with higher contradiction have more evidence of their invalidity.

- *Assumption* shows how many "guesses" an axiom makes due to the OWA. Axioms with lower assumption are less suspicious according to Occam's razor, see Section 1.1.

Definition 5.14 (Basic measures)**.** Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. The *basic coverage, support, contradiction, assumption* of α are defined, respectively, as follows:

$$\begin{aligned} bcov(\alpha, \mathcal{O}) &:= |inst(C,\ \mathcal{O})| \\ bsup(\alpha, \mathcal{O}) &:= |inst(C \sqcap D,\ \mathcal{O})| \\ bcnt(\alpha, \mathcal{O}) &:= |inst(C \sqcap \neg D,\ \mathcal{O})| \\ basm(\alpha, \mathcal{O}) &:= |inst(C,\ \mathcal{O}) \cap inst(?D,\ \mathcal{O})| \end{aligned}$$

In the following, we omit "basic" in the names of the basic measures for the sake of brevity when it is clear from the context. According to Definition 5.14, the basic measures count individuals as follows: coverage counts instances of C; support counts common instances of C and D; contradiction counts instances of C which are instances of $\neg D$; assumption counts instances of C which are neither known instances of D nor $\neg D$. Hence, coverage is simply the sum of support, contradiction, and assumption, see Lemma 5.9.

Lemma 5.9. *Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. Then*

$$bsup(\alpha, \mathcal{O}) + bcnt(\alpha, \mathcal{O}) + basm(\alpha, \mathcal{O}) = bcov(\alpha, \mathcal{O}).$$

Proof. $bsup(\alpha, \mathcal{O}) + bcnt(\alpha, \mathcal{O}) + basm(\alpha, \mathcal{O})$
$= |inst(C \sqcap D, \mathcal{O})| + |inst(C \sqcap \neg D, \mathcal{O})| + |inst(C, \mathcal{O}) \cap inst(?D, \mathcal{O})|$
(by Definition 5.14)

$= |inst(C, \mathcal{O}) \cap inst(D, \mathcal{O})| + |inst(C, \mathcal{O}) \cap inst(\neg D, \mathcal{O})| + |inst(C, \mathcal{O}) \cap inst(?D, \mathcal{O})|$ (by Lemma 5.6)

$= |inst(C,\ \mathcal{O})|$ (an individual is either D, $\neg D$, or $?D$)

$= bcov(\alpha, \mathcal{O})$ (by Definition 5.14). □

Thus, support is presumably a *positive* measure, i.e. higher values indicate better quality of a hypothesis, while contradiction and assumption are presumably *negative* ones, i.e. lower values indicate better quality of a hypothesis. Since, coverage is the sum of support, contradiction, and assumption, it is neither positive nor negative. Please note that support is a symmetric measure, i.e. it is the same for $C \sqsubseteq D$ and $D \sqsubseteq C$, while others are not.

Please note that the basic measures only consider the "forward" direction of an axiom $C \sqsubseteq D$, i.e. how many instances of C are also instances of D. According to the semantics of DLs, an axiom $C \sqsubseteq D$ has also the "backward" direction, i.e. how many instances of $\neg D$ are also instances of $\neg C$. In the following, we introduce axiom measures that take this into account.

It is also worth noticing that the basic measures are based on the instance function that counts only named individuals. The semantics of DLs allows for anonymous individuals, i.e. those that are not explicitly named in the ontology. In some cases, anonymous individuals could be counted that would make the measures considerably more complex, both conceptually and computationally. In other cases, it is not even possible to count all anonymous individuals, see Example 5.13.

Example 5.13. Consider the ontology $\mathcal{O} := \{(\exists R.A)(a)\}$. The concept A has zero instances in $\mathcal{O}$: $|inst(A,\ \mathcal{O})| = 0$. However, there is one element (anonymous individual) which is A in every model of $\mathcal{O}$. Consider another ontology $\mathcal{O}' := \mathcal{T} \cup \mathcal{A}$, where $\mathcal{T} := \{A \sqsubseteq \exists R.A,\ A \sqsubseteq \leq 1R^-.A\}$ and $\mathcal{A} := \{A(a),\ (\leq 0R^-.A)(a)\}$. The concept A has one instance in $\mathcal{O}'$: $|inst(A,\ \mathcal{O}')| = 1$. However, in every model of $\mathcal{O}'$, there are infinitely many elements that belong to A.

While coverage, support, and contradiction have their counterparts in ARM, assumption does not. We introduce assumption to respect the OWA such that an axiom, even if it has no counterexamples, can assume many facts that may be wrong. Example 5.14 illustrates the basic measures.

Example 5.14. Consider the Kinship data shown in Table 5.1 (which is the same as Table 3.2). It respects the OWA: a question mark "?" shows that it is unknown

whether an individual is an instance of a concept (indicated by "X") or it is not (indicated by a blank space). Consider the axioms $\alpha_1 := \exists marriedTo.\top \sqsubseteq Mother$ and $\alpha_2 := \exists hasChild.\top \sqsubseteq Mother$. Their basic measures are calculated as follows:

$$\begin{aligned} bsup(\alpha_1, \mathcal{O}) &= |\{Victoria, Penelope\}| = 2 & bsup(\alpha_2, \mathcal{O}) &= 2 \\ bcnt(\alpha_1, \mathcal{O}) &= |\{James, Chris, Arthur\}| = 3 & bcnt(\alpha_2, \mathcal{O}) &= 2 \\ basm(\alpha_1, \mathcal{O}) &= |\{Margaret\}| = 1 & basm(\alpha_2, \mathcal{O}) &= 0 \\ bcov(\alpha_1, \mathcal{O}) &= 2 + 3 + 1 = 6 & bcov(\alpha_2, \mathcal{O}) &= 4 \end{aligned}$$

According to the basic measures, α_2 has better quality than α_1 because its support is the same but its contradiction and assumption are lower (better).

	Man	$Woman$	$Father$	$Mother$	$\exists hasChild.\top$	$\exists marriedTo.\top$
$Charlotte$		X		?	?	?
$James$	X		X		X	X
$Victoria$		X		X	X	X
$Chris$	X		X		X	X
$Penelope$		X		X	X	X
$Margaret$		X		?	?	X
$Arthur$	X		?		?	X

Table 5.1: Kinship data under OWA

The basic measures can be defined for a RI $R \sqsubseteq S$ in the same way as it is done for a GCI $C \sqsubseteq D$. The only difference is that, instead of returning instances of a concept C, the instance function would return instances of a role R, i.e. individual pairs (a, b) which are connected by R according to the ontology $\mathcal{O}$.

The basic measures depend on the number of individuals occurying in an ontology. In order to compare quality of axioms with respect to different ontologies, one can normalise the basic measures to the fixed range $[0, 1]$. This can be done simply via dividing their values by the total number of individuals, see Definition 5.15. Moreover, the normalised basic measures can be given probabilistic meanings that facilitates their understanding.

Definition 5.15 (Normalised basic measures)**.** Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. The *normalised basic coverage, support, contradiction, assumption* of α are defined as follows:

$$bcov_{[0,1]}(\alpha, \mathcal{O}) := \frac{bcov(\alpha, \mathcal{O})}{|in(\mathcal{O})|}, \qquad bsup_{[0,1]}(\alpha, \mathcal{O}) := \frac{bsup(\alpha, \mathcal{O})}{|in(\mathcal{O})|},$$
$$bcnt_{[0,1]}(\alpha, \mathcal{O}) := \frac{bcnt(\alpha, \mathcal{O})}{|in(\mathcal{O})|}, \qquad basm_{[0,1]}(\alpha, \mathcal{O}) := \frac{basm(\alpha, \mathcal{O})}{|in(\mathcal{O})|}.$$

Normalised coverage is the sum of normalised support, contradiction, and assumption, see Lemma 5.10.

Lemma 5.10. $bcov_{[0,1]}(\alpha, \mathcal{O}) = bsup_{[0,1]}(\alpha, \mathcal{O}) + bcnt_{[0,1]}(\alpha, \mathcal{O}) + basm_{[0,1]}(\alpha, \mathcal{O})$.

Proof. It is a consequence of Definition 5.15 and Lemma 5.9. □

The normalised basic measures can be given probabilistic meanings, i.e. be represented via probabilities, see Lemma 5.11.

Lemma 5.11. *Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. Then*

$$bcov_{[0,1]}(\alpha, \mathcal{O}) = \mathbf{P}_{\mathcal{O}}(C), \qquad bsup_{[0,1]}(\alpha, \mathcal{O}) = \mathbf{P}_{\mathcal{O}}(C, D),$$
$$bcnt_{[0,1]}(\alpha, \mathcal{O}) = \mathbf{P}_{\mathcal{O}}(C, \neg D), \qquad basm_{[0,1]}(\alpha, \mathcal{O}) = \mathbf{P}_{\mathcal{O}}(C, ?D).$$

Proof. We only proof for support as others are analogous:
$bsup_{[0,1]}(\alpha, \mathcal{O}) = \frac{bsup(\alpha, \mathcal{O})}{|in(\mathcal{O})|}$ (by Definition 5.15)
$= \frac{|inst(C \sqcap D, \mathcal{O})|}{|in(\mathcal{O})|}$ (by Definition 5.14)
$= \frac{|inst(C, \mathcal{O}) \cap inst(D, \mathcal{O})|}{|in(\mathcal{O})|}$ (by Lemma 5.6)
$= \mathbf{P}_{\mathcal{O}}(C, D)$ (by Definition 5.13). □

Thus, the normalised basic measures have the following probabilistic meanings: coverage, support, contradiction, assumption are the probabilities of an individual to be an instance of C, C and D, C and $\neg D$, C and neither D nor $\neg D$, respectively.

5.3.1.3 Composite Basic Measures

The basic measures capture basic aspects of statistical quality of an axiom. In order to capture further aspects of statistical quality, we can benefit from related

work in ARM, see Section 2.2.5. Indeed, an axiom in DLs is similar to an association rule in ARM. Hence, we can attempt to transfer quality notions that are identified in ARM to DLs via adapting rule quality measures. The challenge of that adaptation is to respect the OWA of DLs, i.e. to consider the fact that there is $?C$ in addition to C and $\neg C$. Hence, the fact that an individual a is not an instance of C can be interpreted in two possible ways: a is an instance of $\neg C$; a is neither an instance of C nor $\neg C$, i.e. it belongs to $?C$. We translate rule measures to axiom measures via replacing $\mathbf{P}_{\mathcal{D}}(X_1, \ldots, X_n)$ with respective $\mathbf{P}_{\mathcal{O}}(\mathring{C}_1, \ldots, \mathring{C}_n)$ as follows:

- if a rule measure contains *no* negations, then each occurrence of X_i is replaced with C_i;
- if a rule measure contains negations, then each occurrence of X_i is replaced with C_i, each occurrence of $\neg X_i$ is replaced with $\neg C_i$ in the first translation and by $?C_i$ in the second translation.

Thus, if a rule measure contains no negations, it has one translation. If a rule measure contains negations, it has two translations that interpret negations differently: the first one as $\neg C_i$ and the second one as $?C_i$. Following this procedure, we translate confidence, lift, and conviction from Section 2.2.5 as follows, see Definition 5.16. Please notice that, since conviction contains negations, it has two translations: $bconv^{\neg}$ and $bconv^{?}$.

Definition 5.16 (Composite basic measures)**.** Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. The *basic confidence, lift, conviction* of α are defined, respectively, as follows:

$$bconf(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(C, D)}{\mathbf{P}_{\mathcal{O}}(C)}, \qquad blift(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(C, D)}{\mathbf{P}_{\mathcal{O}}(C) \cdot \mathbf{P}_{\mathcal{O}}(D)}$$

$$bconv^{\neg}(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(C) \cdot \mathbf{P}_{\mathcal{O}}(\neg D)}{\mathbf{P}_{\mathcal{O}}(C, \neg D)} \qquad bconv^{?}(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(C) \cdot \mathbf{P}_{\mathcal{O}}(?D)}{\mathbf{P}_{\mathcal{O}}(C, ?D)}.$$

The composite basic measures capture further aspects of statistical quality of an axiom in DLs and have meanings similar to their counterparts in ARM as follows.

- *Confidence* measures the proportion of instances of C which are also instances of D.

- *Lift* measures how frequently an individual is an instance of both C and D in comparison to the case when C and D are statistically independent events.

- *Negated conviction* ($bconv^{\neg}$) measures how frequently an individual is an instance of both C and $\neg D$ when C and $\neg D$ are statistically independent events in comparison to the actual frequency of being an instance of both C and $\neg D$.

- *Assumed conviction* ($bconv^{?}$) measures how frequently an individual is an instance of C and unknown instance of D when C and $?D$ are statistically independent events in comparison to the actual frequency of being an instance of C and unknown instance of D.

Confidence is in the range $[0, 1]$ and, hence, normalised by definition. Lift and conviction are in the range $[0, \infty]$. All measures are expected to be positive, i.e. higher values indicate better quality. While lift is symmetric, others are not. The composite basic measures can be written using basic measures, see Lemma 5.12.

Lemma 5.12. *Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. Then*

$$bconf(\alpha, \mathcal{O}) = \frac{bsup(\alpha, \mathcal{O})}{bcov(\alpha, \mathcal{O})}, \qquad blift(\alpha, \mathcal{O}) = \frac{bsup(\alpha, \mathcal{O})}{bcov(\alpha, \mathcal{O}) \cdot \mathbf{P}_{\mathcal{O}}(D)},$$
$$bconv^{\neg}(\alpha, \mathcal{O}) = \frac{bcov(\alpha, \mathcal{O}) \cdot \mathbf{P}_{\mathcal{O}}(\neg D)}{bcnt(\alpha, \mathcal{O})}, \quad bconv^{?}(\alpha, \mathcal{O}) = \frac{bcov(\alpha, \mathcal{O}) \cdot \mathbf{P}_{\mathcal{O}}(?D)}{asm(\alpha, \mathcal{O})}.$$

Proof. We only proof for confidence as others are analogous:
$bconf(\alpha, \mathcal{O}) = \frac{\mathbf{P}_{\mathcal{O}}(C,D)}{\mathbf{P}_{\mathcal{O}}(C)}$ (by Definition 5.16)
$= \frac{bsup_{[0,1]}(\alpha,\mathcal{O})}{bcov_{[0,1]}(\alpha,\mathcal{O})}$ (by Lemma 5.11)
$= \frac{bsup(\alpha,\mathcal{O})}{bcov(\alpha,\mathcal{O})}$ (by Definition 5.15). □

Other rule measures can be translated from ARM in the same way. For rule measures with negations other ways of translation are possible, i.e. some $\neg X_i$ can be replaced with $\neg C_i$ and others by $?C_i$. However, this translations may produce measures which are harder to interpret due to the mixed meanings of negation. Like the basic measures, the composite basic measures can be defined for a RI in the same way as they are defined for a GCI. Example 5.15 illustrates the composite basic measures.

Example 5.15. Recall Example 5.14 where we use the Kinship data and calculate the basic measures of two axioms: $\alpha_1 := \exists marriedTo.\top \sqsubseteq Mother$ and $\alpha_2 := \exists hasChild.\top \sqsubseteq Mother$. We now calculate the composite basic measures of these axioms using Lemma 5.12. We first calculate required probabilities (M stands for $Mother$): $\mathbf{P}_{\mathcal{O}}(M) = \frac{2}{7},\ \mathbf{P}_{\mathcal{O}}(\neg M) = \frac{3}{7},\ \mathbf{P}_{\mathcal{O}}(?M) = \frac{2}{7}$.

Then, we use the computed probabilities along with the basic measures in Example 5.14 to calculate the composite measures.

$$bconf(\alpha_1, \mathcal{O}) = \frac{2}{6} = \frac{1}{3} \qquad bconf(\alpha_2, \mathcal{O}) = \frac{2}{4} = \frac{1}{2}$$

$$blift(\alpha_1, \mathcal{O}) = \frac{2}{6 \cdot \frac{2}{7}} = \frac{7}{6} \qquad blift(\alpha_2, \mathcal{O}) = \frac{2}{4 \cdot \frac{2}{7}} = \frac{7}{4}$$

$$bconv^{\neg}(\alpha_1, \mathcal{O}) = \frac{6 \cdot \frac{3}{7}}{3} = \frac{6}{7} \qquad bconv^{\neg}(\alpha_2, \mathcal{O}) = \frac{4 \cdot \frac{3}{7}}{2} = \frac{6}{7}$$

$$bconv^{?}(\alpha_1, \mathcal{O}) = \frac{6 \cdot \frac{2}{7}}{1} = \frac{12}{7} \qquad bconv^{?}(\alpha_2, \mathcal{O}) = \frac{4 \cdot \frac{2}{7}}{0} = \infty$$

Thus, according to the composite basic measures, α_2 has better quality than α_1, since all its measures are higher and one is the same. This coincides with the evaluation by the basic measures.

5.3.1.4 Main Measures

Let us consider $\alpha := C \sqsubseteq D$ and $\alpha' := \neg D \sqsubseteq \neg C$. According to the semantics of DLs, $\alpha \equiv \alpha'$, which is commonly called the law of *contraposition*. The axiom α' is called the *contrapositive* of α. Hence, not only does the axiom $C \sqsubseteq D$ imply that all instances of C are instances of D but also that all instances of $\neg D$ are instances of $\neg C$. Please note that this is different for association rules in ARM, see Section 2.2.5. All statistical measures defined so far do not take this fact into account since they assume that α and α' are independent and generally return different values for them.

In order to respect the semantics of DLs better and take the law of contraposition into account, we define *main measures*. Let us consider $\overline{\alpha} := C \sqcup \neg D \sqsubseteq \neg C \sqcup D$ which is yet another syntactic variation of $\alpha := C \sqsubseteq D$, i.e. $\alpha \equiv \overline{\alpha}$ (this is easy to show, e.g. via truth tables). In contrast to α, the LHS and RHS of $\overline{\alpha}$ can be used to retrieve all instances relevant for quality measuring of both α and α'. We call $\overline{\alpha}$ the *cover*[3] of α, as its LHS and RHS "cover" all relevant instances.

[3]The cover of an axiom should not be confused with a covering axiom in OWL.

We refine the basic measures in Definition 5.14 as follows, see Definition 5.17.

Definition 5.17 (Main Measures)**.** Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI, and $\overline{\alpha} := C \sqcup \neg D \sqsubseteq \neg C \sqcup D$ the cover of α. The *main coverage, support, contradiction, assumption* of α are defined, respectively, as follows:

$$cov(\alpha, \mathcal{O}) := bcov(\overline{\alpha}, \mathcal{O})$$
$$sup(\alpha, \mathcal{O}) := bsup(\overline{\alpha}, \mathcal{O})$$
$$cnt(\alpha, \mathcal{O}) := bcnt(\overline{\alpha}, \mathcal{O})$$
$$asm(\alpha, \mathcal{O}) := basm(\overline{\alpha}, \mathcal{O})$$

In the following we omit "main" from the names of the main measures when it is clear from the context. In comparison to the basic measures, see Definition 5.14, their respective main measures additionally count individuals relevant for the contrapositive. They have similar meanings: support is positive; contradiction and assumption are negative; coverage is neither positive nor negative; support is symmetric and others are not. Example 5.16 shows how a measure can differ from its basic measure.

Example 5.16. Recall Example 5.14 where we evaluate the following axiom via the basic measures:

$$\alpha_2 := \exists hasChild.\top \sqsubseteq Mother.$$

According to the calculated values, α_2 makes no guesses, i.e. its basic assumption $basm(\alpha_2, \mathcal{O}) = 0$. However, its main assumption $asm(\alpha_2, \mathcal{O}) = |\{Arthur\}| = 1$. Indeed, as $Arthur$ is an instance of $\neg Mother$, the axiom α_2 assumes that $Arthur$ has no children, i.e. he is an instance of $\neg(\exists hasChild.\top)$. This is not counted by the basic assumption of α_2. As a result, α_2 is actually worse according to its main assumption than it is according to its basic assumption.

Main coverage, like basic coverage, equals the sum of main support, contradiction, and assumption, see Lemma 5.13.

Lemma 5.13. $cov(\alpha, \mathcal{O}) = sup(\alpha, \mathcal{O}) + cnt(\alpha, \mathcal{O}) + asm(\alpha, \mathcal{O})$.

Proof. Follows from Definition 5.17 and Lemma 5.9. □

In contrast to the basic measures, the main measures always return the same values for an axiom and its contrapositive, see Lemma 5.14. In other words, the main measures respect the semantics of DLs better than the basic measures.

Lemma 5.14. *Let $\alpha' := \neg D \sqsubseteq \neg C$ be the contrapositive of α. Then*

$$cov(\alpha, \mathcal{O}) = cov(\alpha', \mathcal{O}) \qquad sup(\alpha, \mathcal{O}) = sup(\alpha', \mathcal{O})$$
$$cnt(\alpha, \mathcal{O}) = cnt(\alpha', \mathcal{O}) \qquad asm(\alpha, \mathcal{O}) = asm(\alpha', \mathcal{O})$$

Proof. Follows from Definition 5.14 because any axiom α and its contrapositive α' have the same cover $\overline{\alpha}$. □

The main measures can be represented via their respective basic measures, see Lemma 5.15.

Lemma 5.15. *Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI, and $\alpha' := \neg D \sqsubseteq \neg C$ the contrapositive of α. Then*

(i) $cov(\alpha, \mathcal{O}) = bcov(\alpha, \mathcal{O}) + bcov(\alpha', \mathcal{O}) - bcnt(\alpha, \mathcal{O})$
(ii) $sup(\alpha, \mathcal{O}) = bsup(\alpha, \mathcal{O}) + bsup(\alpha', \mathcal{O})$
(iii) $cnt(\alpha, \mathcal{O}) = bcnt(\alpha, \mathcal{O}) = bcnt(\alpha', \mathcal{O})$
(iv) $asm(\alpha, \mathcal{O}) = basm(\alpha, \mathcal{O}) + basm(\alpha', \mathcal{O})$

Proof. (i) Let $\overline{\alpha} := C \sqcup \neg D \sqsubseteq \neg C \sqcup D$ be the cover of α. Then

$cov(\alpha, \mathcal{O}) = bcov(\overline{\alpha}, \mathcal{O})$ (by Definition 5.17)

$= |inst(C \sqcup \neg D, \mathcal{O})|$ (by Definition 5.14)

$= |inst(C, \mathcal{O}) \cup inst(\neg D, \mathcal{O})|$ (by Lemma 5.6)

$= |inst(C, \mathcal{O})| + |inst(\neg D, \mathcal{O})| - |inst(C, \mathcal{O}) \cap inst(\neg D, \mathcal{O})|$
(since $|S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2|$ for any sets S_1, S_2)

$= |inst(C, \mathcal{O})| + |inst(\neg D, \mathcal{O})| - |inst(C \sqcap \neg D, \mathcal{O})|$ (by Lemma 5.6)

$= bcov(\alpha, \mathcal{O}) + bcov(\alpha', \mathcal{O}) - bcnt(\alpha, \mathcal{O})$ (by Definition 5.14).

(ii) is proved similarly to (i) using $(C \sqcup \neg D) \sqcap (\neg C \sqcup D) \equiv (C \sqcap D) \sqcup (\neg C \sqcap \neg D)$.

(iii) is proved similarly to (i) using $(C \sqcup \neg D) \sqcap (C \sqcap \neg D) \equiv C \sqcap \neg D$.
By Lemma 5.14 $bcnt(\alpha, \mathcal{O}) = bcnt(\alpha', \mathcal{O})$.

(iv) $asm(\alpha, \mathcal{O}) = basm(\overline{\alpha}, \mathcal{O})$ (by Definition 5.17)

$= |inst(C \sqcup \neg D, \mathcal{O}) \cap inst(?(\neg C \sqcup D), \mathcal{O})|$ (by Definition 5.14)

$= |(inst(C, \mathcal{O}) \cup inst(\neg D, \mathcal{O})) \cap (inst(?(\neg C), \mathcal{O})$
$\cup inst(?D, \mathcal{O}) \setminus inst(\neg C, \mathcal{O}) \setminus inst(D, \mathcal{O}))|$ (by Lemma 5.6)

$= |(inst(C, \mathcal{O}) \cup inst(\neg D, \mathcal{O})) \cap (inst(?(\neg C), \mathcal{O}) \cup inst(?D, \mathcal{O}))|$
(since instances of $\neg C$ and D are removed by $inst(C, \mathcal{O}) \cup inst(\neg D, \mathcal{O})$)

$= |(inst(C, \mathcal{O}) \cap inst(?D, \mathcal{O})) \cup (inst(\neg D, \mathcal{O}) \cap inst(?C, \mathcal{O}))|$
(by propositional distribution)

$= |inst(C, \mathcal{O}) \cap inst(?D, \mathcal{O})| + |inst(\neg D, \mathcal{O}) \cap inst(?C, \mathcal{O})|$
$- |inst(C, \mathcal{O}) \cap inst(?D, \mathcal{O}) \cap inst(\neg D, \mathcal{O}) \cap inst(?C, \mathcal{O})|$
(since $|S_1 \cup S_2| = |S_1| + |S_2| - |S_1 \cap S_2|$ for any sets S_1, S_2)

$= |inst(C, \mathcal{O}) \cap inst(?D, \mathcal{O})| + |inst(\neg D, \mathcal{O}) \cap inst(?C, \mathcal{O})|$
(since an individual cannot be C and $?C$ simultaneously)

$= basm(\alpha, \mathcal{O}) + basm(\alpha', \mathcal{O})$ (by Definition 5.14).

□

Lemma 5.15 implies that it is sufficient to have a function calculating the basic measures and reuse that function to calculate the main measures. This is one of the reasons why we have presented and discussed the basic measures above. Another reason is that, while respecting the semantics better, the main measures require counting additional instances, i.e. they are more computationally costly. In Chapter 9, we investigate the difference in computation time between the main and basic measures.

As a consequence of Lemma 5.15, each main measure is greater than or equal to its respective basic measure. In other words, the basic measures are approximations from below for their respective main measures, see Lemma 5.16.

Lemma 5.16.

$$cov(\alpha, \mathcal{O}) \geq bcov(\alpha, \mathcal{O}) \qquad sup(\alpha, \mathcal{O}) \geq bsup(\alpha, \mathcal{O})$$
$$cnt(\alpha, \mathcal{O}) = bcnt(\alpha, \mathcal{O}) \qquad asm(\alpha, \mathcal{O}) \geq basm(\alpha, \mathcal{O})$$

Proof. Follows from Lemma 5.15 since $bcov(\alpha', \mathcal{O}) \geq bcnt(\alpha', \mathcal{O})$ and $bcnt(\alpha', \mathcal{O}) = bcnt(\alpha, \mathcal{O})$. □

Like the basic measures in Definition 5.15, the main measures can be normalised via dividing them by the total number of individuals, see Definition 5.18.

As for the basic measures, normalising the main measures facilitates their understanding and gives them probabilistic meanings.

Definition 5.18 (Normalised main measures)**.** Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. The *normalised main coverage, support, contradiction, assumption* are defined as follows:

$$cov_{[0,1]}(\alpha, \mathcal{O}) := \frac{cov(\alpha, \mathcal{O})}{|in(\mathcal{O})|}, \qquad sup_{[0,1]}(\alpha, \mathcal{O}) := \frac{sup(\alpha, \mathcal{O})}{|in(\mathcal{O})|},$$
$$cnt_{[0,1]}(\alpha, \mathcal{O}) := \frac{cnt(\alpha, \mathcal{O})}{|in(\mathcal{O})|}, \qquad asm_{[0,1]}(\alpha, \mathcal{O}) := \frac{asm(\alpha, \mathcal{O})}{|in(\mathcal{O})|}.$$

The normalised main measures are represented via probabilities as follows, see Lemma 5.17.

Lemma 5.17. *Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI, $\overline{C} := C \sqcup \neg D$, $\overline{D} := \neg C \sqcup D$. Then*

$$\begin{array}{lll} cov_{[0,1]}(\alpha, \mathcal{O}) & = \mathbf{P}_{\mathcal{O}}(\overline{C}) & = \mathbf{P}_{\mathcal{O}}(C) + \mathbf{P}_{\mathcal{O}}(\neg D) - \mathbf{P}_{\mathcal{O}}(C, \neg D) \\ sup_{[0,1]}(\alpha, \mathcal{O}) & = \mathbf{P}_{\mathcal{O}}(\overline{C}, \overline{D}) & = \mathbf{P}_{\mathcal{O}}(C, D) + \mathbf{P}_{\mathcal{O}}(\neg C, \neg D) \\ cnt_{[0,1]}(\alpha, \mathcal{O}) & = \mathbf{P}_{\mathcal{O}}(\overline{C}, \neg\overline{D}) & = \mathbf{P}_{\mathcal{O}}(C, \neg D) \\ asm_{[0,1]}(\alpha, \mathcal{O}) & = \mathbf{P}_{\mathcal{O}}(\overline{C}, ?\overline{D}) & = \mathbf{P}_{\mathcal{O}}(C, ?D) + \mathbf{P}_{\mathcal{O}}(\neg D, ?C) \end{array}$$

Proof. Let $\alpha' := \neg D \sqsubseteq \neg C$, $\overline{\alpha} := \overline{C} \sqsubseteq \overline{D}$. We only show for support as others are analogous:

$sup_{[0,1]}(\alpha, \mathcal{O}) = \frac{sup(\alpha, \mathcal{O})}{|in(\mathcal{O})|}$ (by Definition 5.18)
$= \frac{bsup(\overline{\alpha}, \mathcal{O})}{|in(\mathcal{O})|}$ (by Definition 5.17)
$= bsup_{[0,1]}(\overline{\alpha}, \mathcal{O})$ (by Definition 5.15)
$= \mathbf{P}_{\mathcal{O}}(\overline{C}, \overline{D})$ (by Lemma 5.11)
$= \frac{bsup(\alpha, \mathcal{O}) + bsup(\alpha', \mathcal{O})}{|in(\mathcal{O})|}$ (by Lemma 5.15)
$= bsup_{[0,1]}(\alpha, \mathcal{O}) + bsup_{[0,1]}(\alpha', \mathcal{O})$ (by Definition 5.15)
$= \mathbf{P}_{\mathcal{O}}(C, D) + \mathbf{P}_{\mathcal{O}}(\neg C, \neg D)$ (by Lemma 5.11). □

5.3.1.5 Composite Main Measures

Like the basic measures, the composite basic measures in Definition 5.16 can be refined to respect the semantics of DLs better, i.e. take the law of contraposition into account. For this purpose, we define *composite main measures*. Like the

composite basic measures in comparison to the basic measures, the composite main measures are aimed at capturing further aspects of statistical quality of an axiom in comparison to the main measures. As for the main measures, given an axiom $\alpha := C \sqsubseteq D$, we use its cover $\overline{\alpha} := C \sqcup \neg D \sqsubseteq \neg C \sqcup D$. In Definition 5.16, we simply substitute C for $C \sqcup \neg D$ and D for $\neg C \sqcup D$, see Definition 5.19.

Definition 5.19 (Composite main measures)**.** Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI, $\overline{\alpha} := \overline{C} \sqsubseteq \overline{D}$ the cover of α, where $\overline{C} := C \sqcup \neg D$, $\overline{D} := \neg C \sqcup D$. The *main confidence, lift, conviction* of α are defined, respectively, as follows:

$$conf(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(\overline{C}, \overline{D})}{\mathbf{P}_{\mathcal{O}}(\overline{C})}, \qquad lift(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(\overline{C}, \overline{D})}{\mathbf{P}_{\mathcal{O}}(\overline{C}) \cdot \mathbf{P}_{\mathcal{O}}(\overline{D})}$$

$$conv^{\neg}(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(\overline{C}) \cdot \mathbf{P}_{\mathcal{O}}(\neg \overline{D})}{\mathbf{P}_{\mathcal{O}}(\overline{C}, \neg \overline{D})} \qquad conv^{?}(\alpha, \mathcal{O}) := \frac{\mathbf{P}_{\mathcal{O}}(\overline{C}) \cdot \mathbf{P}_{\mathcal{O}}(?\overline{D})}{\mathbf{P}_{\mathcal{O}}(\overline{C}, ?\overline{D})}.$$

Like the main measures, the composite main measures respect the law of contraposition, i.e. treat an axiom α as being equivalent to its contrapositive α', see Lemma 5.18.

Lemma 5.18. *Let $\alpha' := \neg D \sqsubseteq \neg C$ be the contrapositive of α. Then*

$$conf(\alpha, \mathcal{O}) = conf(\alpha', \mathcal{O}) \qquad lift(\alpha, \mathcal{O}) = lift(\alpha', \mathcal{O})$$

$$conv^{\neg}(\alpha, \mathcal{O}) = conv^{\neg}(\alpha', \mathcal{O}) \qquad conv^{?}(\alpha, \mathcal{O}) = conv^{?}(\alpha', \mathcal{O})$$

Proof. Follows from Definition 5.19 because any axiom α and its contrapositive α' have the same cover $\overline{\alpha}$. □

Like the composite basic measures, the composite main measures are representable via the main measures and, hence, via the basic measures, see Lemma 5.19.

Lemma 5.19. *Let $\mathcal{O}$ be an ontology, $\alpha := C \sqsubseteq D$ a GCI. Then*

$$\text{(i) } conf(\alpha, \mathcal{O}) = \frac{sup(\alpha, \mathcal{O})}{cov(\alpha, \mathcal{O})}, \qquad \text{(ii) } lift(\alpha, \mathcal{O}) = \frac{sup(\alpha, \mathcal{O})}{cov(\alpha, \mathcal{O}) \cdot \mathbf{P}_{\mathcal{O}}(\overline{D})},$$

$$\text{(iii) } conv^{\neg}(\alpha, \mathcal{O}) = cov_{[0,1]}(\alpha, \mathcal{O}), \quad \text{(iv) } conv^{?}(\alpha, \mathcal{O}) = \frac{cov(\alpha, \mathcal{O}) \cdot \mathbf{P}_{\mathcal{O}}(?\overline{D})}{asm(\alpha, \mathcal{O})}.$$

Proof. (i) $conf(\alpha, \mathcal{O}) = \frac{\mathbf{P}_{\mathcal{O}}(\overline{C}, \overline{D})}{\mathbf{P}_{\mathcal{O}}(\overline{C})}$ (by Definition 5.19)

$= \frac{sup_{[0,1]}(\alpha, \mathcal{O})}{cov_{[0,1]}(\alpha, \mathcal{O})}$ (by Lemma 5.17)

$= \frac{sup(\alpha,\mathcal{O})}{cov(\alpha,\mathcal{O})}$ (by Definition 5.18)

(ii) is analogous to (i).

(iii) $conv^{\neg}(\alpha,\mathcal{O}) = \frac{\mathbf{P}_{\mathcal{O}}(\overline{C})\cdot\mathbf{P}_{\mathcal{O}}(\neg\overline{D})}{\mathbf{P}_{\mathcal{O}}(\overline{C},\neg\overline{D})}$ (by Definition 5.19)

$= \frac{\mathbf{P}_{\mathcal{O}}(\overline{C})\cdot\mathbf{P}_{\mathcal{O}}(\neg\overline{D})}{\mathbf{P}_{\mathcal{O}}(C,\neg D)}$ (by Lemma 5.17)

$= \frac{\mathbf{P}_{\mathcal{O}}(\overline{C})\cdot\mathbf{P}_{\mathcal{O}}(C\sqcap\neg D)}{\mathbf{P}_{\mathcal{O}}(C,\neg D)}$ (by substituting $\overline{D}$ with $\neg C \sqcup D$)

$= \frac{\mathbf{P}_{\mathcal{O}}(\overline{C})\cdot\mathbf{P}_{\mathcal{O}}(C,\neg D)}{\mathbf{P}_{\mathcal{O}}(C,\neg D)}$ (by Lemma 5.8)

$= \mathbf{P}_{\mathcal{O}}(\overline{C})$

$= cov_{[0,1]}(\alpha,\mathcal{O})$ (by Lemma 5.17)

(iv) is analogous to (i).

□

Please notice that the results of Lemma 5.19 are structurally similar to the results of Lemma 5.12, excluding the case (iii). Using Lemma 5.17, the composite main measures can also be written via the probabilities of C and D instead of $\overline{C}$ and $\overline{D}$.

While the main measures and the composite main measures take the law of contraposition into account, they do not respect other consequences of the semantics of DLs. More specifically, they do not guarantee to return the same value for all semantically equivalent axioms, see Example 5.17.

Example 5.17. Consider the axioms $\alpha_1 := A \sqcap B \sqsubseteq \bot$ and $\alpha_2 := A \sqsubseteq \neg B$. They are equivalent, i.e. $\alpha_1 \equiv \alpha_2$. The covers of α_1 and α_2 are, respectively, as follows:

$$\begin{aligned} \overline{\alpha}_1 &:= \top \sqsubseteq \neg A \sqcup \neg B \\ \overline{\alpha}_2 &:= A \sqcup B \sqsubseteq \neg A \sqcup \neg B \end{aligned}$$

Please notice that they are syntactically different, i.e. $\overline{\alpha}_1 \neq \overline{\alpha}_2$, while equivalent, i.e. $\overline{\alpha}_1 \equiv \overline{\alpha}_2$, since $\alpha_1 \equiv \alpha_2$. In particular, their LHSs, i.e. $\top$ and $A \sqcup B$, differ both syntactically and semantically. Consequently, they can have different instances that would result in different values for main measures, e.g. coverage, confidence, lift. For example, consider the ontology $\mathcal{O} := \{A(a), \neg B(a), C(b)\}$ (the TBox is empty).

5.3.2 Axiom Set Measures

The axiom measures defined above are rather simple: their calculation only requires instance retrieval. However, they are not *semantically faithful*: they do not return the same value for all logically equivalent axioms, i.e. $\alpha \equiv \alpha'$ does *not* imply $q(\alpha, \mathcal{O}) = q(\alpha', \mathcal{O})$, where q is an axiom measure. For example, this happens for the basic measures if $\alpha = C \sqsubseteq D$ and $\alpha' = \neg D \sqsubseteq \neg C$. While the main measures take this case into account, they are still not semantically faithful, see Example 5.17. Please note that the latter rather ascertains a general property of the aforementioned measures and does not imply that they are useless. They may still capture useful quality aspects that are hard to quantify in a fully semantically faithfull way. In fact, as our case study in Chapter 9 shows, these measures turn out to be crucial.

The axiom measures implicitly make the UNA, see Section 2.1.6, since they assume that all individuals are different while counting instances. This can be fixed by modifying the instance function, see Definition 5.12, such that it checks equality of individuals pairwise and counts equal individuals as one. However, this modification makes quality measuring computationally harder.

In addition, the axiom measures are only applicable to an axiom, but not to a set of axioms, i.e. an arbitrary hypothesis. At first, this seems easy to fix. For each axiom measure q, a respective axiom set measure q' can be defined as follows:

$$q'(H, \mathcal{O}) := \sum_{\alpha \in H} q(\alpha, \mathcal{O}).$$

However, q' is not semantically faithful because axioms are not independent within the set and can interact with each other, see Example 5.18. In the following, we give more examples of complex interactions within an axiom set that influence its quality.

Example 5.18. Consider the hypotheses $H := \{A \sqsubseteq B,\ B \sqsubseteq C,\ A \sqsubseteq C\}$ and $H' := \{A \sqsubseteq B,\ B \sqsubseteq C\}$ which are equivalent, i.e. $H' \equiv H$, but $q'(H, \mathcal{O}) \neq q'(H', \mathcal{O})$ in general. As a corner case, one can imagine a hypothesis H that consists of n syntactic variations of the same axiom α: its quality $q'(H, \mathcal{O}) = n \cdot q(\alpha, \mathcal{O})$ is deceptively high.

Preserving quality under equivalence, i.e. semantic faithfulness, is a good property for a quality measure because it ensures that all syntactic variations of a hypothesis are evaluated equally. In other words, no syntactic variation can be

evaluated higher than others and, consequently, perturb the hypothesis ordering. This may hide other interesting hypotheses.

We aim at defining semantically faithful measures for an axiom set (and hence for an axiom). In the following, we introduce two measures: *fitness* and *braveness*. Intuitively, fitness is similar to support: it accounts for the number of "facts" supporting the hypothesis. Braveness resembles assumption: it counts the number of "guesses" made by the hypothesis in the ABox due to the OWA.

5.3.2.1 Preliminary Definitions

To define fitness and braveness, we use some auxiliary notions: ontology projection, hypothesis assumption set, and ABox description length. These are defined below.

Ontology projection In order to formalise fitness, we need to specify which "facts" can be counted as supporting ones for a hypothesis. In DLs, besides explicit facts (ABox assertions), there are infinitely many implicit facts. Since we are unable to consider all of them, some focus is necessary. Such focus is provided by concepts $\mathbb{C}$ and roles $\mathbb{R}$ of interest which an ontology is "projected" to, see Definition 5.20.

Definition 5.20 (Ontology projection)**.** The *projection* $\pi(\mathcal{O}, \mathbb{C}, \mathbb{R})$ of an ontology $\mathcal{O}$ to concepts $\mathbb{C}$ and roles $\mathbb{R}$ is defined as follows:

$$\begin{aligned}
\pi_C(\mathcal{O}, \mathbb{C}) &:= \{C(a) \mid \mathcal{O} \models C(a) \ \wedge \ C \in \mathbb{C}^{\neg} \ \wedge \ a \in in(\mathcal{O})\} \\
\pi_R(\mathcal{O}, \mathbb{R}) &:= \{R(a,b) \mid \mathcal{O} \models R(a,b) \ \wedge \ R \in \mathbb{R} \ \wedge \ a, b \in in(\mathcal{O})\} \\
\pi(\mathcal{O}, \mathbb{C}, \mathbb{R}) &:= \pi_C(\mathcal{O}, \mathbb{C}) \ \cup \ \pi_R(\mathcal{O}, \mathbb{R})
\end{aligned}$$

where $\mathbb{C}^{\neg} := \{neg(C) \mid C \in \mathbb{C}\}$; $neg(C) := D$ if there is D such that $C = \neg D$ and $neg(C) := \neg C$ otherwise.

Thus, an ontology projection consists of (entailed) concept assertions over $\mathbb{C}^{\neg}$, which is $\mathbb{C}$ *closed under negation*, and role assertions over $\mathbb{R}$. It is essentially a fully *materialised* ABox with respect to $\mathbb{C}^{\neg}$ and $\mathbb{R}$, i.e. all implicit assertions of $\mathbb{C}^{\neg}$ and $\mathbb{R}$ in $\mathcal{O}$ are made explicit. Given concepts $\mathbb{C}$ and roles $\mathbb{R}$, equivalent ontologies have the same projections, see Lemma 5.20.

Lemma 5.20. *Let $\mathcal{O}$ and $\mathcal{O}'$ be ontologies, $\mathbb{C}$ concepts, $\mathbb{R}$ roles. Then, $\mathcal{O} \equiv \mathcal{O}'$ implies $\pi(\mathcal{O}, \mathbb{C}, \mathbb{R}) = \pi(\mathcal{O}', \mathbb{C}, \mathbb{R})$.*

Proof. Follows from Definition 5.20 because equivalent ontologies have the same models and, hence, the same entailments. □

Since a projection is an ABox, it can be illustrated as a graph, see Example 5.19.

Example 5.19. Consider the ontology $\mathcal{O} :=$ Kinship from Example 2.3. Concepts $\mathbb{C}$ and roles $\mathbb{R}$ are as follows (F, M, c, m stand for $Father$, $Mother$, $hasChild$, $marriedTo$, respectively):

$$\mathbb{C} := \{F,\ M,\ \exists c.\top\},\ \mathbb{R} := \{m,\ c\}.$$

The projection of $\mathcal{O}$ to $\mathbb{C}$ and $\mathbb{R}$ is shown in Figure 5.1. Please compare Figure 5.1 with Figure 2.1 and notice that, while some assertions are ignored, e.g. the assertions of $Human$ and $hasParent$, others are made explicit, e.g. the assertions of $\exists c.\top$ and c. The assertions of $\exists c.\top$ encode information of having a c relation which is lifted from the instance to concept level.

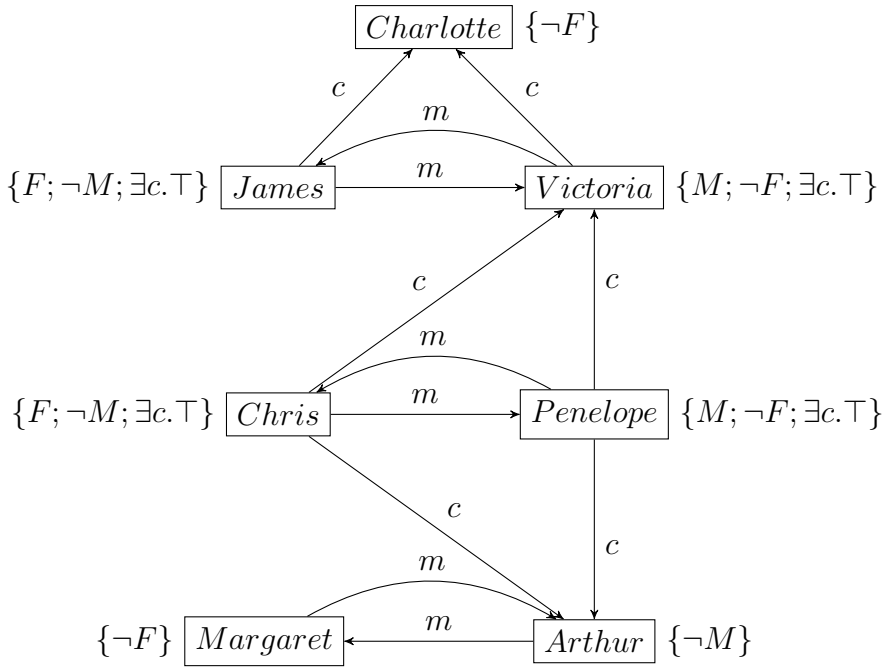


Figure 5.1: Projection for Example 5.19

Hypothesis Assumption Set Similarly to fitness, in order to formalise braveness, we first need to specify what "guesses" a hypothesis can make. Similarly to data projection, those "guesses" are assertions and there are infinitely many of them. Therefore, a focus is required which is again given by concepts $\mathbb{C}$ and roles $\mathbb{R}$, see Definition 5.21.

