Concept Similarity and Agent's Preferences in Description Logics: Computations and Applications

Teeradaj RACHARAK

Japan Advanced Institute of Science and Technology

Doctoral Dissertation

Concept Similarity and Agent's Preferences in Description Logics: Computations and Applications

Teeradaj RACHARAK

Supervisor: Professor Satoshi TOJO

School of Information Science
Japan Advanced Institute of Science and Technology

June, 2018

To my parents who have taught me the value of learning

Abstract

Concept similarity measure, as investigated in this thesis, aims at identifying a degree of commonality of two given concepts and is often regarded as a generalization of the classical reasoning problem of equivalence. That is, any two concepts are equivalent if and only if their similarity degree is one. We formally investigate this notion in Description Logics (DLs). Its results provide a basis for computational methods of identifying the commonalities and the discrepancies between two concepts. Our methods of concept similarity measure are proven to be tractable. To this end, they are thereby restricted to the DLs which do not provide all Boolean operators such as $\mathcal{FL}_0$ and $\mathcal{ELH}$ to avoid inheriting NP-hardness from propositional logic.

Similarity judgment used by human beings often involve preferences and needs in practice. More specifically, when two concepts are not logically equivalent or totally similar, they may rely on subjective factors *e.g.* the agent's preferences. Here, we formally define a formal notion of concept similarity under such subjective factors called *concept similarity measure under preference profile* and identify a set of its desirable properties. These properties relate to the question "what could be good preference-based similarity measures?". To exemplify the developments, we suggest computational techniques for $\mathcal{FL}_0$ and $\mathcal{ELH}$, and also, prove their inherited properties. Two algorithmic procedures for our developed measure sim^{π} are introduced for the top-down and bottom-up implementations, respectively, and their computational complexities are intensively studied. We also discuss the usefulness of our proposed developments to potential use cases.

Analogical reasoning is a complex process based on a comparison between two pairs of concepts or states of affairs (*aka.* the source and the target) for characterizing certain features from the source to the target. To exploit our results of concept similarity measure, we investigate such kind of reasoning that analogical conclusions can be derived from the similarity between DL concepts. Two approaches for the implementation of analogical reasoning are explored. Each is formulated from the study of philosophical understanding called *argumentation schemes* where patterns of non-deductive reasoning are analyzed. Finally, we demonstrate that the analogical argument used in the case of Silkwood v. Kerr-McGee Corporation is reconstructible from the proposed formalisms.

Keywords: Concept Similarity Measure, Semantic Web Ontology, Preference Profile, Description Logics, Analogical Reasoning

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Satoshi Tojo, for his guidance, support, patience, and encouragement throughout my PhD course. His technical and editorial advice was essential to the completion of this thesis. He has taught me innumerable lessons on academic researching. I am and will always be most appreciative of his kindness to help me learn, grow, and move forward.

Grateful acknowledgement is extended to Dr. Nguyen Duy Hung and Dr. Boontawee Suntisrivaraporn for their supports and encouragement when I have to conduct this research at Sirindhorn International Institute of Technology, Thammasat University, Thailand according to JAIST-NECTEC-SIIT dual doctoral degree program. Cordial gratitude is continued to Dr. Katsuhiko Sano for his teaching on the foundation of mathematical logic. The knowledge I have learned from him provides me the basis on logical thinking, which is important to work on this research. In addition, I am deeply indebted Dr. Prachya Boonkwan for his kind proofreading on several of our publications. His suggestions and correction of writing style have improved my writing skill.

The sincere friendship of the members of Tojo laboratory and Nguyen laboratory are much appreciated and have led to many interesting discussions during my life at JAIST.

I am also obliged to JAIST-NECTEC-SIIT dual degree program for the scholarship which facilitated my study at JAIST.

Last but not least, I would like to express my warmest appreciation to my family, Mr. Boontham Racharak and Mrs. Supatra Racharak, for their love, care, patience, understanding, and spiritual supports.

Table of Contents

This dissertation was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and Sirindhron International Institute of Science and Technology, Thammasat University.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is an Ontology?

The word *ontology* is used by different communities under different interpretations. Generally speaking, it can be classified based on two ways of the usage [1], *i.e.* as an uncountable noun and as a countable noun. The former one appears in the field of philosophy. It is worth noting that Aristotle first defined Ontology as the study of attributes that belong to things because of their very nature [2]. This interpretation may not require an existence of actual realities. For instance, ones may study the Ontology of pegasi and other fictitious entities.

On the other hand, the latter one (*i.e.* an ontology) may mean a special kind of information object or computational artifact. In other words, an ontology is referred to a set of relevant entities and relations reflecting an observation. This interpretation prominently appears in computer science. For example, ones may model a human resource department by consisting of three entities, *viz.* Person, Manager, and Researcher. It is worth noting that entities may be called *concepts*. The backbone of an ontology is a generalization/specialization hierarchy of concepts. Hence, it makes perfect sense to say that Person is a super-concept of Manager and Researcher. Also, cooperativeWith may be drawn as a relation between persons. Each concrete person working in the department is considered as an instance of each corresponding concept.

In 1993, Gruber [3] originally defines the notion of an ontology as an explicit specification of a conceptualization. Later, it is redefined by Borst [4] that an ontology is a formal specification of a shared conceptualization. There are two assumptions from this definition: **(1)** the conceptualization should express a shared view among parties (rather than an individual view) and **(2)** the conceptualization should be expressed in a formal machine readable format. In 1998, Studer *et al.* [5] merge these two definitions, *i.e.* an ontology is "a formal, explicit specification of a shared conceptualization". As a computer scientist, this thesis perceives the notion of an ontology in this intuition.

1.2 Description Logics as Ontology Language

Description Logics (DLs) [6–8] are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. Based on these characteristics, DLs are well-suited for an ontology representation language. The name *description logics* is coined based on the fact that the application domain is described by concept *descriptions*, *i.e.* expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. Furthermore, DLs differ from their predecessors (*e.g.* semantic networks and frames) in the way that DLs are equipped with a formal and *logic*-based semantics.

For the interested reader with some background in mathematical logic, we will relate examples to First Order Logic (FOL) in square brackets. DLs are characterized by the constructors they provide to build complex concepts from atomic ones. Formal definitions are given in Chapter 3. Let us illustrate that a concept of "an inflammation that has location on endocardium tissue" can be expressed as follows:

$$\mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Endocardium}$$

$[\mathsf{Inflammation}(x) \wedge \exists y(\mathsf{hasLocation}(x,y) \wedge \mathsf{Endocardium}(y))]$ where Inflammation and Endocardium are atomic concepts. Also, hasLocation is an atomic role and $\sqcap, \exists$ are concept constructors. We note that an atomic role relates an instance of a concept to another.

In addition to this description formalism, DLs are also equipped with a terminological formalism and an assertional formalism. Generally, terminological axioms can be used to: **(1)** introduce an abbreviation of a complex concept (for stating the "necessary and sufficient" conditions of a concept), *e.g.*

$$\mathsf{Endocarditis} \equiv \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Endocardium}$$

$[\forall x(\mathsf{Endocarditis}(x) \leftrightarrow \mathsf{Inflammation}(x) \wedge \exists y(\mathsf{hasLocation}(x,y) \wedge \mathsf{Endocardium}(y)))]$ meaning that an "endocarditis is definitely an inflammation that has location on endocardium tissue"; **(2)** state the "necessary" conditions for being a particular concept[1], *e.g.*

$$\mathsf{Inflammation} \sqsubseteq \mathsf{Disease} \sqcap \exists \mathsf{actsOn}.\mathsf{Tissue}$$

$[\forall x(\mathsf{Inflammation}(x) \rightarrow \mathsf{Disease}(x) \wedge \exists y(\mathsf{actsOn}(x,y) \wedge \mathsf{Tissue}(y)))]$ meaning that an "inflammation is a disease that acts on a tissue". Or, **(3)** it may be used to supplement a constraint like "vitamin K1 has a function as a catalyst":

$$\mathsf{VitaminK1} \sqsubseteq \exists \mathsf{hasFunction}.\mathsf{Catalysing}$$

$[\forall x(\mathsf{VitaminK1}(x) \rightarrow \exists y(\mathsf{hasFunction}(x,y) \wedge \mathsf{Catalysing}(y)))]$. Another example of constraint statements is "$\exists \mathsf{hasChild}.\mathsf{Human} \sqsubseteq \mathsf{Human}$" $[\forall x(\exists y(\mathsf{hasChild}(x,y) \rightarrow \mathsf{Human}(x)))]$

[1] Sometimes, we might know some necessary (but not sufficient) conditions for membership.

saying that only humans can have human children. Furthermore, terminological axioms can be used to state role inclusion statements such as:

$$\mathsf{hasExactLocation} \sqsubseteq \mathsf{hasLocation}$$

$[\forall x \forall y(\mathsf{hasExactLocation}(x, y) \rightarrow \mathsf{hasLocation}(x, y))]$ which states that having an exact location implies that having a location. A set of terminological axioms is called a TBox.

The assertional formalism can be used to state properties of an individual. This intuitive idea coincides with ground facts in FOL. For instance, one may assert that "*bob* is taking antifungal antibiotics":

$$\mathsf{onMedication}(\mathit{bob}, \mathit{anonym1}); \mathsf{AntifungalAntibiotics}(\mathit{anonym1})$$

A set of assertional axioms is called an ABox. In general, an ontology can be represented by a set of terminological axioms and a set of assertional axioms.

The suitability of DLs as ontology representation languages is not only caused by their well-defined languages; but also, is caused by their ability to support various forms of reasoning services. Reasoning is important to ensure the quality of an ontology and to make implicit information in an ontology explicit. For instance, any sound DL reasoners should infer that an endocarditis is a disease from the previous example ontology.

Reasoning can be employed in different phases of an ontology life cycle. For instance, it may used during an ontology design to test whether a concept is non-contradictory or to obtain explicit relationship. In particular, one usually wants to compute the concept hierarchy, *i.e.* the partial ordering of named concepts based on the subsumption relationship. Information on which concept is a specialization of another, and which concepts are synonyms, can be used in the design phase to test whether axioms described in the ontology have intended consequences [9]. Though this subsumption hierarchy inevitably benefits ontology modeling, it merely gives binary responses, *i.e.* inferring a concept is subsumed by another concept or not. Certain pairs of concepts may share commonality even though they are not subsumed. This leads to an amount of research effort on measuring *concept similarity*.

It is worth noting that applications may not use the same ontology. Thus, the issue of ontology integration can be benefited from reasoning services as well. Integration may be supported by asserting inter-ontology relationships. After that, one might want to compute the integrated concept hierarchy or find similar pairs of integrated concepts. Missing subsumption relationships or unsound concept similarity may be a sign of incorrect or incomplete inter-ontology assertions, which should be corrected or completed by a knowledge engineer.

Reasoning can be also employed at the deployment phase. For example, when searching for an information stored in an ontology, it can be useful to consider not only perfect matches; but also, matches with respect to more general/specific terms or similar terms. However, the requirement on the efficiency of reasoning (*e.g.* the practical performance) should be much more strict than in the design phase and the integration phase.

On the availability of a well-defined semantics and the reasoning support, DLs are ideal candidates for ontology representation languages. For instance, OIL [10], DAML+OIL

[11, 12], OWL [13], and OWL 2 [14] are all based on DLs. The advantage of this close connection is that the basic research in DLs and implementation experiences can be directly exploited. Among these languages, OWL and OWL 2 are adopted as the standard (W3C recommendation) for ontology representation languages. There are three sub-languages of OWL (OWL 2), *viz.* OWL Lite (OWL 2 Lite), OWL DL (OWL 2 DL), and OWL Full (OWL 2 Full). OWL DL and OWL 2 DL are basically DLs. Hence, DLs are the logical underpinnings of the DL flavor of OWL and OWL 2. OWL 2 also has three profiles, *viz.* OWL 2 EL, OWL 2 QL, and OWL 2 RL.