Definition 5.21 (Hypothesis assumption set)**.** The *assumption set* $\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$ of a consistent hypothesis H in an ontology $\mathcal{O}$ given concepts $\mathbb{C}$ and roles $\mathbb{R}$ is defined as follows:

$$\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) := \pi(\mathcal{O} \cup H, \mathbb{C}, \mathbb{R}) \setminus \pi(\mathcal{O}, \mathbb{C}, \mathbb{R}).$$

Thus, an assumption set[4] is the difference between the projection of $\mathcal{O} \cup H$ and the projection of $\mathcal{O}$. In other words, it is a set of concept assertions over $\mathbb{C}^{\neg}$ and role assertions over $\mathbb{R}$ which an ontology alone does not entail but, together with a hypothesis, does entail. Consequently, an assumption set never overlaps with a projection. Lemma 5.21 states the properties of an assumption set.

Lemma 5.21. *Let $\mathcal{O}$ be an ontology, H a hypothesis, $\mathbb{C}$ concepts, $\mathbb{R}$ roles. Then*

(i) $\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) \cap \pi(\mathcal{O}, \mathbb{C}, \mathbb{R}) = \emptyset$*;*

(ii) $\mathcal{O} \models H$ *implies* $\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \emptyset$*;*

(iii) $\psi(\emptyset, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \emptyset$*.*

Proof. (i) follows immediately from Definition 5.21.

(ii) $\mathcal{O} \models H$ implies $\mathcal{O} \cup H \equiv \mathcal{O}$. Hence $\pi(\mathcal{O} \cup H, \mathbb{C}, \mathbb{R}) = \pi(\mathcal{O}, \mathbb{C}, \mathbb{R})$ by Lemma 5.20. Therefore $\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \emptyset$ by Definition 5.21.

(iii) follows from (ii).

□

Please notice that an assumption set is only defined for a hypothesis consistent with the ontology, i.e. $\mathcal{O} \cup H$ is consistent, since otherwise entailments are not useful.[5] Since an assumption set is an ABox, it can be illustrated as a graph, see Example 5.20.

[4]The assumption set of a hypothesis should not be confused with its measured assumption even though these capture similar notions. We describe the relationship between them in the following.

[5]An inconsistent ontology entails everything.

Example 5.20. Consider the ontology $\mathcal{O} := \mathsf{Kinship}$, sets $\mathbb{C}$ and $\mathbb{R}$ from Example 5.19, and the hypothesis

$$H := \{\neg F \sqsubseteq M,\ M \sqsubseteq \exists c.\top\},$$

where F, M, c stand for $Father$, $Mother$, $hasChild$. The assumption set of H is as follows:

$$\begin{aligned}\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \{&M(Charlotte), M(Margaret), F(Arthur),\\ &(\exists c.\top)(Charlotte), (\exists c.\top)(Margaret)\}.\end{aligned}$$

It is illustrated in Figure 5.2. $Charlotte$ and $Margaret$ are assumed to be instances of M, since they are instances of $\neg F$. In addition, they are assumed to have some children, i.e. to be instances of $\exists c.\top$, since they are assumed instances of M. Please notice that, due to contraposition, $Arthur$ is assumed to be an instance of F since he is an instance of $\neg M$.

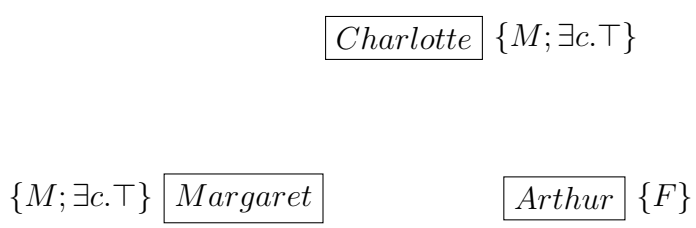


Figure 5.2: Assumption set for Example 5.20

Role assertions (edges) can be assumed in the same way due to RIs. For example, imagine if the relation c between $Chris$ and $Arthur$ is unknown and consider the hypothesis $H' := \{m \circ c \sqsubseteq c\}$ (m stands for $marriedTo$) that states "if x is married to y and y has a child z, then x has a child z". Then, the relation c between $Chris$ and $Arthur$ would be assumed by H'.

An assumption set resembles the main assumption measure in Definition 5.17. One might expect that the size of the assumption set of an axiom α equals its main assumption, i.e. $|\psi(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R})| = asm(\alpha, \mathcal{O})$. However, this is not true because, in contrast to the main assumption measure, the assumption set is semantically faithful, i.e. the same for all equivalent hypotheses, see Lemma 5.22.

Lemma 5.22. *Let $\mathcal{O}$ be an ontology; $\mathbb{C}$ and $\mathbb{R}$ concepts and roles, respectively; H and H' hypotheses consistent with $\mathcal{O}$. Then, $H \equiv H'$ implies $\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \psi(H', \mathcal{O}, \mathbb{C}, \mathbb{R})$.*

Proof. By Definition 5.21 $\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) := \pi(\mathcal{O} \cup H, \mathbb{C}, \mathbb{R}) - \pi(\mathcal{O}, \mathbb{C}, \mathbb{R})$. If $H \equiv H'$, then $\mathcal{O} \cup H \equiv \mathcal{O} \cup H'$. Hence $\pi(\mathcal{O} \cup H, \mathbb{C}, \mathbb{R}) = \pi(\mathcal{O} \cup H', \mathbb{C}, \mathbb{R})$ by Lemma 5.20, $\psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \psi(H', \mathcal{O}, \mathbb{C}, \mathbb{R})$ by Definition 5.21. □

Another difference between the size of an assumption set and the main assumption measure is that ψ depends on $\mathcal{O}, \mathbb{C}, \mathbb{R}$. More specifically, an axiom $\alpha := C \sqsubseteq D$ can assume more assertions of $\mathbb{C}$ ($\mathbb{R}$) due to the TBox of $\mathcal{O}$, see Example 5.21.

Example 5.21. Consider the ontology $\mathcal{O} := \{B \sqsubseteq C,\ A(a), B(a), A(b)\}$, concepts $\mathbb{C} := \{A, B, C\}$, roles $\mathbb{R} := \emptyset$, and axiom $\alpha := A \sqsubseteq B$. Its main assumption is $asm(\alpha, \mathcal{O}) = |\{b\}| = 1$, while its assumption set size is $|\psi(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R})| = |\{C(a), B(b), C(b)\}| = 3$.

Description Length In order to evaluate the statistical quality of a set of axioms, we use the idea that axioms can encode regularities in the data. Those regularities can be used to "compress" the data, i.e. to present it in a shorter way. This is the fundamental principle of the *minimum description length induction* [CW94, VL08] in ML. According to it, the better a hypothesis fits the data, the shorter description of the data it provides. In other words, facts that become redundant in the presence of a hypothesis support it. Thus, we need to measure redundancy of the data (ABox) with respect to a hypothesis in DLs.

A standard way of measuring description length in ML is using syntactic measures. However, syntactic measures do not respect interactions between axioms within the set. Therefore, we introduce a semantic measure of description length,[6] see Definition 5.22.

Definition 5.22 (Description length)**.** The *description length* of an ABox $\mathcal{B}$ with respect to an ontology $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ is defined as follows:

$$dlen(\mathcal{B}, \mathcal{O}) := min\{\ell(\mathcal{B}') \mid \mathcal{B}' \cup \mathcal{O} \equiv \mathcal{B} \cup \mathcal{O}\}.$$

Please notice that an ABox $\mathcal{B}$ is different from an ABox $\mathcal{A}$ of $\mathcal{O}$. An ontology $\mathcal{O}$ can contain a TBox only, i.e. $\mathcal{O} = \mathcal{T}$, and can be empty, i.e. $\mathcal{O} = \emptyset$. Informally, the description length of $\mathcal{B}$ with respect to $\mathcal{O}$ is the length of a minimal ABox $\mathcal{B}'$ which, if paired with $\mathcal{O}$, is equivalent to $\mathcal{B}$ paired with $\mathcal{O}$. In other words, it is

[6] Description length of an ABox should not be confused with syntactic length of a hypothesis.

the length of a *non-redundant* counterpart of $\mathcal{B}$ given $\mathcal{O}$. Hence, the description length can be used to measure redundancy of any ABox $\mathcal{B}$ given an ontology $\mathcal{O}$ as follows: $\ell(\mathcal{B}) - dlen(\mathcal{B}, \mathcal{O})$. In general, a minimal ABox $\mathcal{B}'$ is not unique and not a subset of $\mathcal{B}$, see Example 5.22.

Example 5.22. Consider the ABox $\mathcal{B}$ in Figure 5.1 and the TBox $\mathcal{T}$ of Kinship, see Example 2.3. A minimal ABox $\mathcal{B}'$ for $\mathcal{B}$ with respect to $\mathcal{T}$ (not the full ontology) is shown in Figure 5.3 (F, M, c, m stand for $Father$, $Mother$, $hasChild$, $marriedTo$, respectively). Please notice that some information becomes redundant and can be discarded in the presence of the TBox: $\neg M$ because $\mathcal{T} \models F \sqsubseteq \neg M$, $\neg F$ because $\mathcal{T} \models M \models \neg F$, m because $\mathcal{T} \models m \sqsubseteq m^-$ (a symmetric role). The size of $\mathcal{B}'$ is $\ell(\mathcal{B}') = 20$, hence $dlen(\mathcal{B}, \mathcal{T}) = 20$. A minimal ABox is not unique: each $hasChild$ (c) relation can be replaced with the $hasParent$ (p) relation going in the opposite direction because $\mathcal{T} \models c \sqsubseteq p^-$.

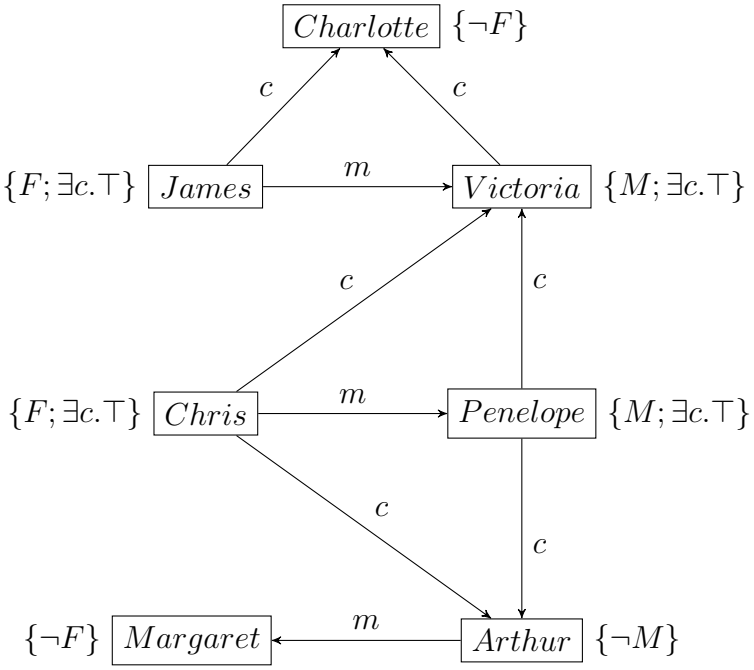


Figure 5.3: Minimal ABox for Example 5.22

Description length has the following properties: (i) description length is the same with respect to all equivalent ontologies; (ii) equivalent ABoxes have the same description length; (iii) the description length given a stronger ontology cannot be lower than the one given a weaker ontology, see Lemma 5.23.

Lemma 5.23. *Let $\mathcal{O}$, $\mathcal{O}_1$, $\mathcal{O}_2$ be ontologies, $\mathcal{B}$, $\mathcal{B}_1$, $\mathcal{B}_2$ ABoxes. Then*

(i) $\mathcal{O}_1 \equiv \mathcal{O}_2$ *implies* $dlen(\mathcal{B}, \mathcal{O}_1) = dlen(\mathcal{B}, \mathcal{O}_1)$*;*

(ii) $\mathcal{B}_1 \equiv \mathcal{B}_2$ *implies* $dlen(\mathcal{B}_1, \mathcal{O}) = dlen(\mathcal{B}_2, \mathcal{O})$*;*

(iii) $\mathcal{O}_1 \rhd \mathcal{O}_2$ *implies* $dlen(\mathcal{B}, \mathcal{O}_1) \leq dlen(\mathcal{B}, \mathcal{O}_2)$.

Proof. (i) $dlen(\mathcal{B}, \mathcal{O}_1) = min\{\ell(\mathcal{B}') \mid \mathcal{B}' \cup \mathcal{O}_1 \equiv \mathcal{B} \cup \mathcal{O}_1\}$
(by Definition 5.22)

$= min\{\ell(\mathcal{B}') \mid \mathcal{B}' \cup \mathcal{O}_2 \equiv \mathcal{B} \cup \mathcal{O}_2\}$
(since $\mathcal{O}_1 \equiv \mathcal{O}_2$ implies $\mathcal{B} \cup \mathcal{O}_1 \equiv \mathcal{B} \cup \mathcal{O}_2$)

$= dlen(\mathcal{B}, \mathcal{O}_2)$ (by Definition 5.22).

(ii) (by contradiction) Let $\mathcal{B}'_1$ and $\mathcal{B}'_2$ be minimal ABoxes for $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, given $\mathcal{O}$. Assume $dlen(\mathcal{B}_1, \mathcal{O}) \neq dlen(\mathcal{B}_2, \mathcal{O})$. Hence $\ell(\mathcal{B}'_1) \neq \ell(\mathcal{B}'_2)$ by Definition 5.22. If $\ell(\mathcal{B}'_1) < \ell(\mathcal{B}'_2)$, then $\mathcal{B}'_2$ is not minimal for $\mathcal{B}_2$ because $\mathcal{B}'_1 \cup \mathcal{O} \equiv \mathcal{B}_2 \cup \mathcal{O}$ (by Definition 5.22 and $\mathcal{B}_1 \equiv \mathcal{B}_2$). Hence $\ell(\mathcal{B}'_1) > \ell(\mathcal{B}'_2)$. However, $\ell(\mathcal{B}'_1) > \ell(\mathcal{B}'_2)$ cannot happen which can be shown by analogy with $\ell(\mathcal{B}'_1) < \ell(\mathcal{B}'_2)$. Therefore $\ell(\mathcal{B}'_1) = \ell(\mathcal{B}'_2)$. Hence $dlen(\mathcal{B}_1, \mathcal{O}) = dlen(\mathcal{B}_2, \mathcal{O})$ by Definition 5.22.

(iii) (by contradiction) Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be minimal ABoxes for $\mathcal{B}$ given $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. Assume $dlen(\mathcal{B}, \mathcal{O}_1) > dlen(\mathcal{B}, \mathcal{O}_1)$. Hence $\ell(\mathcal{B}_1) > \ell(\mathcal{B}_2)$ by Definition 5.22. Then

$\mathcal{B}_1 \cup \mathcal{O}_1 \equiv \mathcal{B} \cup \mathcal{O}_1$ (by Definition 5.22)

$\mathcal{B}_1 \cup \mathcal{O}_1 \cup \mathcal{O}_2 \equiv \mathcal{B} \cup \mathcal{O}_1 \cup \mathcal{O}_2$ (by the semantics of DLs)

$\mathcal{B}_1 \cup \mathcal{O}_1 \cup \mathcal{O}_2 \equiv \mathcal{B}_2 \cup \mathcal{O}_1 \cup \mathcal{O}_2$ (since $\mathcal{B}_2 \cup \mathcal{O}_2 \equiv \mathcal{B} \cup \mathcal{O}_2$ by Definition 5.22)

$\mathcal{B}_1 \cup \mathcal{O}_1 \equiv \mathcal{B}_2 \cup \mathcal{O}_1$ (since $\mathcal{O}_1 \models \mathcal{O}_2$ by Definition 5.9 and $\mathcal{O}_1 \cup \mathcal{O}_2 \equiv \mathcal{O}_1$ by the semantics of DLs)

$\mathcal{B} \cup \mathcal{O}_1 \equiv \mathcal{B}_2 \cup \mathcal{O}_1$ (since $\mathcal{B}_1 \cup \mathcal{O}_1 \equiv \mathcal{B} \cup \mathcal{O}_1$ by Definition 5.22)

Hence, $\mathcal{B}_1$ is not minimal for $\mathcal{B}$ given $\mathcal{O}_1$. Therefore $\ell(\mathcal{B}_1) \leq \ell(\mathcal{B}_2)$. Hence $dlen(\mathcal{B}, \mathcal{O}_1) \leq dlen(\mathcal{B}, \mathcal{O}_2)$.

□

Thus, description length is a semantically faithful measure. Moreover, description length does *not* make the UNA since it measures the size of a minimal ABox: if some individuals are known to be equal, their assertions are minimised.

5.3.2.2 Fitness and Braveness

We use the ontology projection to count facts supporting a hypothesis, i.e. to measure its fitness, and the assumption set of the hypothesis to count its guesses, i.e. to measure its braveness. In this context the description length provides a semantically faithful way of counting. The fitness and braveness of a hypothesis are defined as follows, see Definition 5.23.

Definition 5.23 (Fitness and braveness)**.** Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, $\mathbb{C}$ concepts, $\mathbb{R}$ roles, H a hypothesis consistent with $\mathcal{O}$, $\pi := \pi(\mathcal{O}, \mathbb{C}, \mathbb{R})$ the projection of $\mathcal{O}$, $\psi_H := \psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$ the assumption set of H. Then, the *fitness* and *braveness* of H are defined as follows:

$$\begin{aligned} fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) &:= dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup H) \\ bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) &:= dlen(\psi_H, \mathcal{O}) \end{aligned}$$

According to Definition 5.23, the fitness of a hypothesis measures how much the projection shrinks in the presence of the hypothesis. The more it shrinks, the more redundant the data becomes and the better the hypothesis "explains" it. Thus, fitness shows the *explanatory power* of a hypothesis. While measuring how well a hypothesis fits known facts (the projection), fitness assumes that the hypothesis is correct on unknown facts, i.e. its guesses are right.

The braveness of a hypothesis is the number of its "native" guesses which are counted by the description length of the hypothesis assumption set. Indeed, as Example 5.21 shows, some guesses of a hypothesis can be the result of its interaction with the TBox. Therefore, we should minimise an assumption set with respect to the ontology.

Intuitively, fitness resembles support and braveness resembles assumption, see Definition 5.17. Nonetheless, there are noticeable differences. In contrast to the axiom measures, fitness and braveness count the length of assertions, while support and assumption count individuals. In addition, fitness is asymmetric, i.e. swapping the LHS and RHS of an axiom can change the value, while support is symmetric. In contrast to support and assumption, fitness and braveness *avoid* the UNA since they are based on description length avoiding it.

Although fitness seems to be similar to support, it has a different meaning: fitness evaluates how well a hypothesis explains the ABox given the TBox. More specifically, the TBox can enforce a hypothesis such that the latter explains more

facts and, hence, has higher fitness together with the TBox than alone. The TBox can also explain many facts by itself and make a hypothesis less supported. In other words, fitness evaluates how many facts a hypothesis together with the TBox can explain that the TBox alone cannot explain.

Another difference with the axiom measures is that fitness and braveness are only defined for a hypothesis consistent with the ontology. While this is unavoidable for assumption, it can be relaxed for fitness: a hypothesis H should only be consistent with $\pi \cup \mathcal{T}$, i.e. the projection along with the TBox instead of the ontology. It is a relaxation because $\pi \cup \mathcal{T}$ can be weaker than $\mathcal{O}$, since π is determined by $\mathbb{C}$ and $\mathbb{R}$. Therefore, H can be consistent with $\pi \cup \mathcal{T}$ and inconsistent with $\mathcal{O}$ (but not vice versa).

5.3.2.3 Examples of Fitness and Braveness

Let us illustrate fitness and braveness by examples comparing them with support and assumption. Example 5.23 shows how these measures are calculated for the hypothesis in Example 5.20 given the ontology Kinship.

Example 5.23. Consider the projection π of the ontology $\mathcal{O} :=$ Kinship, see Example 5.19. Its description length with respect to the TBox $\mathcal{T}$ of $\mathcal{O}$ is $dlen(\pi, \mathcal{T}) = 20$, see Example 5.22. Consider the hypothesis in Example 5.20 (F, M, c stand for $Father$, $Mother$, $hasChild$, respectively):

$$H := \{\neg F \sqsubseteq M,\ M \sqsubseteq \exists c.\top\}.$$

The description length of π with respect to $\mathcal{T} \cup H$ is $dlen(\pi, \mathcal{T} \cup H) = 18$ because assertions of $\exists c.\top$ for $Penelope$ and $Victoria$ become redundant as they are instances of M. A possible minimal ABox is the one shown in Figure 5.3, where assertions of $\exists c.\top$ are discarded for $Penelope$ and $Victoria$. Thus

$$fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup H) = 2.$$

Please notice that the axiom $\alpha_1 := \neg F \sqsubseteq M$ does not make any assertions redundant: assertions of $\neg F$ are redundant due to $\mathcal{T}$ since $\mathcal{T} \models M \sqsubseteq \neg F$. This

is different for support:

$$fit(\{\alpha_1\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 0$$
$$sup(\alpha_1, \mathcal{O}) = |inst(\neg F \sqcap M, \mathcal{O})| + |inst(F \sqcap \neg M, \mathcal{O})| = 4$$

Thus, the support of α_1 is higher than its fitness. The axiom $\alpha_2 := M \sqsubseteq \exists c.\top$ has the same fitness and support:

$$fit(\{\alpha_2\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 2$$
$$sup(\alpha_2, \mathcal{O}) = |inst(M \sqcap \exists c.\top, \mathcal{O})| = 2$$

The assumption set ψ_H of H is listed in Example 5.20. Its description length $dlen(\psi_H, \mathcal{O}) = 5$ as $\mathcal{O}$ does not force H to make additional guesses (see Example 5.21 where it does). Thus

$$bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\psi_H, \mathcal{O}) = 5.$$

The braveness of α_1 and α_2 coincides with the assumption of α_1 and α_2, respectively. However, the sum of the assumptions of α_1 and α_2 is lower than the braveness of $H = \{\alpha_1, \alpha_2\}$ because guesses of α_1 cause additional guesses of α_2:

$$bra(\{\alpha_1\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 3,\ asm(\alpha_1, \mathcal{O}) = 3,$$
$$bra(\{\alpha_2\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 0,\ asm(\alpha_2, \mathcal{O}) = 0,$$
$$asm(\alpha_1, \mathcal{O}) + asm(\alpha_2, \mathcal{O}) < bra(\{\alpha_1, \alpha_2\}, \mathcal{O}, \mathbb{C}, \mathbb{R})$$

Thus, axioms within the set can make more guesses than they make separately.

Example 5.24 shows other differences between fitness and support.

Example 5.24. Consider the ontology $\mathcal{O} :=$ Kinship, see Example 2.3. Suppose we are only interested in evaluating hypotheses consisting of GCIs, i.e. we set $\mathbb{R} := \emptyset$, and the following concepts $\mathbb{C}$ are of interest to us (H, M, p stand for $Human$, Man, $hasParent$, respectively):

$$\mathbb{C} := \{H, \exists p.M, \exists p.H\}.$$

Since $\mathbb{R} := \emptyset$,[7] the projection π of $\mathcal{O}$ to $\mathbb{C}$ can be illustrated as a table, see

[7] We would normally include p in $\mathbb{R}$ as it occurs in $\mathbb{C}$ but do not do so for the sake of brevity.

Table 5.2. A question mark "?" shows that it is unknown whether an individual is an instance of a concept (indicated by "X") or it is not (indicated by a blank space). It is the same notation as we used above, e.g. in Table 5.1.

	H	$\exists p.M$	$\exists p.H$
$Charlotte$	X	X	X
$James$	X	?	?
$Victoria$	X	X	X
$Chris$	X	?	?
$Penelope$	X	?	?
$Margaret$	X	?	?
$Arthur$	X	X	X

Table 5.2: Kinship projection for Example 5.24

Consider the axiom $\alpha := H \sqsubseteq \exists p.H$ stating that "every human has a parent who is a human". Let us calculate the fitness and support for α. Please recall that $\mathcal{T} \models M \sqsubseteq H$. Hence $\mathcal{T} \models \exists p.M \sqsubseteq \exists p.H$. Therefore $dlen(\pi, \mathcal{T}) = 10$ and $dlen(\pi, \mathcal{T} \cup \{\alpha\}) = 10$ because all assertions of $\exists p.H$ are redundant due to $\mathcal{T}$. Consequently, the hypothesis $\{\alpha\}$ does not explain any additional facts, while the axiom α does have supporting instances. The fitness and support of α are as follows:

$$fit(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup \{\alpha\}) = 0$$
$$sup(\alpha, \mathcal{O}) = |inst(H \sqcap \exists p.H,\ \mathcal{O})| = 3$$

Thus, fitness shows how many additional facts a hypothesis explains given the TBox, while support simply counts supporting instances. Since α explains the same facts as $\mathcal{T}$ does, the fitness of α equals zero, while its support equals 3.

5.3.2.4 Properties of Fitness and Braveness

In contrast to axiom measures, fitness and braveness are semantically faithful, i.e. return the same value for all equivalent hypotheses. In other words, fitness and braveness are preserved under equivalence, see Theorem 5.1.

Theorem 5.1 (Fitness and braveness preserved under equivalence)**.** *Let $\mathcal{O}$ be an ontology, $\mathbb{C}$ concepts, $\mathbb{R}$ roles, H and H' hypotheses. Then, $H \equiv H'$ implies*

(i) $fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = fit(H', \mathcal{O}, \mathbb{C}, \mathbb{R})$ *and*

(ii) $bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = bra(H', \mathcal{O}, \mathbb{C}, \mathbb{R})$.

Proof. Let $\pi := \pi(\mathcal{O}, \mathbb{C}, \mathbb{R})$, $\psi_H := \psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$, $\psi_{H'} := \psi(H', \mathcal{O}, \mathbb{C}, \mathbb{R})$.

(i) $fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup H)$ (by Definition 5.23)
$= dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup H')$
(by Lemma 5.23 because $H \equiv H'$ implies $\mathcal{T} \cup H \equiv \mathcal{T} \cup H'$)
$= fit(H', \mathcal{O}, \mathbb{C}, \mathbb{R})$ (by Definition 5.23).

(ii) $bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\psi_H, \mathcal{O})$ (by Definition 5.23)
$= dlen(\psi_{H'}, \mathcal{O})$ (by Lemma 5.22)
$= bra(H', \mathcal{O}, \mathbb{C}, \mathbb{R})$ (by Definition 5.23).

□

Theorem 5.1 guarantees that all syntactic variations of a hypothesis are evaluated equally by fitness and braveness, i.e. no syntactic variation is evaluated higher than others. This implies that a redundant hypothesis always has the same quality as its non-redundant counterpart.

According to Definition 5.23, fitness and braveness are based on counting new entailments in a projection and assumption set, respectively. In this sense, they resemble complexity, even though the latter is a measure of logical quality, see Definition 5.11. Moreover, despite the fact that fitness and braveness are measures of statistical quality, they have similar logical properties as complexity has, see Lemma 5.24.

Lemma 5.24. *Let $\mathcal{O}$ be an ontology, $\mathbb{C}$ concepts, $\mathbb{R}$ roles, H and H' hypotheses.*

(i) *$H' \rhd H$ implies $fit(H', \mathcal{O}, \mathbb{C}, \mathbb{R}) \geq fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$ and $bra(H', \mathcal{O}, \mathbb{C}, \mathbb{R}) \geq bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$;*

(ii) *$\mathcal{O} \models H$ implies $fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 0$ and $bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 0$;*

(iii) *$fit(\emptyset, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 0$, $bra(\emptyset, \mathcal{O}, \mathbb{C}, \mathbb{R}) = 0$;*

(iv) *$fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) \geq 0$, $bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) \geq 0$.*

Proof. (i) For fitness:

$fit(H', \mathcal{O}, \mathbb{C}, \mathbb{R}) - fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$

$= dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup H') - dlen(\pi, \mathcal{T}) + dlen(\pi, \mathcal{T} \cup H)$
(by Definition 5.23)

$= dlen(\pi, \mathcal{T} \cup H) - dlen(\pi, \mathcal{T} \cup H')$

≥ 0 (by Lemma 5.23 because $H' \rhd H$ implies either $\mathcal{T} \cup H' \rhd \mathcal{T} \cup H$ or $\mathcal{T} \cup H' \equiv \mathcal{T} \cup H$ by Lemma 5.3).

For braveness:

$bra(H', \mathcal{O}, \mathbb{C}, \mathbb{R}) - bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$

$= dlen(\psi_{H'}, \mathcal{O}) - dlen(\psi_H, \mathcal{O})$ (by Definition 5.23)

≥ 0 (since $H' \rhd H$ implies $\psi_H \subseteq \psi_{H'}$ by Definition 5.21).

(ii) For fitness:

$fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup H)$ (by Definition 5.23)

$= dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T})$ (since $\mathcal{O} \models H$ implies $\mathcal{T} \cup H \equiv \mathcal{T}$)

$= 0.$

For braveness:

$bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\psi_H, \mathcal{O})$ (by Definition 5.23)

$= dlen(\emptyset, \mathcal{O})$ (by Lemma 5.21)

$= 0$ (by Definition 5.22).

(iii) follows from (ii) because any $\mathcal{O} \models \emptyset$.

(iv) follows from (i) and (ii) because either $H \rhd \emptyset$ or $H \equiv \emptyset$.

□

Informally, fitness and braveness have the following properties according to Lemma 5.24: (i) a stronger hypothesis cannot have lower fitness or braveness; (ii) the fitness and braveness of an entailed hypothesis equal zero; (iii) fitness and braveness of the empty hypothesis equal zero; (iv) fitness and braveness cannot be below zero. These properties can be explained by the monotonicity[8] of DLs. For example, (iv) holds because adding a hypothesis to the ontology cannot remove its entailments, i.e. this either causes new entailments or makes no changes.

As fitness and braveness capture similar quality characteristics as support and assumption, respectively, it is worthwhile to investigate whether there is a formal

[8] A logic is monotonic if adding a formula to a theory never removes its consequences.

relationship between them. Since support and assumption belong to the axiom measures, we can only compare them with fitness and braveness of a single-axiom hypothesis, see Lemma 5.25.

Lemma 5.25. *Let $\mathcal{O} := \mathcal{A} \cup \mathcal{T}$ be an $\mathcal{EL}$ ontology, $\mathbb{C} := \{A, B\}$ concepts, $\mathbb{R}$ roles, $\alpha := A \sqsubseteq B$, where $A, B \in N_C$. Then*

(i) $fit(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) \leq sup(\alpha, \mathcal{O})$.

(ii) $bra(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) \leq asm(\alpha, \mathcal{O})$.

Proof. (i) $sup(\alpha, \mathcal{O}) = bsup(\alpha, \mathcal{O}) + bsup(\alpha', \mathcal{O})$ (by Lemma 5.15)

$= |inst(A \sqcap B, \mathcal{O})| + |inst(\neg A \sqcap \neg B, \mathcal{O})|$
(by Definition 5.14 since $\alpha' := \neg D \sqsubseteq \neg C$)

$= |inst(A \sqcap B, \mathcal{O})|$ (since $\mathcal{O}$ is in $\mathcal{EL}$ and $A, B \in N_C$).

$fit(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\pi, \mathcal{T}) - dlen(\pi, \mathcal{T} \cup \{\alpha\})$
(by Definition 5.23)

$= \ell(\pi_1) - \ell(\pi_2)$ (by Definition 5.22), where π_1 and π_2 are the minimal ABoxes of π given $\mathcal{T}$ and $\mathcal{T} \cup \{\alpha\}$, respectively.

By Definition 5.20, besides role assertions of $\mathbb{R}$, the projection π only contains concept assertions of A and B. Consider an individual $a \in in(\mathcal{O})$.

- π_1 differs from π_2 for a if and only if $A(a) \in \pi_1$ and $B(a) \in \pi_1$ (since $\alpha := A \sqsubseteq B$).
- $A(a) \in \pi_1$ and $B(a) \in \pi_1$ if and only if $A(a) \in \pi_2$ and $B(a) \notin \pi_2$ (since π_2 is minimal).
- $A(a) \in \pi_1$ and $B(a) \in \pi_1$ implies $\mathcal{O} \models A(a)$ and $\mathcal{O} \models B(a)$ (by Definition 5.20 and Definition 5.22). The reverse is not true, e.g. if $\mathcal{T} \models A \sqsubseteq B$.
- $A(a) \in \pi_2$ and $B(a) \notin \pi_2$ implies $\mathcal{O} \models A(a)$ and $\mathcal{O} \models B(a)$, i.e. $a \in inst(A \sqcap B, \mathcal{O})$.
- $\ell(\pi_1) - \ell(\pi_2) \leq |inst(A \sqcap B, \mathcal{O})|$ (since π_1 differs from π_2 by at most one assertion per instance of $A \sqcap B$).
- $fit(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) \leq sup(\alpha, \mathcal{O})$.

(ii) $asm(\alpha, \mathcal{O}) = basm(\alpha, \mathcal{O}) + basm(\alpha', \mathcal{O})$ (by Lemma 5.15)

$= |inst(A,\ \mathcal{O}) \cap inst(?B,\ \mathcal{O})| + |inst(\neg B,\ \mathcal{O}) \cap inst(?A,\ \mathcal{O})|$
(by Definition 5.14 since $\alpha' := \neg D \sqsubseteq \neg C$)

$= |inst(A,\ \mathcal{O}) \cap inst(?B,\ \mathcal{O})|$ (since $\mathcal{O}$ is in $\mathcal{EL}$ and $A, B \in N_C$).

$bra(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen(\psi_\alpha, \mathcal{O})$ (by Definition 5.23).

By Definition 5.21 the assumption set ψ_α can only contain concept assertions of B. Consider an individual $a \in in(\mathcal{O})$.

- $B(a) \in \psi_\alpha$ if and only if $\mathcal{O} \cup \{\alpha\} \models B(a)$ and $\mathcal{O} \not\models B(a)$ (by Definition 5.21).
- $\mathcal{O} \cup \{\alpha\} \models B(a)$ and $\mathcal{O} \not\models B(a)$ if and only if $\mathcal{O} \models A(a)$ and $\mathcal{O} \not\models B(a)$ (since $\alpha := A \sqsubseteq B$).
- $B(a) \in \psi_\alpha$ if and only if $a \in inst(A, \mathcal{O})$ and $a \in inst(?B, \mathcal{O})$ (since $\mathcal{O}$ is in $\mathcal{EL}$).
- $|\psi_\alpha| = |inst(A,\ \mathcal{O}) \cap inst(?B,\ \mathcal{O})|$.
- $dlen(\psi_\alpha, \mathcal{O}) \leq |inst(A,\ \mathcal{O}) \cap inst(?B,\ \mathcal{O})|$ (since $dlen(\psi_\alpha, \mathcal{O}) \leq \ell(\psi_\alpha) = |\psi_\alpha|$ and some individuals can be equal).
- $bra(\{\alpha\}, \mathcal{O}, \mathbb{C}, \mathbb{R}) \leq asm(\alpha, \mathcal{O})$.

□

Lemma 5.25 states that support and assumption are upper bounds for fitness and braveness, respectively, if certain conditions hold: an ontology is in $\mathcal{EL}$, a hypothesis is an atomic subsumption, concepts of interest are its signature. These restrictions are necessary because of the differences between the respective measures.

To be more specific, fitness and braveness are based on measuring the length of ABoxes, while support and assumption count individuals. For example, if an ontology is in $\mathcal{ALC}$, the projection and assumption set can contain assertions of negated concepts which are longer than assertions of atomic concepts, e.g. $\ell(\neg A(a)) = 2$ and $\ell(A(a)) = 1$. In contrast, an individual a is counted just once as an instance of $\neg A$. A hypothesis is restricted to be an atomic subsumption (not a GCI) for the same reason. Concepts of interest only contain the signature of an atomic subsumption because additional concepts can bring additional supporting facts for fitness.

Roles $\mathbb{R}$ are not restricted since they cannot affect the result. Lemma 5.25 can be stated for an atomic role inclusion $\alpha := R \sqsubseteq S$, where $R, S \in N_R$, and proved analogously. The only difference is that concepts $\mathbb{C}$ should be restricted, e.g. $\mathbb{C} := \emptyset$, because they can affect the result, e.g. consider $\mathbb{C} := \{\exists R.\top,\ \exists S.\top\}$ that increases the fitness of α.

5.4 Summary

We have defined hypothesis quality measures for all quality dimensions, i.e. readability, logical quality, and statistical quality. The readability measures are length and role depth. These are the standard measures used in the DL literature. We have straightforwardly extended them to some complex concepts and roles.

The logical quality measures are consistency, informativeness, redundancy, logical strength, complexity, and dissimilarity. Some of these measures capture similar notions from the related work. Redundancy is based on axiom superfluity investigated in [HPS08]. Logical strength (weakness) is similar to specificity (generality) used in CDL for concepts, see Section 3.2.1. Concept similarity is explored in [APS14, EPT15]. The notion of complexity has not been investigated so far, to the best of our knowledge. While some measures are easy to define, e.g. consistency and informativeness, others are tricky, particularly complexity. These measures, except consistency and informativeness (see Section 3.2.3), have not been used for evaluating hypothesis quality in OL.

The statistical quality measures comprise the axiom measures, designed for an axiom, and the axiom set measures, designed for a set of axioms. The axiom measures, except assumption, are based on the rule quality measures from ARM, see Section 2.2.5. In contrast to the rule measures, they are suited for DLs, i.e. respect the OWA and take the law of contraposition into account. The axiom set measures are fitness and braveness which fully respect the DL semantics, i.e. return the same values for all equivalent hypotheses. They have no analogous notions in the related work, to the best of our knowledge.

We have investigated properties of the introduced quality measures which are summarised in Table 5.3. Some measures satisfy more properties than others. For statistical measures, we consider all properties as favourable. Any negative measure can easily be converted to positive by simply reversing its sign, i.e. multiplying its value by -1. There are symmetric and asymmetric measures that

respect and do not respect, respectively, the direction of an implication. Other properties are *semantic*, i.e. show how semantically faithful a measure is in DLs.

Quality dimension	**Quality measure**	**Property**						
		S	**P**	**A**	**E**	**C**	**N**	**T**
Readability	Syntactic length	✓					✓	
	Role depth	✓					✓	
Logical	Consistency	✓	✓	✓	✓	✓	✓	✓
	Informativeness	✓	✓	✓		✓	✓	✓
	Redundancy	✓		✓			✓	
	Logical strength	✓	✓	✓	✓	✓	✓	
	Dissimilarity		✓				✓	
	Complexity	✓	✓	✓	✓	✓	✓	✓
Statistical	Basic support		✓					
	Basic lift		✓					
	Basic contradiction			✓		✓		
	Basic assumption			✓				
	Basic confidence		✓	✓				
	Basic conviction		✓	✓				
	Main support		✓			✓		
	Main lift		✓			✓		
	Main contradiction			✓		✓		
	Main assumption			✓		✓		
	Main confidence		✓	✓		✓		
	Main conviction		✓	✓		✓		
	Fitness	✓	✓	✓	✓	✓	✓	✓
	Braveness	✓		✓	✓	✓	✓	✓

Table 5.3: Introduced quality measures and their properties (✓– has a property): **S** – designed for a set of axioms; **P** – positive, i.e. higher values mean better quality; **A** – asymmetric, i.e. generally returns different values for $C \sqsubseteq D$ and $D \sqsubseteq C$; **E** – returns the same value for all equivalent hypotheses; **C** – returns the same value for an axiom and its contrapositive; **N** – does not make the UNA; **T** – fully considers TBox.

As Table 5.3 shows, the main measures satisfy more properties than the basic measures.[9] In comparison to the main measures, fitness and braveness satisfy all semantic properties: they are designed for a set of axioms, preserved under equivalence, do not make the UNA, and fully consider the given TBox. The latter means that the TBox can interact with a hypothesis beyond materialising

[9]We do not list the basic and main coverage in Table 5.3 as these are rather auxiliary measures that are used to define others and both of them are neither positive nor negative.

the ABox, see Example 5.23 and Example 5.24.

Unlike the statistical measures, other measures satisfying fewer properties are not necessarily less faithful: they are simply defined so to capture the respective notions. In particular, the readability measures are concerned with syntactic evaluation of a hypothesis and do not satisfy any semantic properties by definition. With regard to the logical measures, redundancy tests syntactic superfluity of a hypothesis, i.e. whether there is an equivalent hypothesis of a shorter length. Hence, it is senseless (and impossible) to make redundancy preserved under equivalence. Additionally, as discussed in Section 5.2.3, there are reasons to define redundancy irrespective of the TBox. The same is true for logical strength. An example when a measure satisfying more properties than another measure is, in fact, more semantically faithful is complexity in comparison to dissimilarity.

As some statistical measures have better properties than others, should we use only better measures and disregard others? In particular, fitness and braveness have better properties than other statistical measures. So, should we just use fitness and braveness? As Example 5.24 shows, we should not since fitness captures different from support quality characteristics. Another concern is computational performance since more faithful measures are expected to be more expensive to compute. In Chapter 6, we discuss how to compute all the proposed measures.

It is also important to note that the introduced measures are of different significance for hypothesis evaluation. In other words, some measures are *primary* and should be used in all cases, while others are rather *secondary* and provide supplementary information about hypothesis quality. Since we investigate the problem of learning from data, a hypothesis should certainly be evaluated using the ABox. Therefore, the primary measures are key statistical measures. These are support, assumption, contradiction, and confidence for single-axiom hypotheses; fitness and braveness for multi-axiom hypotheses. The secondary measures supply other useful information, e.g. consistency, informativeness, redundancy, etc. One should keep this in mind while selecting measures for hypothesis evaluation in DL-Miner, see Figure 4.1. In addition, if the supplementary measures are included in the hypothesis ordering, they can perturb it, see Chapter 8.

Chapter 6

Computing Hypothesis Quality Measures

In Chapter 5, we have defined the hypothesis quality measures. In this chapter we show how to compute these measures. Indeed, while some measures are rather straightforward to compute, e.g. length, consistency, basic support, others are not, e.g. redundancy, complexity, fitness, braveness. In addition, since a hypothesis space is usually vast, we should evaluate each hypothesis as efficiently as possible.

As a reminder, the architecture of DL-Miner consists of the following functional blocks, see Figure 4.1: Ontology Cleaner, Hypothesis Constructor, Hypothesis Evaluator, and Hypothesis Sorter. Hypothesis Evaluator takes (a subset of) the defined quality measures as input parameters. In this chapter we describe how Hypothesis Evaluator computes these measures.

6.1 Preliminary Definitions

In order to explain how the measures are computed, we need to define some auxiliary notions. In Chapter 5, we frequently use sets $\mathbb{C}$ and $\mathbb{R}$ of concepts and roles. Computing some measures, e.g. dissimilarity, requires figuring out subsumption relations within each of these sets according to the given ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$. In other words, we need to obtain the following sets of entailments:

$$ch(\mathcal{T}, \mathbb{C}) := \{C \sqsubseteq D \mid \mathcal{T} \models C \sqsubseteq D \ \wedge \ C, D \in \mathbb{C}\},$$
$$rh(\mathcal{T}, \mathbb{R}) := \{R \sqsubseteq S \mid \mathcal{T} \models R \sqsubseteq S \ \wedge \ R, S \in \mathbb{R}\}.$$

Thus, we need to check $\mathcal{T} \models C \sqsubseteq D$ and $\mathcal{T} \models R \sqsubseteq S$ for every $C, D \in \mathbb{C}$ and $R, S \in \mathbb{R}$. If sets $\mathbb{C}$ and $\mathbb{R}$ contain only concept and role names, i.e. $\mathbb{C} \subset N_C$, $\mathbb{R} \subset N_R$, the task is accomplished by computing the concept and role hierarchies, see Definition 2.10. These can be computed efficiently using the standard means since reasoners are specifically optimised for computing concept and role hierarchies.

Nonetheless, sets $\mathbb{C}$ and $\mathbb{R}$ can contain complex concepts and roles. Therefore, computing concept and role hierarchies can be insufficient. Since checking entailments for role inclusions seems to be cheap, we directly check all remaining entailments in $rh(\mathcal{T}, \mathbb{R})$. The same procedure, however, can be much more costly for computing $ch(\mathcal{T}, \mathbb{C})$ because checking entailments for concept inclusions seems to be harder and there are usually many more complex concepts in $\mathbb{C}$ than complex roles in $\mathbb{R}$.

Considering the aforementioned factors, we suggest the following optimisation for computing $ch(\mathcal{T}, \mathbb{C})$. The idea is that, if we name complex concepts from $\mathbb{C}$ in the TBox $\mathcal{T}$, we can benefit from optimisations of reasoners for computing concept hierarchies. Let $na(C, \mathcal{T})$ be the function that returns for a concept C a fresh and unique name from N_C with respect to $\mathcal{T}$. Then, for each complex concept $C \in \mathbb{C}$ we add the auxiliary definition $na(C, \mathcal{T}) \equiv C$ to $\mathcal{T}$ as follows:

$$ce(\mathcal{T}, \mathbb{C}) := \mathcal{T} \cup \{na(C, \mathcal{T}) \equiv C \mid C \in \mathbb{C} \backslash N_C\}.$$

Given a TBox $\mathcal{T}$ and a set $\mathbb{C}$ of concepts, the function $ce(\mathcal{T}, \mathbb{C})$ extends $\mathcal{T}$ with auxiliary definitions for complex concepts in $\mathbb{C}$. Let $\mathcal{T}^+ := ce(\mathcal{T}, \mathbb{C})$. Then, a reasoner can be run on $\mathcal{T}^+$ to compute the concept hierarchy $ch(\mathcal{T}^+)$, see Definition 2.10, of $\mathcal{T}^+$ instead of $\mathcal{T}$. As a result, instead of checking $\mathcal{T} \models C \sqsubseteq D$ for every $C, D \in \mathbb{C}$, we simply query the concept hierarchy of $\mathcal{T}^+$ which can be easily done using the OWL API.[1] Formally, a reasoning operation checking whether $\mathcal{T} \models C \sqsubseteq D$ is replaced by the query checking whether $A_C \sqsubseteq A_D \in ch(\mathcal{T}^+)$, where $A_C := na(C, \mathcal{T})$, $A_D := na(D, \mathcal{T})$. Thus, the function $ch(\mathcal{T}, \mathbb{C})$ can be rewritten as follows:

$$ch(\mathcal{T}, \mathbb{C}) := ch(\mathcal{T}) \cup \{C \sqsubseteq D \mid na(C, \mathcal{T}) \sqsubseteq na(D, \mathcal{T}) \in ch(ce(\mathcal{T}, \mathbb{C}))\}.$$

We normally use this definition of $ch(\mathcal{T}, \mathbb{C})$ and assume that the concept hierarchy of the extended TBox is computed. As said above, $rh(\mathcal{T}, \mathbb{R})$ is computed

[1] `http://owlapi.sourceforge.net`

directly.

In the following, we use the notion of a unique representative of equivalent concepts. Let $eq(D, \mathcal{T}, \mathbb{C}) := \{C \in \mathbb{C} \mid \mathcal{T} \models C \equiv D\}$. Thus, it is the set of all concepts in $\mathbb{C}$ which are equivalent to a concept D given a TBox $\mathcal{T}$. Let $ur(S)$ be the function that returns some fixed element $e \in S$, i.e. if $e = ur(S)$, then $e' = ur(S)$ implies $e' = e$. Then, a unique representative is defined as follows, see Definition 6.1.

Definition 6.1 (Unique representative)**.** Let $\mathcal{T}$ be a TBox and $\mathbb{C}$ a set of concepts. A concept C is called the *unique representative* for a concept D given $\mathcal{T}$ and $\mathbb{C}$ if $C = urc(D, \mathcal{T}, \mathbb{C})$, where $urc(D, \mathcal{T}, \mathbb{C}) := ur(eq(D, \mathcal{T}, \mathbb{C}))$. Let $urc(\mathcal{T}, \mathbb{C}) := \{urc(D, \mathcal{T}, \mathbb{C}) \mid D \in \mathbb{C}\}$ and $urc(\mathbb{C}) := urc(\emptyset, \mathbb{C})$.

Thus, $urc(D, \mathcal{T}, \mathbb{C}) = D'$ implies $\mathcal{T} \models D' \equiv D$ which, in turn, implies $urc(D, \mathcal{T}, \mathbb{C}) = urc(D', \mathcal{T}, \mathbb{C})$. Given a TBox $\mathcal{T}$ and a set $\mathbb{R}$ of roles, a role R is called the unique representative for a role S if $R = urr(S, \mathcal{T}, \mathbb{R})$, where $urr(S, \mathcal{T}, \mathbb{R})$ is defined analogously.

6.2 Computing Readability Measures

Syntactic length, see Definition 5.1, and role depth, see Definition 5.3, are simple syntactic measures that can be computed straightforwardly. They do not use reasoning and do not require any optimisations. The situation is different for redundancy, see Definition 5.7, which we discuss in detail.

6.2.1 Detecting and Eliminating Redundancy

According to Definition 5.7, a hypothesis is redundant if it contains an axiom which is redundant or has redundant parts. Hence, in order to detect redundancy, it is sufficient to find at least one such axiom. The task of eliminating redundancy is harder since all such axioms must be either removed (if an axiom is redundant) or replaced by shorter axioms (if an axiom contains redundant parts) such that the resulting hypothesis is equivalent to the original one and as short as possible.

Detecting and eliminating redundancy is investigated in [HPS08, Hor11]. Superfluous (redundant) parts of axioms, see Definition 5.7, are detected and eliminated with the help of *structural transformations*. In particular, the π transformation[2] transforms a set of axioms such that the resulting set, besides the given axioms, contains all weaker and shorter forms of those axioms, see Definition 19 at page 177 in [Hor11] for details. It avoids introducing new terms in the output and defined for the DL $\mathcal{SHOIQ}$. Similarly to justifications, this transformation can be used to detect and eliminate redundancy of a hypothesis.

Let $pi(\mathcal{T})$ be the function computing the π transformation [Hor11] of a TBox $\mathcal{T}$. Given a hypothesis H, we compute $pi(H)$. Then, we search for a shorter hypothesis $H_{min} \subseteq pi(H)$ such that $H_{min} \equiv H$, see Algorithm 1. Instead of checking all possible subsets of $pi(H)$, the algorithm iteratively attempts to remove the longest axiom from $pi(H)$. If the removed axiom is not redundant in the current set, it is added back to the set.

Algorithm 1 *getMinimalHypothesis*(H)

```
1:  inputs
2:    H: a hypothesis
3:  outputs
4:    H_min: a shortest hypothesis equivalent to H
5:  do
6:    H_π ← pi(H)      % compute the π transformation of H
7:    H_min ← (α_1, ..., α_k), where α_i, i ∈ {1, ..., k}, are all axioms of H_π in
      descending order of their length
8:    for each i ∈ {1, ..., k} do
9:      H_min ← H_min \ {α_i}    % remove an axiom
10:     if H_min ⊭ α_i then
11:       H_min ← H_min ∪ {α_i}    % if an axiom is not redundant, add it back
12:     end if
13:   end for
14:   return H_min
```

Please notice that, at each iteration i of the loop (Lines 8 – 13), the current set of axioms H_{min} either decreases, if α_i is redundant, or stays unchanged, if α_i is not redundant. In other words, $H^{i+1}_{min} \subseteq H^{i}_{min}$. Once the algorithm terminates, it returns a non-redundant hypothesis H_{min} equivalent to H, see Lemma 6.1.

Lemma 6.1. *Given a hypothesis H, Algorithm 1 returns a non-redundant hypothesis H_{min} equivalent to H.*

[2]The π transformation should not be confused with a projection π.

Proof. (By contradiction) Assume H_{min} is redundant. Hence, by Definition 5.7, there is $\alpha \in H_{min}$ which is redundant (i) in H_{min} or has redundant parts (ii). The case (i) means that $H_{min} \backslash \{\alpha\} \models \alpha$ after the termination. However, this is not possible because of Lines 9 – 12. The case (ii) means that there is α' such that $\ell(\alpha') < \ell(\alpha)$ and $H_{min} \cup \{\alpha'\} \backslash \{\alpha\} \equiv H_{min}$. Since the π transformation produces all weaker and shorter forms of α including α', α cannot be in H_{min} due to Lines 9 – 12. Thus, there is no such $\alpha \in H_{min}$. □

If the found hypothesis H_{min} is shorter (by syntactic length) than the original hypothesis H, i.e. $\ell(H_{min}) < \ell(H)$, then H is redundant. The found hypothesis H_{min} is a non-redundant counterpart of H. Please note that Algorithm 1 only guarantees to eliminate redundancy which is caused by excessive length, see Definition 5.7, but there are other causes of redundancy in DLs, see Example 5.8.

Algorithm 1 returns a single non-redundant hypothesis. However, a non-redundant hypothesis is not unique in general, see Example 6.1. As long as there is no need to obtain all non-redundant hypotheses, the result of Algorithm 1 is satisfactory.

Example 6.1. Consider the hypothesis $H := \{A \equiv B,\ A \sqsubseteq C,\ B \sqsubseteq C\}$. It is redundant since either second or third axiom can be removed, i.e. there are two non-redundant hypotheses for H: $H_1 := \{A \equiv B,\ A \sqsubseteq C\}$ and $H_2 := \{A \equiv B,\ B \sqsubseteq C\}$.

6.3 Computing Logical Quality Measures

The logical quality measures are computed using the standard reasoning services of DLs. Consistency, see Definition 5.5, is straightforwardly computed by checking the consistency of the ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ augmented with a hypothesis H, i.e. $\mathcal{O} \cup H$. Informativeness, see Definition 5.6, is computed by checking whether a hypothesis H contains axioms entailed by the TBox $\mathcal{T}$, i.e. for each $\alpha \in H$ we check whether $\mathcal{T} \models \alpha$. If there is at least one axiom which is found to be entailed by $\mathcal{T}$, then H is uninformative. As discussed in Section 5.2.2, an uninformative hypothesis H can be transformed to the informative hypothesis H' by simply removing all entailed axioms, i.e. $H' = \{\alpha \in H \mid \mathcal{T} \not\models \alpha\}$, and ensuring that $H' \neq \emptyset$.

Given two hypotheses H and H', logical strength, see Definition 5.9, is easily computed by checking $H \models H'$ and $H' \models H$. By Definition 5.9, if $H \models H'$

and $H' \not\models H$, then $H \rhd H'$; if $H \not\models H'$ and $H' \models H$, then $H' \rhd H$; otherwise H and H' are incomparable. However, computing a partial order $(\mathbb{H}, \rhd)$ on the hypothesis space $\mathbb{H}$ requires additional care because $\mathbb{H}$ can be large. The naive algorithm checks logical strength for every unordered pair of hypotheses from $\mathbb{H}$ and runs in quadratic time with respect to $|\mathbb{H}|$. There are algorithms to efficiently compute a partial order, i.e. to build its Hasse diagram, using transitivity of $\rhd$, see [BHN+92]. The efficiency of such algorithms, however, depends on the structure of $\mathbb{H}$: the longer $\rhd$-chains are, the more checks the algorithm can avoid and the more efficient it is in comparison to the naive algorithm.

6.3.1 Computing Dissimilarity

According to Definition 5.10, the dissimilarity of an axiom is determined by the number of common and distinct subsumers that the left-hand side (LHS) and right-hand side (RHS) of the axiom have in a set $\mathbb{C}$ of concepts given a TBox $\mathcal{T}$. In order to use more information for dissimilarity, by default we include in $\mathbb{C}$ all concept names from $\mathcal{T}$, i.e. in the following we assume that $cn(\mathcal{T}) \subseteq \mathbb{C}$. By Definition 5.10, the set of subsumers of a concept D in a set $\mathbb{C}$ of concepts given a TBox $\mathcal{T}$ is defined as follows:

$$subs(D, \mathcal{T}, \mathbb{C}) := \{C \in \mathbb{C} \cup \{D\} \mid \mathcal{T} \models D \sqsubseteq C\}.$$

A naive algorithm for computing subsumers checks $\mathcal{T} \models D \sqsubseteq C$ for each $C \in \mathbb{C}$. However, as discussed in Section 6.1, this can be inefficient because a set $\mathbb{C}$ can be large. Therefore, we use optimisations of reasoners to efficiently compute all necessary entailments, see Algorithm 2.

We are normally interested in computing dissimilarity for all hypotheses in the set $\mathbb{H}$. However, we would like to avoid extending and classifying the TBox for each hypothesis in $\mathbb{H}$. Algorithm 2 can be straightforwardly adjusted such that the latter is done just once for all hypotheses in $\mathbb{H}$. More specifically, instead using D as an input concept, the algorithm would use a set $\mathbb{D}$ of input concepts that consists of LHSs and RHSs of all axioms appearing in $\mathbb{H}$. Lines 8 – 10 would then use $\mathbb{D}$ instead of $\{D\}$ and Lines 11 – 16 would be run in the loop over all concepts in $\mathbb{D}$.

Computing the dissimilarity of a RI is analogous to the dissimilarity of a GCI. In Algorithm 2 the concept hierarchy CH is replaced by the role hierarchy RH

Algorithm 2 *computeSubsumers*$(D, \mathcal{T}, \mathbb{C})$

```
1: inputs
2:    D: a concept
3:    ℂ: a set of concepts
4:    𝒯: a TBox
5: outputs
6:    subs: the set of subsumers of D in ℂ given 𝒯
7: do
8:    𝒯⁺ ← ce(𝒯, ℂ ∪ {D})      % extend 𝒯 with auxiliary definitions
9:    classify 𝒯⁺               % build the concept hierarchy of 𝒯⁺
10:   CH ← ch(𝒯, ℂ ∪ {D})      % gather necessary entailments
11:   subs ← {D}
12:   for each C ∈ ℂ do
13:      if D ⊑ C ∈ CH then
14:         subs ← subs ∪ {C}   % add a subsumer
15:      end if
16:   end for
17:   return subs
```

which we compute via $rh(\mathcal{T},\ \mathbb{R}\cup\{S\})$ without extending the TBox with auxiliary definitions for complex roles.

6.3.2 Computing Complexity

Definition 5.11 of complexity requires finding *new* entailments, i.e. those which are entailed by the ontology with a hypothesis but not by the ontology alone. As the set of all new entailments is infinite in general, we only consider a certain finite subset of them, i.e. only those which are subsumptions between concepts in a finite set $\mathbb{C}$ (roles in a finite set $\mathbb{R}$). New entailments can be efficiently computed using the optimisation technique suggested in Section 6.1 as a difference of the concept (and role) hierarchy of $\mathcal{T} \cup H$ and $\mathcal{T}$:

$$com(H, \mathcal{T}, \mathbb{C}, \mathbb{R}) \ := \ |(ch(\mathcal{T} \cup H,\ \mathbb{C}) \backslash ch(\mathcal{T}, \mathbb{C})) \ \cup \ (rh(\mathcal{T} \cup H,\ \mathbb{R}) \backslash rh(\mathcal{T}, \mathbb{R}))|.$$

Algorithm 3 implements computing complexity. Like Algorithm 2 for computing subsumers, it can be straightforwardly adjusted to the case of multiple hypotheses. More specifically, the extension $\mathcal{T}^+$ is built and classified just once for all hypotheses. For each hypothesis $H \in \mathbb{H}$ the concept hierarchy is updated and new entailments are found, i.e. Lines 11 – 14 are run in the loop.

Algorithm 3 *computeComplexity*$(H, \mathcal{T}, \mathbb{C}, \mathbb{R})$

inputs
  H: a hypothesis
  $\mathcal{T}$: a TBox
  $\mathbb{C}$, $\mathbb{R}$: sets of concepts and roles
outputs
  $com(H, \mathcal{T}, \mathbb{C}, \mathbb{R})$: the complexity of H
do
  $\mathcal{T}^+ \leftarrow ce(\mathcal{T}, \mathbb{C})$ % extend $\mathcal{T}$ with auxiliary definitions
  classify $\mathcal{T}^+$ % build the concept and role hierarchy of $\mathcal{T}^+$
  $CH \leftarrow ch(\mathcal{T}, \mathbb{C})$; $RH \leftarrow rh(\mathcal{T}, \mathbb{R})$
  classify $\mathcal{T}^+ \cup H$
  $CH_H \leftarrow ch(\mathcal{T} \cup H, \mathbb{C})$
  $RH_H \leftarrow rh(\mathcal{T} \cup H, \mathbb{R})$
  $S \leftarrow (CH_H \backslash CH) \cup (RH_H \backslash RH)$ % find new entailments
  return $|S|$

6.4 Computing Statistical Quality Measures

Like the logical measures, the statistical measures contain measures which are straightforward to compute, i.e. the axiom measures, and measures which are not, i.e. the axiom set measures.

6.4.1 Computing Axiom Measures

The axiom measures essentially compare instances of the LHS with instances of the RHS of an axiom, see Section 5.3.1. Computing these only requires finding all instances of a concept C in an ontology $\mathcal{O}$, i.e. instance retrieval. Given an axiom $C \sqsubseteq D$ and an ontology $\mathcal{O}$, the basic axiom measures, see Definition 5.14 and Definition 5.16, can be calculated after retrieving instances for each of the concepts C, D, and $\neg D$. For example, the basic support of $C \sqsubseteq D$ is simply the number of instances that concepts C and D have in common.

The main axiom measures, see Definition 5.17 and Definition 5.19, like the basic ones, only require instance retrievals. By Lemma 5.15 and Lemma 5.19, the main measures of an axiom $C \sqsubseteq D$ can be written via the basic measures using its contrapositive $\neg D \sqsubseteq \neg C$. Therefore, in order to compute the main measures, we need to retrieve instances of C, D, $\neg D$, and $\neg C$. Thus, one additional instance retrieval is required.

6.4.2 Computing Fitness and Braveness

Unlike the axiom measures, the axiom set measures, i.e. fitness and braveness, are not straightforward to compute. By Definition 5.23, computing fitness and braveness requires computing the description length $dlen(\mathcal{B}, \mathcal{O})$, see Definition 5.22, of an ABox $\mathcal{B}$ with respect to an ontology $\mathcal{O}$, i.e. finding a minimal ABox $\mathcal{B}'$. This can be approached using hitting set tree algorithms [Rei87], where tree nodes represent ABoxes and edges are labelled with assertions. The root node is the initial ABox $\mathcal{B}$. The tree construction includes the following steps.

1. Given a node $\mathcal{B}$, choose an assertion $\alpha \in \mathcal{B}$ such that α does not connect $\mathcal{B}$ to any of the successors of $\mathcal{B}$.

2. Check whether $\mathcal{O} \cup \mathcal{B} \backslash \{\alpha\} \models \alpha$ holds, or perform a *superfluity check* for α.

3. If it does, create a new node $\mathcal{B}' := \mathcal{B} \backslash \{\alpha\}$ since α is superfluous and can be discarded. If it does not, create an empty node $\mathcal{B}' := \emptyset$ since α is not superfluous and all subsets of $\mathcal{B}' := \mathcal{B} \backslash \{\alpha\}$ can be pruned from the search, i.e. the node is not expanded further.

4. Connect $\mathcal{B}$ to $\mathcal{B}'$ via the edge α.

The algorithm terminates once all leaf nodes are empty, i.e. no assertion can be discarded any more. Non-leaf nodes of the smallest size contain minimal ABoxes. Please notice that such algorithm returns a minimal subset $\mathcal{B}'$ of the given ABox $\mathcal{B}$, which is not necessarily a shortest ABox. Hence, it is an approximation. However, the described algorithm can be infeasible even for moderate ABoxes since it requires exponentially many superfluity checks in the worst case and each check is expensive.

Therefore, we suggest a different approach. We compute approximations for fitness and braveness considering that we can estimate the size of a minimal ABox $\mathcal{B}'$ avoiding costly superfluity checks. More specifically, superfluity can be checked just once for each assertion in $\mathcal{B}$ and each check can be done cheaply. The idea is illustrated by Example 6.2.

Example 6.2. Consider the TBox $\mathcal{T}$ and the ABox $\mathcal{B}$:

$$\mathcal{T} := \{A \sqsubseteq B\}, \quad \mathcal{B} := \{A(a), B(a)\}.$$

Since $\mathcal{T} \models A \sqsubseteq B$ and $A(a) \in \mathcal{B}$, the assertion $B(a)$ is superfluous in $\mathcal{B}$ and can be discarded. Thus, the minimal ABox $\mathcal{B}' := \{A(a)\}$. Thus, we use the concept hierarchy of $\mathcal{T}$ and assertions in $\mathcal{B}$ to check superfluity. Nevertheless, one should be careful if the TBox contains equivalent concepts. In this case some assertions can be mistakenly discarded. Consider the TBox $\mathcal{T}' := \{A \equiv B\}$. Given $\mathcal{T}'$, any of the assertions $A(a)$ and $B(a)$ is superfluous in $\mathcal{B}$, but not both of them simultaneously. Thus, there are two minimal ABoxes in this case: $\mathcal{B}' := \{A(a)\}$ and $\mathcal{B}'' := \{B(a)\}$.

Since a given ABox $\mathcal{B}$ contains complex concept assertions in general, computing the simple concept hierarchy is insufficient. Therefore, we again use the concept hierarchy of the extended TBox, i.e. the optimisation technique from Section 6.1.

Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, $\mathbb{C}$ a set of concepts (closed under negation as in Definition 5.20), $\mathbb{R}$ a set of roles, and $\mathcal{B}$ an ABox such that $D(a) \in \mathcal{B}$ implies $D \in \mathbb{C}$ and $S(a,b) \in \mathcal{B}$ implies $S \in \mathbb{R}$, i.e. all assertions in $\mathcal{B}$ are "covered" by $\mathbb{C}$ and $\mathbb{R}$. Then, considering Example 6.2, we define the redundancy of concept assertions (CAs) in $\mathcal{B}$ given $\mathcal{O}$ as follows:

$$
\begin{aligned}
red_C(\mathcal{B}, \mathcal{O}, \mathbb{C}) := \{ & D(a) \in \mathcal{B} \mid D(a) \in \mathcal{A} \ \vee \ \exists C \in \mathbb{C} \ : \\
& \text{(i)} \ (C \sqsubseteq D \in ch(\mathcal{T}, \mathbb{C}) \ \wedge \ D \sqsubseteq C \notin ch(\mathcal{T}, \mathbb{C}) \ \wedge \ C(a) \in \mathcal{B} \cup \mathcal{A}) \vee \\
& \text{(ii)} \ (C \sqsubseteq D \in ch(\mathcal{T}, \mathbb{C}) \ \wedge \ D \sqsubseteq C \in ch(\mathcal{T}, \mathbb{C}) \ \wedge \ D \notin urc(\mathcal{T}, \mathbb{C}))\}.
\end{aligned}
$$

Informally, the function $red_C(\cdot)$[3] returns a set of CAs redundant due to other CAs. Clearly, if a CA $D(a)$ from $\mathcal{B}$ is also contained in $\mathcal{O}$, then it is superfluous in $\mathcal{B}$. Otherwise, the cases (i) and (ii) include superfluous assertions illustrated by Example 6.2. Please notice that no reasoning is used, once the concept hierarchy of the extended TBox is computed.

As the ABox $\mathcal{B}$ can contain superfluous role assertions, we also define the redundancy $red_R(\cdot)$ of role assertions (RAs) which is analogous to the redundancy $red_C(\cdot)$ of CAs, but uses the function $rh(\cdot)$ instead of $ch(\cdot)$. It returns a set of

[3]The symbol "$\cdot$" stands for the arguments of the function if they are clear from the context or irrelevant.

RAs redundant due to other RAs as follows:

$$\begin{aligned} red_R(\mathcal{B},\mathcal{O},\mathbb{R}) := \{S(a,b) \in \mathcal{B} \mid S(a,b) \in \mathcal{A} \ \vee \ \exists R \in \mathbb{R} \ : \\ \text{(i)} \ (R \sqsubseteq S \in rh(\mathcal{T},\mathbb{R}) \ \wedge \ S \sqsubseteq R \notin rh(\mathcal{T},\mathbb{R}) \ \wedge \ R(a,b) \in \mathcal{B} \cup \mathcal{A}) \vee \\ \text{(ii)} \ (R \sqsubseteq S \in rh(\mathcal{T},\mathbb{R}) \ \wedge \ S \sqsubseteq R \in rh(\mathcal{T},\mathbb{R}) \ \wedge \ S \notin urr(\mathcal{T},\mathbb{R}))\}. \end{aligned}$$

Nevertheless, the functions $red_C(\cdot)$ and $red_R(\cdot)$ do not account for all cases of superfluity. In particular, they consider CAs and RAs independently. However, those can interact with each other which causes additional assertions to be superfluous, see Example 6.3.

Example 6.3. Consider the TBox $\mathcal{T}$ and the ABox $\mathcal{B}$:

$$\begin{aligned} \mathcal{T} &:= \{\exists R.A \sqsubseteq B, \ A \sqsubseteq \forall R.B\}, \\ \mathcal{B} &:= \{A(a), A(b), B(a), B(b), R(a,b)\}. \end{aligned}$$

The assertion $B(a)$ is superfluous in $\mathcal{B}$ because a is connected to b, an instance of A, via the role assertion $R(a,b)$ and the axiom $\exists R.A \sqsubseteq B$ in $\mathcal{T}$ states that having at least one R-successor which is an instance of A implies being an instance of B. In addition, the assertion $B(b)$ is superfluous because a is an instance of A connected to b via the role assertion $R(a,b)$ and the axiom $A \sqsubseteq \forall R.B$ in $\mathcal{T}$ states that being an instance of A implies having only R-successors which are instances of B. Thus, $\mathcal{B}' := \{A(a), A(b), R(a,b)\}$ is the (only) minimal ABox of $\mathcal{B}$.

In order to respect the interactions shown in Example 6.3, we define the redundancy $red_{CR}(\cdot)$ which returns a set of CAs superfluous due to interacting CAs and RAs. Let us first define the following entailment sets:

$$\begin{aligned} es_\exists(\mathcal{T},\mathbb{C}) &:= \{\exists R.C \sqsubseteq D \mid \mathcal{T} \models \exists R.C \sqsubseteq D \ \wedge \ R \in rn(\mathcal{T}) \ \wedge \ C,D \in \mathbb{C}\}, \\ es_\forall(\mathcal{T},\mathbb{C}) &:= \{C \sqsubseteq \forall R.D \mid \mathcal{T} \models C \sqsubseteq \forall R.D \ \wedge \ R \in rn(\mathcal{T}) \ \wedge \ C,D \in \mathbb{C}\}. \end{aligned}$$

Using the entailment sets $es_\exists(\cdot)$ and $es_\forall(\cdot)$, the redundancy $red_{CR}(\cdot)$ is defined as follows:

$$\begin{aligned} red_{CR}(\mathcal{B},\mathcal{O},\mathbb{C}) := \{D(a) \in \mathcal{B} \mid \\ \text{(i)} \ (\exists R.C \sqsubseteq D \in es_\exists(\mathcal{T},\mathbb{C}) \ \wedge \ R(a,b) \in \mathcal{B} \cup \mathcal{A} \ \wedge \ C(b) \in \mathcal{B} \cup \mathcal{A}) \vee \\ \text{(ii)} \ (C \sqsubseteq \forall R.D \in es_\forall(\mathcal{T},\mathbb{C}) \ \wedge \ R(b,a) \in \mathcal{B} \cup \mathcal{A} \ \wedge \ C(b) \in \mathcal{B} \cup \mathcal{A})\}. \end{aligned}$$

The cases (i) and (ii) cover the superfluous assertions $B(a)$ and $B(b)$, respectively, from Example 6.3. The entailment sets $es_\exists(\cdot)$ and $es_\forall(\cdot)$ are precomputed using a reasoner. Then, they are simply queried while computing $red_{CR}(\cdot)$, i.e. no additional reasoning is required afterwards.

Thus, certain entailment sets need to be precomputed in all cases: $ch(\cdot)$ and $rh(\cdot)$ are required for $red_C(\cdot)$ and $red_R(\cdot)$, respectively; $es_\exists(\cdot)$ and $es_\forall(\cdot)$ are required for $red_{CR}(\cdot)$. Otherwise, reasoning would need to be triggered multiple times for each assertion in the ABox $\mathcal{B}$. More specifically, computing the introduced redundancy functions would require the following number of entailment checks:

- up to $2 \cdot |\mathcal{B}| \cdot |\mathbb{C}|$ for $red_C(\cdot)$;
- up to $2 \cdot |\mathcal{B}| \cdot |\mathbb{R}|$ for $red_R(\cdot)$;
- up to $2 \cdot |\mathcal{B}| \cdot |\mathbb{C}| \cdot |rn(\mathcal{O})|$ for $red_{CR}(\cdot)$.

Reusing the aforementioned entailments makes identifying superfluous assertions relatively independent of $|\mathcal{B}|$ as reasoning is used for $\mathcal{T} \cup \mathbb{H}$. Moreover, we reuse the entailments for all hypotheses in a similar manner as it is suggested for computing dissimilarity and complexity above.

Let us gather all described kinds of redundancy and define the overall redundancy function $red^*(\cdot)$ as follows:

$$red^*(\mathcal{B}, \mathcal{O}, \mathbb{C}, \mathbb{R}) := red_C(\mathcal{B}, \mathcal{O}, \mathbb{C}) \cup red_R(\mathcal{B}, \mathcal{O}, \mathbb{R}) \cup red_{CR}(\mathcal{B}, \mathcal{O}, \mathbb{C}).$$

Once necessary entailment sets are computed, the function $red^*(\cdot)$ is relatively cheap to calculate. Essentially, it covers the cases of superfluity that can be easily identified. Nonetheless, it still does not cover *all* cases of superfluity, see Example 6.4.

Example 6.4. Consider the TBox $\mathcal{T}$ and the ABox $\mathcal{B}$:

$$\mathcal{T} := \{\exists R.(\exists S.A) \sqsubseteq B\},$$
$$\mathcal{B} := \{A(c), B(a), R(a,b), S(b,c)\}.$$

The assertion $B(a)$ is superfluous in $\mathcal{B}$ because a is connected to b which is connected to c, an instance of A, and $\mathcal{T}$ states that having at least one R-successor which has at least one S-successor which is an instance of A implies

being an instance of B. This is not detected by $red^*(\cdot)$.

Example 6.4 shows that "chains" of assertions are not considered by $red^*(\cdot)$. Indeed, it is easy to extend Example 6.4 to chains of arbitrary length. A similar effect of missing a superfluous assertion can be illustrated for $\mathcal{T}$ containing axioms with universal restrictions. In contrast to the above examples, such "misses" seems to be hard to cover since there are infinitely many of them. In principle, the definition of $red^*(\cdot)$ could be adjusted to cover two-step chains as in Example 6.4. However, the computation would require checking many additional entailments. Moreover, if the DL under consideration is more expressive than $\mathcal{ALC}$, additional undetected cases of superfluity can appear, see Example 6.5.

Example 6.5. Consider the TBox $\mathcal{T} := \{\top \sqsubseteq (\leq 1R.\top)\}$ and the ABox $\mathcal{B} := \{A(a), B(b), B(c), R(a,b), R(a,c)\}$. It follows that the individuals b and c are the same. Therefore, either $B(b)$ or $B(c)$ is superfluous in $\mathcal{B}$ which would not be detected by $red^*(\cdot)$.

Thus, the redundancy function $red^*(\cdot)$ finds some, but not necessarily all, superfluous assertions. Hence, it is an *approximation from below* for a maximal set of superfluous assertions. Using $red^*(\cdot)$, we define the approximation $dlen^*(\cdot)$ of the description length $dlen(\cdot)$, see Definition 5.22, as follows:

$$dlen^*(\mathcal{B}, \mathcal{O}, \mathbb{C}, \mathbb{R}) := \ell(\mathcal{B}) - \ell(red^*(\mathcal{B}, \mathcal{O}, \mathbb{C}, \mathbb{R})).$$

As $red^*(\cdot)$ is an approximation from below, $dlen^*(\cdot)$ is an *approximation from above*, i.e. $dlen^*(\cdot) \geq dlen(\cdot)$. Finally, by analogy with Definition 5.23, we define the approximations $fit^*(\cdot)$ and $bra^*(\cdot)$ of fitness and braveness, respectively, see Definition 6.2.

Definition 6.2 (Fitness and braveness approximations)**.** Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, $\mathbb{C}$ concepts, $\mathbb{R}$ roles, H a hypothesis consistent with $\mathcal{O}$, $\pi := \pi(\mathcal{O}, \mathbb{C}, \mathbb{R})$ the projection of $\mathcal{O}$, $\psi_H := \psi(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$ the assumption set of H. Then, the *approximations* of fitness and braveness of H are defined as follows:

$$\begin{aligned} fit^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) &:= dlen^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R}) - dlen^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R}) \\ bra^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) &:= dlen^*(\psi_H, \mathcal{O}, \mathbb{C}, \mathbb{R}) \end{aligned}$$

The approximations in Definition 6.2 can be rewritten using the redundancy $red^*(\cdot)$, see Lemma 6.2.

Lemma 6.2. *Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, $\mathbb{C}$ concepts, $\mathbb{R}$ roles, H a hypothesis consistent with $\mathcal{O}$. Then*

(i) $fit^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \ell(red^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R})) - \ell(red^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R}))$

(ii) $bra^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \ell(\psi_H) - \ell(red^*(\psi_H, \mathcal{O}, \mathbb{C}, \mathbb{R}))$

Proof. (i) $fit^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R}) - dlen^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R})$ (by Definition 6.2)

$= (\ell(\pi) - \ell(red^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R}))) - (\ell(\pi) - \ell(red^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R})))$
(by the definition of $dlen^*(\cdot)$)

$= \ell(red^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R})) - \ell(red^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R}))$.

(ii) $bra^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = dlen^*(\psi_H, \mathcal{O}, \mathbb{C}, \mathbb{R})$ (by Definition 6.2)

$= \ell(\psi_H) - \ell(red^*(\psi_H, \mathcal{O}, \mathbb{C}, \mathbb{R}))$ (by the definition of $dlen^*(\cdot)$).

□

Since $red^*(\cdot)$ underestimates the true redundancy, $fit^*(\cdot)$ can only underestimate $fit(\cdot)$ and $bra^*(\cdot)$ can only overestimate $bra(\cdot)$, respectively, see Lemma 6.3.

Lemma 6.3. *Let $\mathcal{O}$ be an ontology, $\mathbb{C}$ concepts, $\mathbb{R}$ roles, H a hypothesis consistent with $\mathcal{O}$. Then*

(i) $fit^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) \leq fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$

(ii) $bra^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) \geq bra(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$

Proof. (i) $fit^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \ell(red^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R})) - \ell(red^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R}))$ by Lemma 6.2. Due to the monotonicity of DLs, $red^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R})$ either misses the superfluous assertions missed by $red^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R})$ or misses additional assertions. Therefore, $fit^*(\cdot)$ cannot be bigger than $fit(\cdot)$.

(ii) $bra^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R}) = \ell(\psi_H) - \ell(red^*(\psi_H, \mathcal{O}, \mathbb{C}, \mathbb{R}))$ by Lemma 6.2. Since $red^*(\psi_H, \mathcal{O}, \mathbb{C}, \mathbb{R})$ is an approximation from below, $bra^*(\cdot)$ is an approximation from above.

□

Thus, these approximations are pessimistic: we can only underestimate the hypothesis quality. In other words, we tend to evaluate a hypothesis worse than it actually is. The approximations are based on the following simplifying *assumptions*.

- A minimal ABox $\mathcal{B}'$ is a subset of the given ABox $\mathcal{B}$, i.e. $\mathcal{B}' \subseteq \mathcal{B}$.
- The given ABox $\mathcal{B}$ contains no "chains" that, due to the TBox $\mathcal{T}$, make some assertions in $\mathcal{B}$ superfluous, see Example 6.4.
- The DL under consideration is at most as expressive as $\mathcal{ALC}$, see Example 6.5.

If all aforementioned assumptions are true, then the approximations coincide with the exact values of fitness and braveness. On the other hand, the further these assumptions are from the truth, the further the approximations are from the exact values. Informally, the assumptions are true if an ontology is moderately expressive and not structurally complex, i.e. neither contains complex concept assertions nor role assertion "chains" propagating concept information due to the TBox (this is true for the majority of our experimental ontologies described in Chapter 9).

Algorithm 4 computes the approximation of fitness. The projection π is computed by Definition 5.20 using instance retrievals. Please note that, if the statistical axiom measures, e.g. support, are computed before fitness, the results of their instance retrievals are cached and reused to compute the projection (the reverse is also true).

Algorithm 4 *computeFitness*$(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$

inputs
  H: a hypothesis
  $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$: an ontology
  $\mathbb{C}$, $\mathbb{R}$: sets of concepts and roles
outputs
  $fit^*(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$: the fitness approximation of H
do
  $\mathcal{T}^+ \leftarrow ce(\mathcal{T}, \mathbb{C})$ % extend $\mathcal{T}$ with auxiliary definitions
  classify $\mathcal{T}^+$ % build the concept and role hierarchy of $\mathcal{T}^+$
  $CH \leftarrow ch(\mathcal{T}, \mathbb{C})$; $RH \leftarrow rh(\mathcal{T}, \mathbb{R})$
  $es_\exists \leftarrow es_\exists(\mathcal{T}, \mathbb{C})$; $es_\forall \leftarrow es_\forall(\mathcal{T}, \mathbb{C})$
  $\pi \leftarrow \pi(\mathcal{O}, \mathbb{C}, \mathbb{R})$
  $red \leftarrow red^*(\pi, \mathcal{T}, \mathbb{C}, \mathbb{R})$ % use CH, RH, $es_\exists$, $es_\forall$
  update CH, RH, $es_\exists$, $es_\forall$ for $\mathcal{T} \cup H$
  $red_H \leftarrow red^*(\pi, \mathcal{T} \cup H, \mathbb{C}, \mathbb{R})$ % use CH, RH, $es_\exists$, $es_\forall$
  return $\ell(red_H) - \ell(red)$

Algorithm 4 can straightforwardly be altered to efficiently compute fitness for all hypotheses in a set. More specifically, Lines 8 – 13 are performed just once for all hypotheses. The redundancy red_H in the presence of a hypothesis H is recalculated for each hypothesis. It requires checking additional entailments that can be done using optimisations of reasoners, particularly the incremental mode of reasoning. Computing the approximation of braveness is analogous to fitness. The main difference is that the assumption set of a hypothesis needs to be computed instead of the projection.

6.5 Evaluating Hypotheses

We have defined the hypothesis quality measures in Chapter 5 and described how these measures can be computed in this chapter. We are now ready to present the overall hypothesis evaluation algorithm and thus uncover how Hypothesis Evaluator in Figure 4.1 works, see Algorithm 5.

Algorithm 5 *evaluateHypotheses*$(\mathbb{H}, Q, \mathcal{O})$

1: **inputs**
2: $\mathbb{H}$: a finite set of hypotheses
3: Q: a finite set of quality measures
4: $\mathcal{O}$: an ontology
5: **outputs**
6: $qf(H, q)$: the quality function
7: **do**
8: $\mathbb{C} \leftarrow \{C \mid C \sqsubseteq D \in \mathbb{H} \ \vee \ D \sqsubseteq C \in \mathbb{H}\}$ % get concepts
9: $\mathbb{R} \leftarrow \{R \mid R \sqsubseteq S \in \mathbb{H} \ \vee \ S \sqsubseteq R \in \mathbb{H}\}$ % get roles
10: **for** each $H \in \mathbb{H}$ **do**
11: **for** each $q \in Q$ **do**
12: $qf(H, q) \leftarrow computeQuality(H, q, \mathcal{O}, \mathbb{C}, \mathbb{R})$ % evaluate H by q
13: **end for**
14: **end for**
15: **return** $qf(H, q)$

In Algorithm 5, the function $computeQuality(\cdot)$ computes a quality measure q for a hypothesis H. In addition to a hypothesis to be evaluated, the quality measures can require other arguments to be specified, e.g. $sup(H, \mathcal{O})$ requires an ontology $\mathcal{O}$, $fit(H, \mathcal{O}, \mathbb{C}, \mathbb{R})$ requires an ontology $\mathcal{O}$, concepts $\mathbb{C}$, and roles $\mathbb{R}$. Therefore, the arguments $\mathcal{O}$, $\mathbb{C}$, $\mathbb{R}$ in $computeQuality(H, q, \mathcal{O}, \mathbb{C}, \mathbb{R})$ are provided in all cases, but used only if necessary. The function $computeQuality(\cdot)$ also

computes and caches all necessary objects, i.e. $ch(\cdot)$, $rh(\cdot)$, $es_{\exists}(\cdot)$, $es_{\forall}(\cdot)$, $\pi(\cdot)$, $\psi(\cdot)$, $red^{*}(\cdot)$, which are reused when necessary.

Algorithm 5 returns the quality function $qf(H, q)$ that returns the quality value of $H \in \mathbb{H}$ given $q \in Q$. The quality function can be used to rank hypotheses according to their quality values. We will discuss this in detail in Chapter 8.

6.6 Discussion

We have shown how to compute all measures (or their approximations) proposed in Chapter 5. Not surprisingly, some measures are relatively easy to compute while others require elaborated optimisations. Let us locate the quality measures on the "complexity spectrum" in terms of the number of standard reasoning operations, e.g. entailment checks, they require. The easiest measures are length and role depth as they do not require any reasoning. The middle group contains consistency, informativeness, logical strength, redundancy, as they depend only on a TBox and hypothesis, and the statistical axiom measures, as they require only instance retrievals. The most complex measures are dissimilarity, complexity, fitness, and braveness, as they additionally depend on the size of sets $\mathbb{C}$ and $\mathbb{R}$. In particular, fitness and braveness depend on a TBox, ABox, sizes $\mathbb{C}$ and $\mathbb{R}$ through the projection and assumption set.

The aforementioned considerations, however, should not be taken as direct anticipations of computational performance of the respective measures because some reasoning operations can be more costly than others. For example, instance retrieval can be (significantly) more expensive than entailment checking. We evaluate and discuss computational performance of the measures in Chapter 9.

While dissimilarity and complexity are computed exactly, fitness, and braveness are approximated to achieve their feasibility. In order to improve computational performance of the complex measures, we use the optimisation step from Section 6.1. Please note that it can be done just once to compute all complex measures for all hypotheses. One can observe that the measures respecting the DL semantics more than others, i.e. satisfying more properties in Table 5.3, seem to be harder for computation. Thus, although the measures with better semantic properties can be used to evaluate hypotheses thoroughly, they may come with higher computation price.

One should consider the aforementioned factors while selecting which set of

measures to use for hypothesis evaluation. If an ontology is relatively simple, i.e. neither large in size nor hard for reasoning, one can compute all measures in order to evaluate all sides of hypothesis quality. Otherwise, one can use only cheap measures (that we identify in Chapter 9).

Chapter 7

Constructing Hypotheses

Until now, we have discussed how to evaluate hypotheses, i.e. which quality measures can be used and how to compute those measures. However, little attention has been given to the problem of *constructing* hypotheses so far. The latter is discussed in detail in this chapter. From the perspective of the DL-Miner architecture, see Figure 4.1, this chapter uncovers the functionality and optimisations of Hypothesis Constructor.

7.1 Hypothesis Construction at a Glance

At first, the problem of constructing hypotheses seems to be trivial. Given a set $\mathbb{C}$ of concepts, we can generate all possible general concept inclusions (GCIs) using concepts from $\mathbb{C}$ as a left-hand side (LHS) or right-hand side (RHS). Analogously, given a set $\mathbb{R}$ of roles, we can generate all possible role inclusions (RIs) using roles from $\mathbb{R}$. As a result, we get all possible subsumption axioms "connecting" either two different concepts from $\mathbb{C}$ or two different roles from $\mathbb{R}$. Those axioms constitute all possible single-axiom hypotheses. After that, we can generate all possible multi-axiom hypotheses from the generated axioms. This gives the set $\mathbb{H}$ of all hypotheses. Then, we can evaluate each hypothesis $H \in \mathbb{H}$ by quality measures Q.

However, the method described above is infeasible in all but trivial cases. The reason is that the number of hypotheses grows rapidly with the sizes of sets $\mathbb{C}$ and $\mathbb{R}$, see Lemma 7.1.

Lemma 7.1. *Let $\mathbb{C}$ and $\mathbb{R}$ be sets of concepts and roles, respectively. Then, the*

size of the set $\mathbb{H}$ *of all hypotheses generated from* $\mathbb{C}$ *and* $\mathbb{R}$ *is as follows:*

$$|\mathbb{H}| = 2^m, \text{ where } m = |\mathbb{C}|^2 + |\mathbb{R}|^2.$$

Proof. The number of all GCIs generated from $\mathbb{C}$ equals $|\mathbb{C}|^2$ since they include all possible concept pairs from $\mathbb{C}$. Analogously, the number of all RIs equals $|\mathbb{R}|^2$. Thus, the total number of generated axioms is $m := |\mathbb{C}|^2 + |\mathbb{R}|^2$. Since $\mathbb{H}$ is the set of all subsets of those axioms (the power set), its size equals $|\mathbb{H}| = 2^m$. □

For example, given $|\mathbb{C}| = 5$ concepts and $|\mathbb{R}| = 3$ roles, the number of hypotheses equals $|\mathbb{H}| = 17,179,869,184$ according to Lemma 7.1. Thus, even relatively small numbers of concepts and roles lead to a vast hypothesis space (even if we exclude redundant, uninformative, and inconsistent hypotheses).

In order to maintain feasibility, we should prune the hypothesis space. One way to do that is to limit the number of axioms that a hypothesis is allowed to contain. This is reasonable because a shorter hypothesis is presumably easier to parse and understand, considering the results in Section 9.1.1. If the limit is sufficiently small, the number of hypotheses grows significantly slower, see Lemma 7.2.

Lemma 7.2. *Let* $\mathbb{C}$ *and* $\mathbb{R}$ *be sets of concepts and roles, respectively. Then, the size of the set* $\mathbb{H}_n$ *of all hypotheses of at most* n *axioms generated from* $\mathbb{C}$ *and* $\mathbb{R}$ *is as follows:*

$$|\mathbb{H}_n| = \sum_{k=1}^{n} \binom{m}{k}, \text{ where } m = |\mathbb{C}|^2 + |\mathbb{R}|^2.$$

Proof. The total number of axioms is $m := |\mathbb{C}|^2 + |\mathbb{R}|^2$, see Lemma 7.1. The number of combinations to select k items from m items is equal to the binomial coefficient $\binom{m}{k}$. As we can select from 1 to n axioms, the total number of combinations is the sum of the respective binomial coefficients, i.e. $|\mathbb{H}_n| = \sum_{k=1}^{n} \binom{m}{k}$. □

According to Lemma 7.2, the size $|\mathbb{H}_n|$ varies from m for $n = 1$ to $|\mathbb{H}|$ for $n \geq m$, i.e. $m \leq |\mathbb{H}_n| \leq |\mathbb{H}|$. Let us calculate the number of hypotheses for the same sizes $|\mathbb{C}| = 5$ and $|\mathbb{R}| = 3$, but with the limit of $n = 3$ axioms in a hypothesis. The size is $|\mathbb{H}_n| = 6,579$. Thus, $|\mathbb{H}_n|$ is 2,611,319 times smaller than $|\mathbb{H}|$.

Algorithm 6 generates hypotheses containing at most n axioms from sets $\mathbb{C}$ and $\mathbb{R}$. It makes use of the functions $ura(\cdot)$ and $urh(\cdot)$ that return unique

representatives of equivalent axioms and hypotheses, respectively, and defined analogously to $urc(\cdot)$, see Definition 6.1. Thus, hypotheses which are syntactic variations, see Definition 5.8, of unique representatives are discarded.

Algorithm 6 *generateHypotheses*$(\mathbb{C}, \mathbb{R}, n)$

inputs
  $\mathbb{C}$, $\mathbb{R}$: sets of concepts and roles
  $n \geq 1$: a maximal number of axioms in a hypothesis
outputs
  $\mathbb{H}_n$: the set of hypotheses
do
  $S_\mathbb{C} \leftarrow \{C \sqsubseteq D \mid C \neq D \ \wedge \ C, D \in \mathbb{C}\}$ % all concept inclusions
  $S_\mathbb{R} \leftarrow \{R \sqsubseteq S \mid R \neq S \ \wedge \ R, S \in \mathbb{R}\}$ % all role inclusions
  $S \leftarrow S_\mathbb{C} \cup S_\mathbb{R}$
  $S' \leftarrow ura(S)$ % discard syntactic variations
  $\mathbb{H} \leftarrow \emptyset$
  for each k from 1 to n **do**
    $\mathbb{H}_k \leftarrow getCombinations(S', k)$
      % get all subsets that contain exactly k elements
    $\mathbb{H} \leftarrow \mathbb{H} \cup \mathbb{H}_k$
  end for
  $\mathbb{H}' \leftarrow urh(\mathbb{H})$ % discard syntactic variations
  return $\mathbb{H}'$

Nevertheless, Algorithm 6 does not solve the hypothesis construction problem. It requires a set $\mathbb{C}$ of concepts and a set $\mathbb{R}$ of roles to be specified. Until now, we have assumed that these sets are given. For example, they can be specified by a domain expert. Unfortunately, this is a difficult problem on its own and may require significant human effort. In the following, we discuss how to construct sets $\mathbb{C}$ and $\mathbb{R}$ semi-automatically and fully automatically.