1.3 Research in Description Logics

Following [9, 15], we distinguish the development in DLs into five phases as follows.

Phase 0 (1965 - 1980)

This phase is known as the pre-DL phase in which the so-called *semantic networks* [16] and *frame* [17] were introduced. Both were specialized approaches for representing knowledge in a structured way; however, they were criticized due to their lack of formal semantics [18–21]. An approach to overcome this problem was Brachman's *structured inheritance networks* [22]. This formalism hugely contributed to the next development. Indeed, it was realized by the first DL system, *i.e.* KL-ONE.

Phase 1 (1980 - 1990)

This phase is mainly concerned with the implementation of DL systems, such as KL-ONE [23], K-REP [24], and KRYPTON [25]. These systems employed a so-called *structural subsumption algorithm*, in which concept descriptions were normalized for recursively comparing their syntactic structures. On the one hand, this algorithm is very efficient (*i.e.* in polynomial time). On the other hand, they are limited to inexpressive DLs, *i.e.* they cannot detect all subsumption/instance relationships for more expressive DLs. There were formal investigations into the complexity of reasoning in DLs during this phase. For example, it was shown in [26] that adding small expressive power for the representation might cause intractability for the subsumption problem. It was also shown in [27] that using abbreviations for concept descriptions in TBox might make subsumption intractable if the underlying DL used the constructors conjunction and value restriction. Unfortunately, these constructors were supported by DL systems during this phase. According to this negative complexity results, the implementors of CLASSIC (the first industrial-strength DL system) carefully restricted their DL [28, 29].

Phase 2 (1990 - 1995)

This phase involves in an introduction of a new algorithmic paradigm called *tableau-based algorithms* [30, 31]. They work on propositionally closed DLs (*i.e.* DLs with all Boolean operators) and are also usable with expressive DLs. A tableau-based algorithm works as

follows in order to decide the consistency of a knowledge base: **(1)** it tries to construct a model by structurally decomposing the concepts in the knowledge base; **(2)** it infers new constraints on the elements of this model; **(3)** it stops either when all attempts to build a model failed with obvious contradictions or when reaching a canonical model. This phase also investigates the complexity of reasoning in various DLs [31–33] and observes that DLs are closely related to modal logics [34].

Phase 3 (1995 - 2000)

This phase is characterized by the development of inference procedures for very expressive DLs, either based on the *tableau-based approach* [35,36] or a translation into modal logics [37–40]. Highly optimized systems, *e.g.* FaCT [41] and RACE [42], showed that tableau-based algorithms for expressive DLs performed a good practical performance even on some large knowledge bases. This phase also studies the relationship to modal logics [37,43], the relationship to decidable fragments of First Order Logic [44–48], and the applications in databases (*e.g.* schema reasoning, query optimization, and database integration) [49–51].

Phase 4 (2000 - 2005)

Industrial strength DL systems employing very expressive DLs and tableau-based algorithms (*e.g.* [28, 42, 52]) were developed with applications like the Semantic Web, knowledge representation and integration in medical and bio-informatics systems in mind. In this phrase, the sub-languages of OWL *viz.* OWL DL and OWL Lite became an official W3C recommendation[1]. Furthermore, other approaches were also investigated. For example, ones might employ optimized translation procedures for converting DLs into first-order predicate logic and then applied appropriate first-order resolution provers (*cf.* [53–57]). Another foundation investigation could be found in [8, 55, 58–61] where automata-based approaches were used to show ExpTime complexity upper-bounds.

Phase 5 (2005 - onward)

We are currently in this phase. Here, more expressive DLs with highly-optimized tableau-based methods were proposed *e.g.* the approach in [62] as a basis for the new Web Ontology Language OWL 2[2]. More light-weight DLs were also investigated and were proposed as profiles of OWL 2[3]. For instance, the $\mathcal{EL}$ family [63,64] in which the subsumption and the instance problems can be computed in polynomial time. The DL Lite family [65,66] offers polynomial-time algorithms for instance checking in the size of the ABox.

Despite the higher efficiency of available DL systems, they are not capable of providing good explanations for their main uses. In other words, they are good from the viewpoint of providing yes/no answers. According to this problem, some work [67,68] has described methods to extract explanations from the tableau-based algorithms. Moreover, proof

[1]https://www.w3.org/TR/owl-features/
[2]https://www.w3.org/TR/owl2-overview/
[3]https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

theory was investigated in [69] in which Sequent Calculi and Natural Deduction for some DLs were introduced and were shown to improve the extraction for explanation purpose.

Other problems rather than the classical ones (*i.e.* subsumption checking, instance checking, and consistency checking) have started to gain interest. For instance, query answering (*cf.* [70–73]) aims at answering conjunctive queries w.r.t. DL knowledge base. Axiom pinpointing (*cf.* [24, 43, 74–76]) enables finding the axioms which are responsible for a given consequence. Given a set of desired consequences, modularization (*cf.* [77–79]) extracts a part of DL knowledge base which infers such consequences. Other non-classical problems include least common subsumers, most specific concepts, and concept similarity which is belonged to our primary interest of this thesis. We review the current issues on concept similarity in DLs in the next section.

1.4 Two Issues on Concept Similarity in Description Logics

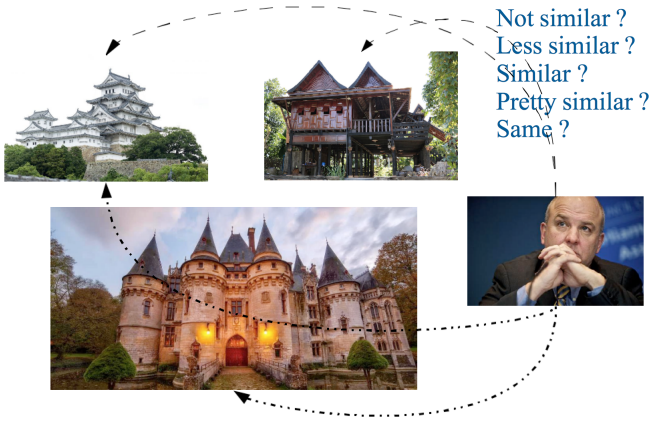


Figure 1.1: Concept similarity in general sense

Concept similarity refers to human judgments of a degree to which a pair of concepts in question is similar. Figure 1.1 depicts the intuitive understanding of this notion. The figure illustrates two possible scenarios where an agent is judging the similarity between the Japanese castle and the Thai-styled house (see the fine-dashed arrows) as well as the Japanese castle and the European castle (see two-dots-one-dash arrows). As seen in the figure, the agent may assign a value from a range (*viz.* not similar, less similar, similar, pretty similar, and same) reflecting his/her judgment to each pair (*i.e.* the Japanese castle vs. the Thai-styled house and the Japanese castle vs. the European castle).

Measures of concept similarity are computational approaches attempting to imitate the human judgments of concept similarity. This subject has been widely studied in many years and is central to functioning of many techniques *e.g.* ontology matching and ontology learning. However, it is still hindered by the following issues.

Issue 1: Formal Development of Concept Similarity in DLs

Concept similarity is widely studied in various fields, *e.g.* psychological science, computer science, artificial intelligence, and linguistic literature. For example, Tversky [82] studied a feature model where common and distinguishing features were used to derived the degree of similarity. The traditional edge-based approach estimates the degree of similarity from the distance/edge length between nodes, *e.g.* as in [83,84]. Or, similarity may be stemmed from the similarity between documents as appeared in natural language processing [85,86]. It is worth observing that, with a few exceptions like [87, 88], those approaches usually ignore the ontological constraints and relationship defined in an ontology.

This thesis deliberately investigates well-designed approaches for developing measures of concept similarity in DLs. A well-defined notion of concept similarity is introduced and concrete measures based on the notion are developed according to our defined notion.

It is worth mentioning here that "concept similarity" developed in this work should not be confused with "the logic for concepts and similarity" considered in [80, 81]. Precisely, we concentrate on how similar two concepts of an ontology is whereas they rather focus on more expressive description languages toward similarity.

Issue 2: Adaptability with Agent's Preferences

When two concepts in question are not *totally similar*, the degree of similarity may vary depending on subjective factors (*e.g.* the agent's preferences). This issue can be illustrated in the following example.

Example 1.1. (Based on Figure 1.1) It is reasonable to conclude that the Japanese castle is totally different to the Thai-styled house and the Japanese castle is also totally different to the European castle. These situations may happen if the agent considers merely the style of each construction, *i.e.* the agent makes judgment based on his/her needs and preferences. Particularly, the usage of preferences typically depends on target applications. For instance, if the agent is finding a building using castle-styled architecture, the degree between the Japanese castle and the European castle should be higher than the degree between the Japanese castle and the Thai-styled house. Using the needs and preferences for judgment usually appear in domain-specific knowledge base and the development of recommendation systems based on the agent's preferences.

1.5 Objectives

The primary objective of the thesis is to provide well-designed techniques for solving the two issues. This goal is further developed into the following objectives:

1. To propose well-defined notions of concept similarity in DLs. These well-defined notions can be divided into two parts, *i.e.* the basic notion of concept similarity and its extended notion for concept similarity under the agent's preferences;

2. To develop concrete measures based on the proposed notions. We restricts our attention on traceable measures. Thus, sub-Boolean DLs *viz.* $\mathcal{FL}_0$ and $\mathcal{ELH}$ are considered for our concrete developments; and

3. To demonstrate potential applications of our proposed measures.

To fulfill these objectives, this thesis makes the following main contributions:

1. The well-defined notion of concept similarity measure in DLs, which is defined as a function mapping from a concept pair to a unit interval ($0 \leq x \leq 1$ for any real number x), and also, the corresponding concrete measures *e.g.* $\sim^s$ (*cf.* Chapter 4);

2. The identification of preferential aspects (called *preference profile*) relevant to concept similarity measure in DLs and the refined notion called *concept similarity measure under preference profile* in DLs, which is defined as a function mapping from a concept pair under preference profile π to a unit interval (*cf.* Chapter 5);

3. The well-designed concrete measures $\overset{\pi}{\sim}s$ and sim^π for the DLs $\mathcal{FL}_0$ and $\mathcal{ELH}$, respectively, and the proofs of their underlying properties (*cf.* Chapter 5);

4. Two algorithmic procedures for implementing the measure sim^π and the empirical evaluation w.r.t. realistic ontologies, for example, showing how the proposed measure yields more intuitive results in the medical ontology SNOMED CT than using other measures (*cf.* Chapter 5);

5. The discussion of potential applications in knowledge engineering, and also, the developed formalisms for inferring conclusions from analogy which are based on the proposed concept similarity measure under preference profile (*cf.* Chapter 5 - 6);

1.6 Thesis Structure

The remainder of the thesis is organized as follows:

Chapter 2 briefly summarizes the background in predicate logic, particularly first-order logic. It basically introduces three basic elements of logic *viz.* syntax, semantics, and proof theory. It also introduces semantic tableau as a proof system which can be be used to prove if a formula is a logical consequence of a set of formulae. In this chapter, we mention that first-order logic is not decidable; hence, it is not appropriate to construct knowledge base systems based on full first-order logic. According to this negative result, decidable subsets of first-order logic are studied in the framework of Description Logics.

Chapter 3 is dedicated to most of preliminaries about Description Logics that are frequently referred by other chapters. It introduces the basics of Description Logics, reasoning services, the most commonly used reasoning techniques, in particular tableau-based and structural subsumption algorithms, and the computational complexity of important basic reasoning services.

Chapter 4 discusses the problem of *concept similarity* and the existing approaches dealing with the problem. It should be noted that the problem of concept similarity is not new and has been widely studied in various domains. However, they still lack in formal development as discussed in Section 1.4. In this chapter, we revisit and re-define in a more formal way for DLs. Concretely, a problem of concept similarity can be seen as a generalization of concept equivalence and we denote the notion by $\sim$. With this viewpoint, $\sim$ can be seen as a function which maps two equivalent concepts to 1 and totally dissimilar concepts to 0. Next, concrete measures of $\sim$ for sub-Boolean DLs, *viz.* $\mathcal{FL}_0$ and $\mathcal{ELH}$, are investigated and their underlying properties are discussed. In contrast to expressive DLs, sub-Boolean DLs are inherently tractable by nature. This fact has motivated us to take a look into them closely in the thesis, especially how ones can generalize the notion of concept equivalence for developing concrete measures. This chapter is mainly summarized from our published work [89, 90].

Similarity measures might be personalized when they are applied for agent-based situations such as recommendation systems. Selecting an appropriate measure is one way of personalizing. In chapter 5, we present an alternative approach by generalizing $\sim$ w.r.t. the notion of the agent's preferences. To achieve this, we propose a formal development of preferential aspects called *preference profile* (denoted by π) which can play an important role in concept similarity under the agent's preferences. A refined notion called *concept similarity under preference profile* (denoted by $\overset{\pi}{\sim}$) is developed by equipping $\sim$ with π. Intuitively, $\overset{\pi}{\sim}$ is a general notion which maps a concept pair under a preference profile π to a unit interval. To develop concrete measures of $\overset{\pi}{\sim}$, previously developed measures of $\sim$ are investigated and are generalized w.r.t each aspect of preference profile. As a result, the measure $\sim^s$ is generalized to $\overset{\pi}{\sim} s$ for the DL $\mathcal{FL}_0$ and the measure sim is generalized to sim^π for the DL $\mathcal{ELH}$. Several underlying properties *e.g.* the computational complexity and desirable properties are investigated. This chapter also investigates two algorithmic procedures for the measure sim^π (*cf.* Appendix B) and empirical evaluation is carried out w.r.t. realistic ontologies such as the medical ontology SNOMED CT (*cf.* Appendix A). This chapter is mainly summarized from our published work [90–94]

Chapter 6 considers the problem of analogical reasoning, which is one of potential applications of the proposed measure. Roughly, analogical reasoning is a form of non-deductive reasoning in which a conclusion is inferred based on the similarity of concepts or states of affairs. This chapter proposes two different approaches which are based on the underlying principle *i.e.* the argumentation scheme for argument from analogy. The first approach combines answer set programming (*cf.* Subsection 6.2.1) with concept similarity measure under preference profile whereas the second approach introduces another formalism of argument-based logic programming which considers three types of rules *viz.* strict, defeasible, and similarity rules. Analogical reasoning is often used by human beings in real-life situations, especially when humans encounter an unseen situation. In this chapter, we demonstrate that realistic arguments which have been supported by analogies between concepts can be reconstructed from our proposed formalisms. There are extremely interesting relations between analogical reasoning and defeasible argumentation (*cf.* Appendix C). We have covered some of them in this chapter. This chapter is mainly summarized

from our published work [95, 96].

Chapter 7 reviews the work presented and the extent to which the stated objectives have been met. The significance of the major results is summarized, and perspectives for further research are sketched.

Chapter 2

Preliminaries: Predicate Logic

In this chapter, we briefly describe the basics of predicate logic, particularly *first-order logic*. Most of the content in this chapter is summarized from [97].

2.1 Quantifiers

Predicate logic differs from propositional logic[2] in a sense that means of talking about objects in a domain are introduced. For example, statements of the form "all even numbers are a sum of two odd primes" can be impractical to formulate in the language of propositional logic *e.g.* $\varphi \to \sigma$ and could have no reasons why it must be true. Hence, predicate logic allows to use *variables* for ranging over objects and quantifiers *viz.* $\forall$ and $\exists$ (for "for all" and "there exists", respectively). We give a few examples as follows:

- $\exists x P(x)$ represents "there is an x with property P";
- $\forall y P(y)$ represents "for all y P holds";
- $\forall \epsilon(\epsilon > 0 \to \exists n(\frac{1}{n} < \epsilon))$ represents "for all positive ϵ there is an n such that $\frac{1}{n} < \epsilon$.

In this chapter, variables are ranged over elements of a given universe but not over properties, relations, or properties of properties, *etc.*. In this way, the predicate logic discussed in this chapter is called first-order logic (or elementary logic).

2.2 Structures

Definition 2.1. A *structure* is an ordered sequence $\langle A, R_1, \ldots, R_n, F_1, \ldots, F_m, \{c_i | i \in I\}\rangle$, where A is a non-empty set, $R_1, \ldots, R_n$ are *relations* on A, $F_1, \ldots, F_m$ are *functions* on A, the $c_i (i \in I)$ are elements of A (*constants*).

Structures are often denoted by Gothic capitals: $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}, \ldots$. As for examples, the field of real numbers is represented by $\langle \mathbb{R}, +, \cdot, ^{-1}, \{0, 1\}\rangle$ and the ordered set of natural

[2] `https://en.wikipedia.org/wiki/Propositional_calculus`

numbers is represented by $\langle \mathbb{N}, < \rangle$. However, it is more traditional to write down the constants instead of the set of constants *e.g.* $\langle \mathbb{R}, +, \cdot, ^{-1}, 0, 1 \rangle$ instead of $\langle \mathbb{R}, +, \cdot, ^{-1}, \{0, 1\} \rangle$.

If we overlook the special properties of the relations and operations (*e.g.* commutativity of addition on the reals), then what remains is the type of a structure, which is given by the number of relations, functions (or operations), and their respective arguments, plus the cardinality of constants. We define this generalization as follows.

Definition 2.2. The *similarity type* of a structure $\mathfrak{A} = \langle A, R_1, \ldots, R_n, F_1, \ldots, F_m, \{c_i | i \in I\}$ is a sequence $\langle r_1, \ldots, r_n; a_1, \ldots, a_m; \kappa \rangle$ where $R_i \subseteq A^{r_i}$, $F_j : A^{a_j} \to A$, $\kappa = |\{c_i | i \in I\}|$ (cardinality of I).

For example, the similarity types of the previous two examples are $\langle -; 2, 2, 1; 2 \rangle$ and $\langle 2; -; 0 \rangle$, respectively. We call A a *universe* of $\mathfrak{A}$. If $R \subseteq A$, then we call R a property (or unary relation). If $R \subseteq A^2$, then we call R a binary relation. If $R \subseteq A^n$, then we call R an n-ary relation.

Considering the limiting cases of relations and functions *i.e.* 0-ary relations and functions. An 0-ary relation is a subset of $A^\emptyset$. Since $A^\emptyset = \{\emptyset\}$, there are two such relations *viz.* $\emptyset$ and $\{\emptyset\}$ (considered as ordinals: 0 and 1). These can be also seen as truth values, which plays the role of the interpretation of propositions. On the other hand, 0-ary functions can play the role of constants. A 0-ary function is a mapping from $A^\emptyset$ into A *i.e.* a mapping from $\{\emptyset\}$ into A. Since the mapping has a singleton as domain, it can be identified with the range.

2.3 The Language of a Similarity Type

Let us fix the similarity type $\langle r_1, \ldots, r_n; a_1, \ldots, a_m; \kappa \rangle$ (where $r_i \geq 0$ and $a_j > 0$) in this section for convenience. The alphabet consists of the following symbols:

1. Predicate symbols: $P_1, \ldots, P_n, \stackrel{\circ}{=}$
2. Function symbols: $f_1, \ldots, f_m$
3. Constant symbols: $\overline{c_i}$ for $i \in I$
4. Variables: $x_0, x_1, x_2, \ldots$ (countably many)
5. Connectives: $\vee, \wedge, \to, \neg, \leftrightarrow, \bot, \forall, \exists$
6. Auxiliary symbols: $(,), ,$

where $\vee, \wedge, \to, \neg, \leftrightarrow, \bot$ are as defined for the propositional logic and $\forall, \exists$ are read as the *universal* and *existential quantifier*. In the following, we define two syntactical categories.

Definition 2.3. TERM is the smallest set X with the properties:

1. $\overline{c_i} \in X (i \in I)$ and $x_i \in X (i \in \mathbb{N})$,

2. $t_1, \ldots, t_{a_i} \in X \Longrightarrow f_i(t_1, \ldots, t_{a_i}) \in X$, for $1 \leq i \leq m$.

TERM is our set of terms.

Definition 2.4. FORM is the smallest set X with the properties:

1. $\bot \in X$; $P_i \in X$ if $r_i = 0$; $t_1, \ldots, t_{r_i} \in \text{TERM} \Longrightarrow P_i(t_1, \ldots, t_{r_i}) \in X$; $t_1, t_2 \in \text{TERM} \Longrightarrow t_1 \stackrel{\circ}{=} t_2 \in X$,
2. $\varphi, \psi \in X \Longrightarrow (\varphi \square \psi)$, where $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$,
3. $\varphi \in X \Longrightarrow (\neg\varphi) \in X$,
4. $\varphi \in X \Longrightarrow ((\forall x_i)\varphi, (\exists x_i)\varphi) \in X$.

FORM is our set of formulae. The formulae introduced in 1. are called *atoms*. The case of 0-ary predicate symbols (also introduced in 1.) are called *proposition symbols*.

As discussed in Section 2.2, a proposition symbol can be seen as a 0-ary relation *i.e.* either 0 or 1. This also corresponds to the practice of propositional logic for interpreting a proposition as either true or false. We also allow a special proposition $\bot$ for the false proposition.

As for the logical connectives, there are the basic propositional logical connectives *viz.* $\wedge, \vee, \rightarrow, \leftrightarrow$ and the newly introduced quantifiers *viz.* $\forall, \exists$. They can be used to form new formulae. To indicate the scope of a quantifier, we find the matching brackets *i.e.* given $((\forall x)\varphi)$ and $((\exists x)\varphi)$, we say that φ is the "scope" of a quantifier.

Properties of terms and formulae are established by inductive procedures *i.e.* first we deal with the atoms and then we proceed to deal with the composite parts. These procedures can be used to prove properties of terms and formulae.

Lemma 2.1. Let $A(t)$ be a property of terms. If $A(t)$ holds for variable of constant t, and if $A(t_1), A(t_2), \ldots, A(t_n) \Longrightarrow A(f(t_1, \ldots, t_n))$, for all function symbol f, then $A(t)$ holds for all $t \in \text{TERM}$.

Lemma 2.2. Let $A(\varphi)$ be a property of formulae. If

1. $A(\varphi)$ for atomic φ,
2. $A(\varphi), A(\psi) \Longrightarrow A(\varphi \square \psi)$,
3. $A(\varphi) \Longrightarrow A(\neg\varphi)$,
4. $A(\varphi) \Longrightarrow A((\forall x_i)\varphi), A((\exists x_i)\varphi)$ for all i, then $A(\varphi)$ holds for all $\varphi \in \text{FORM}$.

To improve readability, we may sometimes delete the outer brackets around $\forall x$ and $\exists x$ and join strings of quantifiers *e.g.* $\forall x_1 x_2 \exists x_3 x_4$.