7.2 Top-down Construction

The simplest way of building concepts $\mathbb{C}$ and roles $\mathbb{R}$ is just selecting all concept and role names from the ontology, i.e. $\mathbb{C} := cn(\mathcal{O})$, $\mathbb{R} := rn(\mathcal{O})$. However, using these sets for further hypothesis generation, see Algorithm 6, leads to hypotheses that only contain atomic subsumptions. In other words, the result of learning is solely concept and role hierarchies. Although such hypotheses are presumably useful as a good starting point, there may be many more useful hypotheses to acquire since DLs offer expressivity far beyond atomic subsumptions, see Table 2.1.

To acknowledge the expressivity of DLs, we should develop a procedure that constructs complex concepts and roles besides atomic ones. A possible way to do that is building concepts $\mathbb{C}$ and roles $\mathbb{R}$ from some "seed" signature using certain construction rules. We call this *top-down construction*.

7.2.1 Selecting a Seed Signature

A seed signature can be specified by a domain expert. For example, one can be interested in acquiring hypotheses about some terms of interest. These terms, however, can be difficult to pick if the signature (vocabulary) of the ontology is large or the domain expertise is scarce, i.e. it is hard to decide which terms are likely to be related to each other. If this is the case, a seed signature Σ can simply be a set of all concept and role names occurring in the ontology, i.e. $\Sigma := \mathit{crn}(\mathcal{O})$.

Nonetheless, including all concept and role names in the seed signature can be excessive. Some of them can occur only in the TBox and have no "evidence" in the ABox. Hypotheses built from such terms can hardly be evaluated using the ABox because it contains no information about them. Therefore, in order to reduce the hypothesis space, it is reasonable to discard those terms from the seed signature. While identifying them, we should be careful because the TBox can semantically "connect" some terms to the ABox, see Example 7.1.

Example 7.1. Consider the ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$, where $\mathcal{T} := \{A \sqsubseteq B,\ C \sqsubseteq D\}$ and $\mathcal{A} := \{A(a), A(b)\}$. Although there is only one concept name A occurring in $\mathcal{A}$, the axiom $A \sqsubseteq B$ in $\mathcal{T}$ implies that the individuals a and b are instances of B. Hence, B is present in $\mathcal{A}$ implicitly. On the contrary, the concept names C and D have no connection to $\mathcal{A}$ and therefore can be excluded from the seed signature.

Data-connected terms can be identified by the means of the standard DL semantics. We use the modular structure of the ontology for this purpose. Formally, given an ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$, we select a seed signature Σ as follows:

$$\Sigma := \mathit{crn}(\mathcal{M}) \cup \{\top\},\ \text{where } \mathcal{M} := \bot\text{-}\mathit{module}(\mathcal{O},\ \mathit{crn}(\mathcal{A})).$$

The function $\bot\text{-}\mathit{module}(\mathcal{O}, \sigma)$ returns the $\bot$-module, see [GHKS08], of an ontology $\mathcal{O}$ given a signature σ. Informally, a module is a subset of the ontology which preserves all knowledge about the signature. It is aimed to be as small as possible but can contain some excessive axioms, i.e. it is not necessarily minimal.

As the seed signature Σ includes all concept and role names from the module $\mathcal{M}$, it may still contain some useless terms. Yet, the expectation is that the majority of them are contained in the excluded set $crn(\mathcal{O})\backslash crn(\mathcal{M})$. We always include $\top$ in Σ because all individuals are its instances, i.e. it always has evidence in the data.

7.2.2 Constructing Concepts from Templates

Given a seed signature Σ, we can use various construction rules to build complex concept and role expressions of $\mathbb{C}$ and $\mathbb{R}$. In the following, we discuss only concept construction as role construction is analogous. We define construction rules in the form of *templates* which are structurally complex concepts where concept and role names are variables.

We can obtain a set $\mathbb{D}_g$ of concepts conforming to a template g by substituting its variables with concept and role names from Σ. Clearly, a set $\mathbb{D}_g$ can contain many concepts that are syntactic variations of other concepts. Analogously to Definition 5.8 for hypotheses, we define syntactic variations for concepts, see Definition 7.1.

Definition 7.1 (Syntactic variation)**.** A concept C' is called a *syntactic variation* of a concept C if $C' \neq C$ and $C' \equiv C$.[1]

We would like to avoid syntactic variations of concepts since they produce syntactic variations of hypotheses. Given a set of templates G, the set $\mathbb{C}$ of concepts is obtained from G using Σ as follows:

$$\mathbb{C} := \bigcup_{g \in G} \mathbb{C}_g, \text{ where } \mathbb{C}_g := urc(\mathbb{D}_g).$$

Thus, the set $\mathbb{C}_g$ contains all unique representatives, see Definition 6.1, for $\mathbb{D}_g$ and, hence, no syntactic variations. Example 7.2 illustrates constructing concepts from templates.

Example 7.2. Consider the signature $\Sigma = \{\top, A, B, R\}$, where $A, B \in N_C$, $R \in N_R$, and templates $G = \{X,\ X \sqcap Y,\ \exists S.X\}$. The concept and role names are as follows: $\Sigma_C := \Sigma \cap N_C = \{\top, A, B\}$, $\Sigma_R := \Sigma \cap N_R = \{R\}$. The concept variables X, Y in G are substituted with concept names A, B and the role variable S in

[1] The statement $C' \equiv C$ is not an axiom but the abbreviation of $\emptyset \models C' \equiv C$.

G is substituted with the role name R. The set $\mathbb{C}$ of concepts is constructed as follows:

$$\begin{aligned}\mathbb{C} &= \{X \mid X \in \Sigma_C\} \cup \{X \sqcap Y \mid X, Y \in \Sigma_C\} \cup \{\exists S.X \mid X \in \Sigma_C \ \wedge \ S \in \Sigma_R\} \\ &= \{\top,\ A,\ B,\ A \sqcap B,\ \exists R.\top,\ \exists R.A,\ \exists R.B\}.\end{aligned}$$

Please notice that the concepts $A \sqcap \top$, $A \sqcap A$, $B \sqcap A$, etc. are not in $\mathbb{C}$ because they are syntactic variations of their respective unique representatives.

Algorithm 7 implements the top-down construction of concepts from templates. The top-down construction of roles from templates is analogous and skipped for the sake of brevity.

Algorithm 7 *buildConceptsTopDown*(Σ, G)

inputs
  Σ: a finite set of terms such that $\top \in \Sigma$
  G: a finite set of templates for concepts
outputs
  $\mathbb{C}$: the set of concepts
do
  $\Sigma_C \leftarrow \Sigma \cap N_C$ % get concept names
  $\Sigma_R \leftarrow \Sigma \cap N_R$ % get role names
  $\mathbb{C} \leftarrow \emptyset$
  for each $g \in G$ **do**
    $\mathbb{D}_g \leftarrow buildConceptsForTemplate(g, \Sigma_C, \Sigma_R)$ % build concepts using g
    $\mathbb{C}_g \leftarrow urc(\mathbb{D}_g)$ % remove syntactic variations
    $\mathbb{C} \leftarrow \mathbb{C} \cup \mathbb{C}_g$
  end for
  return $\mathbb{C}$

In essence, the top-down construction is a semi-automatic procedure since it requires a seed signature and templates to be specified, see Algorithm 7. While a seed signature can be selected automatically, see Section 7.2.1, templates should be defined by a human expert which possesses domain knowledge and basic ontology engineering skills. This is a trivial task if it is known which templates are required or interesting, e.g. atomic concepts, pairwise conjunctions, simple existential restrictions as in Example 7.2. However, it can be hard to know which templates are likely to produce useful hypotheses. If a set of templates is too small, i.e. easily definable, the resulting set $\mathbb{C}$ of concepts can miss many useful concepts. On the other hand, a large set of templates can be tedious to define.

In addition, it increases the hypothesis space tremendously, see Lemma 7.2.

7.3 Bottom-up Construction

Since the hypothesis space grows rapidly with the number of concepts $\mathbb{C}$, we should attempt to make $\mathbb{C}$ as small as possible and ensure that good hypotheses are not missing as a consequence of that minimisation. In other words, we need to predict which concepts are *suitable*, i.e. can potentially be useful for constructing good hypotheses. Instead of employing a domain expert to get this job done, we can consult the data (ABox) and make informed guesses. We call the process of constructing suitable concepts from the data *bottom-up construction*. In contrast to the top-down construction guided by a human, the bottom-up construction is guided by the data, i.e. it is data-driven.

In principle, the bottom-up construction can be done via the top-down construction, see Algorithm 7, that additionally tests the suitability of concepts using the data and removes unsuitable ones. However, this approach is probably infeasible. Indeed, the high expressivity of DLs requires specifying a large number of templates and leads to an enormous amount of concepts, unless a seed signature is tiny. This is true even for inexpressive DLs, e.g. $\mathcal{EL}$. For example, given n concept and m role names, a number of all $\mathcal{EL}$ complex concepts of length up to 5 grows as fast as $O(n^3 + n^2 \cdot m^2 + n \cdot m^4)$ (and a number of all GCIs grows quadratically faster). Many constructed concepts are likely to be useless because they are generated ignoring the ABox. Therefore, we design an approach that does not require templates and produces *only* suitable concepts via consulting the ABox.

7.3.1 Enumerating Concepts via Refinement Operators

The space of concepts can be enumerated (and generated if necessary) without using templates. Instead of defining multiple templates of increasing expressivity, we can use *refinement operators* to traverse the space of concepts.

As a reminder, refinement operators are commonly used in Concept Description Learning (CDL), see Section 3.2.1. A *downward refinement operator* ρ for $\mathcal{DL}$ specifies a set $\rho(C)$ of specialisations of a concept C in $\mathcal{DL}$. In the following,

we only consider downward refinement operators and omit "downward" for brevity. A concept C' is called a *specialisation* of a concept C if $C' \sqsubseteq C$.[2] We do not require $C \not\sqsubseteq C'$, i.e. a concept is a specialisation of itself, due to the reasons that become clear in the following.

Each specialisation $C' \in \rho(C)$ can further be specialised by the refinement operator ρ. Thus, if we start from the most general concept $\top$, we can enumerate all specialisations by repeatedly applying ρ, see Example 3.1. Clearly, we should use a seed signature in order to produce useful specialisations.

A refinement operator can satisfy certain properties that inform how useful it is. In particular, refinement operators can be *proper* and *complete*, see Definition 7.2.

Definition 7.2 (Proper and complete refinement operator [LH10])**.** A refinement operator ρ for $\mathcal{DL}$ is called

- *proper* if for every C, C' in $\mathcal{DL}$, $C' \in \rho(C)$ implies $C' \not\equiv C$;
- *complete* if for every C, C' in $\mathcal{DL}$, $C' \sqsubseteq C$ and $C \not\sqsubseteq C'$ implies there is C'' such that $C'' \equiv C'$ and $C'' \in \rho^n(C)$;

where $\rho^n(C)$ denotes a chain of n applications of ρ starting from C, $n \geq 1$.

Thus, a proper operator ρ never produces a specialisation $C' \in \rho(C)$ which is equivalent to the specialised concept C, i.e. ρ guarantees $C \not\sqsubseteq C'$. A complete operator guarantees that all specialisations *modulo equivalence*, i.e. at least their syntactic variations, can be reached from the specialised concept via some chain of applications of the operator. Properness and completeness are desirable properties for us because we would like to avoid syntactic variations, see Section 7.2.2, and ensure that we do not miss suitable concepts.

In [LH10], the authors prove that proper and complete refinement operators exist for the DLs $\mathcal{ALC}$, $\mathcal{ALCQ}$, $\mathcal{SHOIN}$, $\mathcal{SROIQ}$. Nevertheless, they design such operator only for $\mathcal{ALC}$ (and hence less expressive DLs such as $\mathcal{EL}$). Besides properness and completeness, there are other desirable properties of a refinement operator: finiteness and non-redundancy [LH10]. These are handled not by operator's design, but by a CDL algorithm that restricts the operator in some way, e.g. limits the maximal length of a specialisation.

[2]The statement $C' \sqsubseteq C$ is not an axiom but the abbreviation of $\emptyset \models C' \sqsubseteq C$.

In addition, the operator for $\mathcal{ALC}$ in [LH10] is defined such that it generates specialisations only in negation normal form (NNF).[3] It also guarantees to generate shortest specialisations. Thus, besides being proper and complete, it holds two additional properties. We call such operator *suitable*, see Definition 7.3.

Definition 7.3 (Suitable refinement operator)**.** A proper and complete refinement operator ρ for $\mathcal{DL}$ is called *suitable* if the following properties hold:

- for every C, C' in $\mathcal{DL}$, $C' \in \rho(C)$ implies C' is in NNF;
- for every C in $\mathcal{DL}$, C is in NNF implies there is $C' \in \rho^n(\top)$, where $n \geq 0$, such that $C' \equiv C$ and $\ell(C') \leq \ell(C)$.

Please notice that a proper and complete refinement operator ρ that holds the first property can still lack the second one, i.e. be not suitable, see Example 7.3.

Example 7.3. Consider a proper and complete refinement operator ρ that produces specialisations only in NNF such that for every $A \in N_C$ $A \sqcap A \in \rho^n(\top)$ for some n and $A \notin \rho^m(\top)$ for all m. Informally, ρ does not generate specialisations which are atomic concepts A. This does not affect its properness and completeness because it generates (superfluous) syntactic variations $A \sqcap A$. Thus, specialisations of ρ are not shortest. Therefore, ρ is not suitable by Definition 7.3. Please notice that Definition 7.3 does not require that ρ must avoid superfluous specialisations, i.e. ρ' generating A along with $A \sqcap A$ is suitable.

Intuitively, a refinement operator ρ is suitable if ρ "works well" for length $\ell(\cdot)$, i.e. ensures that useful specialisations are not pruned if we restrict the maximal length ℓ_{max} of specialisations in order to make their set finite. What that means exactly and why it is important is clarified in the following. If a refinement operator ρ is proper and complete, the suitability conditions for ρ is rather easy to ensure by its design (as it is done in [LH10]). In the following, we use only suitable refinement operators. Please note that, although we employ refinement operators to enumerate concepts, we do not use any CDL algorithm as it is supervised and solves a different problem.

7.3.2 Concept Support

Refinement operators allow for enumerating concepts using only a seed signature, i.e. templates or similar human-defined inputs are not required. Nevertheless,

[3] A DL concept is in NNF if the negation operator $\neg$ is only applied to concept names, e.g. $\neg A \sqcup \neg B$ is in NNF and $\neg(A \sqcap B)$ is not in NNF.

they can still produce too many specialisations, unless a seed signature is tiny. Therefore, we consult the ABox in order to enumerate only suitable concepts. Thus, the data informs the choice of concepts and only those are specialised via operators.

Let us discuss when a concept is *suitable* and how it can be identified. By suitable concepts we mean those which are at least minimally supported by the data and therefore can potentially be the LHS or RHS of an axiom in a good hypothesis. Since there are generally many concepts to check, the procedure of testing the suitability of a concept should be computationally efficient. We suggest to identify suitable concepts (and unsuitable ones) based on the number of their instances, or *support*, see Definition 7.4.

Definition 7.4 (Concept support)**.** Let $\mathcal{O}$ be an ontology, C a concept. The support p of C in $\mathcal{O}$ is defined as follows:

$$p(C, \mathcal{O}) := |inst(C, \mathcal{O})|.$$

Concepts with very low support are deemed to have insufficient evidence in the data, i.e. unsuitable for hypothesis construction. The support of a concept determines an upper bound for the basic support of any single-axiom hypothesis with that concept on the LHS or RHS, see Lemma 7.3.

Lemma 7.3. *Let $\mathcal{O}$ be an ontology, C a concept. For any hypothesis $H := \{C \sqsubseteq D\}$ or $H := \{D \sqsubseteq C\}$ it holds that $bsup(H, \mathcal{O}) \leq p(C, \mathcal{O})$.*

Proof. By Definition 5.14 $bsup(H, \mathcal{O}) = |inst(C \sqcap D,\ \mathcal{O})|$, by Definition 7.4 $p(C, \mathcal{O}) := |inst(C, \mathcal{O})|$. By Lemma 5.6 $|inst(C \sqcap D,\ \mathcal{O})| \leq |inst(C, \mathcal{O})|$. Therefore $bsup(H, \mathcal{O}) \leq p(C, \mathcal{O})$. □

From now on, we call a concept *suitable* if its support exceeds a certain threshold (discussed in the following) and *unsuitable* otherwise.

7.3.3 DL-Apriori: a Concept Mining Algorithm

Definition 7.4 of concept support can be used to *efficiently* construct suitable concepts via refinement operators. In particular, the majority of unsuitable concepts can be detected in advance and pruned from the search *a priori*. This becomes possible due to the mechanics of refinement operators. Indeed, a specialisation of a concept cannot have more instances than the concept has, see Lemma 7.4.

Lemma 7.4 (Anti-monotone property of concepts)**.** *Let $\mathcal{O}$ be an ontology, C, D concepts. Then, $C \sqsubseteq D$ implies $p(C, \mathcal{O}) \leq p(D, \mathcal{O})$.*

Proof. Follows from Definition 5.12 because $C \sqsubseteq D$ implies that every instance of C is also an instance of D. □

Considering Definition 7.4, Lemma 7.4 implies that, if a concept is unsuitable, then all its specialisations are also unsuitable. Therefore, they can be safely pruned from the search. We call Lemma 7.4 the *anti-monotone property of concepts* because it resembles Lemma 2.1 which is the anti-monotone property of itemsets in Association Rule Mining, ARM, see Section 2.2.5. Thus, it is essentially the same property which we have defined for DL concepts.

We combine refinement operators and the anti-monotone property of concepts in one algorithm. We call it DL-Apriori because, like Apriori, see Section 2.2.5, it makes use of the anti-monotone property to prune unsuitable candidates *a priori*. Specifically, DL-Apriori methodically traverses concepts via refinement operators and discards multiple unsuitable concepts using the anti-monotone property instead of explicitly testing the suitability for each of them, i.e. it performs an *informed enumeration*.

DL-Apriori, see Algorithm 8, operates as follows. It uses two sets of concepts. The first one, $\mathbb{C}$, is the final set of suitable concepts. The second one, $\mathbb{D}$, is the set of concepts yet to be specialised which is initialised to contain $\top$. The main loop of DL-Apriori consists of the following steps. First, we pick and remove from $\mathbb{D}$ a concept C to be specialised and add C to the final set $\mathbb{C}$. Then, using a suitable refinement operator ρ for $\mathcal{DL}$, we generate the set ρ_C of all specialisations of C, where each specialisation is built from Σ and is not longer than ℓ_{max} (therefore ρ_C is finite). After that, we discard specialisations in ρ_C which are syntactic variations, taking $\mathbb{C}$ and $\mathbb{D}$ into account. Then, we check the support of each specialisation and identify suitable ones, i.e. those whose support is at least p_{min}. These specialisations are added to the concepts $\mathbb{D}$ to be further specialised. The steps are made in this order because discarding syntactic variations saves checking their support that can be computationally costly. In addition, it ensures that each concept is specialised only once thus saving even more checks. Once the set $\mathbb{D}$ is empty, DL-Apriori terminates and returns the final set $\mathbb{C}$.

The run cycle of DL-Apriori can be illustrated as the construction of a *specialisation tree* where nodes are concepts and edges connect specialisations to

Algorithm 8 DL-Apriori $(\mathcal{O}, \Sigma, \mathcal{DL}, \ell_{max}, p_{min})$

inputs
  $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$: an ontology
  Σ: a finite set of terms such that $\top \in \Sigma$
  $\mathcal{DL}$: a DL for concepts
  ℓ_{max}: a maximal length of a concept such that $1 \leq \ell_{max} < \infty$
  p_{min}: a minimal concept support such that $0 < p_{min} \leq |in(\mathcal{O})|$
outputs
  $\mathbb{C}$: the set of suitable concepts
do
  $\mathbb{C} \leftarrow \emptyset$ % initialise the final set of suitable concepts
  $\mathbb{D} \leftarrow \{\top\}$ % initialise the set of concepts yet to be specialised
  $\rho \leftarrow getOperator(\mathcal{DL})$ % initialise a suitable operator ρ for $\mathcal{DL}$
  while $\mathbb{D} \neq \emptyset$ **do**
    $C \leftarrow pick(\mathbb{D})$ % pick a concept C to be specialised
    $\mathbb{D} \leftarrow \mathbb{D} \backslash \{C\}$ % remove C from the concepts to be specialised
    $\mathbb{C} \leftarrow \mathbb{C} \cup \{C\}$ % add C to the final set
    $\rho_C \leftarrow specialise(C, \rho, \Sigma, \ell_{max})$ % specialise C using ρ
    $\mathbb{D}_C \leftarrow \{D \in urc(\rho_C) \mid \nexists D' \in \mathbb{C} \cup \mathbb{D} : D' \equiv D\}$ % discard variations
    $\mathbb{D} \leftarrow \mathbb{D} \cup \{D \in \mathbb{D}_C \mid p(D, \mathcal{O}) \geq p_{min}\}$ % add suitable specialisations
  end while
  return $\mathbb{C}$

the respective specialised concept. The construction begins from the root node $\top$. It repeatedly specialises every leaf node which satisfies the restrictions and is not a syntactic variation. Once there is no such leaf node, the algorithm terminates. All nodes, except leaf nodes, of the constructed tree constitute the final set of suitable concepts. Example 7.4 illustrates the bottom-up concept construction using DL-Apriori.

Example 7.4. Consider the ontology $\mathcal{O}$ used in Example 3.1.

$$\begin{aligned}\mathcal{O} := \{&Man \sqsubseteq \neg Woman,\ hasParent \sqsubseteq hasChild^{-},\\ &Man(Arthur),\ Man(Chris),\ Man(James),\\ &Woman(Penelope),\ Woman(Victoria),\\ &Woman(Charlotte),\ Woman(Margaret),\\ &hasParent(Charlotte, James),\ hasParent(Charlotte, Victoria),\\ &hasParent(Victoria, Chris), hasParent(Victoria, Penelope)\\ &hasParent(Arthur, Penelope),\ hasParent(Arthur, Chris)\}.\end{aligned}$$

We illustrate DL-Apriori with the following input parameters: the seed signature is $\Sigma := \{\top, Man, Woman, hasChild\}$, the target DL is $\mathcal{EL}$ (the refinement operator can only use $\mathcal{EL}$ constructors, i.e. $\exists$ and $\sqcap$), the minimal support is $p_{min} := 1$ (every concept must have at least one instance), the maximal length is $\ell_{max} := 4$ (length of every concept must not exceed 4). Figure 7.1 shows the constructed tree, where M, W, c stand for Man, $Woman$, $hasChild$, respectively. All leaf nodes of the constructed tree are omitted for the sake of brevity, i.e. all nodes (including new leaf nodes) in Figure 7.1 are suitable.

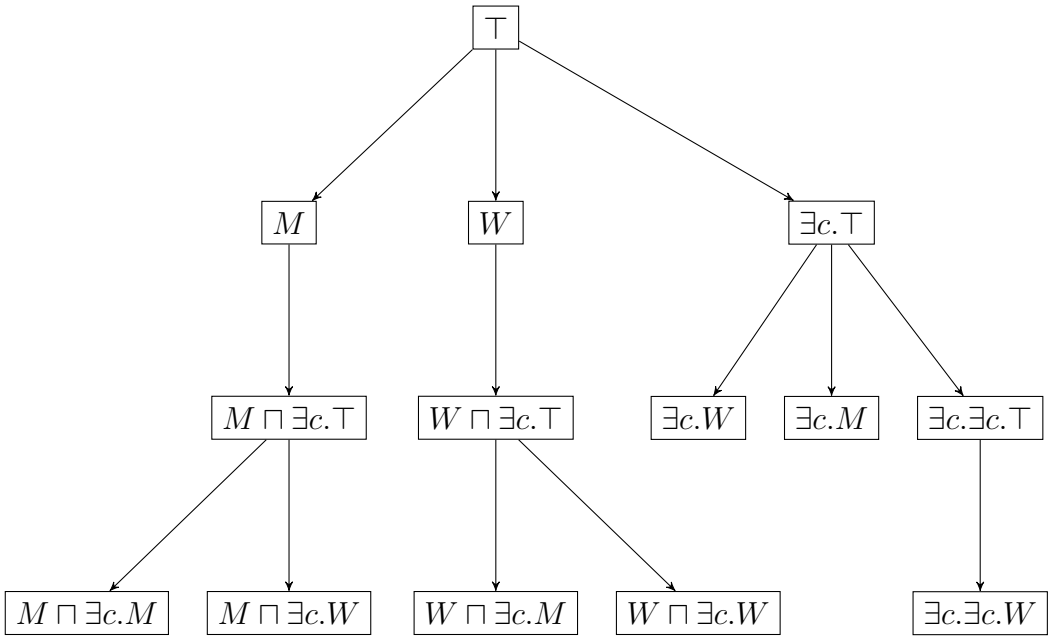


Figure 7.1: Specialisation tree constructed by DL-Apriori

Let us examine the leaf nodes of the tree in Figure 7.1. The concept $C_1 := W \sqcap \exists c.W$, a descendant of $W \sqcap \exists c.\top$, represents all women that have at least one daughter. Its length $\ell(C_1) = 4$ and support $p(C_1, \mathcal{O}) = 2$ since C_1 has two instances, *Penelope* and *Victoria*. The concept $C_2 := \exists c.\exists c.W$ represents all grandparents that have at least one granddaughter. Its length $\ell(C_2) = 3$ and support $p(C_1, \mathcal{O}) = 2$ since C_2 has two instances, *Penelope* and *Chris*. All specialisations of C_1 and C_2 violate the maximal length restriction. Although the concepts $M \sqcap W$ and $\exists c.\exists c.M$ satisfy the maximal length restriction, they are absent because they do not meet the minimal support, i.e. they have no instances. The concept $\exists c.W$ does not have the descendant $\exists c.W \sqcap W$ because it is a syntactic variation of $W \sqcap \exists c.W$ which is a descendant of $W \sqcap \exists c.\top$. Another descendant $\exists c.W \sqcap M$ of $\exists c.W$ and descendants of $\exists c.M$ are absent due to the

same reason. All nodes of the tree in Figure 7.1 constitute the set $\mathbb{C}$ of concepts returned by DL-Apriori.

Let us discuss the parameters of DL-Apriori and how they affect the constructed set $\mathbb{C}$ of concepts. We call the parameter values *legal* if they satisfy the constraints of Algorithm 8. The seed signature Σ plays the same role as in the top-down construction. It can be automatically selected as discussed in Section 7.2.1 and should include $\top$. The target DL $\mathcal{DL}$ determines the expressivity of concepts in $\mathbb{C}$, i.e. which DL constructors are allowed. In Example 7.4 we use the $\mathcal{EL}$ refinement operator which produces specialisations via $\mathcal{EL}$ constructors. In principle, any DL is acceptable for which a suitable refinement operator exists, e.g. $\mathcal{ALC}$.

The constraint ℓ_{max} specifies the maximal (finite) length of concepts under consideration. It can significantly reduce the search space and the resulting set $\mathbb{C}$ of concepts and hence improve the computational performance of the algorithm. Considering the results of the experiment in Section 9.1.1, people tend to use short concepts. Therefore, in order to avoid constructing hardly readable concepts, we suggest to set ℓ_{max} to sufficiently small values, e.g. $\ell_{max} := 5$.

The minimal support p_{min} determines the minimal number of instances that a concept must have to be considered as suitable. It is required to be from 1 to $|in(\mathcal{O})|$ because, if $p_{min} < 1$, then it does not play any role and, if $p_{min} > |in(\mathcal{O})|$, then no concept satisfies it. This threshold should be chosen carefully because the support of a concept is just an estimate of concept's usefulness for hypothesis construction. More specifically, it should be sufficiently low to avoid missing potentially useful concepts. The safest value is 1, i.e. all concepts having at least one instance are suitable, and can be always chosen under the absence of information about quantity of individuals. However, if it is known that an ontology contains numerous individuals, slightly increasing the threshold may help to rule out many unpromising concepts and thus decrease the search space massively. In this sense, it may be helpful to view p_{min} as a "noise threshold", i.e. concepts which do not reach it can be seen as "noisy" and unlikely to be useful, e.g. $M \sqcap W$ in Example 7.4.

7.3.4 Correctness, Completeness, and Termination of DL-Apriori

DL-Apriori, see Algorithm 8, always *terminates*. Another important property of DL-Apriori is that it guarantees to return all concepts modulo equivalence satisfying the constraints, i.e. it is *complete*, and only concepts satisfying the constraints, i.e. it is *correct*. The constraints are given not only by the minimal support p_{min} and maximal length ℓ_{max} but also by the target $\mathcal{DL}$ and seed signature Σ.

We prove these properties in Lemma 7.5. In order to make its presentation and proof easier to grasp, let us first discuss what reasons could cause Algorithm 8 to violate the aforementioned properties and consider some examples. In particular, we discuss what could break termination and completeness.

Algorithm 8 does not terminate if it enumerates an infinite set of specialisations or specialises the same concept infinitely many times. Hence, to ensure termination, we need to prevent these cases. The first case can be prevented by the finite input parameters Σ and ℓ_{max}. The second case can be prevented via avoiding infinite loops during traversal of the specialisation tree, i.e. when a concept or its syntactic variation is visited infinitely many times.

Algorithm 8 looses completeness if it misses some concepts. Clearly, this can happen if a refinement operator ρ is not complete. This can also happen if ρ is complete but not suitable. Consider $\ell_{max} := 2$ and ρ from Example 7.3. Despite the fact that ρ is complete, it does not generate specialisations which are atomic concepts A but generates $A \sqcap A$ instead. Since $\ell_{max} := 2$, specialisations of the form $A \sqcap A$ violate ℓ_{max} and, consequently, the algorithm misses all atomic concepts A. Another example is generating specialisations $\forall R.C \sqcap \forall R.D$ instead of $\forall R.(C \sqcap D)$ given $\ell_{max} := 4$.

In addition, we should consider completeness only for concepts in NNF because some of them can be rewritten in a shorter syntactic form. For example, if $\ell_{max} := 4$, Algorithm 8 misses the concept $\neg(A \sqcap B)$ because its NNF $\neg A \sqcup \neg B$, generated by ρ, violates ℓ_{max}, even if ρ is complete. If we choose a normal form different from NNF, we should change Definition 7.3 of a suitable refinement operator respectively such that the aforementioned cases are prevented. Thus, a suitable refinement operator should "work well" for a chosen normal form and definition of length in the DL it is designed for.

Lemma 7.5 (Correctness, completeness, termination)**.** *Let $\mathcal{O}$, Σ, $\mathcal{DL}$, ℓ_{max}, p_{min} be legal parameters of* DL-Apriori*, where $\mathcal{DL} \leq \mathcal{SROI}$ and ρ is a suitable refinement operator for $\mathcal{DL}$. Let (i) – (iv) be the following conditions for a concept C:*

(i) *C is in NNF and in $\mathcal{DL}$;*

(ii) *$\widetilde{C} \subseteq \Sigma$;*

(iii) *$\ell(C) \leq \ell_{max}$;*

(iv) *$p(C, \mathcal{O}) \geq p_{min}$.*

Then, all following properties hold for DL-Apriori*:*

- *it* terminates*;*
- *it is* correct*: it returns a set $\mathbb{C}$ of concepts such that $C \in \mathbb{C}$ implies C satisfies (i) – (iv);*
- *it is* complete*: if a concept D satisfies (i) – (iv), then there is $C \in \mathbb{C}$ such that $C \equiv D$.*

Proof. (Termination) The set of all concepts in $\mathcal{SROI}$ over Σ which are not longer than ℓ_{max} is finite. Hence, the specialisation tree that Algorithm 8 traverses is finite. Additionally, the algorithm visits each concept at most once: this is ensured by checking whether the concept or its syntactic variation is already visited, see Line 18 in Algorithm 8. Therefore, Algorithm 8 always terminates.

(Correctness) The concept $\top$ obviously satisfies (i) – (iv). Every $C \in \mathbb{C} \backslash \{\top\}$ satisfies (i) because a refinement operator ρ is suitable for $\mathcal{DL}$. C satisfies (ii) – (iv) due to Lines 17 and 19 in Algorithm 8. Therefore, Algorithm 8 is correct.

(Completeness) Assume a concept D satisfies (i) – (iv). Since ρ is complete, by Definition 7.2 there is some concept C such that $C \equiv D$ and $C \in \rho^n(\top)$. Since ρ is suitable, C satisfies (i) and (ii) by Definition 7.3. Moreover $\ell(C) \leq \ell(D)$ by Definition 7.3. Therefore, if D satisfies (iii), then C satisfies (iii), i.e. $\ell(C) \leq \ell(D) \leq \ell_{max}$. Since $C \equiv D$ and D satisfies (iv), C satisfies (iv), i.e. $p(C, \mathcal{O}) = p(D, \mathcal{O}) \geq p_{min}$. Thus, C satisfies (i) – (iv) and hence $C \in \mathbb{C}$. Therefore, Algorithm 8 is complete.

□

Please notice that the properties of DL-Apriori, see Lemma 7.5, do not hold for DLs with number restrictions $\geq k.C$ and $\leq k.C$, e.g. $\mathcal{SROIQ}$. The reason is that, by Definition 5.1, syntactic length of $\geq k.C$ and $\leq k.C$ is the same for any k. Hence, given a finite Σ, the maximal length restriction ℓ_{max} does not guarantee to make the number of specialisations finite. In particular, infinite specialisation chains are possible, e.g. $\geq (k+n).C = \rho^n(\geq k.C)$, where $n = \infty$. This can be fixed by adapting Definition 5.1 such that $\ell(\geq k.C) = \ell(\leq k.C) := f(k) + \ell(C)$, where $f(k)$ is some increasing function of k, i.e. $f(k+1) > f(k)$. Alternatively, we can add the parameter k_{max} to Algorithm 8 which bounds k in number restrictions, like ℓ_{max} bounds length of concepts. Both ways regain termination, correctness, and completeness of DL-Apriori for $\mathcal{SROIQ}$, but complicate the presentation.

7.4 Discussion

In order to construct concepts, we can use either the top-down, see Algorithm 7, or bottom-up construction, see Algorithm 8. The top-down construction requires templates to be defined by a human and does not consult the data. This may cause missing many useful concepts and constructing an overwhelming number of useless concepts. In contrast, the bottom construction does not require templates, as it methodically enumerates all concepts, and produces only suitable concepts given the suitability threshold. Therefore, for concepts, we favour the bottom-up construction.

To the best of our knowledge, refinement operators for specialising roles are not investigated. Therefore, we do not define DL-Apriori for roles as it is done for concepts. Instead, we use the top-down construction, see Algorithm 7, analogously implemented for roles. In practice, roles, unlike concepts, can easily be generated via the top-down construction. The reason is that DLs normally allow for much greater structural variability in concepts than in roles. As a result, templates are usually easier to specify for roles than for concepts and the top-down construction produces much fewer roles than concepts. If there are too many roles constructed, we use the notion of role suitability and discard unsuitable roles. Role suitability is defined analogously to concept suitability using instances of a role, or its support, i.e. individual pairs that are connected by the role according to the ontology.

We automatically select a seed signature as described in Section 7.2.1 for constructing concepts and roles. Once concepts and roles are constructed, we use them to build hypotheses via Algorithm 6. We normally set the maximal number n of axioms in a hypothesis to be a sufficiently small number because the hypothesis space grows quickly with n, see Lemma 7.2. Moreover, it is hard for a human expert to comprehend and judge large sets of axioms.

Chapter 8

DL-Miner: a Hypothesis Mining Algorithm

We have discussed hypothesis construction and evaluation and proposed approaches that can be used to solve these problems. In the architecture of DL-Miner, see Figure 4.1, these approaches are implemented by two core functional blocks: Hypothesis Constructor and Hypothesis Evaluator.

In this chapter, we discuss two remaining functional blocks of DL-Miner: Ontology Cleaner and Hypothesis Sorter. The purpose of Ontology Cleaner is to prepare the ontology for processing, i.e. identify inconsistency and repair the ontology. The purpose of Hypothesis Sorter is to order hypotheses according to their quality values measured by the respective quality measures. Finally, we combine all functional blocks in one system and their subroutines in one algorithm. We discuss its parameters, properties, and implementation.

8.1 Handling Noisy Data

Until now, we have assumed that any input ontology is consistent. On the one hand, this assumption makes sense because high quality, curated ontologies should be consistent. For example, most ontologies in the BioPortal repository[1] are consistent, see Section 9.1.1. On the other hand, inconsistent ontologies do exist. Moreover, an ABox is sometimes automatically extracted from the data, but not manually engineered. In this case, inconsistencies can easily creep in since the

[1] `http://bioportal.bioontology.org/`

data can contain noise or errors. Therefore, it is worthwhile to consider the situation when an input ontology is inconsistent.

8.1.1 Handling Inconsistent Data

If an ontology is inconsistent, it is not sensible to use reasoning with respect to that ontology. Consequently, no techniques based on such reasoning work correctly. In particular, it is not possible to reasonably evaluate hypotheses. More specifically, most logical measures, i.e. informativeness, consistency, dissimilarity, complexity, are useless. The statistical measures are also worthless as they use instance retrieval. Although one can try to approximate their values by ignoring the TBox, such approximations can be considerably misleading. Not only does hypothesis evaluation become hardly possible, but so does hypothesis construction. Indeed, since the bottom-up construction uses instance checking, it cannot be used for an inconsistent ontology, unless instance checking is reasonably approximated. Thus, given an inconsistent ontology, the core functional blocks do not operate appropriately.

Instead of ignoring an inconsistent input, we can attempt to *repair* it, i.e. make the ontology consistent by eliminating causes of inconsistency. As repairing ontologies manually can be a tedious and error-prone task, automated approaches to assist repairing are investigated [HS05, HPS09]. The approach in [HPS09] uses *justifications*, see Definition 8.1.

Definition 8.1 ([HPS09])**.** Let $\mathcal{O}$ be an ontology and η an entailment of $\mathcal{O}$, i.e. $\mathcal{O} \models \eta$. A set $\mathcal{J}$ of axioms is called a *justification* for η in $\mathcal{O}$ if $\mathcal{J} \subseteq \mathcal{O}$, $\mathcal{J} \models \eta$, and for every $\mathcal{J}' \subsetneq \mathcal{J}$ it holds that $\mathcal{J}' \not\models \eta$.

Informally, a justification for a given entailment is a minimal subset of the ontology that entails the entailment. Justifications provide explanations for an entailment as they contain only axioms causing that entailment. Multiple justifications can exist for a given entailment.

Although justifications can help to repair the inconsistency manually, they are not straightforward to employ for automatic repair [HPS09]. In practice, the manual repair can be difficult because one needs to remove axioms from the ontology such that *all* justifications for inconsistency, i.e. $\mathcal{O} \models \top \sqsubseteq \bot$, are broken. Of course, one can simply remove or weaken all axioms that occur in at

least one justification. However, such approach can remove a lot of axioms that do not need to be excluded to prevent inconsistency.

In our scenario, we assume that an ontology is inconsistent because of some erroneous, contradictory assertions in the ABox. We also assume that such assertions constitute a small fraction (noise) of the ABox. Otherwise, the ABox is unlikely to be useful for evaluating hypotheses anyway. Under these assumptions, we can repair an inconsistent ontology via removing all assertions that occur in at least one justification for inconsistency. We call those assertions a *data repair*, see Definition 8.2.

Definition 8.2 (Data repair)**.** Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an inconsistent ontology, where $\mathcal{T}$ is consistent. Let $just(\mathcal{O}, \eta)$ be the set of all justifications for an entailment η in $\mathcal{O}$. The *data repair* $rep(\mathcal{O})$ of $\mathcal{O}$ is defined as follows:

$$rep(\mathcal{O}) := \mathcal{A} \cap \bigcup just(\mathcal{O},\ \top \sqsubseteq \bot).$$

The ontology $\mathcal{O}' := \mathcal{O} \backslash rep(\mathcal{O})$ is called *repaired*.

The repaired ontology $\mathcal{O}'$ is the result of removing the repair $rep(\mathcal{O})$ from the inconsistent ontology $\mathcal{O}$. Please notice that the data repair $rep(\mathcal{O})$ is unique because the set of all justifications for an entailment is unique. The repaired ontology $\mathcal{O}'$ is guaranteed to be consistent, see Lemma 8.1.

Lemma 8.1. *Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an inconsistent ontology, where $\mathcal{T}$ is consistent. Then, $\mathcal{O}' := \mathcal{O} \backslash rep(\mathcal{O})$ is consistent.*

Proof. Let $S := just(\mathcal{O},\ \top \sqsubseteq \bot)$. Since $\mathcal{T}$ is consistent, each $\mathcal{J} \in S$ overlaps with $\mathcal{A}$, i.e. $\mathcal{J} \cap rep(\mathcal{O}) \neq \emptyset$. Hence, each $\mathcal{J} \in S$ is "broken" in $\mathcal{O}'$ and hence $\mathcal{O}'$ is consistent. □

Clearly, a data repair can contain excessive axioms, i.e. it is not minimal. In principle, it can be improved to include only axioms necessary for breaking all justifications. In the most extreme case, all justifications share the same axiom. In this case, it is sufficient to include only that axiom in the repair because, once it is removed from the ontology, no justification remains to cause inconsistency. Nevertheless, finding a minimal data repair is computationally more expensive. It requires judging axioms in all justifications, which can be numerous for a large ABox. Thus, it is easier to throw out all axioms which are potentially responsible

for inconsistency. Since an inconsistent ontology contains presumably few noisy assertions, the number of removed axioms due to the repair is small in comparison to the size of the ontology. Algorithm 9 implements the data repair.

Algorithm 9 *repairOntology*($\mathcal{O}$)

inputs
  $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$: an ontology
outputs
  $\mathcal{O}'$: the repaired ontology
do
  $S \leftarrow just(\mathcal{O}, \top \sqsubseteq \bot)$ % compute all justifications for inconsistency
  $\mathcal{O}_\bot \leftarrow \emptyset$
  for each $\mathcal{J} \in S$ **do**
    $\mathcal{O}_\bot \leftarrow \mathcal{O}_\bot \cup \mathcal{J}$ % collect axioms of all justifications
  end for
  $\mathcal{A}_\bot \leftarrow \mathcal{A} \cap \mathcal{O}_\bot$ % retain only ABox axioms
  $\mathcal{O}' \leftarrow \mathcal{O} \backslash \mathcal{A}_\bot$
  return $\mathcal{O}'$

8.1.2 Handling Consistent But Incorrect Data

Even if an ontology is consistent, it can still contain some erroneous data. Those errors are harder to detect than inconsistencies. In particular, such data can cause good hypotheses to be inconsistent with the ontology. In principle, we can consider the ontology augmented with the hypothesis as a new ontology and repair it as discussed above in Section 8.1.1. Formally, given a consistent ontology $\mathcal{O}$ and a hypothesis H such that $\mathcal{O}_H := \mathcal{O} \cup H$ is inconsistent, we compute $\mathcal{O}'_H := \mathcal{O} \backslash rep(\mathcal{O}_H)$ and use $\mathcal{O}' := \mathcal{O}'_H \backslash H$ to evaluate H. Nonetheless, this can be considerably more expensive computationally than handling an inconsistent input (which is already costly) because it needs to be done for every inconsistent hypothesis.

There are other kinds of errors in the data which are not possible to detect using inconsistency. Interestingly, those can be spotted by a domain expert exploring the acquired hypotheses, as we discuss in Chapter 9. This opens another potential application for DL-Miner.

8.2 Ordering and Ranking Hypotheses

Once hypotheses $\mathbb{H}$ are evaluated, the quality function $qf(H, q)$ is returned by Algorithm 5. Then, we can order hypotheses by their quality values using $qf(\cdot)$. Since we evaluate hypotheses by multiple quality measures in general, there are different ways to order them.

8.2.1 Single-Measure Ordering

The easiest way to order hypotheses $\mathbb{H}$ is just sorting them by a single quality measure $q \in Q$, see Definition 8.3.

Definition 8.3 (Single-measure ordering)**.** Let $\mathbb{H}$ be a hypothesis space, Q a set of positive[2] quality measures, $qf(H, q)$ the quality function, where $H \in \mathbb{H}$, $q \in Q$. Given a quality measure $q \in Q$, a hypothesis H is *better* on q than a hypothesis H', written as $H' \prec_q H$, if $qf(H', q) < qf(H, q)$.

Informally, given a quality measure, one hypothesis is better than another if its quality value is greater. Please note that any negative measure can be turned into a positive one by simply reversing its sign, i.e. multiplying it by -1. One can compute $(H, \prec_q)$ for each quality measure $q \in Q$ and navigate through the hypotheses $\mathbb{H}$ switching the measures when necessary. The best hypotheses are those which have the maximal value of q among all.

As we generally evaluate hypotheses by multiple quality measures, one hypothesis can be better than another hypothesis on one measure and be worse on another measure. In this case, it can be hard to conclude which one is better overall, i.e. hypotheses are *incomparable*. Thus, one can opt to order hypotheses not by each measure separately, but by a set of measures jointly.

In order to consider all measures Q simultaneously, we can aggregate them into a single, collective measure q_0 and then use the single-measure ordering $\prec_{q_0}$, see Definition 8.3. Based on $\prec_{q_0}$, we can rank hypotheses, i.e. define a ranking function $rf(H)$, such that a hypothesis of the highest rank has the maximal value of q_0.

Since the quality measures have different scales, each measure $q \in Q$ should first be normalised. More specifically, all values returned by $qf(\cdot)$ should be

[2] A measure is positive if a greater value indicates a hypothesis of better quality. Otherwise, it is negative. This should not be confused with the sign of a measure, i.e. whether it is greater or below zero.

scaled such that each measure $q \in Q$ can be compared on the same scale with other measures. We denote the scaled quality function as $\widehat{qf}(\cdot)$. In addition, we can specify a weight $w_q \geq 0$ for each measure $q \in Q$ that indicates how important it is in the set. If all measures in Q are equally important, we can simply set $w_q = 1$ for each $q \in Q$. Various aggregation schemes can be used. For example, the collective quality measure can be defined as follows:

$$q_0(H) := \sum_{q \in Q} w_q \cdot \widehat{qf}(H, q).$$

Nonetheless, some quality measures are not straightforward to scale because they have unbounded values, e.g. lift, conviction. In addition, even if scaled, the measures can have different distributions of values. As a result, a measure having mostly low values becomes unimportant since it has little influence on the collective measure.

8.2.2 Multi-Measure Ordering

As mentioned above, if we consider multiple quality measures, hypotheses can be incomparable. Nevertheless, some hypotheses are *comparable*. A hypothesis H can be compared to another hypothesis H' if H is better than H' on some measures and not worse on others. In this case, we say that a hypothesis H *dominates* a hypothesis H'. Thus, it is possible to compare hypotheses using the dominance relation, see Definition 8.4, instead of a collective quality measure.

Definition 8.4 (Multi-measure ordering)**.** Let $\mathbb{H}$ be a hypothesis space, Q a set of positive quality measures, $qf(H, q)$ the quality function, where $H \in \mathbb{H}$, $q \in Q$. A hypothesis H *dominates* a hypothesis H', written as $H' \prec H$, if

- there is $q \in Q$ such that $qf(H', q) < qf(H, q)$;
- for all $q' \in Q$, $q' \neq q$ implies $qf(H', q') \leq qf(H, q')$.

Two hypotheses H and H' are called *comparable* if $H' \prec H$ or $H \prec H'$. Otherwise, H and H' are called *incomparable*, denoted as $H' \parallel H$.

Clearly, if there is only one quality measure, i.e. $Q = \{q\}$, the multi-measure ordering $(\mathbb{H}, \prec)$ coincides with the single-measure ordering $(\mathbb{H}, \prec_q)$. Figure 8.1 illustrates comparable and incomparable hypotheses.

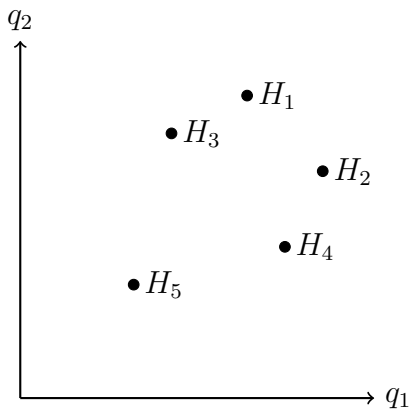


Figure 8.1: Comparable hypotheses: $H_5 \prec H_3$, $H_3 \prec H_1$, $H_5 \prec H_4$, $H_4 \prec H_2$; incomparable hypotheses: $H_1 \parallel H_2$, $H_3 \parallel H_4$, $H_1 \parallel H_4$, $H_2 \parallel H_3$.

In order to identify best hypotheses across all measures, we can use the standard notion of a *Pareto front* [Deb01]. The Pareto front is the set of all hypotheses which are not dominated by any other hypothesis. Such hypotheses are called *optimal*. If a hypothesis H is optimal, there is no other hypothesis H' which improves some quality of H without degrading others. Formally, the Pareto front is defined as follows:

$$pareto(\mathbb{H}, Q, qf) := \{H \in \mathbb{H} \mid \nexists H' \in \mathbb{H} \ : \ H \prec H'\}.$$

In Figure 8.1, the hypotheses H_1 and H_2 are optimal, i.e. they constitute the Pareto front. As discussed in Section 4.1.3, it is reasonable to consider not only optimal hypotheses but suboptimal ones as well, i.e. navigate through the ordering $(\mathbb{H}, \prec)$. In order to support the navigation, we can slice the hypothesis space $\mathbb{H}$ into layers, or *ranks*, of incomparable hypotheses such that hypotheses of a higher rank dominate hypotheses of a lower rank. The Pareto front contains hypotheses of the highest rank (and only those hypotheses). Algorithm 10 ranks hypotheses with respect to the dominance relation.

Algorithm 10 computes the Pareto front $\mathbb{H}_p$ of the current set $\mathbb{H}$ of hypotheses and assigns each hypothesis $H \in \mathbb{H}_p$ the current rank r. Then, it removes the Pareto front $\mathbb{H}_p$ from $\mathbb{H}$ and increases the rank r. The algorithm terminates once $\mathbb{H}$ is empty. Algorithm 10 always terminates, see Lemma 8.2.

Lemma 8.2. *Algorithm 10 is terminating.*

Proof. If $\mathbb{H}$ is empty, the algorithm never enters the main loop and terminates.

Algorithm 10 *rankHypotheses*$(\mathbb{H}, Q, qf)$

```
1: inputs
2:    ℍ: a finite set of hypotheses
3:    Q: a finite set of quality measures
4:    qf(H, q): a quality function, where H ∈ ℍ, q ∈ Q
5: outputs
6:    rf(H): the ranking function, where H ∈ ℍ
7: do
8:    r ← 0                    % initialise the current rank
9:    while ℍ ≠ ∅ do
10:      ℍ_p ← pareto(ℍ, Q, qf)      % compute a new Pareto front
11:      for each H ∈ ℍ_p do
12:         rf(H) ← r      % set the rank for each hypothesis in the front
13:      end for
14:      ℍ ← ℍ \ ℍ_p      % remove the current front
15:      r ← r + 1      % increase the current rank
16:   end while
17:   return rf(H)
```

If $\mathbb{H}$ is not empty, we show that there is always at least one optimal hypothesis since $\mathbb{H}$ is finite. Assume there are no optimal hypotheses in $\mathbb{H}$. Then, for every $H \in \mathbb{H}$ there is $H' \in \mathbb{H}$ such that $H \prec H'$. Pick any $H_1 \in \mathbb{H}$. Then, there is $H_2 \in \mathbb{H}\backslash\{H_1\}$, such that $H_1 \prec H_2$. In turn, there is $H_3 \in \mathbb{H}\backslash\{H_1, H_2\}$ such that $H_2 \prec H_3$. Let $k := |\mathbb{H}|$. Then, there is $H_k \in \mathbb{H}'$, where $\mathbb{H}' := \mathbb{H}\backslash\bigcup_{i=1}^{k-1}\{H_i\} = \{H_k\}$ such that $H_{k-1} \prec H_k$. Thus, H_k is optimal and the contradiction is obtained. Therefore, there is always an optimal hypothesis in a finite $\mathbb{H}$. Consequently, the Pareto front is never empty and at least one hypothesis is removed from $\mathbb{H}$ at each iteration of the main loop. Hence, the loop terminates after at most $k := |\mathbb{H}|$ iterations. □

Algorithm 10 would rank the hypotheses in Figure 8.1 as follows: $rf(H_1) = 0$, $rf(H_2) = 0$, $rf(H_3) = 1$, $rf(H_4) = 1$, $rf(H_5) = 2$. The ranking function $rf(\cdot)$ can be viewed as a quality measure derived from other quality measures, i.e. a collective quality measure.

It is important to note that ordering hypotheses by the dominance relation has some limitations. In particular, if multiple measures (many more than 2) are used to compare hypotheses, a hypothesis easily reaches the top rank if it is best on just one measure, regardless of all other measures. Therefore, we should carefully select measures to order hypotheses. More specifically, we can use the

set Q of measures to evaluate hypotheses $\mathbb{H}$ and then select the set $Q' \subseteq Q$ of *critical measures* to order $\mathbb{H}$.

Considering aggregation problems of the single-measure ordering, see Section 8.2.1, we prefer to use the multi-measure ordering for hypothesis ranking. As the critical measures Q' for ranking, we usually use support, assumption, and confidence. Algorithm 10 can be straightforwardly adjusted to take these considerations into account.

8.3 Putting All The Pieces Together: DL-MINER

We have discussed all tasks that DL-MINER is designed to accomplish. For the reader's convenience, we remind the architecture of DL-MINER, see Figure 8.2, and how it is covered in the thesis (please note that the order is different).

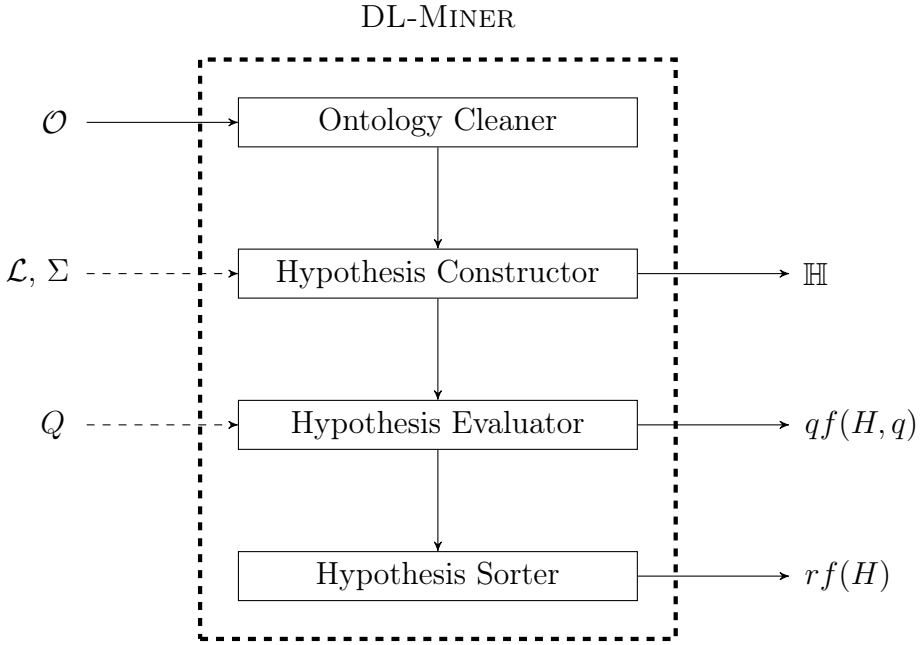


Figure 8.2: Architecture of DL-MINER

In Chapter 5, we have proposed a range of quality measures that can be used to rigorously evaluate hypotheses in DLs. In Chapter 6, we have described how these measures can be computed. These two chapters uncover the functionality of Hypothesis Evaluator, see Figure 8.2. In Chapter 7, we have discussed how hypotheses can be constructed and proposed a data-driven algorithm called

DL-Apriori. The latter chapter explicates the functionality of Hypothesis Constructor. In the current Chapter 8, we have discussed how to handle an inconsistent input ontology, see Section 8.1, and how to order and rank the evaluated hypotheses using (multiple) quality measures, see Section 8.2. Thus, Section 8.1 and Section 8.2 describe the functionality of Ontology Cleaner and Hypothesis Sorter, respectively.

The aforementioned chapters and sections present algorithms aimed at impementing the respective tasks. Now, in order to implement the architecture shown in Figure 8.2, we combine all respective algorithms in one algorithm called DL-Miner.

8.3.1 The DL-Miner Algorithm

The input parameters of DL-Miner, see Algorithm 11, are an ontology $\mathcal{O}$, a seed signature Σ, a language bias $\mathcal{L}$, and a set Q of quality measures. If a seed signature Σ is not provided, it is selected automatically as described in Section 7.2.1. A language bias $\mathcal{L}$ is defined as follows, see Definition 8.5.

Definition 8.5 (Language bias)**.** A *language bias* is a tuple $\mathcal{L} := (\mathcal{DL}, \ell_{max}, p_{min}, G_R, n)$, where

- $\mathcal{DL} \leq \mathcal{SROI}$ is a DL which is used for constructing concepts and for which a suitable refinement operator exists;
- ℓ_{max} is a finite maximal length of a concept such that $\ell_{max} \geq 1$;
- p_{min} is a minimal concept support such that $0 < p_{min} \leq |in(\mathcal{O})|$;
- G_R is a finite set of templates that specify the shape of constructed roles;
- n is a maximal number of axioms in a hypothesis such that $n \geq 1$.

A language bias $\mathcal{L}$ specifies constraints for hypotheses, i.e. determines which of them are constructed by the algorithm. It includes the parameters $\mathcal{DL}$, p_{min}, ℓ_{max} for concept construction, see Algorithm 8 and Lemma 7.5, and the parameter G_R for role construction, see Algorithm 7 which is straightforwardly adapted for roles. The parameter n of $\mathcal{L}$ specifies how many axioms a hypothesis is permitted to contain.

Given an ontology $\mathcal{O}$, Algorithm 11 checks its consistency and, if it is inconsistent, repairs it. The result is the consistent ontology $\mathcal{O}'$. Then, the algorithm

Algorithm 11 DL-MINER $(\mathcal{O}, \Sigma, \mathcal{L}, Q)$

inputs
  $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$: an ontology
  Σ: a finite set of terms such that $\top \in \Sigma$
  $\mathcal{L} := (\mathcal{DL}, \ell_{max}, p_{min}, G_R, n)$: a language bias, see Definition 8.5
  Q: a finite set of quality measures
outputs
  $\mathbb{H}$: the set of hypotheses
  $qf(H, q)$: the quality function, where $H \in \mathbb{H}$, $q \in Q$
  $rf(H)$: the ranking function, where $H \in \mathbb{H}$
do
  $\mathcal{O}' \leftarrow repairOntology(\mathcal{O})$ % see Algorithm 9
  $\mathbb{C} \leftarrow$ DL-APRIORI $(\mathcal{O}', \Sigma, \mathcal{DL}, \ell_{max}, p_{min})$ % see Algorithm 8
  $\mathbb{R} \leftarrow buildRolesTopDown(\Sigma, G_R)$ % see Algorithm 7
  $\mathbb{H} \leftarrow generateHypotheses(\mathbb{C}, \mathbb{R}, n)$ % see Algorithm 6
  $qf \leftarrow evaluateHypotheses(\mathbb{H}, Q, \mathcal{O}')$ % see Algorithm 5
  $rf \leftarrow rankHypotheses(\mathbb{H}, Q, qf)$ % see Algorithm 10
  return $(\mathbb{H}, qf, rf)$

constructs concepts $\mathbb{C}$ in the bottom-up way and roles $\mathbb{R}$ in the top-down way, according to the language bias $\mathcal{L}$. The concepts $\mathbb{C}$ and roles $\mathbb{R}$ are used to generate hypotheses $\mathbb{H}$. After that, the hypotheses $\mathbb{H}$ are evaluated by the quality measures Q. As a result, the quality function $qf(\cdot)$ is computed. Finally, $qf(\cdot)$ is used to rank hypotheses by their quality, i.e. to compute the ranking function $rf(\cdot)$. The algorithm returns the hypotheses $\mathbb{H}$, quality function $qf(\cdot)$, and ranking function $rf(\cdot)$. Figure 8.3 places these subroutines on the architecture of DL-MINER.

8.3.2 Correctness, Completeness, and Termination

We now discuss the properties of DL-MINER. Algorithm 11 is *correct* in the sense that it never returns an incorrect output, i.e. violating the constraints given by the input parameters. Correctness ensures that DL-MINER *only* returns hypotheses which conform to the language bias $\mathcal{L}$ and use the signature Σ. Correctness of DL-MINER is similar to correctness of Algorithm 8.

Algorithm 11 is *complete*, i.e. it returns and evaluates *all* hypotheses modulo equivalence which conform to the language bias $\mathcal{L}$ and use the signature Σ. Completeness ensures that potentially useful hypotheses are not missed. Completeness of Algorithm 11 is similar to completeness of Algorithm 8.

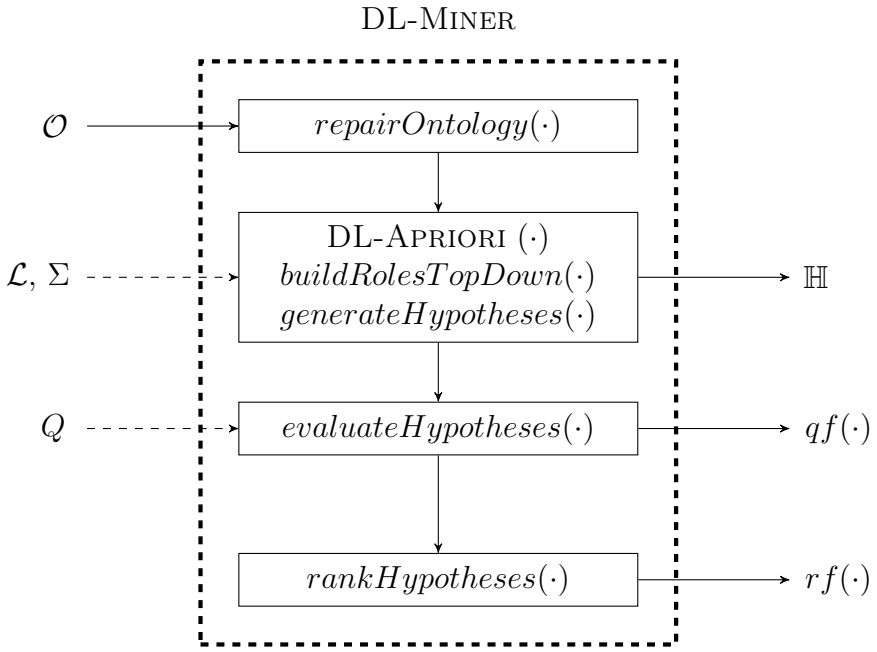


Figure 8.3: Architecture of DL-Miner with subroutines

Another property of Algorithm 11 is that it always *terminates*. This ensures that the algorithm returns an output (even though it may take long) for any *legal* input parameters, i.e. satisfying the respective constraints of Algorithm 11. The properties of correctness, completeness, and termination of Algorithm 11 follow from the same properties of its subroutines, see Theorem 8.1.

Theorem 8.1 (Correctness, completeness, termination)**.** *Let* $\mathcal{O}$, Σ, $\mathcal{L} := (\mathcal{DL}, \ell_{max}, p_{min}, G_R, n)$, Q *be legal parameters of* DL-Miner*. Let (i) – (iii) be the following conditions for a hypothesis* H*:*

(i) H *conforms to* $\mathcal{L}$*;*

(ii) H *is in NNF;*

(iii) $\widetilde{H} \subseteq \Sigma$*.*

Then, all following properties hold for DL-Miner*:*

- *it* terminates*;*
- *it is* correct*: it returns a set* $\mathbb{H}$ *of hypotheses such that* $H \in \mathbb{H}$ *implies* H *satisfies (i) – (iii);*

- *it is* complete*: if a hypothesis H' satisfies (i) – (iii), then there is $H \in \mathbb{H}$ such that $H \equiv H'$.*

In addition, it returns the quality function $qf(H, q)$ and ranking function $rf(H)$, where $H \in \mathbb{H}$, $q \in Q$.

Proof. (Termination) Algorithm 11 is terminating because all its subroutines are terminating. Algorithm 9 terminates because there are finitely many justifications for any ontology, see Definition 8.1, since a power set of a finite set is finite. Algorithm 8 terminates by Lemma 7.5. Algorithm 7 terminates because Σ and G_R are finite sets. Algorithm 6 terminates because $\mathbb{C}$ and $\mathbb{R}$ are finite sets, n is a finite number. Algorithm 5 terminates because $\mathbb{H}$ is a finite set and every quality measure is computed in a finite time. Algorithm 10 terminates by Lemma 8.2.

(Correctness, completeness) By Lemma 7.5, Algorithm 8 is correct and complete for $\mathcal{DL}$, Σ, ℓ_{max}, p_{min} and constructs a set $\mathbb{C}$ of concepts in NNF. Algorithm 7 constructs a set $\mathbb{R}$ of all roles such that $R \in \mathbb{R}$ if and only if R conforms to a template from G_R and $\widetilde{R} \subseteq \Sigma$. Given $\mathbb{C}$ and $\mathbb{R}$, Algorithm 6 generates a set $\mathbb{H}$ of all hypotheses that have at most n axioms. Therefore, Algorithm 10 is correct and complete.

□

8.3.3 What Hypotheses Can DL-Miner Mine?

The language bias $\mathcal{L} := (\mathcal{DL}, \ell_{max}, p_{min}, G_R, n)$ is a parameter of DL-Miner which, along with the seed signature Σ, specifies the set $\mathbb{H}$ of hypotheses under consideration. The language bias $\mathcal{L}$ is a "flexible" parameter because it allows for constructing concepts of arbitrary shapes (up to a certain length). In the following, we give some examples of $\mathcal{L}$ and show what types of hypotheses the algorithm subsequently constructs. In particular, we fix $\ell_{max} := \ell$ and $p_{min} := p$, but vary $\mathcal{DL}$, G_R, and n, see Table 8.1.

As Table 8.1 shows, the language bias $\mathcal{L}_1$ produces $\mathcal{EL}$ GCIs where each concept is not longer than ℓ. In addition to the GCIs of $\mathcal{L}_1$, $\mathcal{L}_2$ adds RIs where each role conforms to the templates $G_R := \{R,\ R \circ R\}$, i.e. RIs of the form $R \sqsubseteq S$ and $R \circ R \sqsubseteq S$, where $R, S \in N_R$. In addition to those, $\mathcal{L}_3$ produces $\mathcal{ALC}$ GCIs (not longer than ℓ) and RIs with inverse roles. Finally, the language bias $\mathcal{L}_4$ generates the same axioms as $\mathcal{L}_3$ does, but additionally combines them in pair sets, i.e. allows for two-axiom hypotheses.

	Language bias			Types of hypotheses
	$\mathcal{DL}$	G_R	n	
$\mathcal{L}_1$	$\mathcal{EL}$	$\emptyset$	1	$\{\alpha\}$, where $\alpha := C_1 \sqsubseteq C_2$, C_1, C_2 are concepts from $\mathcal{EL}(\ell)$
$\mathcal{L}_2$	$\mathcal{EL}$	$\{R,\ R \circ R\}$	1	$\{\alpha\}$, where $\alpha := C_1 \sqsubseteq C_2$ or $\alpha := S_1 \sqsubseteq S_2$, C_1, C_2 are concepts from $\mathcal{EL}(\ell)$, $S_1, S_2 \in G_R$, $R \in N_R$
$\mathcal{L}_3$	$\mathcal{ALC}$	$\{R,\ R^-,\ R \circ S\}$	1	$\{\alpha\}$, where $\alpha := C_1 \sqsubseteq C_2$ or $\alpha := S_1 \sqsubseteq S_2$, C_1, C_2 are concepts from $\mathcal{ALC}(\ell)$, $S_1, S_2 \in G_R$, $R, S \in N_R$
$\mathcal{L}_4$	$\mathcal{ALC}$	$\{R,\ R^-,\ R \circ S\}$	2	$\{\alpha_1\}$, $\{\alpha_1, \alpha_2\}$, where $\alpha_i := C_1 \sqsubseteq C_2$ or $\alpha_i := S_1 \sqsubseteq S_2$, $i \in \{1, 2\}$, C_1, C_2 are concepts from $\mathcal{ALC}(\ell)$, $S_1, S_2 \in G_R$, $R, S \in N_R$

Table 8.1: Language biases and resulting types of hypotheses

Considering Table 8.1, if we denote a set of hypotheses of $\mathcal{L}_i$, $i \in \{1, 2, 3, 4\}$, by $\mathbb{H}_i$, then the following holds: $\mathbb{H}_1 \subseteq \mathbb{H}_2 \subseteq \mathbb{H}_3 \subseteq \mathbb{H}_4$. We say that a language bias $\mathcal{L}_2$ is more *expressive* than $\mathcal{L}_1$ if $\mathbb{H}_1 \subseteq \mathbb{H}_2$. We can continue Table 8.1 and define new language biases that are more expressive than the listed ones, i.e. we can further extend G_R, increase n, and use a more expressive $\mathcal{DL}$, e.g. $\mathcal{SROI}$. Since suitable refinement operators are so far designed for $\mathcal{DL} \leq \mathcal{ALC}$, the current implementation of DL-Miner is able to construct hypotheses of the following shape:

- $\{\alpha_1, \ldots, \alpha_n\}$, where $\alpha_i := C_1 \sqsubseteq C_2$ or $\alpha_i := S_1 \sqsubseteq S_2$, $i \in \{1, \ldots, n\}$, C_1, C_2 are concepts from $\mathcal{ALC}(\ell)$, $S_1, S_2 \in G_R$.

Example 8.1 shows some of the hypotheses acquired by DL-Miner for the Kinship ontology. We have eyeballed the acquired hypotheses and picked those that have a good quality and look interesting.

Example 8.1. Consider the Kinship ontology in Example 2.3. We run DL-Miner with the following parameters:

$\Sigma := \{Man, Woman, Father, Mother, hasChild, marriedTo\}$,

$\mathcal{DL} := \mathcal{ALC}$, $\ell_{max} := 4$, $p_{min} := 1$, $G_R := \emptyset$, $n := 1$, $Q :=$ {support, assumption, confidence}.

Given these input parameters, DL-MINER mines 536 hypotheses whose confidence exceeds 0.9. The following are some examples of them:

$$Woman \sqcap \exists hasChild.\top \sqsubseteq Mother \quad (H_1)$$
$$Man \sqcap \exists hasChild.\top \sqsubseteq Father \quad (H_2)$$
$$\exists hasChild.\top \sqsubseteq \exists marriedTo.\top \quad (H_3)$$
$$\exists marriedTo.\top \sqsubseteq \exists hasChild.\top \quad (H_4)$$
$$\exists marriedTo.Woman \sqsubseteq Man \quad (H_5)$$
$$\exists marriedTo.Mother \sqsubseteq Father \quad (H_6)$$
$$Father \sqsubseteq \exists marriedTo.(\exists hasChild.\top) \quad (H_7)$$
$$Mother \sqsubseteq \exists marriedTo.(\exists hasChild.\top) \quad (H_8)$$
$$\exists hasChild.\top \sqsubseteq Mother \sqcup Father \quad (H_9)$$
$$\exists hasChild.\top \sqsubseteq Man \sqcup Woman \quad (H_{10})$$
$$\exists hasChild.\top \sqsubseteq Father \sqcup Woman \quad (H_{11})$$

The hypotheses H_1 and H_2 provide descriptions for $Mother$ and $Father$. The hypotheses H_3 and H_4 indicate interesting correlations in the data: being married implies having children and vice versa. The hypotheses H_1, H_2, H_3, H_4 are already discussed in Chapter 3, as they can be obtained by other approaches. In addition to these hypotheses, DL-MINER acquires the hypotheses H_5, H_6, H_7, H_8, H_9, H_{10}, H_{11} (and many others) due to the flexible language bias. The hypothesis H_5 provides a description for Man, while the hypotheses H_6 – H_9 encode additional knowledge about $Father$ and $Mother$.

The hypotheses H_{10} and H_{11} show some issues that can arise while mining hypotheses using DL-MINER. More specifically, the hypothesis H_{10} seems carrying no useful knowledge in comparison to H_9. The reason is that the input ontology Kinship (its TBox) does not capture that everyone is either a man or woman, i.e. $\mathsf{Kinship} \not\models \top \sqsubseteq Man \sqcup Woman$. If the ontology captured this information, the hypothesis H_9 would be uninformative and would not be mined (as DL-MINER returns only informative hypotheses by default). On the other hand, the hypothesis H_{11} seems to be superfluous given H_9 since $\mathsf{Kinship} \models Mother \sqsubseteq Woman$. Thus, acquired hypotheses can appear to be superfluous due to a poor input TBox and due to other hypotheses considered in the context of the given TBox.

8.3.4 DL-Miner in Ontology Learning Dimensions

In Chapter 3 we introduced the ontology learning (OL) dimensions and located OL approaches in these dimensions, see Table 3.3. We now place DL-Miner in the OL dimensions so that it can be compared with other approaches, see Table 8.2.

	Input Data	**Target Knowledge**	**Semantics**	**Supervision**
CDL	ABox, TBox, target A	$A \equiv C$, where $A \in N_C$, C is a concept from $\mathcal{ALC}(\ell)$	OWA, partial consideration of TBox	supervised: positive and negative examples; unsupervised: CWA
SSI	RDF	$\bigsqcap_{C' \in X} C' \sqsubseteq \bigsqcap_{D' \in Y} D'$, where $X, Y \subseteq \mathbb{C}$ and $\mathbb{C} := \{A, \exists R.A, \exists R^-.A \mid A \in N_C \wedge R \in N_R\}$; $R \sqsubseteq S$, $R \circ R \sqsubseteq R$, where $R, S \in N_R$	CWA, no TBox	unsupervised
KBC	ABox, TBox	$\bigsqcap_{C' \in X} C' \sqsubseteq \bigsqcap_{D' \in Y} D'$, where X, Y are sets of concepts from $\mathcal{FLE}(\delta)$	OWA, full consideration of TBox	supervised: interactive learning where a domain expert verifies axioms or provides counterexamples
BelNet	ABox, TBox	$C \sqsubseteq D$, $C \sqcap D \sqsubseteq \bot$, where $C, D \in \mathbb{C}$ and $\mathbb{C} := \{A, \exists R.\top \mid A \in N_C \wedge R \in N_R\}$	CWA, mostly disregards TBox	unsupervised
DL-Miner	ABox, TBox	$\{\alpha_1, \ldots, \alpha_n\}$, where $\alpha_i := C_1 \sqsubseteq C_2$ or $\alpha_i := S_1 \sqsubseteq S_2$, $i \in \{1, \ldots, n\}$, C_1, C_2 are concepts from $\mathcal{ALC}(\ell)$, $S_1, S_2 \in G_R$	OWA, full consideration of TBox	unsupervised

Table 8.2: DL-Miner and other OL approaches in OL dimensions

As Table 8.2 shows, in comparison to other approaches, DL-Miner advances the expressivity of hypotheses under consideration. In contrast to CDL, complex concepts are permitted to be on both the LHS and RHS of a GCI, see H_3, H_4, H_9 in Example 8.1. In contrast to SSI, KBC, and BelNet, DL-Miner can acquire more expressive GCIs with concepts from $\mathcal{ALC}(\ell)$ and more expressive RIs.

Besides constructing expressive hypotheses, DL-Miner evaluates them rigorously. Multiple measures are used to evaluate different aspects of hypothesis quality. Not only an axiom can be evaluated, but also a set of axioms jointly.

One of the main differences between DL-Miner and other approaches is that

it respects the standard semantics of DLs and fully considers the TBox. In particular, the conventional OWA is respected. As a result, it evaluates hypotheses more cautiously than approaches using the CWA, i.e. SSI and BelNet, whose quality measures can be misleading. An evident example of this is the evaluation of disjointness axioms, see Example 8.2.

Example 8.2. Consider the ontology $\mathcal{O} := \{A(a_1), \ldots, A(a_m), B(b_1), \ldots, B(b_n)\}$. Under the CWA the individuals $a_1, \ldots, a_m$ are assumed to be the instances of $\neg B$ and the individuals $b_1, \ldots, b_n$ are assumed to be the instances of $\neg A$. As a consequence, the disjointness axiom $\alpha := A \sqsubseteq \neg B$ (A is disjoint from B) is assumed to be of high quality. However, it is possible that information in $\mathcal{O}$ is just incomplete and many of $a_1, \ldots, a_m$ are unknown instances of B, as well as many of $b_1, \ldots, b_n$ are unknown instances of A. In this case, the quality of α is significantly overestimated and misleading. In contrast, under the OWA α is evaluated cautiously: $sup(\alpha, \mathcal{O}) = 0$, $asm(\alpha, \mathcal{O}) = m + n$. For comparison, under the CWA $sup(\alpha, \mathcal{O}^\neg) = m + n$, $asm(\alpha, \mathcal{O}^\neg) = 0$, where $\mathcal{O}^\neg := \mathcal{O} \cup \{\neg B(a_1), \ldots, \neg B(a_m), \neg A(b_1), \ldots, \neg A(b_n)\}$.

By its design DL-MINER is unsupervised, i.e. it constructs and evaluates all hypotheses without any human intervention. No training examples are required from a domain expert. A user only interacts with the output of the algorithm, i.e. at the final stage once all hypotheses are constructed and evaluated.

According to Theorem 8.1, DL-MINER guarantees completeness, i.e. constructs and evaluates all hypotheses satisfying the input restrictions. In this sense, the approach resembles KBC, see Section 3.2.3, that also guarantees a certain form of completeness, see Lemma 3.1. Please note that completeness is rather hard to achieve for a flexible language bias that permits arbitrary concepts. To be more specific, completeness is easy to achieve if concepts are specified by some templates G_C, see Example 8.3.

Example 8.3. Consider the templates $G_C := \{X, \exists S.X\}$. We can easily generate all GCIs of the form $A \sqsubseteq B$, $A \sqsubseteq \exists R.B$, $\exists R.A \sqsubseteq B$, $\exists R.A \sqsubseteq \exists R.B$, where $A, B \in N_C$, $R \in N_R$. In other words, we straightforwardly achieve completeness for these forms of GCIs specified by G_C.

However, as discussed in Section 7.2.2, it is hard to enumerate all shapes of GCIs, where concepts are from some $\mathcal{DL}$ and at most as long as ℓ_{max}. A brute-force procedure that generates *all* such concepts is doomed even for inexpressive

DLs, e.g. $\mathcal{EL}$. Clearly, it can result in many unsuitable concepts unnecessary increasing the hypothesis space.

8.4 Implementation of DL-Miner

DL-Miner (including all its subroutines) is implemented in Java (version 8.91) using the OWL API[3] (version 3.5.0). As said above, the implementation currently supports DLs up to $\mathcal{ALC}$ since suitable refinement operators are available for these DLs. To perform the required reasoning tasks, we use Pellet [SPG+07] (version 2.3.1) as it shows better performance on our experimental ontologies than other popular reasoners that support the OWL API. The implementation is publicly available.[4] In the following, we discuss optimisations used and possible user interaction scenarios for DL-Miner.

8.4.1 Optimisations and Heuristics

In general, DL-Miner constructs a vast set $\mathbb{H}$ of hypotheses. According to Lemma 7.2, the size of $\mathbb{H}$ grows rapidly with the number of concepts in $\mathbb{C}$ and roles in $\mathbb{R}$. In practice, these sets, particularly $\mathbb{C}$, can be large. Indeed, the algorithm constructs all concepts modulo equivalence satisfying the input constraints. Consequently, if the constraints are not extremely limiting, a set $\mathbb{C}$ of concepts is likely to be large. This results in a set $\mathbb{H}$ of hypotheses which is even larger. Since, besides constructing hypotheses, DL-Miner also evaluates them, this can be computationally costly for a vast set $\mathbb{H}$.

8.4.1.1 Incomplete Construction of Hypotheses

There are several ways to deal with a vast set $\mathbb{H}$ of hypotheses. One possibility is to construct a smaller set $\mathbb{H}' \subsetneq \mathbb{H}$ of hypotheses, i.e. sacrifice completeness. As the size of $\mathbb{H}$ is normally determined by the size of $\mathbb{C}$, we can construct only *most* promising concepts $\mathbb{C}' \subsetneq \mathbb{C}$ amongst all suitable concepts $\mathbb{C}$ (where promisingness is estimated by concept support). The standard heuristic which can be employed for this task is *beam search*. Beam search expands only k currently most promising nodes in a search tree instead of expanding all of them, where k is a predefined

[3] `http://owlapi.sourceforge.net`
[4] `https://github.com/slava-sazonau/dlminer`

number. DL-Apriori, see Algorithm 8, can be easily adjusted to perform beam search such that only k most promising specialisations are added to the queue of candidates for further refinement. One can also limit the maximal role depth of a concept in addition to the maximal length. This is reasonable if we consider the experimental results in Section 9.1.1.

Another possibility for reducing a set $\mathbb{H}$ of hypotheses is to handle redundancy while constructing concepts $\mathbb{C}$. To be more specific, some concepts can contain superfluous parts with respect to the TBox, see Example 8.4.

Example 8.4. Consider the ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$, where the TBox $\mathcal{T} := \{A \sqsubseteq B,\ B \sqsubseteq C\}$. Assume we are specialising the concept A. Then, the specialisations of A include $A \sqcap B$ and $A \sqcap C$. However, $\mathcal{T} \models A \equiv A \sqcap B$ and $\mathcal{T} \models A \equiv A \sqcap C$. In other words, the concepts $A \sqcap B$ and $A \sqcap C$ contain the superfluous parts B and C, respectively, with respect to $\mathcal{T}$.

Specialisations with superfluous parts can be detected via subsumption checking. Given a concept C, a specialisation $C' \in \rho(C)$ of C contains superfluous parts with respect to the TBox $\mathcal{T}$ if $\mathcal{T} \models C \equiv C'$. Such a specialisation is not specialised further and not included in the final set $\mathbb{C}$ of concepts. DL-Apriori can be straightforwardly altered to take this into account. Please notice that a *proper* refinement operator ρ handles superfluous parts automatically if the TBox is empty. Since the TBox is not empty in general, we need to check specialisations with respect to it in order to avoid concepts with superfluous parts.

Another way of reducing the number of hypotheses is using the logical and readability measures to filter out deficient hypotheses. In particular, if a hypothesis is inconsistent, uninformative, or redundant, it is sensible to discard it. In other words, we save computing other quality measures for such hypotheses. Please recall that syntactic variations are already avoided during hypothesis construction in Algorithm 6.

8.4.1.2 Incomplete Evaluation of Hypotheses

As an alternative to neglecting some hypotheses, we can construct all hypotheses $\mathbb{H}$ but evaluate them only partially. The simplest way is to pick cheap quality measures and only evaluate hypotheses by those measures ignoring others. In this case, however, we risk overlooking some important information about hypothesis quality.

A more careful approach is to evaluate all hypotheses by cheap quality measures, identify a certain number $k < |\mathbb{H}|$ of most promising ones based on their values, and fully evaluate those hypotheses. Hence, cheap measures can act as heuristics for promisingness of hypotheses. Instead of specifying a number k of hypotheses, we can specify some thresholds for heuristics and only completely evaluate hypotheses that satisfy those thresholds. The basic axiom measures, such as support and confidence, can be used as heuristics.

Finally, hypothesis evaluation can be run as an *anytime algorithm*[5] such that it aims at evaluating all hypotheses but can be interrupted at any time to return hypotheses evaluated so far. In this case, it is important to evaluate promising hypotheses as soon as possible. As above, promising hypotheses can be identified by the basic axiom measures acting as heuristics.

8.4.2 User Interaction

Once DL-Miner terminates, it returns hypotheses $\mathbb{H}$, the quality function $qf(\cdot)$, and ranking function $rf(\cdot)$. There are many possible ways for a domain expert to use the output of DL-Miner. In the simplest scenario, she uses the ranking function to explore hypotheses. She starts from the best rank, i.e. optimal hypotheses, and then reviews hypotheses of the next rank. The expert proceeds so until a certain number of hypotheses is reviewed.

The domain expert can use the quality values directly to navigate through the set of hypotheses. She picks some quality measure of interest, e.g. confidence, and orders all hypotheses by that measure. Then, she reviews the hypotheses in descending order of quality values and stops once a certain number of hypotheses is reviewed.

The expert can also combine both aforementioned ways of interaction, i.e. use the ranking and quality function simultaneously. More specifically, she can order hypotheses in each rank by a quality measure of choice. The latter is more complex but more reliable because the ranking ensures that dominating hypotheses are reviewed before dominated ones.

In order to reduce the total number of hypotheses to be examined, the user can skip those which do not meet minimal quality thresholds. For example, the confidence threshold $conf_{min} = 0.9$ selects all hypotheses that have a confidence

[5] An anytime algorithm is an algorithm that returns a valid result even if it is interrupted before it ends. The longer it runs, the better result it is expected to return.

value greater than 0.9 (which is relatively high). Such thresholds, however, should be chosen cautiously as they may prune useful hypotheses. We investigate usefulness of hypotheses acquired by DL-Miner and gain insight into user interaction in our case study with domain experts in Section 9.4.2.

8.5 Summary

This chapter concludes conceptual contributions of this thesis. We have proposed an approach for General Terminology Induction. Its architecture consists of four functional blocks: Ontology Cleaner, Hypothesis Constructor, Hypothesis Evaluator, and Hypothesis Sorter. We have described respective techniques and algorithms throughout the thesis and combined all of them in the algorithm called DL-Miner in this chapter. We have proved its correctness, completeness, and termination. We have compared its general properties with ones of related OL approaches. We have suggested optimisations and heuristics for DL-Miner and discussed some user interaction scenarios. The next chapter is devoted to in-depth empirical evaluation of DL-Miner and case studies.

Chapter 9

Evaluation of DL-Miner

We have discussed the design, implementation, and properties of DL-Miner, an approach for General Terminology Induction, see Definition 4.2. This chapter is devoted to the empirical evaluation of the approach. We start by describing the data used in our experiments. In the first experiment, we investigate correlations between hypothesis quality measures and computational performance of the algorithm. Then, we compare DL-Miner with related approaches to Ontology Learning (OL). Finally, we run case studies with human experts to gain insight into usefulness of hypotheses acquired by DL-Miner and its potential applications.

All experiments are implemented in Java (version 8.91) using the OWL API (version 3.5.0). The experiments are executed on the following machine: Linux Ubuntu 14.04.2 LTS (64 bit), Intel Core i5-3470 3.20 GHz, 8 GB RAM. We use Pellet (version 2.3.1) for reasoning.

9.1 Experimental Data

As part of our experimental data, we have chosen BioPortal[1] which is a large repository of biomedical ontologies constantly curated and maintained by domain experts. In other words, BioPortal is a corpus of great significance for the community. We first analyse BioPortal and describe axioms occurying in it with the aim to justify our conjecture, see Section 5.1, that people tend to write short and readable axioms while building ontologies.

[1] `http://bioportal.bioontology.org`

In order to evaluate DL-Miner, we construct two disjoint corpora of ontologies. The first one, called *handpicked corpus*, consists of ontologies hand-picked from related work. The second one, called *principled corpus*, is automatically constructed from BioPortal. We define suitable ontology metrics and then use them to describe ontologies in both corpora.

9.1.1 BioPortal and Axiom Diversity

BioPortal (snapshot 27.01.2015 [MP15]) contains 329 ontologies of various sizes and expressivity. There are 311 consistent ontologies, 7 inconsistent ontologies, and the rest have caused the reasoner, Pellet (version 2.3.1), to halt due to an internal error. In order to gain insight into readability and motivate some design choices made for the readability measures in Section 5.1, we design and conduct an experiment. We aim at answering the following research question:

RQ Do domain experts use short and simple axioms more frequently than long and complex axioms?

If the research question can be answered positively, shorter and simpler hypotheses should be preferred as suggested in Section 5.1.

9.1.1.1 Experimental Design

In order to answer the research question, we measure syntactic complexity of each TBox axiom, i.e. a GCI or RI, in the corpus. We only consider TBox axioms because a hypothesis can contain only TBox axioms. To measure the complexity of an axiom, we calculate the standard metrics commonly used in DLs and OWL, i.e. the number of complex concepts occurring in the axiom, the number of existential restrictions, universal restrictions, conjunctions, disjunctions, negations, etc. In addition, we measure the length, see Definition 5.1, and role depth, Definition 5.3, of an axiom. We also record the ontology that contains the axiom.

9.1.1.2 Results

In order to interpret the results correctly, we need to take the corpus bias into account. As we have found out, 81 out of 329 ontologies do not use complex concepts, i.e. they are used to model solely concept and role hierarchies. Since

they constitute ≈ 25% of the corpus, it is considerably biased towards easy ontologies. To make the analysis more rigorous, we exclude all ontologies which do not use complex concepts from the results. Thus, 248 ontologies are retained. We gather all axioms of those ontologies, which results in 9,133,219 axioms in total, and extract their metrics. Table 9.1 shows the proportion of axioms using a particular DL constructor.

DL constructor	C	$\exists R.C$	$C \sqcap D$	$\forall R.C$	$C \sqcup D$	$\neg C$
Axioms, %	99.73	67.82	1.15	0.46	0.09	0.01

Table 9.1: Use of DL constructors by axioms in BioPortal (ontologies without complex concepts are excluded)

According to Table 9.1, 99.73% of axioms are concept inclusions. Hence, all role inclusions constitute just 0.27% of axioms. GCIs constitute around 69% of axioms. Interestingly, almost all of them, 98.2%, use existential restrictions that occur in 67.82% of axioms overall. This is much more than all other constructors (in descending order): conjunctions occur in 1.15%, universal restrictions in 0.46%, disjunctions in 0.09%, negations in 0.01% of all axioms. In addition, disjointness axioms, which constitute 1.22% of all axioms, augment the fraction of conjunctions, if interpreted as $C \sqcap D \sqsubseteq \bot$, or the fraction of negations, if interpreted as $C \sqsubseteq \neg D$.[2]

While Table 9.1 shows how frequently common DL constructors are used in the axioms, it does not show how complex the axioms are. To investigate this, we measure the length and depth of the axioms using Definition 5.1 and Definition 5.3, respectively. In particular, we compare the proportions of short and long axioms, the proportions of shallow and deep axioms, see Table 9.2.

	mean	**mode**	**5%**	**25%**	**50%**	**75%**	**95%**	**99%**	**99.9%**
length	2.63	3	2	2	3	3	3	3	5
depth	0.69	1	0	0	1	1	1	1	3

Table 9.2: Length and role depth of axioms in BioPortal (ontologies without complex concepts are excluded)

In Table 9.2 mean and mode are the standard statistical notions which are calculated across all gathered axioms. For length, the mean is 2.63 and mode is

[2]In the OWL API disjointness axioms are handled not as concept inclusions, but as a separate type of axioms.

3, i.e. the most frequently appearing length of axioms is 3. For depth, the mean is 0.69 and mode is 1, i.e. the most frequently appearing depth of axioms is 1. Considering the results in Table 9.1, those are likely to be axioms with existential restrictions.

The numbers with the "%" sign are percentiles.[3] As Table 9.2 shows, 99% of axioms have the length at most 3 and the role depth at most 1, 99.9% of axioms have the length at most 5 and the role depth at most 3. Hence, both long and deep axioms are used extremely rarely. Thus, we can answer the research question *positively*: domain experts do tend to write short and simple axioms.