Example 2.1. Example of a language of type $\langle 2; 2, 1; 1\rangle$. Let $L, \stackrel{\circ}{=}$ be predicate symbols, p, i be function symbols, and $\overline{e}$ be a constant symbol. Then, $x_0, p(x_1, x_2), p(\overline{e}, \overline{e}), i(x_7)$ are examples of terms. Furthermore, $(x_0 \stackrel{\circ}{=} x_1 \rightarrow x_1 \stackrel{\circ}{=} x_0), \forall x_0 \forall x_1 (x_0 \stackrel{\circ}{=} x_1 \rightarrow \neg L(x_0, x)1))$ are examples of formulae. $\square$

The value of a term and a formula is uniquely determined by the values of its parts. This allows us to find its value in finitely many steps.

Definition 2.5 (Recursion on TERM)**.** Let *Var* and *Const* be a set of variables and constant symbols, respectively. Let A be a universe, $H_0 : \mathit{Var} \cup \mathit{Const} \to A$, $H_i : A^{a_i} \to A$, then there is a unique mapping $H : \mathrm{TERM} \to A$ such that:

- $H(t) = H_0(t)$ for variable or constant t,
- $H(f_i(t_1, \ldots, t_{a_i})) = H_i(H(t_1), \ldots, H(t_{a_i}))$.

Definition 2.6 (Recursion on FORM)**.** Let At be a set of atoms and A be a universe. Let $H_{at} : At \to A$, $H_\square : A^2 \to A$ (where $\square \in \{\vee, \wedge, \to, \leftrightarrow\}$), $H_\neg : A \to A$, $H_\forall : A \times \mathbb{N} \to A$, $H_\exists : A \times \mathbb{N} \to A$. Then, there is a unique mapping $H : \mathrm{FORM} \to A$ such that:

- $H(\varphi) = H_{at}(\varphi)$ for atomic φ,
- $H(\varphi \square \psi) = H_\square(H(\varphi), H(\psi))$,
- $H(\neg\varphi) = H_\neg(H(\varphi))$,
- $H(\forall x_i \varphi) = H_\forall(H(\varphi), i)$,
- $H(\exists x_i \varphi) = H_\exists(H(\varphi), i)$.

Next, we distinguish between "free" and "bound" variables.

Definition 2.7. The set $FV(t)$ of free variables of term t is defined by:

- $FV(x_i) \coloneqq \{x_i\}$,
  $FV(\overline{c_i}) \coloneqq \emptyset$,
- $FV(f(t_1, \ldots, t_n)) \coloneqq FV(t_1) \cup \cdots \cup FV(t_n)$.

Definition 2.8. The set of $FV(\varphi)$ of free variables of formula φ is defined by:

- $FV(P(t_1, \ldots, t_p)) \coloneqq FV(t_1) \cup \cdots \cup FV(t_p)$,
  $FV(t_1 \stackrel{\circ}{=} t_2) \coloneqq FV(t_1) \cup FV(t_2)$,
  $FV(\bot) = FV(P) \coloneqq \emptyset$ for propositional symbol P,
- $FV(\varphi \square \psi) \coloneqq FV(\varphi) \cup FV(\psi)$,
  $FV(\neg\varphi) \coloneqq FV(\varphi)$,
- $FV(\forall x_i \varphi) = FV(\exists x_i \varphi) \coloneqq FV(\varphi) - \{x_i\}$.

Definition 2.9. t or φ is called *closed* if $FV(t) = \emptyset$ or $FV(\varphi) = \emptyset$, respectively. A closed formula is also called a *sentence*. $TERM_c$ denotes the set of closed terms and *SENT* denotes the set of sentences.

It is worth observing that the same variable may occur free and bound. For example, $\forall x_1(x_1 \mathrel{\dot{=}} x_2) \to P(x_1)$ contains x_1 both free and bound.

We also have substitution operators for terms and for formulae in predicate calculus.

Definition 2.10. Let s, t be terms. Then, a substitution of term t for x in term s (denoted by $s[t/x]$) is defined as follows.

- $y[t/x] \coloneqq y$ if $y \not\equiv x$; or t if $y \equiv x$,
  $c[t/x] \coloneqq c$,
- $f(t_1, \ldots, t_p)[t/x] \coloneqq f(t_1[t/x], \ldots, t_p[t/x])$.

In the above, $y \equiv x$ refers to "x and y are the same variables".

Definition 2.11. Let φ be a formula. Then, a substitution of term t for x in formula φ (denoted by $\varphi[t/x]$) is defined as follows:

- $\bot[t/x] \coloneqq \bot$,
  $P[t/x] \coloneqq P$ for proposition P,
  $P(t_1, \ldots, t_p)[t/x] \coloneqq P(t_1[t/x], \ldots, t_p[t/x])$,
  $(t_1 \mathrel{\dot{=}} t_2)[t/x] \coloneqq t_1[t/x] \mathrel{\dot{=}} t_2[t/x]$,
- $(\varphi \mathbin{\square} \psi)[t/x] \coloneqq \varphi[t/x] \mathbin{\square} \psi[t/x]$,
  $(\neg\varphi)[t/x] \coloneqq \neg\varphi[t/x]$,
- $(\forall y\varphi)[t/x] \coloneqq \forall y\varphi[t/x]$ if $x \not\equiv y$; otherwise $\forall y\varphi$,
  $(\exists y\varphi)[t/x] \coloneqq \exists y\varphi[t/x]$ if $x \not\equiv y$; otherwise $\exists y\varphi$.

Definition 2.12. Let \$ be a proposition symbol. Then, a substitution of formula φ of \$ in formula σ (denoted by $\sigma[\varphi/\$]$) is defined as follows:

- $\sigma[\varphi/\$] \coloneqq \sigma$ if $\sigma \not\equiv \$$ for atomic \$; otherwise φ,
- $(\sigma_1 \mathbin{\square} \sigma_2)[\varphi/\$] \coloneqq \sigma_1[\varphi/\$] \mathbin{\square} \sigma_2[\varphi/\$]$,
  $(\neg\sigma_1)[\varphi/\$] \coloneqq \neg\sigma_1[\varphi/\$]$,
  $(\forall y\varphi)[\varphi/\$] \coloneqq \forall y\sigma[\varphi/\$]$,
  $(\exists y\varphi)[\varphi/\$] \coloneqq \exists y\sigma[\varphi/\$]$.

Example 2.2. Let $t_1 \coloneqq p(x_1, x_2)$ and $t_2 \coloneqq i(x_3)$. Therefore, $t_2[t_1/x_1] = i(x_3)$. □

It is worth noticing that Definition 2.11 forbids substitution for bound variables. Unfortunately, there is another case we have to be aware of *i.e.* when some variables become bound after substitution. To deal with this case, we define *t is free for x in φ* to indicate that the free variables of t are not going to be bound after substitution in φ.

Definition 2.13. *t is free for x in φ* if

- φ is atomic,

- $\varphi \coloneqq \varphi_1 \square \varphi_2$ (or $\varphi \coloneqq \neg\varphi_1$) and t is free for x in φ_1 and φ_2 (φ_1, respectively),
- ($\varphi \coloneqq \exists y\psi$ or $\varphi \coloneqq \forall y\psi$) and ($y \notin FV(t)$ and t is free for x in ψ, where $x \not\equiv y$).

Example 2.3. x_2 is free for x_0 in $\exists x_3 P(x_0, x_3)$ whereas $f(x_0, x_1)$ is not free for x_0 in $\exists x_1 P(x_0, x_3)$.

Lemma 2.3. t is free for x in $\varphi \iff$ the variables of t in $\varphi[t/x]$ are not bound by a quantifier.

There are also analogous definition and lemma for the substitution of a formula as follows:

Definition 2.14. *φ is free for $ in σ* if

- σ is atomic,
- $\sigma \coloneqq \sigma_1 \square \sigma_2$ (or $\neg\sigma_1$) and φ is free for $ in σ_1 and σ_2 (σ_1, respectively),
- ($\sigma \coloneqq \exists y\tau$ or $\sigma \coloneqq \forall y\tau$) and ($y \notin FV(\varphi)$ and φ is free for $ in τ, where $ $\not\equiv y$).

Lemma 2.4. φ is free for $ in $\sigma \iff$ the free variables of φ are in $\sigma[\varphi/\$]$ not bound by a quantifier.

For simplicity, we may sometimes write down substitution informally *i.e.* $\varphi(t)$ is the result of replacing x by t in $\varphi(x)$ and $\varphi(t)$ is called a *substitution instance* of $\varphi(x)$.

The language we have introduced so far can be used to describe structures and classes of structures of a given type. The predicate symbols, function symbols, and constant symbols act as names for various relations, operations, and constants, respectively. In the following, we define how each element of $|\mathfrak{A}|$ can be referred individually.

Definition 2.15. The *extended language*, $L(\mathfrak{A})$, of $\mathfrak{A}$ is obtained from the language L, of the type of $\mathfrak{A}$, by adding constant symbols for all elements of $\mathfrak{A}$. We denote the constant symbol, belonging to $a \in |\mathfrak{A}|$, by $\overline{a}$.

2.4 Semantics

Interpretation is the art of relating syntactic objects (*i.e.* strings of symbols) and state of affairs in reality. That is, a statement σ is true in a structure if it is actually the case that σ applies. For example, the sentence "snow is white" is true if snow is actually white.

We start by giving an example in the structure of integer numbers. That is, let structure $\mathfrak{A} = (\mathbb{Z}, <, +, -, 0))$ with the following alphabet:

- predicate symbols: $\hat{=}, L$;
- function symbols: P, M,
- constant symbol: $\overline{0}$.

Table 2.1: Interpretation of closed terms of $\mathfrak{A}$

t	$t^{\mathfrak{A}}$
$\overline{m}$	m
$P(t_1, t_2)$	$t_1^{\mathfrak{A}} + t_2^{\mathfrak{A}}$
$M(t)$	$-t^{\mathfrak{A}}$

As introduced in Definition 2.15, $L(\mathfrak{A})$ has constant symbol $\overline{m}$ for all $m \in \mathbb{Z}$. First, we interpret the closed terms of $L(\mathfrak{A})$ *i.e.* the interpretation $t^{\mathfrak{A}}$ of t is an element of $\mathbb{Z}$ (*cf.* Table 2.1).

Second, we interpret sentences of $L(\mathfrak{A})$ by assigning one of the truth values *viz.* 0 or 1. These can be defined by the valuation function v as follows:

- $v(\bot) = 0$,
- $v(t \mathrel{\dot{=}} s) = 1$ if $t^{\mathfrak{A}} = s^{\mathfrak{A}}$; otherwise 0,
- $v(L(t, s)) = 1$ if $t^{\mathfrak{A}} < s^{\mathfrak{A}}$; otherwise 0,
- $v(\varphi \,\square\, \psi)$ where $\square \in \{\vee, \wedge, \to, \leftrightarrow\}$ and $v(\neg\varphi)$ as defined for the propositional logic,
- $v(\forall x\varphi) = \min\{v(\varphi[\overline{n}/x]) | n \in \mathbb{Z}\}$,
- $v(\exists x\varphi) = \max\{v(\varphi[\overline{n}/x]) | n \in \mathbb{Z}\}$.

Observe that v is uniquely determined by $\mathfrak{A}$. Though it may be more appropriate to use $v_{\mathfrak{A}}$, we may simply drop $\mathfrak{A}$ for convenience. Furthermore, we may write $[\![\varphi]\!]_{\mathfrak{A}}$ for $v_{\mathfrak{A}}(\varphi)$. Following this convention, it may be better to write $[\![t]\!]_{\mathfrak{A}}$ for $t^{\mathfrak{A}}$. However, we may keep both notations and use them interchangeably. We may sometimes drop the subscript if no confusion can be arisen.