9.1.2 Ontology Metrics

In order to describe our experimental ontologies, we define the following metrics which are grouped with respect to the attribute they account for. Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology.

- **Ontology expressivity** The metric $\mathcal{DL}$ shows the expressivity[4] of $\mathcal{O}$, i.e. the DL used.

- **ABox size** The metrics $|in(\mathcal{A})|$, CA, RA count the numbers of individuals, concept and property assertions, respectively, in the ABox $\mathcal{A}$. These jointly describe the size of the ABox, i.e. the higher the values are, the bigger the size is.

- **Average ABox vertex degree** Let $\mathcal{G} := (V, E)$ be the *graph* of the ABox $\mathcal{A}$, where $V := in(\mathcal{A})$, $E := \{\langle a, b\rangle \mid R(a, b) \in \mathcal{A} \ \wedge \ a, b \in V\}$. The *degree* of a vertex a in $\mathcal{G}$ is defined as follows: $deg(a, \mathcal{G}) := |\{\langle a, b\rangle \mid \langle a, b\rangle \in E\}|$. As the first metric for ABox graph complexity, we use the *average vertex degree*, i.e. $deg(\mathcal{A}) := \frac{1}{|V|} \sum_{a \in V} deg(a, \mathcal{G})$.

- **Average ABox connected component size** Besides the average vertex degree of the graph, we measure how many individuals a connected component of the graph contains on average. We say that vertices a and b are *connected* in the ABox graph $\mathcal{G} := (V, E)$ if there is a path between a and b in $\mathcal{G}$. Let $con(a, b, \mathcal{G}) := true$ if a and b are connected in $\mathcal{G}$ and

[3] A percentile indicates the value below which a given percentage of observations fall.

[4] Expressivity is determined via the standard OWL API implementation which performs only syntactic analysis.

$con(a, b, \mathcal{G}) := false$ otherwise. A *connected component* of $a \in V$ is the number of all vertices connected to a in $\mathcal{G}$, i.e. $con(a, \mathcal{G}) := \{b \in V \mid con(a, b, \mathcal{G})\}$. If a and b are connected, then a and b are in the same connected connected component. Thus, we can count the total number of *unique* connected components in $\mathcal{G}$ as the number of unique sets, i.e. $k := |\bigcup_{a \in V} \{con(a, \mathcal{G})\}|$ (please notice that k is the size of the set of sets but not the size of the union set). Then, the *average size* of a connected component is as follows: $con(\mathcal{A}) := \frac{|V|}{k}$.

- **Ontology vocabulary size** The set of all concept and role names of $\mathcal{O}$ is called the *vocabulary* of $\mathcal{O}$, i.e. $voc(\mathcal{O}) := crn(\mathcal{O})$. Thus, the metrics $|voc(\mathcal{A})|$ and $|voc(\mathcal{T})|$ measure the size of the vocabulary of the ABox $\mathcal{A}$ and TBox $\mathcal{T}$, respectively. Please notice that individual names are excluded as they describe the size of the ABox, but not its vocabulary.

- **Jaccard index of ABox and TBox vocabularies** The metric $jvoc(\mathcal{A}, \mathcal{T}) := \frac{|voc(\mathcal{A}) \cap voc(\mathcal{T})|}{|voc(\mathcal{A}) \cup voc(\mathcal{T})|}$ is the *Jaccard index*[5] of the ABox and TBox vocabularies. It shows how many terms the ABox $\mathcal{A}$ and TBox $\mathcal{T}$ share, i.e. the higher the value is, the more terms are shared.

Why are these ontology metrics informative? They capture how many hypotheses can possibly be learned from an ontology and how hard an ontology is for learning. The ABox size, ABox graph complexity, and ABox vocabulary size describe how rich the ABox is. The richer the ABox is, the more hypotheses can possibly be acquired from it. However, finding good hypotheses becomes harder because the hypothesis space is larger. Moreover, there is an evidence that the size of an ontology is a good indicator of computational performance of reasoning [SSB14]. Hence, evaluating the quality of each hypothesis is expected to be computationally more expensive for larger ontologies. Expressivity of an ontology additionally informs how costly reasoning operations can be for that ontology.

The overlap of TBox and ABox vocabularies suggests how much knowledge they share. A low overlap shows that little of the ABox knowledge is represented by the TBox. In this case, acquired hypotheses can potentially "fill the gap". On the other hand, a high overlap may mean that the TBox sufficiently "covers" the knowledge encoded in the ABox. Consequently, little new knowledge can be acquired from the ABox to supplement the TBox. Nonetheless, hypotheses can

[5]The size of the intersection divided by the size of the union

still reveal some missing bits. Please note that all these metrics are syntactic, e.g. they neglect the fact that TBox terms can be connected to the ABox implicitly, see Example 7.1. We use them as a guidance for assessing and comparing ontologies.

9.1.3 Handpicked Corpus

The handpicked corpus is composed of ontologies frequently used in related work, in particular in [FDE08, LABT11]. Additional ontologies are taken from the DL-Learner repository,[6] Protégé OWL repository,[7] and TONES repository.[8]

We have selected the ontologies based on the following criteria: each ontology should contain at least minimal data to learn from (at least 15 individuals and 15 role assertions). We also ensure that a reasoner can handle every ontology. The resulting corpus is available online [Saz17] and shown in Table 9.3 where the metrics are as described in Section 9.1.2.

Ontology	$\mathcal{DL}$	$\lvert in(\mathcal{A})\rvert$	CA	RA	$deg(\mathcal{A})$	$con(\mathcal{A})$	$\lvert voc(\mathcal{A})\rvert$	$\lvert voc(\mathcal{T})\rvert$	$jvoc(\mathcal{A},\mathcal{T})$
alzh	$\mathcal{AL}$	150	106	854	5.7	150	40	0	0
arch	$\mathcal{ALC}$	19	26	26	1.4	3.8	10	13	0.77
carc	$\mathcal{ALC}(\mathcal{D})$	22,372	22,372	40,666	1.8	65.8	113	146	0.77
cin	$\mathcal{ALCOF}$	45	45	76	1.7	45	7	37	0.19
eart	$\mathcal{ALCHOF}(\mathcal{D})$	58	58	55	0.9	14.5	23	2,482	0.01
econ	$\mathcal{ALCH}$	482	649	555	1.2	5.3	29	380	0.04
fam	$\mathcal{AL}$	202	1,052	728	3.6	20.2	18	18	0.56
fin	$\mathcal{ALCOIF}$	17,941	17,941	47,248	2.6	8,970.5	52	76	0.68
heart	$\mathcal{AL}(\mathcal{D})$	280	275	1,080	3.9	280	9	11	0.82
krk	$\mathcal{SHI}$	420	525	1,508	3.6	4	25	40	0.55
lubm	$\mathcal{AL}(\mathcal{D})$	1,555	1,623	4,115	2.6	1,555	26	68	0.38
mam	$\mathcal{AL}(\mathcal{D})$	975	975	2,883	3	975	18	22	0.82
mdm	$\mathcal{ALCHOF}(\mathcal{D})$	112	130	169	1.5	2	82	215	0.38
mut	$\mathcal{AL}(\mathcal{D})$	14,145	14,145	26,533	1.9	61.5	60	91	0.66
ntn	$\mathcal{SHOIN}(\mathcal{D})$	724	724	1,636	2.3	2.8	64	78	0.82
sur	$\mathcal{AL}(\mathcal{D})$	2,979	2,979	6,008	2	175.2	20	49	0.41

Table 9.3: Handpicked corpus

Table 9.3 shows that the ontologies fin, carc, mut have the largest ABoxes. fin has the most complex ABox graph and carc has the largest ABox vocabulary. Please notice that the corpus contains expressive ontologies, e.g. ntn, mdm, fin,

[6] https://github.com/AKSW/DL-Learner

[7] http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library

[8] http://owl.cs.manchester.ac.uk/repository

Eart, along with inexpressive "toy" ontologies, e.g. alzh, fam, heart, lubm. All ontologies are consistent.

9.1.4 Principled Corpus

The principled corpus is automatically constructed from BioPortal. As in Section 9.1.1, we use snapshot 27.01.2015 that consists of 329 ontologies. We select all ontologies which contain at least 100 individuals and 100 role assertions. As for the handpicked corpus, this is to ensure that there is some data in each ontology for learning. The resulting corpus is available online [Saz17] and shown in Table 9.4 where the metrics are as described in Section 9.1.2.

Ontology	$\mathcal{DL}$	$\lvert in(\mathcal{A})\rvert$	CA	RA	$deg(\mathcal{A})$	$con(\mathcal{A})$	$\lvert voc(\mathcal{A})\rvert$	$\lvert voc(\mathcal{T})\rvert$	$jvoc(\mathcal{A},\mathcal{T})$
bof	$\mathcal{ALCF}(\mathcal{D})$	315	267	166	0.5	1.7	78	245	0.32
brid	$\mathcal{SROIN}(\mathcal{D})$	250	255	239	1	22.7	6	1,708	0
clo	$\mathcal{SHIN}(\mathcal{D})$	17,597	17,597	17,762	1	2	8	10	0.80
ctx	$\mathcal{ALCOIN}(\mathcal{D})$	282	282	1,123	4	16.6	129	312	0.41
derm	$\mathcal{ALUF}(\mathcal{D})$	6,101	6,101	6,099	1	3,050.5	2	6,109	0
gly	$\mathcal{SHOIQ}(\mathcal{D})$	1,784	3,011	29,405	16.5	77.6	109	282	0.37
heio	$\mathcal{ALCHIF}(\mathcal{D})$	1,605	4,895	2,982	1.9	20.3	119	134	0.89
iceci	$\mathcal{AL}(\mathcal{D})$	2,229	2,229	243	0.1	1.1	3	2,233	0
icf	$\mathcal{ALCHOIF}(\mathcal{D})$	4,807	3,353	3,244	0.7	3.1	20	1,637	0.01
icps	$\mathcal{SHOIQ}(\mathcal{D})$	763	1,540	701	0.9	1.6	65	567	0.11
mo	$\mathcal{ALEOF}(\mathcal{D})$	698	744	136	0.2	1.1	100	314	0.31
natp	$\mathcal{SHOIN}(\mathcal{D})$	22,012	22,012	20,751	0.9	2.2	538	9,478	0.06
ncit	$\mathcal{SH}(\mathcal{D})$	40,069	0	89,292	2.2	607.1	12	110,815	0
oad	$\mathcal{ALH}$	2,465	0	10,889	4.4	85	178	5,899	0
ogdi	$\mathcal{SHIN}(\mathcal{D})$	363	363	645	1.8	14	162	459	0.33
piero	$\mathcal{ALRI}+$	73,891	71,844	216,637	2.9	671.7	19	107	0.18
sitb	$\mathcal{ALCON}(\mathcal{D})$	163	284	230	1.4	4	103	217	0.47
sse	$\mathcal{SHIF}$	1,323	1,109	1,214	0.9	12.1	223	260	0.86
sso	$\mathcal{ALIF}(\mathcal{D})$	158	159	356	2.3	31.6	16	182	0
sweet	$\mathcal{SHOIN}(\mathcal{D})$	2,152	2,448	794	0.4	1.2	246	4,791	0.05
swo	$\mathcal{ALRI}+(\mathcal{D})$	116	349	13,506	116.4	116.0	15	3,827	0

Table 9.4: Principled corpus

Table 9.4 shows that the ontologies piero, ncit, natp, clo have the largest ABoxes. piero, ncit, gly have relatively complex ABox graphs and natp has the largest ABox vocabulary. For 7 ontologies TBox and ABox signatures do not overlap, e.g. brid, ncit, swo. Please notice that, in comparison to the handpicked

corpus, see Table 9.3, the principled corpus consists of considerably more expressive ontologies. In addition, while all ontologies in the handpicked corpus are consistent, the principled corpus contains two inconsistent ontologies: bof and ogdi.

9.2 Evaluating Quality Measures and Performance

In Chapter 5 we have defined the hypothesis quality measures quantifying readability, logical quality, and statistical quality. While some measures capture similar aspects of quality, others are supposed to be rather orthogonal. As examples of the former, the basic statistical measures are simplified versions of the main statistical measures. As examples of the latter, the logical and statistical measures are designed to evaluate different dimensions of quality; support, assumption, and contradiction are designed to capture independent, complementary aspects of statistical quality.

9.2.1 Research Questions

It is worthwhile to investigate whether the quality measures behave in practice in the same way as it is suggested by their definitions. This can be clarified by examining their mutual correlations. In particular, if there is an evidence that a measure q_1 strongly correlates with a measure q_2, it suggests (but does not imply) that only one of them should be used. Such a fact could be valuable information if computing q_1 is more expensive than q_2 or vice versa.

In addition to correlations between the measures, we investigate how expensive their computation is and which factors influence its performance. That evidence coupled with correlations should inform the choice of measures for evaluating hypotheses. We also compare the computational performance of the subroutines of DL-Miner. Thus, the experiment is aimed at investigating the following research questions.

RQ1 How do the quality measures correlate? Do related measures strongly correlate? Do unrelated measures not correlate?

RQ2 How costly are the quality measures? Which factors influence computational performance of hypothesis evaluation?

RQ3 Which subroutines of DL-Miner are more expensive than others?

9.2.2 Experimental Design

We design the experiment as follows. As experimental data, we use the hand-picked and principled corpus, see Table 9.3 and Table 9.4. We run the experiment on each corpus independently and compare the results. For each ontology $\mathcal{O}$, we run DL-Miner, see Algorithm 11, with the following parameters:

- the seed signature Σ is automatically extracted from $\mathcal{O}$ as suggested in Section 7.2.1;
- the language bias $\mathcal{L}$ is specified as follows:
  - $\mathcal{DL} = \mathcal{ALC}$ (determines what complex concepts are constructed),
  - $p_{min} = 10$ (determines which concepts are deemed to be insufficiently supported by the data and therefore skipped),
  - $\ell_{max} = 4$ (sets the maximal permitted length for concepts),
  - $G_R = \{R,\ R^-,\ R \circ S\}$ (determines what complex roles are constructed),
  - $n = 1$ (sets the maximal number of axioms in a hypothesis);
- the set Q of quality measures comprises all measures defined in Chapter 5.

As we run the experiment for multiple ontologies (including relatively complex ones), we fully evaluate at most 500 random hypotheses *per ontology* for feasibility considerations (the total number of evaluated hypotheses acquired for all ontologies is much greater). Ideally, we would vary this threshold and investigate how results change, i.e. run sensitivity analysis. However, the latter would be hardly feasible as we run the experiment for multiple ontologies. In addition, we have reasons to believe that results would not change significantly for higher thresholds since 500 is a sufficiently high number of hypotheses sampled randomly for each ontology.

For each hypothesis, we record the time it took to evaluate its quality measures. We also record the time of running the subroutines, e.g. concept construction, role construction, hypothesis evaluation, etc. We also capture the time of computing entailments required to compute the complex measures, see Section 6.1.

Once the algorithm terminates, we discard inconsistent, uninformative, and redundant hypotheses from the output as mentioned in Section 8.4.1. For all experimental ontologies, the algorithm has terminated and returned evaluated hypotheses. We first discuss correlations between the quality measures (RQ1) and then computational performance (RQ2, RQ3).

9.2.3 Mutual Correlations of Quality Measures

In order to answer RQ1, we compute mutual correlations of the quality measures across all hypotheses in a corpus. We present the results in the form of a *correlation matrix*, which is a symmetric matrix[9] of correlation coefficients. In addition, for each mutual correlation, we run a statistical significance test with significance level 0.05.[10]

Figure 9.1 shows correlation matrices for the handpicked and principled corpus. Positive correlations are shown in blue, negative correlations are shown in red, colour intensity shows correlation strength, i.e. a higher intensity indicates a higher correlation coefficient (by absolute value). Insignificant correlations are marked by crosses.

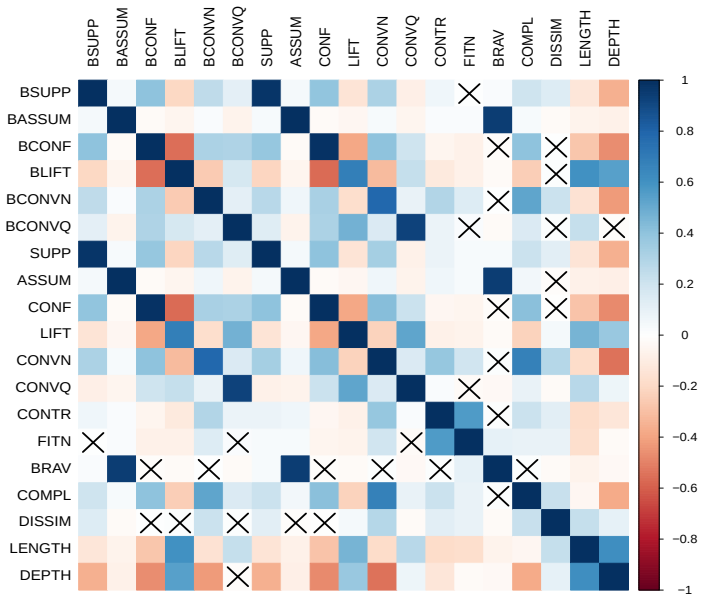
The quality measures are abbreviated in Figure 9.1 as follows: (B)SUPP – (basic) support, (B)ASSUM – (basic) assumption, (B)CONF – (basic) confidence, (B)LIFT – (basic) lift, (B)CONVN – (basic) negated conviction, (B)CONVQ – (basic) assumed conviction, CONTR – contradiction, FITN – fitness, BRAV – braveness, COMPL – complexity, DISSIM – dissimilarity. Please recall that basic contradiction equals main contradiction by Lemma 5.15. Therefore, only main contradiction is shown.

Let us now discuss the results in Figure 9.1. We highlight strong and weak correlations and discuss which of them are expected and unexpected.

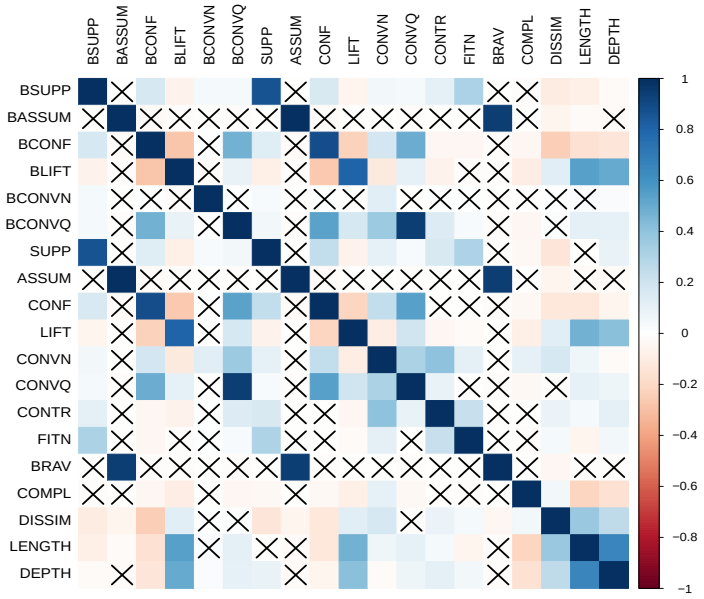
Perhaps, the most evident observation is that all main measures, except negated conviction for the principled corpus, strongly and positively correlate with their basic counterparts. We observe this for both corpora by noticing lines of dark blue squares parallel to the main diagonal. This result is expected because the basic measures are approximations of the respective main measures. Another strong and positive correlation occurs between assumption and braveness which

[9] A symmetric matrix is a square matrix which is equal to its transpose, i.e. its elements are symmetric with respect to the main diagonal.

[10] If the p-value is higher than the significance level, the *null* hypothesis stating that the correlation is insignificant is not rejected.



(a) Handpicked corpus



(b) Principled corpus

Figure 9.1: Mutual correlations of quality measures for handpicked (a) and principled (b) corpus: positive correlations are in blue, negative correlations are in red, crosses mark statistically insignificant correlations (significance level 0.05)

is also expected since these measures count guesses (though differently) that a hypothesis makes in the data.

The difference between the basic and main measures is worth additional investigation. Overall, the basic and main measures differ for $\approx 3.6\%$ of hypotheses in the handpicked corpus and for $\approx 4.1\%$ of hypotheses in the principled corpus. In order to uncover what causes the differences, we have identified ontologies where the basic and main measures differ. Those are 5 and 8 ontologies in the handpicked and principled corpus, respectively. It turns out that all of those ontologies contain explicit or implicit negative information in their ABoxes. More specifically, we have extracted the bottom module of each ontology given the signature of its ABox, see Section 7.2.1, and found out that in all cases this module contains negations or disjointness axioms, i.e negative information. Moreover, it turns out that a module contains negative information mainly for those ontologies where the basic and main measures differ (the exceptions are explained by the fact that hypotheses are sampled randomly and by the parameters of the algorithm). This observation suggests that negative information in the ABox causes differences between the basic and main measures. This also follows from their definitions, see Section 5.3.1.4.

Another noticeable result is that lift (main and basic) positively correlates with length and depth which is rather unexpected. In other words, longer hypotheses are likely to be of higher quality if it is measured by lift. Hence, it seems beneficial to construct and evaluate them. According to Definition 5.16, this shows that a greater length of an axiom increases the chance that its LHS and RHS have common instances divided by the chance that the LHS and RHS are (statistically) independent. Among other observations are the positive correlations between conviction and confidence (particularly for the principled corpus) that capture similar aspects of quality, i.e. the direction and strength of the association between the LHS and RHS of an axiom.

Besides positive correlations, Figure 9.1 shows some negative correlations, e.g. between confidence and lift. By Definition 5.16 confidence equals lift multiplied by the chance of the RHS to have an instance. On the other hand, axioms with the RHS covering more instances are likely to be more confident. Therefore, higher confidence is likely to cause lower lift and vice versa. Since lift positively correlates with length and depth, confidence negatively correlates with these measures.

Another observation worth highlighting is the *insignificant* correlation between

fitness and support for the handpicked corpus, see Figure 9.1a. This is expected since these measures are designed to capture different aspects of hypothesis "fit" to the data, see Example 5.23 and Example 5.24. Please notice that, in fact, fitness and support do correlate for the principled corpus, see Figure 9.1b.

Overall, the principled corpus shows more correlations which are statistically insignificant than the handpicked one. This happens probably because the principled corpus consists of more expressive ontologies than the handpicked one. In particular, as discussed above, its ontologies contain more negative information. As a consequence, the measures tend to diverge more often and occasional correlations are less likely to appear. This also explains why the correlations between the basic and main measures are stronger for the handpicked corpus than for the principled corpus.

Thus, we can answer RQ1 as follows: related measures do correlate significantly, while unrelated measures do not (except the correlations between lift and length and between lift and depth). In particular, strong correlations are evident between the basic and main statistical measures and between assumption and braveness.

In addition, we explore the axiom set measures for multi-axiom hypotheses. For each ontology, we generate and evaluate 500 hypotheses of two axioms ($n = 2$) and 500 hypotheses of three axioms ($n = 3$), where axioms are sampled randomly from single-axiom hypotheses acquired from the ontology. Thus, a multi-axiom hypothesis is obtained via putting together several single-axiom hypotheses. We investigate how frequently the fitness $fit(H, \cdot)$ of a multi-axiom hypothesis H differs from the sum of fitnesses of its axioms, i.e. its aggregated fitness $fit_{sum}(H, \cdot) := \sum_{\alpha \in H} fit(\{\alpha\}, \cdot)$, where "$\cdot$" stands for $\mathcal{O}, \mathbb{C}, \mathbb{R}$. We count the number of ontologies where the values of fit and fit_{sum} differ for at least one generated hypothesis and the total fraction of generated hypotheses with unequal values. Table 9.5 shows the results.

Corpus	**Case**	**Ontologies**	**Hypotheses (%)**
Handpicked (16 ontologies)	$fit > fit_{sum}$	16	48.6
	$fit < fit_{sum}$	14	1.4
Principled (21 ontologies)	$fit > fit_{sum}$	19	43.8
	$fit < fit_{sum}$	20	2.6

Table 9.5: Comparing fitness of a multi-axiom hypothesis with aggregated fitness of its axioms

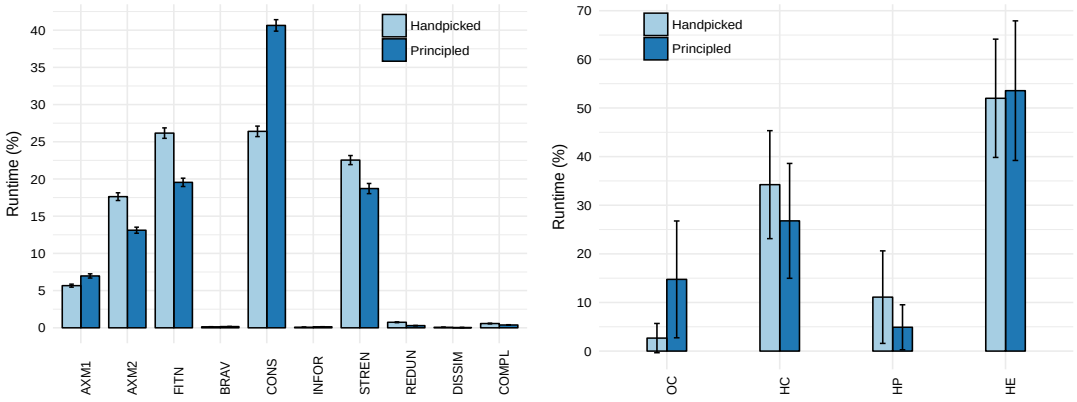
According to Table 9.5, the measures differ for nearly half of all hypotheses and for almost all ontologies. For most of those hypotheses, the fitness is higher than the aggregated fitness. The case $fit > fit_{sum}$ can happen because axioms can "interact", i.e. a set of axioms entails all entailments of its axioms and can additionally have entailments that are not entailed by any of its axioms independently. Hence, a set of axioms can "earn" additional value of fitness. The case $fit < fit_{sum}$ can happen because axioms within the set can repeat entailments counted by fitness, e.g. consider the hypothesis $H := \{A \sqsubseteq C,\ B \sqsubseteq C\}$ and the ontology $\mathcal{O} := \{A(a), B(a), C(a)\}$. As Table 9.5 shows, the first case is much more likely to happen than the second one. Thus, aggregated fitness usually underestimates fitness.

9.2.4 Computational Performance Results

In order to find out which quality measures are more computationally expensive than others (the first part of RQ2), we investigate their relative performance. More specifically, for each hypothesis, we record what fraction of its evaluation time is spent to compute each measure. Then, we aggregate respective fractions for all hypotheses in a corpus and calculate their means and confidence intervals with confidence level 95%. In the same way, we investigate relative performance of subroutines of the algorithm across ontologies (RQ3). Clearly, measuring relative computation time has its advantages and disadvantages. On the one hand, it allows for aggregating results across all ontologies and comparing computation costs irrespective of an input ontology. On the other hand, computation time may vary significantly across ontologies and we are also interested in comparing absolute (not relative) computation time. This will be investigated in the following.

Figure 9.2 shows the results of measuring relative computation times. The abbreviations in Figure 9.2a are as follows: AXM1 – the axiom measures which do not consider negation, i.e. basic support, assumption, confidence, lift, assumed conviction; AXM2 – the axiom measures which consider negation, i.e. main support, assumption, confidence, lift, assumed conviction, contradiction, (basic and main) negated conviction; FITN, BRAV – fitness and braveness; CONS, INFOR, STREN – consistency, informativeness, logical strength; REDUN, DISSIM, COMPL – redundancy, dissimilarity, complexity. As Figure 9.2a shows, the most expensive quality measures are (in descending order) consistency, fitness, logical

strength, and the statistical axiom measures.



(a) Quality measures (b) Subroutines

Figure 9.2: Relative performance of hypothesis quality measures (a) and subroutines (b) of DL-Miner for principled and handpicked corpus

According to Figure 9.2a, consistency is the most expensive measure which is rather unexpected. Please recall that consistency tests whether the union of a hypothesis and the ontology is consistent. Hence, it can be costly if the ontology is large and/or expressive. Indeed, consistency is considerably more costly for the principled corpus than for the handpicked one which is likely to be a consequence of higher expressivity of the former. The higher cost of consistency for the principled corpus decreases the relative contributions of other measures for this corpus, i.e. fitness, logical strength, etc.

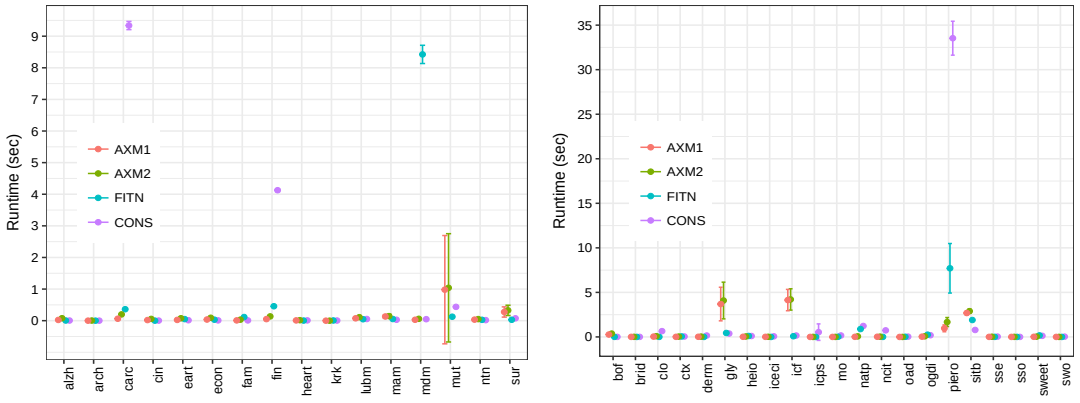
The relatively high computational cost of logical strength is explained by the fact that its performance is measured by comparing a given hypothesis to all others (in the worst case). Hence, it grows with the number of hypotheses to be evaluated. Therefore, it should be compared with other measures cautiously.

As Figure 9.2a shows, considering negation in the statistical axiom measures is relatively expensive. Therefore, given the strong correlation between the basic and main measures, see Figure 9.1, it is sensible to replace the main measures with their basic counterparts in certain cases, particularly if the ontology does not contain negative information in the ABox. On the other hand, if other expensive measures need to be computed, the relative cost of computing all axiom measures is not so big.

The abbreviations in Figure 9.2b stand for the respective running times as follows: OC – ontology parsing and classification; HC – hypotheses construction

including concept and role construction; HP – entailment checks for subsequent computation of the complex quality measures, i.e. fitness, braveness, dissimilarity, and complexity, see Section 6.1; HE – hypothesis evaluation including computing *all* quality measures. As the results show, the most expensive procedure of the algorithm is hypothesis evaluation (given that all quality measures are computed). Its cost can be significantly reduced if some expensive measures, shown in Figure 9.2a, are not computed. The cost of checking entailments for the complex measures is relatively low in comparison to the hypothesis evaluation cost.

In order to gain insight into factors affecting the performance of hypothesis evaluation (the second part of RQ2), we now consider each ontology independently. It is known that ontology properties influence performance of reasoning [GPS12]. In particular, one of the most influential factors is the *size* of an ontology [SSB14]. Hence, ontology properties should also affect the performance of hypothesis evaluation since it uses reasoning. To test this conjecture, we calculate means and confidence intervals of the runtime of all costly measures except logical strength (it is independent of the ontology) per ontology. Figure 9.3 shows the results (please notice that the runtime scales are different).



(a) Handpicked corpus (b) Principled corpus

Figure 9.3: Average performance of hypothesis quality measures per ontology in handpicked corpus (a) and principled corpus (b)

As Figure 9.3 shows (the abbreviations are the same as above), for most ontologies, it is relatively quick to evaluate a hypothesis: the average runtime is just slightly above zero. Yet, there are clear outliers. The outliers for the handpicked corpus, see Figure 9.3a, are carc, fin, mdm, mut. According to Table 9.3, carc, fin, and mut have the largest ABoxes which explains the higher costs of computing

the statistical axiom measures (AXM1 and AXM2) for them. The ontologies carc and fin are relatively expressive that, along with their large ABox size, seems to increase the cost of consistency checking. The high computation cost of fitness for mdm may be caused by its high expressivity and relatively large ABox signature.

The outliers for the principled corpus, see Figure 9.3b, are gly, icf, piero. According to Table 9.4, these ontologies have relatively large ABoxes, particularly piero. Although gly and icf have much smaller ABoxes than piero, they are more expressive and have larger signatures. This may cause higher costs of computing the statistical axiom measures for these ontologies. The low costs for ncit and oad are explained by the absence of concept assertions (CAs) in their ABoxes. The low costs for clo and natp may be caused by the simplicity of their ABoxes. Interestingly, the costs of the axiom measures are higher for sitb than for piero, despite the fact that sitb has a much smaller ABox (but larger vocabulary and higher expressivity) than piero. Thus, the ABox size, structure, signature, and expressivity are likely to affect computational performance of reasoning and, consequently, performance of hypothesis evaluation. Yet, considering the example of sitb, these are not the only factors.

It is worth noticing that runtime differences in computing the statistical axiom measures which do not consider negation (AXM1) and those which do consider negation (AXM2) are relatively small for the outliers in both corpora, see Figure 9.3. Thus, for these ontologies, computing the latter is not (significantly) more expensive than computing the former, despite the results in Figure 9.2a. It is likely that factors other than the presence or absence of negative information, e.g. a large and complex ABox, determine the performance of instance retrievals and, hence, computing the axiom measures for the outliers.

9.2.5 Side-observations of Interest

Besides the numeric evaluation of the quality measures and computational performance of the algorithm, we examine the learned hypotheses by eyeballing them. In other words, we act as domain experts and seek hypotheses which seem interesting. As a guidance, we use the quality values and ranking, i.e. we only search among those hypotheses which are ranked highly. Table 9.6 shows some acquired hypotheses which, according to our opinion, look interesting.

Since all hypotheses in Table 9.6 are informative with respect to their ontologies, the knowledge they capture is not entailed by the respective ontologies.

Ontology	Examples of hypotheses
carc	$\exists hasBond.\top \sqsubseteq \exists hasAtom.\top$
cin	$Movie \sqsubseteq \exists cast.Actor$
eart	$Cyclone \sqsubseteq \exists hasAssociatedPhenomena.AtmosphericCirculation$
fam	$married \circ hasChild \sqsubseteq hasChild$
fin	$OKRunningLoan \sqsubseteq \exists hasLoanStatusValue.(\neg ProblemStatus)$
heart	$Patient \sqcap \exists hasThalValue.ReversableDefect \sqsubseteq \exists hasChestPain.\top$
lubm	$AssociateProfessor \sqsubseteq \exists teaches.TeachingCourse$
mam	$Patient \sqcap \exists hasShape.Irregular \sqsubseteq \exists hasDensity.Illdefined$
ntn	$\forall siblingof.Human \sqsubseteq Human$
gly	$BetaSugar \sqcap \exists hasRingForm.\top \sqsubseteq Pyranose$
oad	$clinicallySimilar \circ hasSeverity \sqsubseteq hasSeverity$
sweet	$PlanetaryLayer \sqsubseteq \exists hasAstronomicalBody.\top$

Table 9.6: Examples of learned hypotheses

Some hypotheses look useful for TBox enrichment. Please notice the role chains in Table 9.6, e.g. $married \circ hasChild \sqsubseteq hasChild$ stating that "if x is married to y and y has a child z, then x has a child z". Clearly, this hypothesis is not true in general but it indicates that many ABox individuals conform to it, given its high quality. Likewise, the hypothesis $\forall siblingof.Human \sqsubseteq Human$ expressing that "everyone who can have only human siblings is a human" is incorrect since it implies that every object without a *siblingof* relation is also a human. Nonetheless, given its high quality, the hypothesis indicates that the ABox individuals are mostly humans. Thus, these examples hint that, besides TBox enrichment, acquired hypotheses can potentially be useful for data analysis and exploration. In the following, we investigate potential use cases by running case studies with domain experts.

9.2.6 Methodological Reflection

Although we believe that the results of this experiment are generalisable, some caution should be exercised. A potential threat to the validity of the results is the choice to evaluate only 500 hypotheses instead of evaluating all hypotheses. This is done to achieve feasibility of the experiment since we run the algorithm for many ontologies. In practice, one is likely to be interested in acquiring hypotheses for one or few ontologies and this restriction can be omitted. Another concern is the input parameters of the algorithm which are set to be the same for all ontologies, but ideally should be ontology-sensitive.

9.3 Comparing DL-Miner with Related Approaches

The following experiments are aimed at comparing DL-Miner with other OL approaches. Please recall that some comparison of designs and capabilities is already done in Section 8.3.4 where we have located the approaches in the OL dimensions, see Table 8.2. Nonetheless, it is worthwhile to investigate whether and how their results differ in practice.

Please note that, since the approaches operate in different settings, direct comparison of their results is not possible. In particular, some approaches are supervised, while others are not. Some respect the standard semantics, while others make the CWA and disregard the TBox. Such differences in design imply that certain hypotheses can be learned by one approach and cannot be learned by another, even if the latter has more expressive language bias than the former.

9.3.1 Comparing DL-Miner with Concept Description Learning

As discussed in Section 3.2.1, Concept Description Learning (CDL) is mainly a supervised approach based on ILP. It normally requires positive and negative examples to be specified in order to learn a concept description that conforms to those examples, i.e. entails (almost) all positive and (almost) no negative examples.

DL-Learner[11] [BLW16] is a popular implementation of CDL. Besides the standard supervised mode, it can operate in an unsupervised mode such that, given an ontology $\mathcal{O}$ and a target concept name A, it attempts to select positive and negative examples for A in $\mathcal{O}$ by applying a form of the CWA, see Section 3.2.1. We use DL-Learner in the unsupervised mode and compare it with DL-Miner. We investigate the following research questions.

RQ1 Can DL-Miner learn hypotheses, i.e. concept definitions, learned by DL-Learner? How many of them are learned?

RQ2 Do concept definitions of DL-Miner and DL-Learner differ? How do they differ?

[11] `http://dl-learner.org`

RQ3 How does the computational performance of DL-Miner and DL-Learner compare?

Please notice that we do not investigate the reverse version of RQ1, i.e. whether DL-Learner can learn hypotheses learned by DL-Miner. The reason is that it is clear that DL-Miner can learn GCIs with a complex LHS and RHS, while DL-Learner cannot because of its design. Besides finding out whether DL-Miner learns any hypotheses of DL-Learner, we quantify how many of them are learned (either explicitly or implicitly).

9.3.1.1 Experimental Design

As above, we use the handpicked and principled corpus as our experimental data, see Table 9.3 and Table 9.4. For each ontology $\mathcal{O}$ and each concept name $A \in cn(\mathcal{O})$, DL-Learner (version 1.1) is run in the unsupervised mode with default parameters to find the best concept description for A. The parameters of DL-Miner which are different from the parameters in Section 9.2 are as follows: $p_{min} = 1$, $\ell_{max} = 6$, the set Q of quality measures includes only the basic measures which do not consider negation, i.e. AXM1, see Section 9.2. Thus, we adjust the input parameters to construct and evaluate more hypotheses.

In order to answer RQ1, one can simply find common hypotheses learned by both algorithms. Since a hypothesis can have many syntactic variations, we make use of the entailment relation $\models$ to compare hypotheses. Instead of finding only exact matches, we find concept definitions of DL-Learner which are entailed by hypotheses of DL-Miner, i.e. learned explicitly or implicitly by DL-Miner. In order to count partly entailed definitions, we write each concept definition $A \equiv C_A$ of DL-Learner as two axioms $A \sqsubseteq C_A$, $C_A \sqsubseteq A$, where $A \in cn(\mathcal{O})$ is a concept name, C_A is a concept description of A. Formally, we are interested in finding the *hit set* of concept definitions, see Definition 9.1.

Definition 9.1 (Hits, misses)**.** Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology; $\mathbb{H}_m$ and $\mathbb{H}_l := \{A \sqsubseteq C_A,\ C_A \sqsubseteq A \mid A \in cn(\mathcal{O})\}$ sets of hypotheses of DL-Miner and DL-Learner,[12] respectively, learned for $\mathcal{O}$; $\mathcal{T}_H := \bigcup \mathbb{H}_m \cup \mathcal{T}$. Then, the *set* $\mathbb{H}_l^{hit} \subseteq \mathbb{H}_l$ *of hits* (or simply *hits*) of $\mathbb{H}_m$ is defined as follows:

$$\mathbb{H}_l^{hit} := \{\alpha \in \mathbb{H}_l \mid \mathcal{T}_H \models \alpha\}.$$

[12] m in $\mathbb{H}_m$ stands for "miner", l in $\mathbb{H}_l$ stands for "learner"

The set $\mathbb{H}_l^{mis} := \mathbb{H}_l \backslash \mathbb{H}_l^{hit}$ is called the *set of misses* (or simply *misses*) of $\mathbb{H}_m$.

Thus, hits are those concept definitions of DL-Learner which are entailed by all hypotheses of DL-Miner together with the given TBox. Let us explain why hits are defined in this way. Firstly, we join hypotheses of DL-Miner with the TBox because DL-Miner only learns informative hypotheses while DL-Learner does not. Secondly, we consider all hypotheses of DL-Miner as a union, despite the fact that they are supposed to be independent of each other. This is done in order to find out whether DL-Miner is capable of learning hypotheses that capture the same knowledge as concept definitions of DL-Learner do. The latter can be true even if the set of all hypotheses of DL-Miner and the set of all concept definitions of DL-Learner do not intersect. Therefore, simple counting of coinciding hypotheses would not be informative. In the following, we compare all hypotheses of DL-Miner against all concept definitions of DL-Learner simultaneously (RQ2).

According to Definition 9.1, $\mathcal{T}_H$ can be inconsistent and, hence, can unfairly "hit" all definitions in $\mathbb{H}_l$. Therefore, in order to check which definitions are actually learned, for each $\alpha \in \mathbb{H}_l$ we test whether there is a consistent subset of $\mathcal{T}_H$ that entails α. More specifically, we first check whether $\alpha \in \mathcal{T}_H$, i.e. learned explicitly. If $\alpha \notin \mathcal{T}_H$, we check whether the module[13] for $\widetilde{\alpha}$, i.e. $\mathcal{M}_\alpha := \bot\text{-}module(\mathcal{T}_H, \widetilde{\alpha})$, is consistent and entails α, i.e. $\mathcal{M}_\alpha \models \alpha$. If this is not true, we assume that $\mathcal{T}_H \not\models \alpha$ (even though α may actually have been learned). As one module can be suitable for checking multiple definitions, we group definitions by their modules in order to optimise such tests. Please note that the described "trick" is made solely for the purpose of comparing hypotheses of DL-Miner with concept definitions of DL-Learner. Normally, we consider all hypotheses of DL-Miner independently of each other and the problem of their joint inconsistency with respect to the TBox does not arise.

In order to figure out differences between hypotheses of DL-Miner and concept definitions of DL-Learner (RQ2), we identify which hypotheses of DL-Miner are responsible for "hitting" definitions of DL-Learner. Indeed, it is possible that only a (small) subset of $\mathbb{H}_m$ is sufficient for entailing $\mathbb{H}_l^{hit}$. Therefore, we find such a minimal subset $\mathbb{H}_m^{hit}$, called a *hitting set*, see Definition 9.2.

Definition 9.2 (Hitting set)**.** Let $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ be an ontology, $\mathbb{H}_l^{hit} \subseteq \mathbb{H}_l$ the

[13]Modules are discussed in Section 7.2.1.

set of hits of $\mathbb{H}_m$. Then, a set $\mathbb{H}_m^{hit} \subseteq \mathbb{H}_m$ is called a *hitting set* for $\mathbb{H}_l^{hit}$ if $\bigcup \mathbb{H}_m^{hit} \cup \mathcal{T} \models \mathbb{H}_l^{hit}$ and for every $\mathbb{H}'_m \subsetneq \mathbb{H}_m^{hit}$ it holds that $\bigcup \mathbb{H}'_m \cup \mathcal{T} \not\models \mathbb{H}_l^{hit}$.

Informally, a hitting set is simply a minimal subset of $\mathbb{H}_m$ which, if joined with the TBox, entails all hits $\mathbb{H}_l^{hit}$. A hitting set can also be defined using justifications, see Definition 8.1. It is similar to a hitting set used in diagnosis [Rei87] and justification-finding algorithms [HPS08]. However, a hitting set of hypotheses of DL-MINER is only used to compare it with DL-LEARNER.

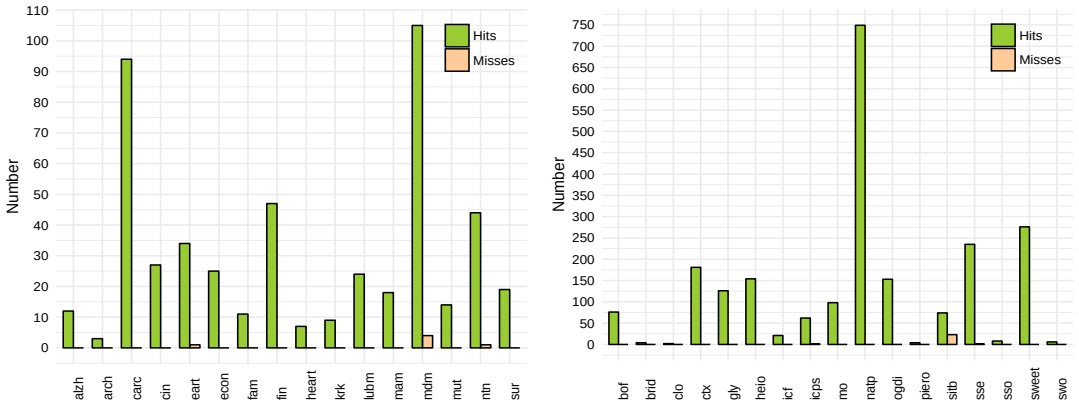
We search for a hitting set using a procedure similar to the redundancy elimination implemented by Algorithm 1, i.e. we remove axioms one by one in descending order of their length and ensure that a resulting subset still entails all the definitions. Since the found hitting set captures all knowledge of the hits (and possibly more), we can compare hypotheses in both sets on the common grounds.