Example 2.4. $(P(P(\overline{2}, \overline{3}), M(\overline{7})))^{\mathfrak{A}} = P(\overline{2}, \overline{3})^{\mathfrak{A}} + M(\overline{7})^{\mathfrak{A}} = (\overline{2}^{\mathfrak{A}} + \overline{3}^{\mathfrak{A}}) + (-\overline{7}^{\mathfrak{A}}) = 2 + 3 + (-7) = 2$. □

Example 2.5. $[\![\overline{2} \mathrel{\dot{=}} \overline{1}]\!] = 0$ since $2 \neq 1$. □

Let us now present a formal definition of interpretation for the general case. Let $\mathfrak{A} = \langle A, R_1, \ldots, R_n, F_1, \ldots, F_m, \{c_i | i \in I\}$ of a given similarity type $\langle r_1, \ldots, r_n; a_1, \ldots, a_m; |I|\rangle$. The corresponding language has predicate symbol $\overline{R}_1, \ldots, \overline{R}_n$, function symbol $\overline{F}_1, \ldots, \overline{F}_m$, and constant symbol $\overline{c}_i$. Moreover, $L(\mathfrak{A})$ has constant symbol $\overline{a}$ for all $a \in |\mathfrak{A}|$.

Definition 2.16. An interpretation of the closed terms of $L(\mathfrak{A})$ in $\mathfrak{A}$ is a mapping $(\cdot)^{\mathfrak{A}} : TERM_c \to |\mathfrak{A}|$ satisfying:

- $\overline{c}_i^{\mathfrak{A}} = c_i$,
  $\overline{a}^{\mathfrak{A}} = a$,
- $(\overline{F}_i(t_1, \ldots, t_p))^{\mathfrak{A}} = F_i(t_1^{\mathfrak{A}}, \ldots, t_p^{\mathfrak{A}})$ where $p = a_i$.

Definition 2.17. Let φ be a sentence of $L(\mathfrak{A})$ in $\mathfrak{A}$. An interpretation of φ is a mapping $[\![\cdot]\!]_{\mathfrak{A}} : SENT \to \{0,1\}$ satisfying:

- $[\![\underline{\bot}]\!]_{\mathfrak{A}} \coloneqq 0$,
  $[\![\overline{R}]\!]_{\mathfrak{A}} \coloneqq R$ (*i.e.* 0 or 1),
- $[\![\overline{R}_i(t_1,\ldots,t_p)]\!]_{\mathfrak{A}} \coloneqq 1$ if $\langle t_1^{\mathfrak{A}},\ldots,t_p^{\mathfrak{A}}\rangle \in R_i$ where $p = r_i$; otherwise 0,
  $[\![t_1 \stackrel{\circ}{=} t_2]\!]_{\mathfrak{A}} \coloneqq 1$ if $t_1^{\mathfrak{A}} = t_2^{\mathfrak{A}}$; otherwise 0,
- $[\![\varphi \wedge \psi]\!]_{\mathfrak{A}} \coloneqq \min([\![\varphi]\!]_{\mathfrak{A}}, [\![\psi]\!]_{\mathfrak{A}})$,
  $[\![\varphi \vee \psi]\!]_{\mathfrak{A}} \coloneqq \max([\![\varphi]\!]_{\mathfrak{A}}, [\![\psi]\!]_{\mathfrak{A}})$,
  $[\![\varphi \to \psi]\!]_{\mathfrak{A}} \coloneqq \max(1 - [\![\varphi]\!]_{\mathfrak{A}}, [\![\psi]\!]_{\mathfrak{A}})$,
  $[\![\varphi \leftrightarrow \psi]\!]_{\mathfrak{A}} \coloneqq 1 - |[\![\varphi]\!]_{\mathfrak{A}} - [\![\psi]\!]_{\mathfrak{A}}|$,
  $[\![\neg\varphi]\!]_{\mathfrak{A}} \coloneqq 1 - [\![\varphi]\!]_{\mathfrak{A}}$
- $[\![\forall x\varphi]\!]_{\mathfrak{A}} \coloneqq \min\{[\![\varphi[\overline{a}/x]]\!]_{\mathfrak{A}} \mid a \in |\mathfrak{A}|\}$
  $[\![\exists x\varphi]\!]_{\mathfrak{A}} \coloneqq \max\{[\![\varphi[\overline{a}/x]]\!]_{\mathfrak{A}} \mid a \in |\mathfrak{A}|\}$

When $[\![\varphi]\!]_{\mathfrak{A}} = 1$, we write $\mathfrak{A} \models \varphi$ which can be read as "φ is true (or valid) in $\mathfrak{A}$". The relation $\models$ is called the *satisfaction relation*.

So far, we have only defined truth for sentences of $L(\mathfrak{A})$. To deal with arbitrary formulae, we extend $\models$ to the following.

Definition 2.18. Let $FV(\varphi) = \{z_1,\ldots,z_k\}$. Then, $Cl(\varphi) \coloneqq \forall z_1 \ldots z_k\varphi$ is the *universal closure* of φ (assume that the order of variable $z_1 \ldots z_k$ is fixed in some way).

Definition 2.19.

1. $\mathfrak{A} \models \varphi$ iff $\mathfrak{A} \models Cl(\varphi)$,
2. $\models \varphi$ iff $\mathfrak{A} \models \varphi$ for all $\mathfrak{A}$ (of the appropriate type),
3. $\mathfrak{A} \models \Gamma$ iff $\mathfrak{A} \models \psi$ for all $\psi \in \Gamma$,
4. $\Gamma \models \varphi$ iff $(\mathfrak{A} \models \Gamma \Longrightarrow \mathfrak{A} \models \varphi)$ where $\Gamma \cup \{\varphi\}$ consists of sentences.

Traditionally, we call $\mathfrak{A}$ a *model* of σ if $\mathfrak{A} \models \sigma$. We call $\mathfrak{A}$ a *model* of Γ if $\mathfrak{A} \models \Gamma$. We say that φ is *true* (or *valid*) if $\models \varphi$. Furthermore, φ is a *semantic consequence* of Γ if $\Gamma \models \varphi$ *i.e.* φ holds in each model of Γ.

The properties of the satisfaction relation are in understandable and convenient correspondence with the intuitive meaning of the connectives (*cf.* the following lemma).

Lemma 2.5. If we restrict ourselves to sentences, then

1. $\mathfrak{A} \models \varphi \wedge \psi \Longleftrightarrow \mathfrak{A} \models \varphi$ and $\mathfrak{A} \models \psi$,
2. $\mathfrak{A} \models \varphi \vee \psi \Longleftrightarrow \mathfrak{A} \models \varphi$ or $\mathfrak{A} \models \psi$,
3. $\mathfrak{A} \models \neg\varphi \Longleftrightarrow \mathfrak{A} \not\models \varphi$,
4. $\mathfrak{A} \models \varphi \to \psi \Longleftrightarrow (\mathfrak{A} \models \varphi \Longrightarrow \mathfrak{A} \models \psi)$,

5. $\mathfrak{A} \models \varphi \leftrightarrow \psi \Longleftrightarrow (\mathfrak{A} \models \varphi \Longleftrightarrow \mathfrak{A} \models \psi)$,

6. $\mathfrak{A} \models \forall x\varphi \Longleftrightarrow \mathfrak{A} \models \varphi[\overline{a}/x]$, for all $a \in |\mathfrak{A}|$,

7. $\mathfrak{A} \models \exists x\varphi \Longleftrightarrow \mathfrak{A} \models \varphi[\overline{a}/x]$, for some $a \in |\mathfrak{A}|$.

The above lemma also says that we can replace the connectives by their analogues in the meta-language and interpret the atoms by checking the relations in the structure.

2.5 Simple Properties of Predicate Logic

We first consider the generalizations of De Morgan's laws as follows.

Theorem 2.1. 1. $\models \neg\forall x\varphi \leftrightarrow \exists x\neg\varphi$;

2. $\models \neg\exists x\varphi \leftrightarrow \forall x\neg\varphi$;

3. $\models \forall x\varphi \leftrightarrow \neg\exists x\neg\varphi$;

4. $\models \exists x\varphi \leftrightarrow \neg\forall x\neg\varphi$.

The following theorem says that the order of quantifiers of the same sort is not relevant and quantification over a variable that does not occur can be deleted.

Theorem 2.2. 1. $\models \forall x\forall y\varphi \leftrightarrow \forall y\forall x\varphi$;

2. $\models \exists x\exists y\varphi \leftrightarrow \exists y\exists x\varphi$;

3. $\models \forall x\varphi \leftrightarrow \varphi$ if $x \notin FV(\varphi)$;

4. $\models \exists x\varphi \leftrightarrow \varphi$ if $x \notin FV(\varphi)$.

Ones may observe from Definition 2.17 that $\forall$ and $\exists$ are generalizations of $\wedge$ and $\vee$, respectively. It is not surprising that $\forall$ and $\exists$ can distribute over $\wedge$ and $\vee$, respectively. However, $\forall$ and $\exists$ can distribute over $\vee$ and $\wedge$ only if a certain condition is met.

Theorem 2.3. 1. $\models \forall x(\varphi \wedge \psi) \leftrightarrow \forall x\varphi \wedge \forall x\psi$;

2. $\models \exists x(\varphi \vee \psi) \leftrightarrow \exists x\varphi \vee \exists x\psi$;

3. $\models \forall x(\varphi(x) \vee \psi) \leftrightarrow \forall x\varphi(x) \vee \psi$ if $x \notin FV(\psi)$;

4. $\models \exists x(\varphi(x) \wedge \psi) \leftrightarrow \exists x\varphi(x) \wedge \psi$ if $x \notin FV(\psi)$.

It is worth noting that $\forall x(\varphi(x)\psi(x)) \rightarrow \forall x\varphi(x) \vee \forall x\psi(x)$ and $\exists x\varphi(x) \wedge \exists x\psi(x) \rightarrow \exists x(\varphi(x) \wedge \psi(x))$ are "not" true.

The following theorem spells out that one can replace a bound variable by a "fresh" one, which enable us to pull out quantifiers from a formula.

Theorem 2.4 (Change of Bound Variables)**.** If x, y are free for z in φ and $x, y \notin FV(\varphi)$, the following holds:

- $\models \exists x\varphi[x/z] \leftrightarrow \exists y\varphi[y/z]$;
- $\models \forall x\varphi[x/z] \leftrightarrow \forall y\varphi[y/z]$.

For example, ones can use the above theorem as follows: $\forall x\varphi(x) \vee \forall x\psi(x) \Longleftrightarrow \forall x\varphi(x) \vee \forall y\psi(y) \Longleftrightarrow \forall xy(\varphi(x) \vee \psi(y))$.

2.6 Semantic Tableau

In the preceding sections, we always look at the logic from a semantic point of view. However, this is not the only possible viewpoint. In this section, we will explore the non-semantic approach *i.e.* by setting up a system for deriving formula φ from a given set Γ of formulae (denoted by $\Gamma \vdash \varphi$). There are many available such systems and we are going to introduce a so-called *semantic tableau* here [98].

Definition 2.20. A *signed formula* is an expression of the form $\mathbf{T}\varphi$ or $\mathbf{F}\varphi$ where φ is a formula. A signed formula $\mathbf{T}\varphi$ is called *true* if φ is true and false otherwise. On the other hand, a signed formula $\mathbf{F}\varphi$ is called *true* if φ is false and true otherwise.

Definition 2.21. A *signed tableau* is a rooted dyadic tree where each node carries a signed formula.

If τ is a signed tableau, an *immediate extension* of τ is a larger tableau τ' obtained by applying a tableau rule to a finite path of τ.

Definition 2.22. A path of a tableau is said to be *closed* if it contains a conjugate pair of formulae *i.e.* $\mathbf{T}\varphi$ and $\mathbf{F}\varphi$. A path of a tableau is said to be *open* if it is not closed. A tableau is said to be *closed* if each of its path is closed.

To show that $\Gamma \vdash \varphi$ by semantic tableau, we form a signed tableau starting with $\mathbf{T}\psi_1, \ldots, \mathbf{T}\psi_k, \mathbf{F}\varphi$, where $\psi_1, \ldots, \psi_k \in \Gamma$. If the tableau is closed, then φ is derivable from Γ. If $\Gamma = \emptyset$ *i.e.* $\vdash \varphi$, we form a signed tableau with $\mathbf{F}\varphi$ and conclude $\vdash \varphi$ holds if the tableau is closed. Figure 2.1 depicts tableau rules for propositional logic. To cope with predicate logic, we have to equip with additional rules for the quantifiers (*cf.* Figure 2.2) where t is a closed term and c is a fresh constant.

Figure 2.3 depicts that we can employ semantic tableau to show $\vdash (\exists x(P(x) \vee Q(x)) \leftrightarrow (\exists xP(x) \vee \exists xQ(x)))$. This holds because the tableau is closed (*cf.* Definition 2.22)

It is worth mentioning that semantic tableau is sound and complete *i.e.* the relation $\models$ and $\vdash$ coincide. We state this in the following theorems.

Theorem 2.5 (Soundness)**.** $\Gamma \vdash \varphi \Longrightarrow \Gamma \models \varphi$.

Theorem 2.6 (Completeness)**.** $\Gamma \models \varphi \Longrightarrow \Gamma \vdash \varphi$.

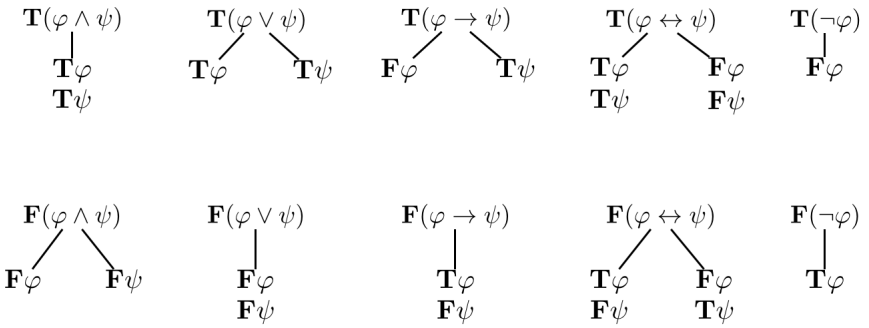


Figure 2.1: Rules used in propositional logic

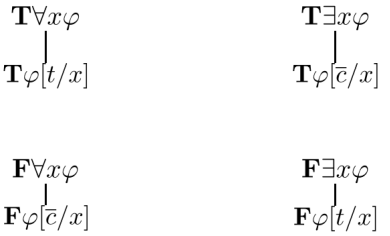


Figure 2.2: Rules used in predicate logic

While semantic tableau in propositional logic is decidable *i.e.* we can test if $\Gamma \vdash \varphi$ for arbitrary Γ and φ, this is not the case for first-order predicate logic. If $\Gamma \not\models \varphi$, then semantic tableau may be not terminated. This is not a deficiency of the tableau method since first-order predicate logic is known to be undecidable (*cf.* Section 3.3). This negative result gives unimpressiveness to use first-order predicate logic to model some knowledge base systems. In the next chapter, we will briefly introduce a decidable fragment of first-order logic which has been immensely studied in the area of ontology development called *Description Logics*.

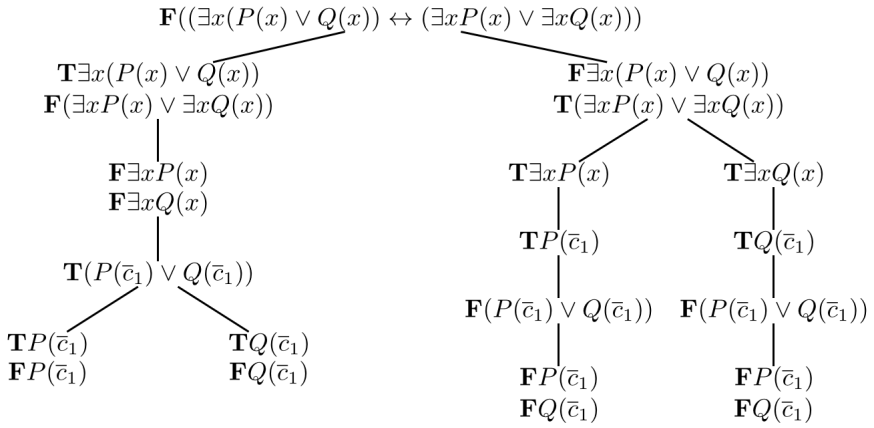


Figure 2.3: Example of using semantic tableau

Chapter 3

Fundamental of Description Logics

Description Logics (DLs) [6–8] are a family of logic-based knowledge representation formalisms that can be used to represent and reason about the knowledge of an application domain in a structured and well-understood way. They are based on a common family of languages, called *description languages*, which specifies a set of *concept constructors* to build concept descriptions and role descriptions. These concept and role descriptions may be used to set up a knowledge base in forms of *terminology* (TBox) and *assertions* (ABox), and also, to *reason* about the content of a knowledge base, by a knowledge representation system based on DLs (or shortly, DL system). Generally, an architecture of DL systems can be sketched as in Figure 3.1.

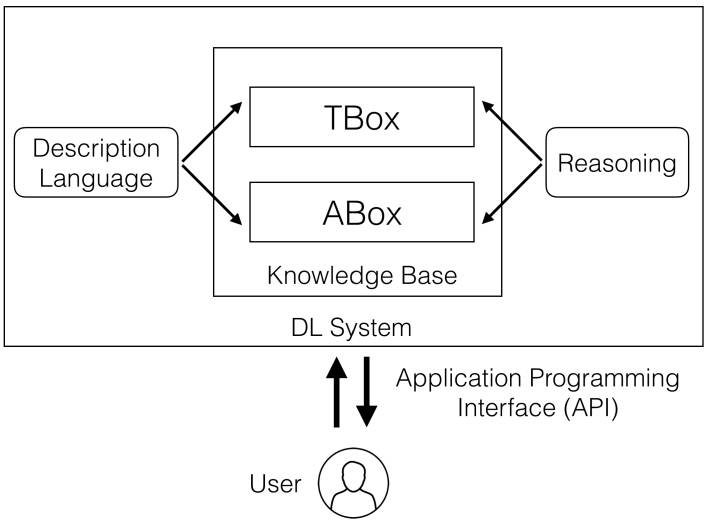


Figure 3.1: Architecture of a DL system

Intuitively, a knowledge base (KB) is composed of two distinct parts, *i.e.* a TBox representing general knowledge about the problem domain and an ABox representing knowledge about a specific situation. As informally exemplified in Chapter 1, a statement like $\mathsf{Endocarditis} \equiv \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Endocardium}$ is contained in the TBox and statements like $\mathsf{onMedication}(bob, anonym1)$ and $\mathsf{AntifungalAntibiotics}(anonym1)$ are

contained in the ABox. A formal discussion on these two parts will be continued in Section 3.2.

Figure 3.1 also shows that the general architecture of a DL system interacts with users via application programming interface (API) for several purposes, *e.g.* querying the knowledge base and modifying the knowledge base by adding/retracting concepts, roles, and assertions. However, many systems, in addition of providing APIs, may also provide an escape hatch by which application programs can operate on a knowledge base in arbitrary ways [6].

In the first section, we introduce core elements of arbitrary description languages that are building-blocks of concept descriptions and role descriptions. Then, we give definitions of ontological axioms which can be distinguished into the two main groups, *viz.* a TBox and an ABox (*cf.* Section 3.2). Section 3.4 describes the most widely used DL reasoning services. Finally, a variety of reasoning algorithms which can be used to deal with the reasoning services are discussed in Section 3.5.

3.1 Description Languages

The foundations of description languages are concept descriptions and role descriptions (concepts and roles, for short, respectively). Intuitively, a concept represents a class of objects sharing common characteristics whereas a role represents a binary relationship between objects. The language for building concepts and roles is a characteristic of each DL system and different systems are distinguished by their description languages. As we shall see soon, the description languages have a model-theoretic semantics. Hence, it is worth noting that concepts and roles can be identified by formulae in First Order Logic or its slight extension in some cases (see Section 3.3 and [6] for such discussions).

Basically, we assume three disjoint sets of concept names CN, role names RN, and individual names Ind. Description languages are distinguished by a set of *concept constructors* they provide. These constructors are used to inductively define concepts and roles. It is obvious that the more concept constructors a particular DL provides, the more expressive concepts and roles can be constructed. In abstract notations, we use A and B to denote atomic concepts, C and D to denote concept descriptions[1], r to denote atomic roles, and a and b to denote individuals. Table 3.1 lists common concept constructors that are widely considered in the literature. The second and the third columns show the syntax and semantics elements, respectively. The attributive language $\mathcal{AL}$ was introduced in [99] as a minimal language that is of practical interest. $\mathcal{AL}$ provides exactly the constructors as in the table except existential quantification ($\exists r.C$).

The first naming scheme for DLs was also proposed in [99]: starting from the DL $\mathcal{AL}$, additional constructors are indicated by appending corresponding letters; *e.g.* $\mathcal{ALC}$ is obtained from $\mathcal{AL}$ by featuring the complement operator ($\neg$) and $\mathcal{ALE}$ is obtained from $\mathcal{AL}$ by adding existential quantification ($\exists r.C$). It is worth noting that $\mathcal{ALC}$ (stands for *Attributive Language with Complement*) is considered as the smallest Boolean-closed DL.[2]

[1]The precise definition of concept description is given later.

[2]Strictly speaking, a DL must provide at least one quantifier, *i.e.* either existential or universal. Thus,

Constructor name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
nominal	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$
limited existential quantification	$\exists r.\top$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e\colon (d,e) \in r^{\mathcal{I}}\}$
existential quantification	$\exists r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e\colon (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
universal restriction	$\forall r.C$	$\{d \in \Delta^{\mathcal{I}} \mid \forall e\colon (d,e) \in r^{\mathcal{I}} \rightarrow e \in C^{\mathcal{I}}\}$

Table 3.1: Syntax and semantics of concept constructors.

There are also a number of interesting sub-Boolean DLs[1], most of which disallow disjunction and (full) negation such as $\mathcal{FL}_0$ and $\mathcal{EL}$. For historical naming reasons, $\mathcal{FL}_0$ is obtained by disallowing atomic negation and limited existential quantification from $\mathcal{AL}$; and also, $\mathcal{EL}$ is obtained by disallowing atomic negation and universal restriction from $\mathcal{AL}$. Both are sub-languages of $\mathcal{AL}$ that are practical interest due to their practical efficiency and sufficient expressivity. The main investigation of this thesis starts from these two languages. In the following, we provide their formal definitions and discuss on their corresponding reasoning techniques later in the remaining of this chapter.

Let $\mathcal{L}$ be a specific DL. We denote the set of concept descriptions for DL $\mathcal{L}$ by $\mathsf{Con}(\mathcal{L})$. In the following, we give formal definitions for the syntax and semantics of $\mathcal{FL}_0$ (*cf.* Definition 3.1) and $\mathcal{EL}$ (*cf.* Definition 3.2).

Definition 3.1 ($\mathcal{FL}_0$ syntax)**.** Let CN be a set of concept names and RN be a set of role names. The sets of $\mathcal{FL}_0$ concept descriptions (denoted by $\mathsf{Con}(\mathcal{FL}_0)$) is the smallest sets such that:

1. If $A \in \mathsf{CN} \cup \{\top\}$, then $A \in \mathsf{Con}(\mathcal{FL}_0)$;
2. If $C, D \in \mathsf{Con}(\mathcal{FL}_0)$ and $r \in \mathsf{RN}$, then $C \sqcap D, \forall r.C \in \mathsf{Con}(\mathcal{FL}_0)$.