9.3.1.2 Comparison Results

We have run DL-MINER and DL-LEARNER on our corpora as described above. DL-MINER has terminated and produced hypotheses for each ontology. A set of hypotheses along with the TBox has turned out to be inconsistent (and we have used the trick described above to identify learned definitions) for 3 ontologies in the handpicked corpus, i.e. arch, mdm, ntn, and for 5 ontologies in the principled corpus, i.e. ctx, glyco, icps, sitb, sse. DL-LEARNER has terminated for each ontology in both corpora but failed to produce any hypotheses for 4 ontologies which all belong to the principled corpus: derm, iceci, ncit, oad. These ontologies are excluded from the results.

It turns out that many concept definitions of DL-LEARNER are, in fact, entailed by the TBox alone, i.e. they are *uninformative*. To be more specific, $\approx 45\%$ for the handpicked corpus and $\approx 43\%$ for the principled corpus of concept definitions are uninformative on average per ontology. Since those concept definitions are actual hits of the TBox, but not DL-MINER, we exclude them from the results because the comparison would unfairly favour DL-MINER otherwise. We address RQ1 by counting concept definitions of DL-LEARNER which are entailed (hits) and not entailed (misses) by hypotheses of DL-MINER along with the TBox (but not by the TBox alone), see Figure 9.4.

As Figure 9.4a shows, in the handpicked corpus, DL-MINER learns 100 % of concept definitions for 13 out of 16 ontologies and more than 97 % of concept definitions for the remaining three ontologies, i.e. eart, mdm, ntn. For eart,



(a) Handpicked corpus

(b) Principled corpus

Figure 9.4: Number of concept definitions of DL-LEARNER entailed (hits) and not entailed (misses) by hypotheses of DL-MINER

1 out of 35 concept definitions is missed because it uses concrete domains, i.e. datatypes[14] in the OWL specification, currently unsupported by DL-MINER. For ntn, 1 out of 45 definitions is missed because the set of hypotheses is inconsistent and the search for a consistent subset entailing that definition has failed. For mdm, 2 out of 109 concept definitions are missed. The first definition is missed because a consistent entailing subset has not been found. The second definition is missed because it contains the concept $C := DiseaseTypes \sqcap \neg CancerTypes \sqcap \neg EndometrialAbnormality$ whose length exceeds the maximal length restriction, i.e. $\ell(C) = 7 > 6 = \ell_{max}$.

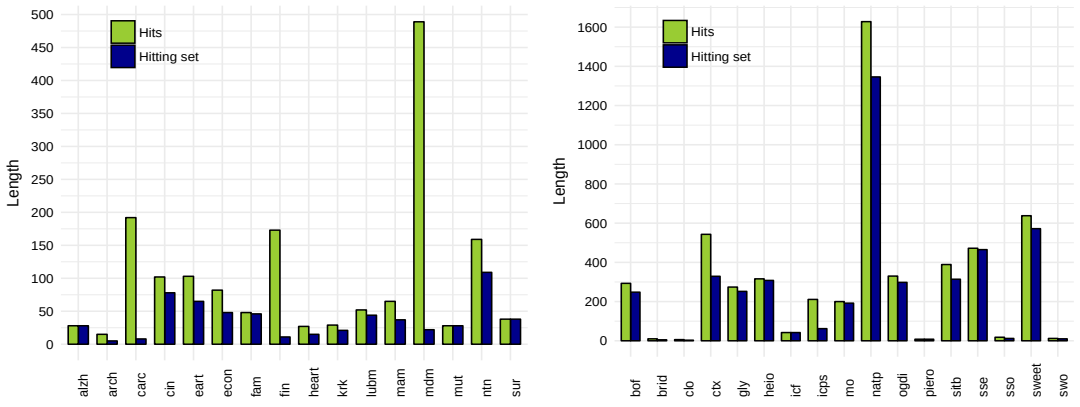
Figure 9.4b shows that, in the principled corpus, DL-MINER learns 100 % of concept definitions for 14 out of 17 ontologies (4 ontologies are excluded because DL-LEARNER has failed to learn any definitions for them). For icps and sse, 1 out of 63 and 1 out of 236 definitions, respectively, are missed because a consistent entailing subset has not been found. For sitb, 23 out of 97 definitions are missed because they contain concepts whose length exceeds the maximal length restriction, e.g. $C := HospitalDepartment \sqcap \neg GeneralDepartment \sqcap \neg OutPatientClinic$.

Thus, the main reason for missing a concept definition by DL-MINER is the maximal length restriction. In comparison, DL-LEARNER also imposes the maximal length restriction but increases it if necessary. Considering the results in Figure 9.4, we argue that increasing the maximal length restriction of DL-MINER would allow it to learn those missing definitions. Another reason for missing a

[14] `https://www.w3.org/TR/owl2-syntax/#Datatypes`

definition is the presence of datatypes in that definition. This can be fixed by extending the current implementation of DL-Miner to support datatypes. Finally, for some definitions, we were unable to check whether they are actually learned by DL-Miner or not because the set of its hypotheses was inconsistent. We considered those definitions as missing, even though they may, in fact, have been learned.

In order to gain insight into differences between concept definitions learned by DL-Miner and DL-Learner (RQ2), we examine the hitting set of DL-Miner, i.e. a minimal subset $\mathbb{H}_m^{hit}$ of its hypotheses $\mathbb{H}_m$ entailing all hits $\mathbb{H}_l^{hit}$, see Definition 9.2. In particular, we intend to find out whether $\mathbb{H}_m^{hit}$ is different from $\mathbb{H}_l^{hit}$. We compare the cumulative length of all hypotheses in both sets, i.e. the length $\ell(\mathbb{H}_m^{hit}) := \sum_{H \in \mathbb{H}_m^{hit}} \ell(H)$ of a hitting set and the length $\ell(\mathbb{H}_l^{hit}) := \sum_{H \in \mathbb{H}_l^{hit}} \ell(H)$ of hits, see Figure 9.5. Please note that, as above, hits of the TBox are excluded from the results.



(a) Handpicked corpus

(b) Principled corpus

Figure 9.5: Cumulative length of concept definitions of DL-Learner (hits) and hypotheses of DL-Miner (hitting set) that entail them

Figure 9.5 shows that the cumulative length of hits differs from the cumulative length of a hitting set. Hence, these sets are not equal. The length of a hitting set never exceeds the length of hits. In fact, it is smaller for most ontologies: for 13 out of 16 ontologies in the handpicked corpus, see Figure 9.5a, and for 15 out of 17 ontologies in the principled corpus, see Figure 9.5b. Interestingly, for some ontologies the hitting set is significantly shorter than its hits, e.g. carc, fin, mdm, ctx, icps.

Thus, the results show that DL-Miner is able to learn shorter hypotheses

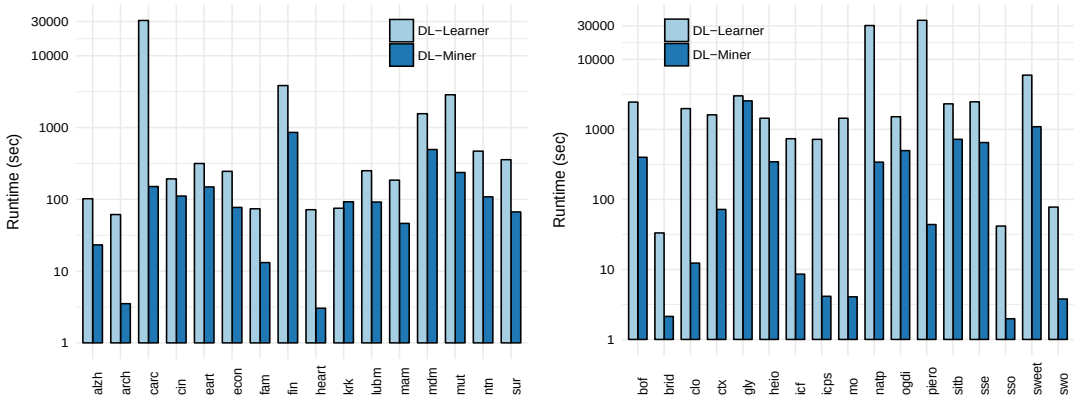
than DL-Learner learns. This happens because DL-Learner is *not* aimed at learning concept definitions minimised with respect to the TBox and minimised with respect to each other. In other words, concept definitions are learned independently and, therefore, tend to repeat knowledge encoded by the TBox and other concept definitions, see Example 9.1. As the results show, they even sometimes replicate the TBox knowledge, i.e. they are uninformative.

Example 9.1. Consider the TBox $\mathcal{T} := \{A \sqsubseteq B\}$ and the concept definition $\alpha := A \equiv B \sqcap C$ viewed as two axioms $\alpha_1 := A \sqsubseteq B \sqcap C$ and $\alpha_2 := B \sqcap C \sqsubseteq A$. The length of the definition is $\ell(\alpha) = \ell(\alpha_1) + \ell(\alpha_2) = 4 + 4 = 8$. However, given $\mathcal{T}$, α is redundant because α_1 can be replaced by shorter $\alpha_1' := A \sqsubseteq C$, i.e. $\mathcal{T} \cup \{\alpha_1', \alpha_2\} \models \alpha$ and $\ell(\alpha_1') + \ell(\alpha_2) = 2 + 4 = 6 < 8 = \ell(\alpha)$. Clearly, the TBox is not the only reason of excessive length since some definitions can repeat knowledge encoded by others, e.g. consider the set of definitions $\{A \equiv B,\ B \equiv C,\ C \equiv A\}$, where any of the definitions is redundant given other two.

In contrast to concept definitions of DL-Learner, hypotheses of DL-Miner are not restricted to be of the form $A \equiv C$. Therefore, the hypothesis $H_1 := \{A \sqsubseteq C\}$ in Example 9.1 can easily be learned by the algorithm. In addition, as we search for a minimal subset, hypotheses which are entailed by others are discarded. This results in a shorter cumulative length of the hitting set in comparison to its hits.

Finally, we compare the total runtime of DL-Miner and DL-Learner for each ontology (RQ3), see Figure 9.6.

According to Figure 9.6, DL-Miner is slower than DL-Learner only for one ontology, i.e. krk, in both corpora. In fact, for several ontologies the former is significantly faster than the latter: consider arch, carc, heart in Figure 9.6a and clo, icf, icps, mo, natp, piero in Figure 9.6b. In order to interpret these results correctly, one should keep in mind two things. Firstly, DL-Miner only computes the cheap basic measures in this experiment. Considering the results in Section 9.2, the total runtime would probably be much greater if we computed some costly quality measures as well. Secondly, to the best of our knowledge, DL-Learner is not currently optimised to learn definitions of multiple concept names in the ontology, i.e. it runs independently for each concept name. To be more specific, the algorithm is likely to process the same concepts, including retrieving their instances, across multiple runs, even though it avoids processing



(a) Handpicked corpus (b) Principled corpus

Figure 9.6: Runtime (logarithmic scale) of DL-Learner and DL-Miner for handpicked (a) and principled (b) corpus

equivalent concepts within a run. In contrast, DL-Miner processes all concepts just once per ontology. Therefore, the difference in performance grows with the number of concept names, as the likelihood of encountering equivalent concepts grows, and with the number of individuals in the ontology, as computational performance of instance retrieval degrades.

9.3.1.3 Side Observations of Interest

As mentioned above, DL-Learner has failed to learn any concept definitions for 4 ontologies in the principled corpus. It turns out that DL-Miner learns not only GCIs for these ontologies but also some concept definitions (which are missed by DL-Learner). For example, for derm[15] the concept definition $RadlexMetaclass \equiv RadlexMetaclass \sqcap \exists IsA.RadlexMetaclass$ (a good example of identifying modelling flaws) is learned by DL-Miner but not learned by DL-Learner.

We have also looked at the quality of concept definitions of DL-Learner using the quality measures introduced in this thesis. It turns out that the assumption of those concept definitions is relatively high on average per ontology. Hence, given a concept definition $A \equiv C$, the concept description C is usually far from being the "perfect description" of A that would have the assumption equal to zero. This is not surprising because the data can be insufficient to find the

[15]The Dermatology Lexicon is a standardized terminology of dermatologic diagnoses, therapies, subroutines, and laboratory tests, see `http://bioportal.bioontology.org`.

"perfect description". For example, consider the task of learning the description of $Father$ from the data containing no information about children. In this case, the best description might be Man and the result definition $Father \equiv Man$ would probably have a high assumption.

9.3.2 Comparing DL-Miner with Unsupervised Approaches

We compare DL-Miner with other unsupervised approaches from Table 8.2, i.e. BelNet (Bayesian Description Logic Network), see Section 3.2.4, and SSI (Statistical Schema Induction), see Section 3.2.2. We do not compare it with KBC (Knowledge Base Completion) because the latter is supervised and requires domain expert guidance, see Section 3.2.3. We investigate the following research question.

RQ Can DL-Miner learn hypotheses learned by BelNet and SSI? How many of them are learned?

9.3.2.1 Experimental Design

In [ZGP+15], the authors evaluate BelNet and compare its results with DL-Learner and GoldMiner.[16] The latter is an implementation of SSI aimed at generating ontologies from RDF data. GoldMiner is not compliant with the OWL API. For each input dataset, it needs to be run using the console commands, requires manually loading the data into a SPARQL[17] endpoint and configuring an auxiliary relational database. Due to these reasons, it is hard to use the tool for multiple OWL ontologies in the same way as we have used DL-Learner. The implementation of BelNet is not available. Therefore, we rely on the results in [ZGP+15] for comparisons.

In [ZGP+15], the authors use 4 ontologies in their experiments which include the ontologies fam, lubm, ntn of the handpicked corpus, see Table 9.3. Instead of comparing the results of the algorithms directly, they manually extend the experimental ontologies with the goal to obtain "gold standard" ontologies. Gold standard ontologies are aimed to be complete in a certain sense, see [ZGP+15] for details.

[16] https://code.google.com/archive/p/gold-miner
[17] SPARQL is a query language for RDF.

The algorithms are run on the experimental ontologies and their results are compared by consulting the respective gold standard ontologies. More specifically, the gold standard ontologies are used to count "correct" and "wrong" hypotheses via the measures called *precision* and *recall*, see Definition 9.3

Definition 9.3 ([ZGP+15])**.** Let $\mathcal{T}'$ and $\mathcal{T}^s$ be a learned TBox and gold standard TBox, respectively. *Precision* and *recall* of $\mathcal{T}'$ given $\mathcal{T}^s$ are calculated as follows:

$$precision(\mathcal{T}', \mathcal{T}^s) := \frac{|\{\alpha \in \mathcal{T}' \mid \mathcal{T}^s \models \alpha\}|}{|\mathcal{T}'|}$$
$$recall(\mathcal{T}', \mathcal{T}^s) := \frac{|\{\alpha \in \mathcal{T}^s \mid \mathcal{T}' \models \alpha\}|}{|\mathcal{T}^s|}$$

Precision counts how many axioms in the learned TBox $\mathcal{T}'$ are entailed by the gold standard TBox $\mathcal{T}^s$, relative to the size of $\mathcal{T}'$. In other words, a high precision ensures that *only* knowledge encoded by $\mathcal{T}^s$ is learned and penalises any additional knowledge acquired, regardless of whether the data supports that knowledge or not. Thus, the more extra knowledge the learned TBox $\mathcal{T}'$ contains, the lower the value of precision is. Hence, precision penalises approaches capable of learning expressive hypotheses. We argue that such approaches should be rewarded, not penalised. Therefore, we exclude this measure (and F-measure calculated using it) from our comparison.

Recall counts how many axioms in the gold standard TBox $\mathcal{T}^s$ are entailed by the learned TBox $\mathcal{T}'$, relative to the size of $\mathcal{T}^s$. In other words, recall evaluates "coverage" of the approach, i.e. the higher the value is, the more axioms in the gold standard TBox are learned (explicitly or implicitly). In fact, recall is the relative number of hits, see Definition 9.1, of the learned TBox $\mathcal{T}'$ in the gold standard TBox $\mathcal{T}^s$, ignoring the fact that $\mathcal{T}'$ can be inconsistent. We deal with inconsistency as in Section 9.3.1 and use this measure for further comparisons.

We use the experimental ontologies from [ZGP+15], i.e. fam, lubm, ntn (for one ontology we have not found its gold standard extension). These ontologies lack negative information in their ABoxes. Since DL-Miner respects the OWA, it does not acquire disjointness axioms for these ontologies, see Example 8.2, in contrast to BelNet and SSI. Therefore, we artificially impose the CWA for these ontologies in order to compare the results with [ZGP+15]. To be more specific, given an ontology $\mathcal{O}$, we extend it as follows: $\mathcal{O}^{\neg} := \mathcal{O} \cup \{\neg A(a) \mid \mathcal{O} \not\models A(a)\}$, where $A \in cn(\mathcal{O})$, $a \in in(\mathcal{O})$. Although such an extension can easily be

inconsistent, this does not happen for the given ontologies.

9.3.2.2 Comparison Results

DL-Miner is run for each ontology with the same parameters as in Section 9.3.1. Once the algorithm terminates for an input ontology $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$, we use the acquired hypotheses $\mathbb{H}_m$ to construct the learned TBox as above, i.e. $\mathcal{T}' := \bigcup \mathbb{H}_m \cup \mathcal{T}$. Then, given the gold standard TBox $\mathcal{T}^s$ for $\mathcal{T}$, we calculate the recall using Definition 9.3 and compare it with the values reported in [ZGP+15]. As $\mathcal{T}'$ is inconsistent for ntn (and only for it), we test whether an axiom of the gold standard ontology is learned like we do for a concept definition in Section 9.3.1. Table 9.7 shows the results.

	BelNet	GoldMiner	DL-Miner
fam	0.83	0.93	1.0
lubm	0.53	0.39	0.88
ntn	0.78	0.86	0.98

Table 9.7: Comparing DL-Miner with BelNet and GoldMiner by recall using gold standard ontologies

According to Table 9.7, DL-Miner shows higher recalls than BelNet and GoldMiner do, i.e. it "hits" more axioms in the gold standard ontologies. All misses in lubm are disjointness axioms where at least one of the concepts has no instances, i.e. no evidence in the data. Those concepts are not considered by the algorithm because the parameter $p_{min} = 1$ sets that a concept must have at least one instance to be considered. All misses in ntn are axioms with nominals, e.g. $\{Jesus\} \sqsubseteq SonOfGod$. Nominals are not supported by the current implementation of DL-Miner.

Thus, as Table 9.7 shows, DL-Miner has outperformed both BelNet and GoldMiner on all three ontologies. This means that DL-Miner can learn hypotheses that BelNet and GoldMiner cannot, which is expected considering their design.

9.3.3 Methodological Reflection

The results show that DL-Miner is able to learn most hypotheses of DL-Learner, BelNet, and GoldMiner (and hypotheses that these tools cannot learn because of their design). Nonetheless, one should consider these results with caution

because of considerable differences between the approaches. For the sake of comparison, DL-Learner has been used in the unsupervised mode, while it is primarily a supervised approach. It is possible that, given carefully prepared training examples, DL-Learner can learn concept definitions which an unsupervised approach cannot, regardless of expressivity of the language bias of the latter. Another important difference is that DL-Miner respects the OWA, while BelNet and GoldMiner rely on the CWA. As a result, some hypotheses are not acquired by DL-Miner, unless the CWA is imposed (and the CWA is not always a right assumption to make if information is incomplete).

9.4 Case Studies

Until now, we have evaluated DL-Miner on our experimental ontologies, i.e. investigated correlations between the quality measures, evaluated its computational performance, compared it with other approaches. In this section, we explore how the approach works in some practical scenarios, whether its results are useful, and gain insight into potential applications.

9.4.1 Using DL-Miner for Rice Fertility Prediction

So far, we have considered DL-Miner as a tool for acquiring knowledge to be reviewed by a human expert. In this case study, we investigate another potential use of DL-Miner – making predictions. We investigate the following research question:

RQ Can DL-Miner be used to obtain useful predictions?

9.4.1.1 Predictions in Description Logics

Let us formalise the notion of predictions in DLs. In principle, hypotheses learned from the data in DLs can act as *prediction rules*. Specifically, if an ontology $\mathcal{O}$ is enriched with hypotheses $\mathbb{H}$ acquired from it, then an individual in $\mathcal{O}$ can become a new instance of some concepts, see Definition 9.4.

Definition 9.4 (Prediction)**.** Let $\mathcal{O}$ be an ontology, $\mathbb{H}$ a set of hypotheses acquired from it, $\mathcal{O}_H := \bigcup \mathbb{H} \cup \mathcal{O}$. An individual $a \in in(\mathcal{O})$ is called a *predicted instance* of a concept C if $\mathcal{O} \not\models C(a)$ and $\mathcal{O}_H \models C(a)$. The class assertion $C(a)$ is called a *prediction*.

Informally, predictions are class assertions that are not entailed by the ontology alone but entailed by the union of the ontology and hypotheses acquired from it. Presumably, in order to make sensible predictions, we should use hypotheses of sufficiently high quality. Please notice that, in Definition 9.4, the union of acquired hypotheses is used for predictions, despite the fact that hypotheses are supposed to be independent of each other. The reason is that hypotheses are considered as being *complementary* but not alternatives. In particular, it is possible that two hypotheses individually do not make a prediction but their union does. Thus, by combining all acquired hypotheses together, we obtain the strongest possible "predictor" (that however causes other problems discussed in the following).

In this case study, we concentrate on the binary classification[18] problem, where for a target concept name $\mathcal{A} \in cn(\mathcal{O})$ we aim to make predictions of the form $\mathcal{O}_H \models A(a)$ and $\mathcal{O}_H \models \neg A(a)$, i.e. classify individuals into two mutually exclusive categories: A and $\neg A$. Nonetheless, turning a set of hypotheses into a classifier (prediction model) is not straightforward due to the following problems.

1. Due to the OWA, an individual $a \in in(\mathcal{O})$ can be left *unpredicted*, i.e. it is possible that $\mathcal{O}_H \not\models A(a)$ and $\mathcal{O}_H \not\models \neg A(a)$. Informally, a set of hypotheses can be insufficient to make a prediction.

2. A individual $a \in in(\mathcal{O})$ can be predicted to be an instance of both mutually exclusive concepts, i.e. it is possible that $\mathcal{O}_H \models A(a)$ and $\mathcal{O}_H \models \neg A(a)$. We call such an individual a *clash*. The presence of a clash implies that $\mathcal{O}_H$ is inconsistent. Informally, a set of hypotheses can be contradictory.

Both problems are rather challenging and we leave their detailed investigation for future work. In this study, we deal with the first problem by simply assigning an unpredicted individual a random label, i.e. either A or $\neg A$. The second problem is handled as follows.

- In order to avoid dealing with inconsistency of $\mathcal{O}_H$, we substitute a concept $\neg A$ for a fresh concept name $A^\neg$ in $\mathcal{O}$ (without loosing any data relevant for the task at hand).

- Given a set $\mathbb{H}$ of hypotheses, we first discard all hypotheses which would be (individually) inconsistent with $\mathcal{O}$, i.e. $\mathcal{O} \cup H \models A(a)$ and $\mathcal{O} \cup H \models A^\neg(a)$

[18] Classification is used in the Machine Learning sense, see also Section 2.2.

implies $H \notin \mathbb{H}$. Then, we identify all clashes by simply retrieving instances of $A \sqcap A^\neg$ in $\mathcal{O}_H$, i.e. $cl(A, \mathcal{O}_H) := inst(A \sqcap A^\neg,\ \mathcal{O}_H)$.

- For each clash $a \in cl(A, \mathcal{O}_H)$, we find all hypotheses that make a prediction for a, i.e. $\mathbb{H}_a := \{H \in \mathbb{H} \mid \mathcal{O} \cup H \models A(a) \ \vee \ \mathcal{O} \cup H \models A^\neg(a)\}$. Please notice that this is a simplification because, in general, multiple hypotheses can be necessary to make a prediction.

- Given $\mathbb{H}_a$, we find hypotheses with the highest confidence in $\mathbb{H}_a$ and, if there are many of those, pick one with the highest support, i.e. $H_a \in \mathbb{H}_a$ such that $conf(H_a, \mathcal{O}) = max\{conf(H, \mathcal{O}) \mid H \in \mathbb{H}_a\}$ and $sup(H_a, \mathcal{O}) = max\{sup(H, \mathcal{O}) \mid H \in \mathbb{H}_a \ \wedge \ conf(H, \mathcal{O}) = conf(H_a, \mathcal{O})\}$.

- Finally, we resolve the clash using H_a, i.e. assume that $\mathcal{O}_H \models A(a)$ if $\mathcal{O} \cup H_a \models A(a)$ and $\mathcal{O}_H \models A^\neg(a)$ if $\mathcal{O} \cup H_a \models A^\neg(a)$. This heuristic is chosen for the sake of simplicity but can, in principle, be based on multiple hypotheses and other quality measures.

9.4.1.2 Experimental Design

To address our research question, we investigate the problem of predicting fertility of rice varieties based on their genetic information. This study is conducted in collaboration with colleagues from the School of Computer Science and Manchester Institute of Biotechnology.[19] We thank Oghenejokpeme Orhobor for the data provided and helpful discussions. The data[20] has been translated to OWL in order to produce the input ontology, called rice. The metrics of rice are shown in Table 9.8.

$\mathcal{DL}$	$\lvert in(\mathcal{A})\rvert$	CA	RA	$deg(\mathcal{A})$	$con(\mathcal{A})$	$\lvert voc(\mathcal{A})\rvert$	$\lvert voc(\mathcal{T})\rvert$	$jvoc(\mathcal{A}, \mathcal{T})$
$\mathcal{ALC}$	252	500	3968	15.75	252	270	0	0

Table 9.8: Metrics of rice

Table 9.8 shows that rice has the empty TBox and relatively small but highly connected ABox. The ontology encodes 248 rice varieties along with their genetic information and states whether each variety is fertile or infertile. In other words, the data is complete with respect to fertility. Moreover, it is perfectly balanced: 124 varieties are fertile and 124 varieties are infertile.

[19] `http://www.mib.ac.uk`

[20] The data is not public and we are not allowed to disclose it due to IP reasons.

We aim at using hypotheses of DL-Miner to predict fertility and infertility of rice varieties using their composite genetic features, i.e. it is a binary classification problem discussed in Section 9.4.1.1. We build a prediction model from hypotheses as described in Section 9.4.1.1. In order to evaluate its usefulness, we measure how accurate a model is, i.e. we estimate its prediction errors via standard evaluation techniques from ML, see Section 2.2.3. In addition, we investigate how parameters of DL-Miner, see Algorithm 11, influence prediction errors. In particular, we concentrate on the minimal concept support p_{min} because it determines how "specific" acquired hypotheses can be. We vary p_{min} in the range from 1 to 20. For each value, we train and test a prediction model via 10-fold cross-validation. Hence, we run DL-Miner 200 times in total. Other parameters are fixed as in Section 9.3.1. We do not investigate another critical parameter ℓ_{max} considering the high computational cost of training and testing a prediction model.

9.4.1.3 Prediction Results

The algorithm has terminated and produced hypotheses in all cases. Thus, 200 prediction models have been trained and tested overall. Let us first discuss some observations. On the one hand, we observe that *no* varieties are left unpredicted, i.e. the case $\mathcal{O}_H \not\models F(v)$ and $\mathcal{O}_H \not\models F^{\neg}(v)$, where F stands for fertility, never happens. On the other hand, we observe clashes for *all* varieties, i.e. the case $\mathcal{O}_H \models F(v)$ and $\mathcal{O}_H \models F^{\neg}(v)$ always happens. This is explained by the fact that DL-Miner is complete, i.e. it produces all hypotheses which have any evidence in the data. In particular, $H_F := \{\top \sqsubseteq F\}$ and $H_{F^{\neg}} := \{\top \sqsubseteq F^{\neg}\}$ are amongst them. These hypotheses predict every variety to be fertile and infertile simultaneously which implies clashes for all varieties. We deal with clashes as described in Section 9.4.1.1. Figure 9.7 shows the prediction results, i.e. mean training and validation errors with confidence intervals (confidence level 95%) for different values of the parameter p_{min}.

As Figure 9.7 shows, the mean training error steadily increases with the minimal concept support p_{min}. The mean validation error slowly decreases with p_{min}, reaches its minimum at $p_{min} = 14$, and rises again. This is a typical behaviour of a prediction model. Low values of p_{min} lead to hypotheses capturing too many details of the data, or *overfitting*. High values of p_{min} result in hypotheses ignoring too many details of the data, or *underfitting*.

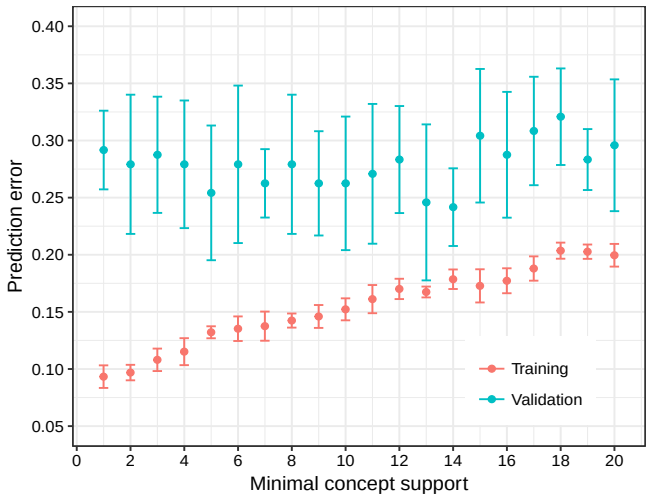


Figure 9.7: Prediction errors of hypotheses acquired by DL-Miner for rice using different minimal concept support

The optimal value of the parameter p_{min} seems to be $p_{min} = 14$, where the mean validation error has its minimum ≈ 0.24. Yet, it is hard to conclude given considerably wide confidence intervals. Nonetheless, given that the upper bound of the confidence interval for $p_{min} = 14$ is 0.2734661, we can conclude with confidence level 95% that the true test error is lower than 0.28 (or the true accuracy is higher than 0.72). Thus, a prediction model demonstrates potential to make reasonable predictions.

Finally, prediction accuracy is not the only goal: a prediction model itself can potentially help domain experts to understand phenomena. To be more specific, hypotheses of high quality may hint at genetic factors that are likely to be indicators of fertility, see Example 9.2.

Example 9.2. Consider one of the hypotheses acquired by DL-Miner for rice:

$$H := \{\exists s2Zygosity.c1Homozygote \sqcap \exists s10Zygosity.c2Homozygote \sqsubseteq Fertile\}$$

The hypothesis states that "if a rice variety has some $s2Zygosity$ relations to $c1Homozygote$ and some $s10Zygosity$ relations to $c2Homozygote$, then it is $Fertile$". If we look at the quality values of this hypothesis, we find out that its support $sup(H, \cdot) = 21$, assumption $asm(H, \cdot) = 3$, and confidence $conf(H, \cdot) = 0.875$, where "$\cdot$" denotes the remaining arguments as above. Thus, 21 out of 24 varieties with genetic features given by the LHS of H are, in fact, fertile. Hence, these genetic factors seem to be good indicators of fertility.

Example 9.2 suggests that, besides prediction accuracy, comprehensibility of hypotheses is also an important concern. In fact, prediction accuracy and comprehensibility are viewed as two orthogonal performance axes of ML models since the 1980s, see [Mic88]. As DLs and OWL are initially designed to be comprehensible knowledge representation formalisms, hypotheses acquired by DL-Miner offer an interesting opportunity for further research in this direction.

Thus, we can answer our research question (RQ) positively: DL-Miner can potentially be used for prediction purposes in DLs. This is sensible when data is naturally representable as a DL ABox, i.e. a labelled graph with typed relations. An example of such data is genetic information about rice varieties used in this case study. Besides predictions, DL-Miner can potentially assist human experts in understanding phenomena. Consequently, it allows for reading and refining the built prediction model as it is comprehensible.

9.4.1.4 Side Observations of Interest

In addition, we observe that a prediction model is more accurate for predicting infertility than fertility. In other words, it seems easier to predict a true negative than a true positive. We use the standard ML measures, *sensitivity* and *specificity* to check this. The first one, also called *true positive rate*, is the fraction of true positives amongst all positives. The second one, also called *true negative rate*, is the fraction of true negatives amongst all negatives. The average values of sensitivity and specificity across all runs of the algorithm are, respectively, as follows: 0.64 and 0.81. This confirms that infertility is, in fact, easier to predict than fertility on average.

Besides using DL-Miner, we have attempted to employ DL-Learner to make predictions. More specifically, we have configured it to learn the best description of the concept name F (fertility) given all instances of F as positive examples and all instances of $\neg F$ as negative examples (DL-Learner does not require a maximal length restriction to be specified but internally increases it when necessary). However, the best description returned by the algorithm is $\top$, i.e. the learned definition is $F \equiv \top$. Clearly, this prediction model has the prediction error of 0.5 and, hence, it is not better than a random guesser. This result is probably caused by the inherent structure of the data. To be more specific, there is no "feature", i.e. a genetic relation, that distinguishes most fertile varieties from infertile ones. All genetic relations seem to be required to produce

a meaningful prediction model. For example, there are at least 252 hypotheses of DL-MINER which are responsible for predictions in the case of $p_{min} = 20$ (and even more hypotheses for the lower values of the parameter). Therefore, to match the accuracy of DL-MINER on the given data, DL-LEARNER probably needs to produce a very long concept description (much longer than the longest descriptions reported in [LH10] that have the average length around 30).

9.4.2 Learning Hypotheses from the US National Transgender Discrimination Survey

Until now, we have been evaluating the approach mainly via the means independent of human opinion. In order to receive human feedback, we run a case study with researchers from the University of Florida (Health Outcomes and Policy College of Medicine) and the University of Arkansas for Medical Sciences (Myeloma Institute). We thank Amanda Hicks and Michael Rutherford for providing the data for this study, participating in it, and giving valuable feedback.

The subject of the study is the ontology,[21] in the following called ntds, created by the domain experts using data from the US National Transgender Discrimination Survey.[22] Table 9.9 shows the metrics of ntds.

$\mathcal{DL}$	$\lvert in(\mathcal{A})\rvert$	CA	RA	$deg(\mathcal{A})$	$con(\mathcal{A})$	$\lvert voc(\mathcal{A})\rvert$	$\lvert voc(\mathcal{T})\rvert$	$jvoc(\cdot)$
$\mathcal{SROIQ}$	169,058	169,058	404,219	2.39	26.08	93	522	0.16

Table 9.9: Metrics of ntds ("$\cdot$" stands for $\mathcal{A}, \mathcal{T}$)

The case study is aimed at investigating whether DL-MINER is able to learn hypotheses which are useful or interesting for the domain experts. According to Table 9.9, ntds is an ontology with a large ABox and expressive TBox. For this ontology, DL-MINER is expected to be useful for finding axioms which are missing in the TBox but supported by the ABox. Presumably, those axioms can help the domain experts to enrich the TBox, reveal modelling errors, or discover new phenomena. In addition, we intend to find out whether and which quality measures are helpful for finding good hypotheses, i.e. indicators of hypothesis usefulness. Thus, we investigate the following research questions.

[21] The ontology is not public yet.

[22] `http://www.transequality.org/issues/national-transgender-discrimination-survey`

RQ1 Can DL-Miner learn hypotheses which are useful or interesting for the domain experts?

RQ2 Which quality measures (if any) are good indicators of hypothesis usefulness?

9.4.2.1 Experimental Design

To answer the research questions, we run DL-Miner to produce hypotheses from the ontology, evaluate them, and ask the domain experts to assess them. Expert opinion needs to be quantified such that we can compare values of expert-assessed quality with values of machine-assessed quality, i.e. hypothesis quality measures. The comparison can then be done via measuring correlations between expert opinion and quality measures.

We run DL-Miner on ntds with the same parameters as in Section 9.2. Since the algorithm generates a large number of hypotheses, we cannot ask the domain experts to judge all of them. Therefore, we sample some specified number of hypotheses that the experts can comfortably handle. Considering possible outcomes of expert feedback, i.e. optimistic, pessimistic, expected, we have estimated that we require judgements of at least 60 hypotheses in order to achieve statistically significant results in the expected case. Those hypotheses are sampled such that one half of them consists of high-quality hypotheses and another half consists of middle- and low-quality hypotheses. To be more specific, we identify high-quality hypotheses using one of the quality measures (we have picked confidence because it is easy to interpret). Then, we randomly sample 30 hypotheses from those and 30 hypotheses from all the rest. This way we ensure both variability of hypothesis quality and sufficient presence of high-quality hypotheses (which are presumably most promising).

We concentrate on two aspects of expert opinion about a hypothesis: validity and interestingness.

- *Validity* assesses whether a hypothesis captures a general truth about the domain and can be perceived as an axiom to be added to the ontology.

- *Interestingness* assesses how interesting a hypothesis is for a domain expert. In contrast to validity, it does not evaluate the general correctness of a hypothesis, but rather expert's curiosity and attention that she pays to it.

Thus, validity and interestingness are different notions. On the one hand, a hypothesis can be valid but uninteresting if it captures unimportant or irrelevant knowledge with respect to the ontology. On the other hand, a hypothesis can be invalid but interesting if it indicates a data bias or modelling error worth investigating, i.e. it looks useful for ontology debugging.

To quantify validity, we ask the domain experts to judge a hypothesis by choosing one of the following three options: *"correct"*, *"wrong"*, *"don't know"*. The first (second) option should be chosen if the hypothesis is mostly correct (wrong) according to the expert's knowledge. The third option should be chosen when it is hard to decide between the first two options. An undecided, or *unknown*, hypothesis is interesting if it seems to capture new domain knowledge or phenomena and encourages new investigations. To quantify interestingness, we ask the experts to rate how interesting a hypothesis is on the linear scale from 0 (lowest) to 4 (highest). We collect feedback using an online survey.

9.4.2.2 User Feedback

Once DL-Miner terminated and returned hypotheses for ntds, we sampled 60 hypotheses and composed a survey as described above. The survey was shared with the domain experts and one of them fully completed it. In the feedback that we received the expert expressed interest in reviewing additional hypotheses, more specific than the ones presented in the survey. She was particularly interested in exploring hypotheses about various gender identities. Thus, we were implicitly given *focus terms*, i.e. a signature of interest Σ, which was previously extracted automatically and included the majority of terms from the ontology.

Considering the expert's response, we decided to make another survey consisting of hypotheses about the focus terms. In the following, we refer to the initial, unfocused survey as *Survey 1* and the follow-up, focused survey as *Survey 2*. To acquire hypotheses for Survey 2, DL-Miner was run with the same parameters as for Survey 1, but with the seed signature Σ consisting of the focus terms (all concept names representing gender identities and all role names). We sampled 60 new hypotheses as above and made Survey 2 from them which was shared with the domain expert. She completed the survey and gave us further feedback.

We present the results of Survey 1 alongside the results of Survey 2 and compare them. Table 9.10 shows the distribution of the expert's answers along the validity and interestingness axes.

	Validity	**Interestingness**				
		0	*1*	*2*	*3*	*4*
Survey 1 (unfocused)	*Wrong*	6	11	30	-	-
	Don't know	-	1	-	2	4
	Correct	-	-	-	6	-
Survey 2 (focused)	*Wrong*	1	-	1	-	5
	Don't know	-	-	-	-	49
	Correct	-	-	-	-	4

Table 9.10: Assessment of hypotheses acquired by DL-Miner for ntds: distribution of answers in Survey 1 and Survey 2 ("-" denotes zero)

As Table 9.10 shows, Survey 1 contains 47 hypotheses deemed to be wrong, 7 hypotheses unknown, and 6 hypotheses correct. The majority of wrong hypotheses are of average interestingness (marked by 2) and the rest of wrong hypotheses are less interesting (marked by 0 or 1). As the domain expert points out in her feedback, wrong hypotheses which are marked by the interestingness of 2 indicate *data bias*, i.e. those are incorrect but strongly supported by the data. According to the results, unknown and correct hypotheses appear to be much more interesting than wrong ones: all of them, except one, have high values of interestingness (marked by 3 and 4). Amongst those, unknown hypotheses are marked to be the most interesting and, according to the expert's response, require further analysis. Overall, 12 out of 60 hypotheses (20%) are found to be interesting.

The results of Survey 2 are much different from the results of Survey 1, see Table 9.10. While most hypotheses in Survey 1 are deemed to be wrong, most hypotheses (49 out of 60) in Survey 2 are marked as unknown. Another noticeable difference is that all hypotheses, except two, in Survey 2 are marked by the highest value of interestingness, i.e. 58 out of 60 hypotheses ($\approx 96.7\%$), including wrong ones, are very interesting in expert's opinion. Moreover, the expert informed us in her response that one of the wrong hypotheses, besides indicating data bias, revealed an *error in the ontology*. Thus, if focus terms are specified by the domain expert, the resulting focused hypotheses appear to be significantly more interesting than unfocused ones. This is not surprising because, by providing focus terms, the expert expresses her interest in exploring hypotheses about those terms. In addition, the expert is likely to inquire into the domain area which she knows less about. As a result, the majority of focused hypotheses are deemed to

be unknown.

Considering the results of Survey 1 and Survey 2, we can conclude that DL-Miner is able to acquire hypotheses that are useful and interesting for domain experts (and answer RQ1 positively). Wrong hypotheses are found to be useful for ontology debugging. Correct hypotheses are helpful for enriching the TBox. Unknown hypotheses can encourage investigations and help to discover new domain knowledge.

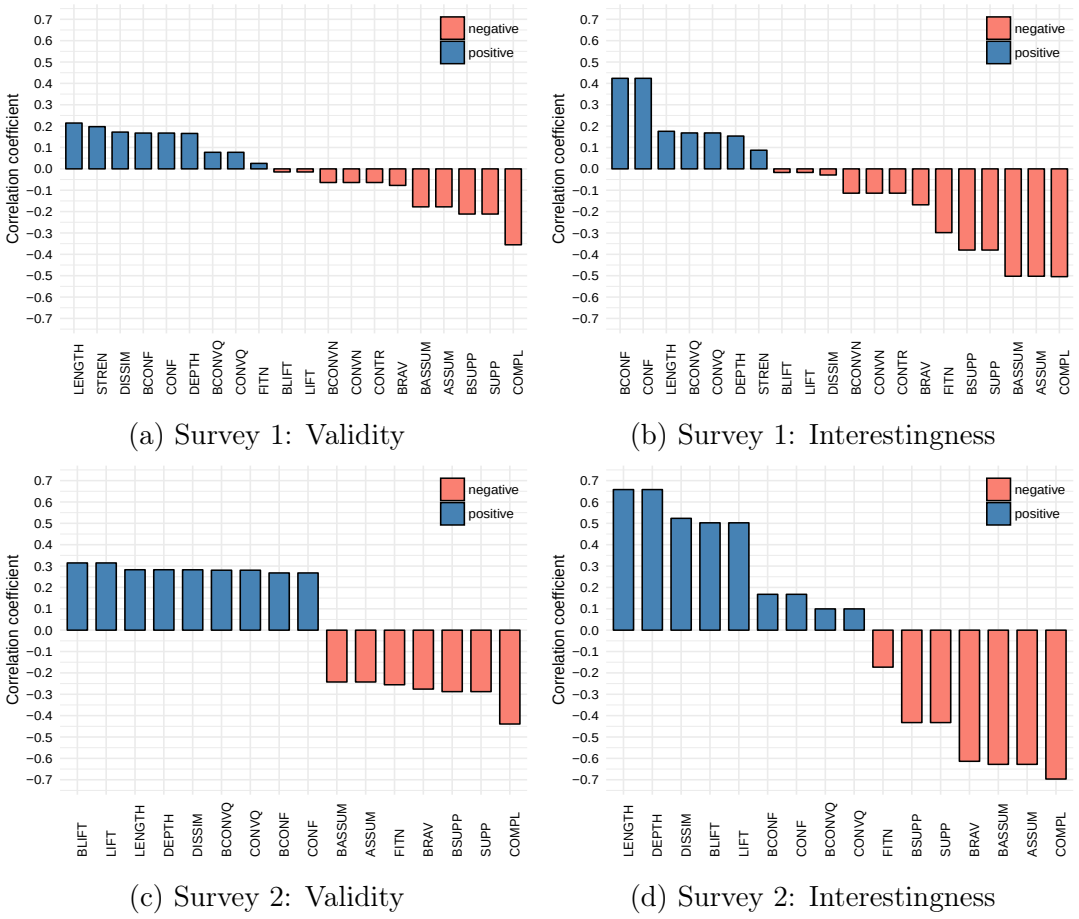
9.4.2.3 Indicators of Hypothesis Usefulness

We have described the usefulness of hypotheses for ntds by collecting their judgements made by the domain expert, i.e. the validity and interestingness scores, see Table 9.10. We now investigate which quality measures indicate hypothesis usefulness (RQ2). As mentioned above, this can be done via estimating correlations between values of machine-assessed quality, i.e. hypothesis quality measures, and values of expert-assessed quality, i.e. validity and interestingness. The results are shown in Figure 9.8.

As Figure 9.8 shows, there are positive and negative correlations, which is to be expected. In other words, there are *positive* and *negative* indicators. Positive (negative) indicators are those measures which show positive (negative) correlations with hypothesis usefulness, i.e. validity or interestingness.

According to Figure 9.8a, there is no best positive indicator for validity in Survey 1: length, strength, dissimilarity, confidence, and depth have approximately equal values. However, complexity seems to be the best negative indicator followed by support and assumption. The fact that support is a negative indicator is rather unexpected since the measure evaluates how much evidence the data contains about the hypothesis. A possible explanation is that hypotheses with more evidence seem to be easier to reject for the domain expert because "counterexamples" are easier to recall from the data.

In contrast to validity, see Figure 9.8a, interestingness in Survey 1 has a clearly best positive indicator which is confidence (basic and main), see Figure 9.8b. In other words, the domain expert treats more confident hypotheses as more interesting ones. Like validity, interestingness has complexity, assumption, and support as its best negative indicators. Fitness turns from a non-indicator for validity to a negative indicator for interestingness. Support appears to be a strong negative indicator for interestingness because hypotheses with high support are



(a) Survey 1: Validity

(b) Survey 1: Interestingness

(c) Survey 2: Validity

(d) Survey 2: Interestingness

Figure 9.8: Correlations (in descending order) between hypothesis quality measures and expert's judgements (4 measures are not shown for Survey 2 because their deviations equal zero and correlation coefficients cannot be calculated)

likely to be "general", i.e. reflecting known, easily seen patterns of the data. Those hypotheses are not as surprising as "specific" ones which, on the contrary, are likely to reflect uncommon, hardly seen patterns of the data.