Definition 3.2 ($\mathcal{EL}$ syntax)**.** Let CN be a set of concept names and RN be a set of role names. The sets of $\mathcal{EL}$ concept descriptions (denoted by $\mathsf{Con}(\mathcal{EL})$) is the smallest sets such that:

1. If $A \in \mathsf{CN} \cup \{\top\}$, then $A \in \mathsf{Con}(\mathcal{EL})$;

the logic with the first five constructors in Table 3.1 is not a DL as it is equivalent to the propositional logic.

[1]Sub-Boolean DLs are DLs that are not equipped with all Boolean operators

2. If $C, D \in \mathsf{Con}(\mathcal{EL})$ and $r \in \mathsf{RN}$, then $C \sqcap D, \exists r.C \in \mathsf{Con}(\mathcal{EL})$.

In the following, we give examples of concepts expressed in $\mathcal{FL}_0$ and $\mathcal{EL}$.

Example 3.1. The concept of Herbivore, whose members are animal which eat only plants, may be expressed as: Animal $\sqcap \forall$eat.Plant. Similarly, the concept of Carnivore, whose members are animal which eats only animals, may be expressed as: Animal $\sqcap \forall$eat.Animal. Finally, the concept of Omnivore which eats animals and plants, may be expressed as: Animal $\sqcap \exists$eat.Animal $\sqcap \exists$eat.Plant. □

Example 3.2. The concept of Endocarditis, whose members are an inflammation which has location on an endocardium tissue, may be expressed using concept names and role names in SNOMED CT as: Inflammation $\sqcap \exists$hasLocation.Endocardium. □

We can agree that the concepts Herbivore and Carnivore are $\mathcal{FL}_0$ concepts because, following Definition 3.1, it is obvious that:

1. Animal is $\mathcal{FL}_0$ concept;
2. $\forall$eat.Plant and $\forall$eat.Animal are $\mathcal{FL}_0$ concepts;
3. Finally, Animal $\sqcap \forall$eat.Plant and Animal $\sqcap \forall$eat.Animal are $\mathcal{FL}_0$ concepts.

On the other hand, the concepts Omnivore and Endocarditis can be shown to be $\mathcal{EL}$ concepts based on the similar steps together with Definition 3.2.

Though sub-Boolean DLs are not very expressive, they are also of theoretical interest due to their tractability. Table 3.2 shows the worst-case complexity of concept satisfiability problem[1] in $\mathcal{ALC}$ and the subsumption problem in $\mathcal{FL}_0$ and $\mathcal{EL}$. It is worth noting that the satisfiability problem is trivial in $\mathcal{FL}_0$ and $\mathcal{EL}$ since any concept expressed in these languages is satisfiable. The table also shows that $\mathcal{EL}$ exhibits the most robust behavior w.r.t. every type of terminology.

Terminology	$\mathcal{ALC}$	$\mathcal{FL}_0$	$\mathcal{EL}$
the empty TBox	PSpace-complete [99]	polynomial [100]	polynomial [101]
acyclic TBoxes	PSpace-complete [99]	coNP-complete [27]	polynomial [102]
general TBoxes	ExpTime-complete [103]	ExpTime-complete [63]	polynomial [104]

Table 3.2: Comparing the Description Logics $\mathcal{ALC}$, $\mathcal{FL}_0$, and $\mathcal{EL}$

It should be also noted that these results are not merely theoretical interest. In fact, they also provide sufficiently expressivity. For instance, the Systematized Nomenclature of Medicine, Clinical Terms (*aka.* SNOMED CT)[2] [105,106] employs $\mathcal{EL}$ with an acyclic TBox extended with role hierarchy axioms. Furthermore, the *Gene Ontology* [107] can be seen

[1]Section 3.4 gives precise definitions of most widely used reasoning services in DLs.
[2]http://bioportal.bioontology.org/ontologies/SNOMEDCT

as an acyclic $\mathcal{EL}$ TBox. These motivate us to find tractable DLs for our targeted problem, *i.e.* concept similarity. It is worth noting that $\mathcal{FL}_0$ and $\mathcal{EL}$ are the minimal candidate DLs to pursue a polynomial complexity since they would not inherit NP-hardness from the propositional logic [15].

Like any DLs, the semantics of $\mathcal{FL}_0$ and $\mathcal{EL}$ concepts are defined through interpretations as shown in the following.

Definition 3.3 (Semantics of $\mathcal{FL}_0$ and $\mathcal{EL}$)**.** An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ of interpretation domain and an interpretation function $\cdot^{\mathcal{I}}$, which assigns to each concept name $A \in \mathsf{CN}$ a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to each role name $r \in \mathsf{RN}$ a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to a concept descriptions by inductive definitions given in the right column of Table 3.1.

An interpretation $\mathcal{I}$ is said to be a *model* of a concept C, or $\mathcal{I}$ models C, iff the interpretation of C in $\mathcal{I}$, *i.e.* $C^{\mathcal{I}}$, is not empty, *i.e.* $C^{\mathcal{I}} \neq \emptyset$.

Example 3.3. Given a concept $\mathsf{Animal} \sqcap \forall \mathsf{eat}.\mathsf{Plant}$, we can find an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $\mathcal{I}$ is a model of the concept as follows:

1. Suppose an interpretation domain $\Delta^{\mathcal{I}} = \{cow_1, cow_2, \text{cow}_3, grass_1\}$;
2. Suppose $\mathsf{Animal}^{\mathcal{I}} = \{cow_1, cow_2, \text{cow}_3\}$, $\mathsf{Plant}^{\mathcal{I}} = \{grass_1\}$, and $\mathsf{eat}^{\mathcal{I}} = \{(cow_1, grass_1), (cow_2, cow_3)\}$;
3. Using the semantics given in Table 3.1, we know $(\forall \mathsf{eat}.\mathsf{Plant})^{\mathcal{I}} = \{cow_1, cow_3\}$;
4. Using the semantics given in Table 3.1, we know $(\mathsf{Animal} \sqcap \forall \mathsf{eat}.\mathsf{Plant})^{\mathcal{I}} = \{cow_1, cow_3\}$. □

Since $(\mathsf{Animal} \sqcap \forall \mathsf{eat}.\mathsf{Plant})^{\mathcal{I}} \neq \emptyset$, then the defined interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of the concept $\mathsf{Animal} \sqcap \forall \mathsf{eat}.\mathsf{Plant}$.

3.2 DL Knowledge Base

We have seen how concept descriptions are built through the use of concept constructors. Now, ones may want to form statements representing the general knowledge about the problem domain and the knowledge about a specific situation. For example,

$$\mathsf{Herbivore} \equiv \mathsf{Animal} \sqcap \forall \mathsf{eat}.\mathsf{Plant}$$

saying that "a herbivore is an animal which eats only plants". Or, the statements

$$\mathsf{Herbivore}(cow_1) \quad \text{and} \quad \mathsf{Herbivore}(cow_3)$$

saying that "cow_1 is a herbivore" and "cow_3 is a herbivore", respectively.

Constructor name	Syntax	Semantics
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept disjointness	$C \sqcap D \sqsubseteq \bot$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$
domain restriction	$\mathsf{domain}(r) \sqsubseteq C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists e\colon (d,e) \in r^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$
range restriction	$\mathsf{range}(r) \sqsubseteq C$	$\{e \in \Delta^{\mathcal{I}} \mid \exists d\colon (d,e) \in r^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$
functionality	$\mathsf{functional}(r)$	$\forall d \in \Delta^{\mathcal{I}}\colon \sharp\{e \in \Delta^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \leq 1$
reflexivity	$\mathsf{reflexive}(r)$	$\forall d \in \Delta^{\mathcal{I}}\colon (d,d) \subseteq r^{\mathcal{I}}$
transitivity	$\mathsf{transitive}(r)$	$\forall d,e,f \in \Delta^{\mathcal{I}}\colon (d,e),(e,f) \in r^{\mathcal{I}} \rightarrow (d,f) \in r^{\mathcal{I}}$
role hierarchy	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_k \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Table 3.3: Syntax and semantics of ontological constructors

These statements can be formulated by using a terminological formalism and an assertional formalism. Interestingly, such a formalism is also characterized by a set of *ontological constructors*. Table 3.3 lists most commonly used constructors in the literature where the middle and the right column show their syntax and semantics.

According to the table, ontological constructors can be divided into three main groups, *viz.* concept axiom constructors, role axiom constructors, and assertion constructors. Ones may notice that the above examples also conform to these three groups. The statement $\mathsf{Herbivore} \equiv \mathsf{Animal} \sqcap \forall \mathsf{eat}.\mathsf{Plant}$ employs the concept definition constructor, which is a concept axiom constructor, and also, $\mathsf{Herbivore}(cow_1)$ and $\mathsf{Herbivore}(cow_3)$ employs the concept assertion, which is one of assertion constructors. We note that, for the rest of the thesis, we adopt these notations (*i.e.* font styles) of concept names, role names, and individual names throughout the thesis.

In the following subsections, we give formal definitions of a DL knowledge base and its basic components. Informally speaking, a terminological formalism (TBox) captures definitions, relations, and constraints about terminology (concepts and roles) whereas an assertional formalism (ABox) captures assertional statements about individuals w.r.t. the TBox. We note that some work in the literature may regard role axioms as another component called RBox due to some specific reasons *e.g.* the style of internalization in TBox does not involve role axioms.

3.2.1 TBox

As aforementioned, a TBox captures general knowledge about the problem domain. It is basically a set of statements about how concepts are related to each other. For example, "endocarditis is an inflammation that has location on endocardium tissue". This can be expressed as $\mathsf{Endocarditis} \equiv \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Endocardium}$. Formally, a TBox is defined as follows.

Definition 3.4 (TBox)**.** Let $\mathcal{L}$ be a specific DL, $A \in \mathsf{CN}$, and $C \in \mathsf{Con}(\mathcal{L})$. Then, $A \equiv C$ and $A \sqsubseteq C$ are called a *concept definition* and a *primitive concept definition*, respectively. Let $\rhd$ denote either $\equiv$ or $\sqsubseteq$. Then, TBox $\mathcal{T}$ is a finite set of (possibly primitive) concept definitions. A concept definition $A \rhd C$ is *unique* if, for each $A \in \mathsf{CN}$, there is at most one concept definition $A \rhd C$ for some $C \in \mathsf{Con}(\mathcal{L})$.

We call A *directly uses* B in $\mathcal{T}$ if $A \rhd B$ occurs in $\mathcal{T}$ and we define *uses* to be the transitive closure of the relation *directly uses*. Then, a concept definition $A \rhd C$ is *cyclic* if A *uses* itself. Otherwise, we call such definition an *acyclic* concept definition. TBox $\mathcal{T}$ is called *unfoldable* if all concept definitions are unique and acyclic definitions.

A concept name P in $\mathcal{T}$ is said to be *undefined* if it is neither fully defined nor primitively defined in $\mathcal{T}$.

An interpretation $\mathcal{I}$ is a model of a concept definition $A \equiv C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$ and is a model of a primitive concept definition $A \sqsubseteq C$ iff $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. $\mathcal{I}$ is a model of $\mathcal{T}$ iff it is a model of every definition $A \rhd C$ in $\mathcal{T}$.