The results in Figure 9.8c are similar to the results in Figure 9.8a: the correlations for validity look almost equally distributed. The main difference is that lift turns from a non-indicator in Survey 1 to a strong positive indicator in Survey 2. One possible reason is that Survey 2 consists of "specific" hypotheses. Such hypotheses are likely to have a higher lift than "general" hypotheses in Survey 1 (since the denominator decreases faster than the numerator, see Definition 5.16).

In comparison to Figure 9.8b for Survey 1, Figure 9.8d for Survey 2 shows considerably stronger correlations for interestingness. This may be caused by the much higher fraction of interesting hypotheses in Survey 2 in comparison to

Survey 1, see Table 9.10. While the best negative indicators remain mostly the same, the best positive indicators change: confidence is not in the top which includes length, depth, dissimilarity, and lift, see Figure 9.8d. Thus, lift and dissimilarity, which are non-indicators for interestingness in Survey 1, become strong positive indicators for it in Survey 2. Both of them may indicate (by high values) "specific" hypotheses comprising Survey 2. The same reason may facilitate length and depth to become the top positive indicators (please recall that length and depth strongly correlate with lift according to Figure 9.1).

Overall, the results in Figure 9.8 show that good positive indicators are confidence for Survey 1 (unfocused), lift, dissimilarity, length, and depth for Survey 2 (focused). Good negative indicators are complexity, assumption, and support. Importantly, the results suggest that there is no single best indicator of hypothesis usefulness. Hence, we probably need to consider multiple quality measures while selecting promising hypotheses.

9.4.2.4 Side Observations of Interest

Besides examining correlations between the hypothesis quality measures and human judgements, i.e. validity and interestingness, we checked the correlation between validity and interestingness. It turns out that, despite being different notions, these correlate relatively strongly: their correlation coefficient equals 0.61 for Survey 1 and 0.39 for Survey 2. The correlation is stronger for Survey 1 because, according to Table 9.10, most interesting hypotheses are either correct or unknown.

9.5 Discussion

The results presented in this chapter look promising. Despite the fact that DL-Miner uses reasoning, it is feasible, even for relatively big and complex ontologies. Its computational performance strongly depends on the choice of quality measures to be computed. This choice should be made considering the input ontology and task at hand. One can choose only cheap quality measures if performance is a concern or detailed hypothesis evaluation is not required.

DL-Miner seems to be capable of learning most hypotheses that other unsupervised approaches can possibly learn. In addition, it is able to learn hypotheses that other approaches cannot learn. This is a consequence of more expressive,

flexible language bias and completeness of DL-Miner.

In our case study on rice fertility prediction, we have found out that acquired hypotheses can be used to build a prediction model capable of making sufficiently good predictions for highly relational genetic data. As we have learned from the case study with domain experts, a hypothesis can be interesting regardless of its validity, i.e. an interesting hypothesis can be correct, wrong, or unknown. Specifically, a correct hypothesis can enrich the TBox; a wrong hypothesis can indicate a modelling error or data bias; an unknown hypothesis can capture an interesting, novel piece of domain knowledge. Focus terms specified by a domain expert can help to acquire hypotheses of higher interestingness. It turns out that there is no one best indicator of hypothesis usefulness, i.e. multiple quality measures probably need to be considered. This is not surprising since a hypothesis in DLs clearly has several quality dimensions. Further investigations are required to make conclusions about possible applications of the approach.

Chapter 10

Summary and Outlook

This thesis focuses on advancing the state-of-the-art of Ontology Learning (OL). In this chapter, we summarise the main results of the thesis and contributions made. We also discuss some outstanding issues and make suggestions for future work.

10.1 Contributions in Brief

This thesis investigates whether and how a general TBox (terminology) can be induced from an ABox in DLs, respecting the standard semantics. The problem is formulated as General Terminology Induction and opens a new perspective on OL. Before the work presented in this thesis, it was not clear how to efficiently construct expressive hypotheses and how to rigorously evaluate them. We think that these are two fundamental questions in OL. The thesis investigates both of them in detail.

We have designed an approach for General Terminology Induction and implemented it in an algorithm, called DL-Miner. In contrast to other approaches, it is *not* a direct adaptation of an existing method from other areas of Artificial Intelligence, but an approach which is specifically designed for DLs respecting the standard semantics and background knowledge. We contribute the foundations of the approach, its implementation, optimisation, and evaluation including case studies.

10.2 Key Ideas

There are several ideas behind the approach presented in this thesis. We consider them as part of our contribution. In the following, we summarise main ideas which, as we think, should be of interest to the OWL and DL community.

- **Measuring multi-dimensional hypothesis quality** In order to rigorously evaluate hypothesis quality, we should ideally consider all its aspects. We propose that there are multiple dimensions of hypothesis quality in DLs. The data, i.e. ABox, can inform statistical quality of a hypothesis, e.g. the number of instances supporting it (support), the number of instances "guessed" by it (assumption), etc. The background knowledge, i.e. existing TBox, can inform logical quality of a hypothesis, e.g. whether it is consistent or informative with respect to the TBox. In addition, some hypotheses are harder to read than others because they are longer or redundant. We introduce multiple quality measures to evaluate these and other aspects of hypothesis quality.

- **Respecting the standard DL semantics** Hypothesis quality measures should respect the standard semantics of DLs. In particular, they should permit its OWA and thus avoid the CWA, unless the latter is rational for the given data. This can be achieved by careful definitions of hypothesis quality measures.

- **Acquiring hypotheses, not solutions** We view OL as a process of constructing and evaluating hypotheses, rather than searching for solutions. The decision which hypotheses are useful should rather be made by a human expert than by an automated procedure. According to our case studies, hypotheses of high quality can be uninteresting and hypotheses of average quality can be interesting. We acquire a set of all suitable hypotheses and use hypothesis quality measures to order and navigate through them.

- **Reaching expressive hypotheses** Instead of focusing on particular types of DL axioms, e.g. concept definitions, we suggest to acquire general axioms, i.e. GCIs and RIs. It is generally unknown which axiom types lead to interesting hypotheses. Therefore, we suggest to target as flexible hypotheses as possible, though they are still limited by a certain language bias to maintain feasibility.

- **Achieving completeness** In order to avoid missing interesting hypotheses, we should ideally construct and evaluate all hypotheses conforming to the given language bias. This is achieved via the principled bottom-up construction of concepts and then generating all hypotheses from those concepts according to the language bias.

As we demonstrate in this thesis, it is possible to implement all of the aforementioned ideas in one algorithm. Moreover, it does not require any supervision or human intervention to deliver its results.

10.3 Main Challenges

Nonetheless, realising the listed ideas was not straightforward. We faced both conceptual and implementation challenges along the way which are discussed below.

10.3.1 Defining Hypothesis Quality Measures

As the approach is aimed at expressive and flexible hypotheses, it is not clear how to rigorously evaluate such hypotheses. Other approaches mainly focus on learning hypotheses of certain types, e.g. concept definitions, disjointness axioms, etc. They can simplify measuring hypothesis quality by considering only those axiom types. In particular, they tend to use quality measures from ML and DM that disregard the OWA of DLs. Moreover, a hypothesis is commonly evaluated against the data, i.e. ABox, only. In other words, there is a lack of measures evaluating a hypothesis against the available background knowledge, i.e. TBox.

We introduce and define the quality measures that evaluate arbitrary (sets of) GCIs and RIs while respecting the standard DL semantics and background knowledge. Some of these measures are straightforward to define as they capture the standard notions in DLs, i.e. consistency, informativeness, logical strength. The statistical axiom measures are based on similar measures from DM but defined to respect the semantics of DLs. Not only do they take the TBox into account while retrieving instances, but also respect the standard OWA and treat GCIs like the classic implications, i.e. being equivalent to their contraposition.

Some measures are aimed at quantifying novel aspects of hypothesis quality, i.e. complexity, fitness, braveness. These are rather difficult to formalise as they,

in turn, require introducing and measuring non-standard notions, e.g. description length (the length of a minimal ABox with respect to a given TBox and ABox). We investigate, both theoretically and empirically, properties of the proposed quality measures and relationships between them.

10.3.2 Computing Hypothesis Quality Measures

As it turns out, not all quality measures are straightforward to compute given their definitions: there are both conceptual and performance challenges. It is important that the measures are computed efficiently because there are generally numerous hypotheses to be evaluated. Eliminating redundancy is challenging because redundancy can take various shapes, i.e. not only whole axioms can be redundant but also their parts. We deal with redundancy of a hypothesis via computing its structural transformation and then finding its minimal subset which is equivalent to the hypothesis.

The naive computation of dissimilarity and complexity is computationally expensive because it requires performing multiple reasoning operations with respect to the TBox. The definition of fitness and braveness requires finding a minimal, shortest ABox with respect to a given TBox and ABox, i.e. considering all reasons of relative redundancy of the former with respect to the latter. For the aforementioned measures, we suggest approximations and optimisations which are based on checking and then reusing certain entailments when necessary.

10.3.3 Constructing Hypotheses

Not only evaluating, but also constructing expressive and flexible hypotheses is challenging. At first, it may not seem so because one can specify axiom shapes of interest (templates) and generate all hypotheses of those shapes in a top-down fashion. This is what the unsupervised OL approaches normally do.

However, this method is likely to be infeasible for expressive hypotheses. Since it is generally unknown which axiom shapes produce good hypotheses, a human expert should specify templates for as many of them as possible. This can be a tedious task even for not expressive DLs. Moreover, some interesting shapes can still be overlooked. On the other hand, templates of all shapes, even if specified, can lead to a tremendous number of hypotheses the majority of which may have no sufficient evidence in the data and, hence, cannot be reasonably evaluated

anyway.

We suggest constructing hypotheses in a bottom-up fashion. Instead of handcrafting templates, we enumerate all concepts in a principled way. In particular, we use the mechanics of refinement operators to enumerate all concepts according to the language bias, i.e. ensure completeness. In order to identify and discard unsuitable concepts, we consult the data. Specifically, we formalise the notion of suitability for a concept based on the instances of that concept in the data and use that notion to prune the majority of unsuitable concepts *a priori*. This procedure is implemented by the algorithm called DL-Apriori.

10.3.4 Handling Enormous Amount of Hypotheses

As our experiments show, for some ontologies, especially ones with a large vocabulary, even the bottom-up construction of concepts results in an enormous number of hypotheses. We suggest several optimisation techniques that can help to narrow down the set of acquired hypotheses. Those techniques can be categorised into two groups: incomplete construction and incomplete evaluation of hypotheses.

As the name suggests, techniques of the first group are used to construct some, but not all, hypotheses, i.e. sacrifice completeness. They include limiting the maximal role depth of a concept in addition to its length, discarding concepts with superfluous parts with respect to the TBox, constructing only most promising concepts via beam search. Techniques of the second group are used to evaluate hypotheses partly. They include evaluating all hypotheses via cheap quality measures and evaluating completely only suitable or most promising hypotheses. In particular, one can consider only consistent, informative, and non-redundant hypotheses. Most promising hypotheses can be identified by cheap quality measures acting as heuristics. Finally, hypothesis evaluation can be run as an anytime algorithm.

Of course, a set of hypotheses can be reduced by adjusting the parameters of DL-Miner. In particular, this can be done by decreasing the maximal concept length, increasing the minimal concept support, or specifying terms of interest, i.e. a seed signature.

10.3.5 Handling Inconsistent Ontologies

If an input ontology is inconsistent, hypotheses cannot be correctly evaluated because most quality measures use reasoning and thus return trivial, useless values. Even the bottom-up construction of concepts makes no sense in this case since all individuals are instances of all concepts. We use an automated procedure to repair an inconsistent ontology based on justifications. It guarantees to return a consistent ontology but may remove too many assertions. The latter should not mislead hypothesis construction and evaluation unless the number of erroneous assertions is relatively big (that would make the results of learning questionable anyway).

10.3.6 Ordering and Ranking Hypotheses

Although multiple measures offer profound evaluation of hypothesis quality, they can also cause difficulties in ordering and navigating through the evaluated hypotheses. The reason is that, if multiple quality measures are used, one hypothesis can be better than another hypothesis on one measure and be worse on another measure, i.e. hypotheses can be incomparable. There are several ways to approach this problem.

Firstly, one can aggregate multiple quality measures into a single, collective measure and rank hypotheses by that measure. This, however, requires normalising quality values to ensure fair comparisons. Such normalisation is not straightforward to perform for some measures. Another concern is that some measures are more important than others. This can be resolved by weighted aggregation schemes where weights for measures should be chosen carefully.

Secondly, one can define a dominance relation on hypotheses such that one hypothesis dominates another hypothesis if it is better on at least one measure and not worse on other measures. We suggest ranking each hypothesis using the standard notion of a Pareto front (though this can cause many hypotheses to be in the same rank). One should carefully choose measures for ranking because a hypothesis reaches the top rank if it is better than all other hypotheses on one measure and worse on all other measures.

Finally, the evaluated hypotheses can be explored without ranking them. A user can navigate through the hypotheses switching between the measures and each time sorting all hypotheses by the chosen measure. Alternatively, a user can

specify some quality thresholds, e.g. the minimal confidence is 0.7, and retrieve all hypotheses satisfying them.

10.3.7 Evaluating DL-Miner

Both the implementation of DL-Miner and its evaluation present new challenges. The first experiment, where we investigate mutual correlations of the quality measures and computational performance of DL-Miner, is rather straightforward to design and execute. On the contrary, the second experiment comparing DL-Miner with other OL approaches is not easy to accomplish. Its main challenge is that the approaches operate in different settings: some of them are supervised, others are unsupervised but neglect the DL semantics. Moreover, all related approaches are aimed at hypotheses that are less expressive than hypotheses that DL-Miner is designed to acquire. These reasons make direct comparisons hardly possible.

In order to compare DL-Miner with DL-Learner, we use the latter in the unsupervised mode, despite the fact that it is a supervised algorithm for CDL. As DL-Learner learns concept definitions, it is not clear how to compare them with hypotheses of DL-Miner which are GCIs in general. We introduce the notion of hits (and misses) to identify concept definitions of DL-Learner which are learned, either explicitly or implicitly, by DL-Miner. As DL-Miner generally produces (significantly) more hypotheses than DL-Learner does, we identify a minimal set of DL-Miner hypotheses, i.e. a hitting set, that capture all knowledge of the hit concept definitions of DL-Learner. Then, we are able to compare both sets on the common grounds.

Nonetheless, the aforementioned comparison strategy incurs additional difficulties. Firstly, a set of hypotheses of DL-Miner can be inconsistent. Hence, it makes no sense to consider which concept definitions of DL-Learner are entailed. As a workaround, we test (using modules) whether there is a consistent subset of hypotheses that entails a given concept definition. Secondly, as research on justifications in OWL and DLs demonstrate, finding a minimal subset (hitting set) of axioms that entails given axioms (hits) is computationally costly. For this purpose, we reuse the procedure of redundancy elimination suggested in this thesis.

Another challenge is comparing DL-Miner with other unsupervised OL approaches, i.e. BelNet and SSI. Since these apply the CWA to learn disjointness

axioms from ontologies where the data lacks any negative information, we are forced to do the same to be able to acquire such hypotheses. Therefore, we explicitly add all negative assertions "assumed" by the CWA to the ABox. Please recall that this is done solely for the sake of comparison.

The case studies also turn out to be challenging to design and run. In the first case study, we investigate whether DL-Miner can be useful for rice fertility prediction. Constructing a prediction model from acquired hypotheses is not straightforward. Firstly, an individual can be left unpredicted due to the OWA. Secondly, an individual can be predicted to be an instance of disjoint concepts, e.g. F (fertile) and $\neg F$ (infertile), i.e. a clash can occur, since a prediction model is built from a set of hypotheses. At it turns out, the first case never happens in our experiment. The second case is handled via finding hypotheses causing the clash and trusting the one which has the highest quality according to confidence and support.

The second case study is aimed at collecting and analysing human opinion about hypotheses acquired by DL-Miner. The first question that arises is how to quantify human opinion in order to make numerical analysis possible. For this purpose, we introduce the notions of validity and interestingness for a hypothesis. The first captures whether a hypothesis is true in the domain and is assessed by a domain expert via choosing one out of three options: "correct", "wrong", "don't know". The second captures how interesting a hypothesis is for a domain expert and is marked on the linear scale from 0 to 4. Another question is how to sample hypotheses for the study since the algorithm usually generates a large number of hypotheses. We sample one half of hypotheses from high-quality ones and another half from all the rest to ensure sufficient presence of promising hypotheses in the study and variability of hypothesis quality. Feedback from a domain expert is collected via online surveys.

10.4 Gained Insights

The thesis gains several insights into the problem of OL which we view in the broad sense as General Terminology Induction. Overall, the results of empirical evaluation of DL-Miner show that the algorithm mainly works as specified by its design. The case studies open opportunities for further investigations. The main empirical outcomes are discussed below.

10.4.1 Mutual Correlations of Hypothesis Quality Measures

The experiments show that the introduced quality measures mainly correlate as expected, considering their definitions and properties. In other words, related measures correlate, while unrelated do not. The main axiom measures positively correlate with their respective basic axiom measures. The differences between them are mainly caused by the presence of negative information in the ABox. Hence, if negative information is not present in the ABox, the main measures can be replaced with their basic counterparts, which also follows from their definitions. While braveness strongly correlates with assumption, the correlation between fitness and support is weak, confirming that these measures capture different notions. There are also some unexpected correlations, e.g. between lift and length, suggesting that longer hypotheses are likely to have higher quality according to lift.

10.4.2 Computational Performance of DL-Miner

As it turns out, consistency of a hypothesis with respect to the ontology is the most expensive quality measure on average. The cost of comparing hypotheses by logical strength is also high, as it grows with the number of hypotheses. Computing the axiom measures that do not consider negation (most basic measures) is considerably more expensive than computing the axiom measures that consider negation (all main measures and some basic ones). As the difference between the main and basic measures is mainly caused by negative information in the ontology, it is worthwhile to compute only the basic measures when negative information is absent. Overall, considering that hypothesis evaluation is the most expensive operation, the computational performance of DL-Miner can be significantly improved if costly measures are skipped. Nonetheless, we advise doing that only if some measures are clearly irrelevant or performance is a concern, e.g. the ontology is particularly hard for reasoning. Otherwise, one would risk overlooking important information about hypothesis quality, e.g. consistency.

10.4.3 Comparing DL-Miner with Other Approaches

As the comparison with related approaches shows, DL-Miner is able to acquire almost all hypotheses learned by other OL approaches. Moreover, DL-Miner acquires many hypotheses that other approaches cannot learn. This is a consequence of realising its design principles. Specifically, DL-Miner is designed to acquire more expressive hypotheses than other approaches can possibly learn, i.e. handle an expressive and flexible language bias. Completeness of DL-Miner with respect to its language bias ensures that the algorithm does not miss any hypotheses conforming to the language bias.

While comparing DL-Miner with DL-Learner, we find out that concept definitions of DL-Learner can, in fact, be represented by (significantly) shorter hypotheses of DL-Miner. Hence, concept definitions, if considered as a set, are prone to contain redundancy. This happens because, as a CDL approach, DL-Learner fixes axiom shapes for its hypotheses, see Example 9.1. Moreover, DL-Learner is not optimised for learning multiple concept definitions. This makes it (sometimes significantly) slower than DL-Miner if the latter uses only the cheap quality measures.

Comparing DL-Miner with BelNet and GoldMiner (the implementation of SSI) requires the adaptation of the CWA because disjointness axioms cannot be acquired by DL-Miner for the experimental ontologies otherwise. On the one hand, this shows the difference between the approaches, i.e. the consequences of respecting and neglecting the standard OWA, see Example 8.2. On the other hand, this demonstrates that DL-Miner can also operate under the CWA which is made by adding respective negative assertions to the ABox. However, this relies on the ability to correctly "close" the ABox.

10.4.4 Making Predictions in Description Logics Using DL-Miner

As the case study on rice fertility prediction shows, hypotheses acquired by DL-Miner can be used to build a prediction model. Such a prediction model exhibits a behaviour typical for ML models, including the effects of underfitting and overfitting. It demonstrates a potential to make accurate predictions as it achieves the accuracy around 76% on highly relational genetic data. Importantly, a prediction model is comprehensible since it consists of hypotheses which are DL axioms.

Hence, it can potentially help domain experts to gain insight into their data and understand phenomena, see Example 9.2.

10.4.5 Usefulness of Hypotheses for Domain Experts

Our case study with the domain experts demonstrates that DL-Miner is able to acquire useful hypotheses. Moreover, the results show that a hypothesis can be useful regardless of its validity, i.e. whether it is correct, wrong, or unknown. Correct hypotheses can enrich the TBox. Wrong hypotheses appear to be useful for ontology debugging as they can help a domain expert to identify modelling errors. Unknown hypotheses can spark investigations and help to discover new phenomena in the domain.

As it turns out, "focused" learning can produce significantly more interesting hypotheses than "unfocused" one. More specifically, if an expert specifies terms of interest, i.e. a seed signature, the resulting hypotheses appear to be much more interesting than the hypotheses acquired using automatically selected terms. This is not surprising since an expert shows her interest in hypotheses about focus terms by providing them. Please note that a "boring" hypothesis can still be useful as it can enrich the TBox. This is why we separate the notions of interestingness and validity in Section 9.4.2.

Another important observation is that there is no single best indicator of hypothesis usefulness amongst the quality measures. Good positive indicators are confidence, lift, dissimilarity, and length. Good negative indicators are complexity, assumption, and support. This shows that we probably need to consider multiple quality measures in order to select useful hypotheses in DLs.

10.5 Future Work

This thesis presents a new, generalised vision of the OL problem and lays the foundations of an approach that seems capable of fulfilling that vision. The proposed methodology and empirical evaluation of the implementation open opportunities for further research in this direction. Some avenues for future work are presented and discussed below.

10.5.1 Advancing DL-Miner

Although the empirical results of DL-Miner look promising, there are still opportunities for further enhancements. In the following, we discuss outstanding issues that need to be addressed.

10.5.1.1 Dealing with Redundancy of Hypotheses

Definition 5.7 and Algorithm 1 deal with redundancy of a hypothesis caused by its excessive length. More specifically, redundant axioms and axioms with redundant parts are detected in a hypothesis and subsequently eliminated from it. Although these are frequent signs of redundancy (also noticed for justifications), there are other, less obvious types of redundancy, see Example 5.7 and Example 5.8.

In order to handle all possible types of redundancy, we need to extend Definition 5.7 accordingly. Besides requiring that a non-redundant hypothesis is shortest amongst all syntactic variations, such definition should also state that all parts of a non-redundant hypothesis are as weak as possible. Detecting and eliminating redundancy according to that definition would probably require a procedure more sophisticated than Algorithm 1. It is also expected that such procedure would be more expensive computationally.

10.5.1.2 Constructing Concepts Beyond $\mathcal{ALC}$

The current implementation of DL-Miner supports constructing complex concepts for $\mathcal{ALC}$ (and hence less expressive DLs). Please recall that it also constructs complex roles given their templates. Hence, depending on role templates, the resulting hypotheses can be as expressive as $\mathcal{SHI}$ to allow for complex role hierarchies and inverses. Nevertheless, there are more expressive DLs which are used to build ontologies. In particular, as mentioned in Section 2.1.4, the current version of OWL is based on $\mathcal{SROIQ}(\mathcal{D})$. Therefore, it is sensible to construct complex concepts as expressive as that.

The capabilities of DL-Miner to construct complex concepts depend on availability of suitable refinement operators that are currently proposed for $\mathcal{ALC}$. Thus, in order to construct concepts beyond $\mathcal{ALC}$, we need to design suitable refinement operators for more expressive DLs (or wait until they are designed by other researchers, e.g. in CDL). Of course, we can generate expressive concepts for any DL using respective concept templates. However, this is likely to make

the algorithm incomplete for that DL, i.e. many interesting hypotheses might be missed. Once a suitable refinement operator is designed for a DL of choice, it can be straightforwardly embedded into the implementation of DL-Miner. The latter may require some additional optimisations since reasoning operations are likely to be more computationally costly for more expressive DLs.

10.5.1.3 Scaling DL-Miner to Large Knowledge Bases

As our experiments show, DL-Miner is computationally feasible even for large domain ontologies. Its performance strongly depends on performance of instance retrieval. Hence, instance retrieval should be as efficient as possible. It is known that, while OWL reasoners are proved to be efficient for large TBoxes, their performance degrades significantly for large ABoxes.

The poor performance of OWL reasoners on large ABoxes has inspired developing hybrid systems, such as PAGOdA [ZCGNH15], that use a standard OWL reasoner to perform TBox reasoning and a specialised, data-focused reasoner (query engine) to perform ABox reasoning, i.e. instance retrieval. A similar approach is implemented in the latest version of DL-Learner [BLW16], where instance retrieval is executed by an optimised RDF query engine.

In order to scale DL-Miner to large knowledge bases, we can take the same path, i.e. replace an OWL reasoner with a data-focused reasoner to perform instance retrievals. This is even more sensible in the light of rapidly growing Linked Data which is essentially a collection of RDF ontologies with large ABoxes. The applicability of DL-Miner would then be expanded and enable learning expressive hypotheses for large RDF knowledge bases.

10.5.2 Investigating Other Quality Measures and Measure Combination Schemes

One of the main questions addressed in this thesis is how to rigorously evaluate expressive hypotheses in DLs. We have discussed three dimensions of hypothesis quality, i.e. readability, logical quality, statistical quality, and introduced several quality measures to evaluate a hypothesis in these dimensions. Although we argue that the proposed quality measures allow for thorough evaluation of a hypothesis, they do not form the "complete list" of hypothesis quality measures. Clearly, there are other possible measures.

Firstly, the notion of readability is not only syntactic, as it is measured by syntactic length and depth. Although redundancy captures some semantic aspects of readability, there are other aspects to consider. For example, some DL constructors, e.g. $\forall R.C$, seem to be harder to understand than others irrespective of their length [WMCM14]. In addition, logical interactions between axioms can come into play [HBPS11]. Clearly, some hypotheses can be easier or harder to understand depending on the TBox.

Amongst the proposed logical measures, logical strength can be defined respecting the TBox. TBox-aware logical strength would facilitate finer, more complete ordering of hypotheses but would be much more costly to compute for a hard TBox. Additional statistical axiom measures can be derived from measures used in ML/DM, e.g. cosine, Gini index, J-measure, etc. [GH06], respecting the standard DL semantics and its OWA as it is done in this thesis.

Finally, other strategies of ranking hypotheses can be investigated. In particular, we can refine the single-measure ordering, see Section 8.2.1. This needs developing techniques for normalising (scaling) the quality measures. There are also various measure aggregation schemes. Considering the results of our case study with the human experts, it is likely that a weighted aggregation scheme is most suitable as some measures are better indicators of hypothesis usefulness than others. The importance, i.e. weight, of each measure could be estimated empirically by running additional case studies using the methodology described in Section 9.4.2. The results would allow estimating the weight of a measure based on its correlation with interestingness and validity of hypotheses.

10.5.3 User Interaction Scenarios

In Section 8.4.2, we suggest several ways how an output of DL-Miner can possibly be used. To recap, a user can explore hypotheses using their ranking, sort them by a measure of choice, select most promising hypotheses via thresholds. Considering the results of our case study with the domain experts, it is not clear which way of interaction is most convenient. To be more specific, although we have revealed some good indicators of hypothesis validity and interestingness, they are not constant, i.e. they differ for the surveys. Therefore, it is hard to conclude which set of measures and, hence, which way of interaction should be used generally. Perhaps, a user should be given different options of interaction with hypotheses so that she can try them out and choose the one which suits

best for the given hypotheses and the task at hand. Further case studies can shed light on this question.

10.5.4 Ontology Completion and Debugging

Besides sequentially examining the acquired hypotheses, a user can use DL-Miner for *interactive ontology completion and debugging*. In the ontology completion scenario, a domain expert judges a certain number of hypotheses and annotates them respectively, e.g. using the validity marks as we do in our case study. She adds some hypotheses, e.g. all correct ones, to the ontology and reruns DL-Miner on the enriched ontology. This gives another set of hypotheses different from the initial one since the added hypotheses become uninformative and may affect the quality values of other hypotheses. Then, she repeats these steps for new hypotheses. She carries on until the ontology is sufficiently enriched, i.e. complete, that can be verified via standard techniques of ontology engineering, e.g. competency questions [UG96, Vra09].

As our case study with the domain experts shows, hypotheses, particularly wrong ones, can be useful for ontology debugging as they can reveal modelling errors. Similarly to the ontology completion scenario, DL-Miner can be used to debug the ontology interactively. More specifically, a user marks hypotheses that indicate some data bias or error worthwhile to fix. Then, she fixes those errors in the ontology and reruns DL-Miner on the fixed ontology. If the error-indicating hypotheses are not in the new set, the errors are fixed. Otherwise, she attempts new fixes and then repeats the test until all error indicators are removed. After that, she may explore and mark the hypotheses once again as fixing some errors might cause other errors.

It is also reasonable to combine ontology completion and debugging into one interactive scenario because enriching an ontology may reveal errors in it. To fix the errors, the ontology debugging scenario can be triggered. Once all errors are fixed, a user can proceed with ontology completion until she detects new errors that need to be fixed. The aforementioned scenarios are subjects of further case studies.

10.5.5 Comprehensible Predictors and Data Analysis Using Description Logics

In Section 9.4.1.1, we have introduced the problem of making predictions in DLs. By Definition 9.4, predictions are all new assertions in the data caused by hypotheses learned from it that act as prediction rules. Using this definition, we can straightforwardly formalise the binary classification problem in DLs as predicting instances of a given concept name A and its negation $\neg A$.

Nevertheless, transforming a set of hypotheses into a binary prediction model (classifier) is not straightforward. Firstly, a set can be too weak, i.e. insufficient to make any prediction for some individuals. Secondly, a set can be too strong (contradictory), i.e. can predict that some individuals are both A and $\neg A$, or clashes. The first problem does not occur in our case study due to completeness of DL-Miner which acquires the hypotheses $H_A := \{\top \sqsubseteq A\}$ and $H_{\neg A} := \{\top \sqsubseteq \neg A\}$. However, these hypotheses mutually imply that all individuals are clashes, i.e. the second problem always occurs.

In principle, we can always handle the first problem by adding the hypotheses H_A and $H_{\neg A}$ to the set of acquired hypotheses. Then, we can concentrate on the second problem which becomes a major issue as all individuals are affected. In this thesis we have proposed a simplified solution to this problem, see Section 9.4.1.1. In short, in order to resolve a clash, we identify hypotheses causing the clash and trust the prediction of the most confident hypothesis (taking its support into account). Of course, it is a simplification since multiple hypotheses can be necessary to make a prediction. In addition, multiple hypotheses can be considered to resolve the clash, e.g. via weighted voting. Moreover, clashes may be harder to resolve for multiple mutually exclusive concepts. Therefore, the problem requires further analysis and empirical investigations.

As it is pointed out in our case study on rice fertility prediction, making predictions is not the only use case for hypotheses acquired by DL-Miner. They are comprehensible since they are encoded in DLs and OWL which are originally designed to be readable and understandable. Hence, hypotheses of high quality can potentially be used by human experts to explore correlations in the data, understand phenomena, identify factors indicating accurate predictions, and, as a consequence, refine a prediction model, see Example 9.2. Further case studies are required to address this conjecture.

Bibliography

[ACKZ09] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-lite family and relations. *Journal of Artificial Intelligence Research*, 36(1):1–69, September 2009.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1995.

[Ang88] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[APS14] Tahani Alsubait, Bijan Parsia, and Uli Sattler. Measuring similarity in ontologies: A new family of measures. In *Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW'14)*, volume 8876 of *Lecture Notes in Computer Science*, pages 13–25, Linköping, Sweden, November 2014. Springer.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, Santiago de Chile, Chile, September 1994. Morgan Kaufmann.

[Bar14] Nick Barrowman. Correlation, causation, and confusion. *The New Atlantis*, (43):23–44, 2014.

[BCM+10] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2nd edition, 2010.

[BGSS07] Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. Completing Description Logic knowledge bases using formal concept analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07)*, pages 230–235, Hyderabad, India, January 2007. Morgan Kaufmann Publishers Inc.

[BHN+92] Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An empirical analysis of optimization techniques for terminological representation systems, or making KRIS get a move on. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 270–281, Cambridge, MA, October 1992. Morgan Kaufmann.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, Secaucus, NJ, USA, 2006.

[BK98] Franz Baader and Ralf Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proceedings of the 22nd Annual German Conference on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140, Bremen, Germany, September 1998. Springer.

[BL12] Lorenz Bühmann and Jens Lehmann. Universal OWL axiom enrichment for large knowledge bases. In *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*, volume 7603 of *Lecture Notes in Computer Science*, pages 57–71, Galway City, Ireland, 2012. Springer.

[BLW16] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. DL-Learner – a framework for inductive learning on the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 39:15–24, 2016.

[BNC00] Liviu Badea and Shan-Hwei Nienhuys-Cheng. A refinement operator for Description Logics. In *Proceedings of the 10th International*

Conference on Inductive Logic Programming (ILP'00), volume 1866 of *Lecture Notes in Computer Science*, pages 40–59, London, UK, 2000. Springer.

[BST07] Franz Baader, Baris Sertkaya, and Anni-Yasmin Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3):392–420, 2007.

[CDL+05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable Description Logics for ontologies. In *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference*, pages 602–607, Pittsburgh, Pennsylvania, USA, July 2005. AAAI/MIT Press.

[CH94] William W. Cohen and Haym Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 121–133, Bonn, Germany, May 1994. Morgan Kaufmann.

[CRH10] Philipp Cimiano, Sebastian Rudolph, and Helena Hartfiel. Computing intensional answers to questions – an Inductive Logic Programming approach. *Data and Knowledge Engineering*, 69(3):261–278, March 2010.

[CW94] Darrell Conklin and Ian H. Witten. Complexity-based induction. *Machine Learning*, 16(3):203–225, 1994.

[Deb01] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[Dis10] Felix Distel. An approach to exploring Description Logic knowledge bases. In *Proceedings of the 8th International Conference on Formal Concept Analysis (ICFCA'10)*, pages 209–224, Agadir, Morocco, March 2010. Springer.

[DLK+08] Pedro Domingos, Daniel Lowd, Stanley Kok, Hoifung Poon, Matthew Richardson, and Parag Singla. Just add weights: Markov Logic for the Semantic Web. In *Revised Selected and Invited Papers*

of Uncertainty Reasoning for the Semantic Web I, ISWC International Workshops, URSW 2005-2007, volume 5327 of *Lecture Notes in Computer Science*, pages 1–25. Springer, 2008.

[EPT15] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. Similarity-based relaxed instance queries. *Journal of Applied Logic*, 13(4):480 – 508, 2015.

[FDE08] Nicola Fanizzi, Claudia D'Amato, and Floriana Esposito. DL-FOIL concept learning in Description Logics. In *Proceedings of the 18th International Conference on Inductive Logic Programming (ILP'08)*, volume 5194 of *Lecture Notes in Computer Science*, pages 107–121, Prague, Czech Republic, 2008. Springer.

[FdE10] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Induction of concepts in web ontologies through terminological decision trees. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'10)*, volume 6321 of *Lecture Notes in Computer Science*, pages 442–457, Barcelona, Spain, September 2010. Springer.

[FV11] Daniel Fleischhacker and Johanna Völker. Inductive learning of disjointness axioms. In *Proceedings of the Confederated International Conference: On the Move to Meaningful Internet Systems (OTM'11)*, volume 7045 of *Lecture Notes in Computer Science*, pages 680–697, Hersonissos, Crete, Greece, October 2011. Springer.

[GH06] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3), September 2006.

[GHKS08] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, pages 273–318, 2008.

[GHM+08] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 6(4):309–322, 2008.

[GPS12] Rafael S. Gonçalves, Bijan Parsia, and Ulrike Sattler. Performance heterogeneity and approximate reasoning in Description Logic ontologies. In *Proceedings of the 11th International Semantic Web Conference (ISWC'12)*, volume 7649 of *Lecture Notes in Computer Science*, pages 82–98, Boston, MA, USA, November 2012. Springer.

[GT07] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[GTHS13] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International World Wide Web Conference (WWW'13)*, pages 413–422, Rio de Janeiro, Rio de Janeiro, Brazil, 2013. International World Wide Web Conferences Steering Committee / ACM.

[GW99] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis*. Springer, 1999.

[HBPS11] Matthew Horridge, Samantha Bail, Bijan Parsia, and Ulrike Sattler. The cognitive complexity of OWL justifications. In *Proceedings of the 10th International Conference on The Semantic Web (ISWC'11)*, volume 7031 of *Lecture Notes in Computer Science*, pages 241–256, Bonn, Germany, October 2011. Springer.

[HG93] Darrell Huff and Irving Geis. *How to Lie With Statistics*. W. W. Norton & Company, 1993.

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67, Lake District, UK, 2006. AAAI Press.

[Hor11] Matthew Horridge. *Justification Based Explanation in Ontologies*. PhD thesis, The University of Manchester, 2011.

[HPS08] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in OWL. In *Proceedings of the 7th International Semantic Web Conference (ISWC'08)*, volume 5318 of *Lecture Notes*

in Computer Science, pages 323–338, Karlsruhe, Germany, October 2008. Springer.

[HPS09] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explaining inconsistencies in OWL ontologies. In *Proceedings of the 3rd International Conference on Scalable Uncertainty Management (SUM'09)*, volume 5785 of *Lecture Notes in Computer Science*, pages 124–137, Washington, DC, USA, September 2009. Springer.

[HPVH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank Van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[HS05] Peter Haase and Ljiljana Stojanovic. Consistent evolution of OWL ontologies. In *Proceedings of the 2nd European Conference on The Semantic Web (ESWC'05)*, volume 3532 of *Lecture Notes in Computer Science*, pages 182–197, Heraklion, Crete, Greece, May 2005. Springer.

[IPF07] Luigi Iannone, Ignazio Palmisano, and Nicola Fanizzi. An algorithm based on counterfactuals for concept learning in the Semantic Web. *Applied Intelligence*, 26(2):139–159, April 2007.

[Jae97] Manfred Jaeger. Relational Bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI'97)*, UAI'97, pages 266–273, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[JT06] Tao Jiang and Ah-Hwee Tan. Mining RDF metadata for generalized association rules. In *Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)*, volume 4080 of *Lecture Notes in Computer Science*, pages 223–233, Kraków, Poland, September 2006. Springer.

[KKS11] Yevgeny Kazakov, Markus Krötzsch, and František Simancík. Concurrent classification of EL ontologies. In *Proceedings of the 10th International Conference on The Semantic Web (ISWC'11)*, volume 7031 of *Lecture Notes in Computer Science*, pages 305–320, Bonn, Germany, October 2011. Springer.

[KLOW14] Boris Konev, Carsten Lutz, Ana Ozaki, and Frank Wolter. Exact learning of lightweight Description Logic ontologies. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*, Vienna, Austria, July 2014. AAAI Press.

[LABT11] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9(1):71–81, 2011.

[Las08] Kathryn Blackmond Laskey. MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, 172(2–3):140–178, 2008.

[LH07] Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the $\mathcal{ALC}$ Description Logic. In *Proceedings of the 17th International Conference on Inductive Logic Programming (ILP'07)*, pages 147–160, Corvallis, OR, USA, June 2007.

[LH10] Jens Lehmann and Pascal Hitzler. Concept learning in Description Logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.

[Lis08] Francesca A. Lisi. Building rules on top of ontologies for the Semantic Web with Inductive Logic Programming. *Theory and Practice of Logic Programming*, 8(3):271–300, 2008.

[Llo87] John W. Lloyd. *Foundations of Logic Programming*. Springer, New York, NY, USA, 2nd edition, 1987.

[LV14] Jens Lehmann and Johanna Völker. *Perspectives on Ontology Learning*, volume 18 of *Studies on the Semantic Web*. IOS Press, 2014.

[Mic88] Donald Michie. Machine learning in the next five years. In *Proceedings of the 3rd European Working Session on Learning (EWSL'88)*, pages 107–122. Pitman, 1988.

[MP15] Nicolas Matentzoglu and Bijan Parsia. *BioPortal Snapshot 27.01.2015*. `https://doi.org/10.5281/zenodo.15667`, February 2015.

[MS01] Alexander Mädche and Steffen Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, March 2001.

[Mug91] Stephen Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.

[Mug95] Stephen Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3):245–286, 1995.

[NL10] Victoria Nebot and Rafael Berlanga Llavori. Mining association rules from Semantic Web data. In *Proceedings of the 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE'10)*, volume 6097 of *Lecture Notes in Computer Science*, pages 504–513, Cordoba, Spain, June 2010. Springer.

[NS13] Nadeschda Nikitina and Sven Schewe. Simplifying Description Logic ontologies. In *Proceedings of the 12th International Semantic Web Conference (ISWC'13)*, volume 8218 of *Lecture Notes in Computer Science*, pages 411–426, Sydney, Australia, October 2013. Springer.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[Qui90] John Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, September 1990.

[RD06] Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, February 2006.

[Rei87] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Rud04] Sebastian Rudolph. Exploring relational structures via $\mathcal{FLE}$. In *Proceedings of the 12th International Conference on Conceptual Structures (ICCS'04)*, volume 3127 of *Lecture Notes in Computer Science*, pages 196–212, Huntsville, AL, USA, July 2004. Springer.

[Rud08] Sebastian Rudolph. Acquiring generalized domain-range restrictions. In *Proceedings of the 6th International Conference on Formal Concept Analysis (ICFCA'08)*, volume 4933 of *Lecture Notes in Computer Science*, pages 32–45, Montreal, Canada, February 2008. Springer.

[Sal06] Wesley C. Salmon. *Four Decades of Scientific Explanation.* University of Pittsburgh Press, 2006.

[Saz17] Viachaslau Sazonau. *General Terminology Induction: Two Corpora of Experimental Ontologies.* `https://doi.org/10.5281/zenodo.291837`, February 2017.

[SCB14] Konstantinos Sechidis, Borja Calvo, and Gavin Brown. Statistical hypothesis testing in positive unlabelled data. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'14)*, volume 8726 of *Lecture Notes in Computer Science*, pages 66–81, Nancy, France, September 2014. Springer.

[Ser09] Baris Sertkaya. OntoComP system description. In *Proceedings of the 22nd International Workshop on Description Logics (DL'09)*, volume 477 of *CEUR Workshop Proceedings*, Oxford, UK, July 2009. CEUR-WS.org.

[SMH08] Rob Shearer, Boris Motik, and Ian Horrocks. HermiT: A highly-efficient OWL reasoner. In *Proceedings of the 5th Workshop on OWL: Experiences and Directions (OWLED'08)*, CEUR Workshop Proceedings, Karlsruhe, Germany, October 2008. CEUR-WS.org.

[Sob15] Elliott Sober. *Ockham's Razors: A User's Manual.* Cambridge University Press, 2015.

[SPG+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, June 2007.

[SSB14] Viachaslau Sazonau, Uli Sattler, and Gavin Brown. Predicting performance of OWL reasoners: Locally or globally? In *Proceedings*

of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14), Vienna, Austria, July 2014. AAAI Press.

[SSB15] Viachaslau Sazonau, Uli Sattler, and Gavin Brown. General Terminology Induction in OWL. In *Proceedings of the 14th International Semantic Web Conference (ISWC'15)*, volume 9366 of *Lecture Notes in Computer Science*, pages 533–550, Bethlehem, PA, USA, October 2015. Springer.

[STP07] Maytal Saar-Tsechansky and Foster Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8:1623–1657, December 2007.

[TH06] Dmitry Tsarkov and Ian Horrocks. FACT++ Description Logic reasoner: System description. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Computer Science*, pages 292–297, Seattle, WA, USA, August 2006. Springer.

[TSK13] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2013.

[UG96] Mike Uschold and Michael Gruninger. Ontologies: principles, methods and applications. *Knowledge Enginerring Review*, 11(2):93–136, 1996.

[VÖ9] Johanna Völker. *Learning Expressive Ontologies: Studies on the Semantic Web*, volume 2. IOS Press, Amsterdam, Netherlands, 2009.

[VL08] Paul M.B. Vitányi and Ming Li. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, 3rd edition, 2008.

[VN11] Johanna Völker and Mathias Niepert. Statistical schema induction. In *Proceedings of the 8th Extended Semantic Web Conference (ESWC'11)*, volume 6643 of *Lecture Notes in Computer Science*, pages 124–138, Heraklion, Crete, Greece, May 2011. Springer.

[Vra09] Denny Vrandecic. Ontology evaluation. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 293–313. Springer, 2009.

[VVSH07] Johanna Völker, Denny Vrandecic, York Sure, and Andreas Hotho. Learning disjointness. In *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, volume 4519 of *Lecture Notes in Computer Science*, pages 175–189, Innsbruck, Austria, June 2007. Springer.

[WMCM14] Paul Warren, Paul Mulholland, Trevor D. Collins, and Enrico Motta. The usability of Description Logics – understanding the cognitive difficulties presented by Description Logics. In *Proceedings of the 11th International Conference on The Semantic Web: Trends and Challenges (ESWC'14)*, volume 8465 of *Lecture Notes in Computer Science*, pages 550–564, Anissaras, Crete, Greece, May 2014. Springer.

[ZCGNH15] Yujiao Zhou, Bernardo Cuenca Grau, Yavor Nenov, and Ian Horrocks. PAGOdA: Pay-as-you-go ABox reasoning. In *Proceedings of the 28th International Workshop on Description Logics (DL'15)*, volume 1350 of *CEUR Workshop Proceedings*, Athens, Greece, June 2015. CEUR-WS.org.

[ZGP+13] Man Zhu, Zhiqiang Gao, Jeff Z. Pan, Yuting Zhao, Ying Xu, and Zhibin Quan. Ontology learning from incomplete Semantic Web data by BelNet. In *Proceedings of the 25th International Conference on Tools with Artificial Intelligence (ICTAI'13)*, pages 761–768, Herndon, VA, USA, November 2013. IEEE Computer Society.

[ZGP+15] Man Zhu, Zhiqiang Gao, Jeff Z. Pan, Yuting Zhao, Ying Xu, and Zhibin Quan. TBox learning from incomplete data by inference in BelNet+. *Knowledge-Based Systems*, 75:30–40, 2015.