Given an unfoldable TBox $\mathcal{T}$, concept names occurring on the left-hand side of a concept definition are called *defined concept names* (denoted by $\mathsf{CN}^{\mathsf{def}}$) whereas the others are called *primitive concept names* (denoted by $\mathsf{CN}^{\mathsf{pri}}$). The name *unfoldable* is motivated by the fact that, in such a TBox $\mathcal{T}$, $\mathcal{T}$ can be transformed into an equivalent one $\mathcal{T}'$ by substituting all the defined concept names in concept descriptions with their definitions until only primitive concept names remain. In particular, for a concept definition defined in $\mathcal{T}$ by an axiom $\mathsf{A} \equiv D$, the procedure is simply to replace A with D whenever if occurs in C, and then to recursively unfold D. For a "primitive" concept definition defined in $\mathcal{T}$ by an axiom $\mathsf{A} \sqsubseteq D$, the procedure is slightly more complex. Whenever A occurs in C, it is replaced with the concept $X \sqcap D$ where X is a new concept name not occurring in $\mathcal{T}$ or C. After that, D is recursively unfolded. We note that X represents the unspecified characteristics that differentiate if from D. Such unfolded concepts which remain only primitive concept names are called *fully expanded* concepts. This transformation is called *unfolding* and we use $\mathsf{Unfold}(C, \mathcal{T})$ to denote that the concept C is unfolded w.r.t. $\mathcal{T}$.

When $\mathcal{T}$ is unfolded to $\mathcal{T}'$, each defined concept name in $\mathcal{T}'$ is an independent concept description in a sense that the TBox itself can be disregard. From a computational point of view, unfoldable TBoxes are interesting since they may allow for the use of simplified reasoning techniques (*cf.* Table 3.2) and reasoning in the presence of a TBox is often harder than that without a TBox (or an empty TBox).

A much more expressive formalism of TBox is called a *general TBox* where each axiom is called a *general concept inclusion*. Informally, a general concept inclusion is a statement like this form: $\exists \mathsf{married}.\mathsf{Human} \sqsubseteq \mathsf{Human}$ saying that "a human is only married to a

human". This general formalism is supported by most state-of-the-art DL reasoners. In the following, we gives a formal definition for a general TBox.

Definition 3.5 (General TBox)**.** Let $\mathcal{L}$ be a specific DL and $C, D \in \mathsf{Con}(\mathcal{L})$. Then, a general concept inclusion (GCI) is of the form $C \sqsubseteq D$. Then, a *general TBox* is a finite set of GCIs.

An interpretation $\mathcal{I}$ is called a model of a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is a model of a general TBox $\mathcal{T}$ iff it is a model of every GCI in $\mathcal{T}$.

It should be also noted that general TBoxes are more general than unfoldable TBoxes since GCIs can be used to express (primitive) concept definitions. In particular, a primitive concept definition is a special form of GCI whereas a concept definition can be expressed by means of two GCIs, *i.e.* $A \equiv C$ with $A \sqsubseteq C$ and $C \sqsubseteq A$.

It is also worth noting that, following [108], the semantics we have studied so far is called *descriptive semantics*. This semantics can produce counter-intuitive results when a TBox contains cyclic dependency. In such a case, the so-called *fixpoint semantics* [40, 108] is recommended to use. However, the descriptive semantics is adopted in this thesis because of its wide acceptance as the most appropriate one [109, 110].

Apart from concept definitions and inclusions, there are also interesting and important ontological constructors. Some of them are listed in the upper part of Table 3.3. In some cases, one constructor can be simulated by another. For instance, a domain restriction $\mathrm{domain}(r) \sqsubseteq C$ can be expressed by the GCI $\exists r.\top \sqsubseteq C$. Also, reflexivity, transitivity, and role hierarchy are special forms of role inclusion[1], *i.e.* $\epsilon \sqsubseteq r$, $r \circ r \sqsubseteq r$, and $r \sqsubseteq s$, respectively. Table 3.4 presents various DLs with their supported constructors, where ✓ denotes optional features that may or may not be supported.

3.2.2 ABox

Intuitively, an ABox describes a specific situation (w.r.t. individuals) of an application domain in terms of concepts and roles. This component is alternatively called *world description*. An ABox can contain two kinds of axiom. That is, the first is for asserting that an individual is an instance of a given concept and the second kind is for asserting that a pair of individuals is an instance of a given role. As seen, a statement "*bob* is a Man and is married to *jane* who is a Professor" can be represented as:

$$(\mathsf{Human} \sqcap \mathsf{Male})(\mathit{bob}) \quad \mathsf{married}(\mathit{bob}, \mathit{jane}) \quad \mathsf{Professor}(\mathit{jane})$$

In the following, we give a formal definition of an ABox.

Definition 3.6 (ABox)**.** Let $\mathcal{L}$ be a specific DL, $C \in \mathsf{Con}(\mathcal{L})$, $r \in \mathsf{RN}$, and $a, b \in \mathsf{Ind}$. Then, an assertional axiom is of the form $C(a)$ or $r(a, b)$. Then, an ABox $\mathcal{A}$ is a finite set of assertional axioms.

[1]Sometimes role axioms are regarded as another component called RBox (*cf.* page 34).

DL dialects	$\mathcal{L}_0$	$\mathcal{EL}$	$\mathcal{FL}_0$	$\mathcal{ELH}$	$\mathcal{ALC}$	$\mathcal{SHIF}$	$\mathcal{SROIQ}$
top concept	✔	✔	✔	✔	✔	✔	✔
bottom concept					✔	✔	✔
conjunction	✔	✔	✔	✔	✔	✔	✔
disjunction					✔	✔	✔
negation					✔	✔	✔
nominal							✔
exist. restrictions		✔		✔	✔	✔	✔
value restrictions			✔		✔	✔	✔
concept definition	✓	✓	✓	✓	✓	✓	✓
concept inclusion	✓	✓	✓	✓	✓	✓	✓
domain restriction		✓	✓	✓	✓	✓	✓
concept disjointness					✓	✓	✓
range restriction					✓	✓	✓
functionality						✔	✔
reflexivity							✔
transitivity						✔	✔
role hierarchy				✔		✔	✔
role inclusion							✔
concept assertion	✓	✓	✓	✓	✓	✓	✔
role assertion		✓	✓	✓	✓	✓	✔

Table 3.4: Logical constructors in various DLs

An interpretation $\mathcal{I}$ is a model of $C(a)$[1] iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and is a model of $r(a,b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ iff it is a model of every axiom in $\mathcal{A}$.

An Abox $\mathcal{A}$ also has a special interpretation $\mathcal{I}_{\mathcal{A}} = (\Delta^{\mathcal{I}_{\mathcal{A}}}, \cdot^{\mathcal{I}_{\mathcal{A}}})$ called *canonical interpretation* [6]. In general, any canonical model should be a representative and small model (which can be extended to other models). Such a model can be constructed according to the following definition.

Definition 3.7 (Canonical Interpretation)**.** Let $\mathcal{A}$ be an ABox and $\Delta^{\mathcal{I}_{\mathcal{A}}}$ be a non-empty finite set of the domain. Then, a *canonical interpretation* $\mathcal{I}_{\mathcal{A}}$ is induced by $\mathcal{A}$ as follows:

1. The domain $\Delta^{\mathcal{I}_{\mathcal{A}}}$ of $\mathcal{I}_{\mathcal{A}}$ consists of all individual names occurring in $\mathcal{A}$;
2. For all atomic concepts A, we define $A^{\mathcal{I}_{\mathcal{A}}} = \{x \mid A(x) \in \mathcal{A}\}$; and
3. For all roles r, we define $r^{\mathcal{I}_{\mathcal{A}}} = \{(x,y) \mid r(x,y) \in \mathcal{A}\}$.

There are two common assumptions about the ABox. The first one is the so-called *unique name assumption* (UNA), *i.e.* if $a, b \in \mathsf{Ind}$ are distinct individual names, then

[1]Several other notations for writing axioms can be found in the literature, *e.g.* $a : C$ and $\langle a, b\rangle : r$

$a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The second one is called the *open world assumption*. This is inherent in the fact that an ABox may have many models, only some of them are constrained by assertions. For example, $\mathcal{A} = \{\mathsf{Student}(ken)\}$ expresses that *ken* is a $\mathsf{Student}$ in all models. In particular, *ken* is the only one student in some models whereas there are other students in other models.

3.2.3 Knowledge Base

Having defined an TBox and an ABox, we are now ready to give the formal definition of a DL ontology. As aforementioned, an ontology consists of two parts, *viz.* a terminological part and an assertional part. Though the assertional part often boils down to an ABox $\mathcal{A}$ as defined above, the terminological part is more flexible. For example, some DLs only allow for an unfoldable TBox, some may allow for a general TBox, and also, different DLs may employ different logical constructors (*cf.* Table 3.4).

Definition 3.8 (Knowledge Base)**.** A DL knowledge base (KB) is a pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ iff $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{I}$ is a model of $\mathcal{A}$. Let $\mathcal{T}$, $\mathcal{A}$, and φ denote a TBox, an ABox, and a (terminological or assertional) axiom, respectively. Conventionally, we write $\mathcal{I} \models \mathcal{K}$, $\mathcal{I} \models \mathcal{T}$, $\mathcal{I} \models \mathcal{A}$, and $\mathcal{I} \models \varphi$ to denote that $\mathcal{I}$ is a model of $\mathcal{K}$, $\mathcal{T}$, $\mathcal{A}$, and φ, respectively.

It should be also noted that when a knowledge base does not contain an ABox, *i.e.* $\mathcal{A} = \emptyset$, we may denote the knowledge base by $\mathcal{T}$. This case may appear in practice since some ontologies, *e.g.* the medical ontology SNOMED CT, do not have the ABox.

3.3 Description Logics as First Order Fragments

As shown in Section 1.2, it is easily seen that both TBox axioms and ABox axioms can be translated into FOL formulae in a satisfiability-preserving way; and thus, DLs can be seen as notational variants of fragments of FOL. Indeed, it is well-known that most of DLs are fragments of FOL. This means that DL interpretations have the same structure as FOL interpretations if we view individual names as constants, concept names as unary predicates, and role names as binary predicates. Under this viewpoint, we can design a syntactical translation τ which, applied to a DL axiom α, yields a FOL formula $\tau(\alpha)$ such that the model sets of α and $\tau(\alpha)$ coincide. In the following, we discuss the mentioned translation and relate the results to well-known decidable fragments of FOL.

3.3.1 First Order Translation

We provide the definition of ϕ which outputs FOL formulae corresponding to DL $\mathcal{ALCH}$[1]. Technically, every $\mathcal{ALCH}$ knowledge base KB can be translated via ϕ to a theory ϕ(KB)

[1]Similar translations can be applied for more expressive DLs but we omit to discuss here.

in FOL. Hence, we define

$$\phi(\text{KB}) = \bigcup_{\alpha \in \text{KB}} \phi(\alpha) \tag{3.1}$$

i.e. we translate each axiom in KB separately to a FOL formula. Since axioms in DL knowledge base are inductively constructed from concept descriptions together with ontological constructors, two translation steps are to be done *viz.* a translation for concept description level and another one for ontological axiom level.

First, we define auxiliary function $\phi_C(y)$ for translating any concept C to a FOL formula, where y is a new variable. Each type of concepts is translated as follows:

$$\begin{gathered}
\phi_A(y) = A(y) \text{ iff } A \text{ is atomic concept.} \\
\phi_{C \sqcup D}(y) = \phi_C(y) \vee \phi_D(y) \\
\phi_{C \sqcap D}(y) = \phi_C(y) \wedge \phi_D(y) \\
\phi_{\neg C}(y) = \neg \phi_C(y) \\
\phi_{\exists r.C}(y) = \exists x.r(y,x) \wedge \phi_C(x) \\
\phi_{\forall r.C}(y) = \forall x.r(y,x) \rightarrow \phi_C(x)
\end{gathered}$$

Second, we define ϕ to translate each DL axiom α in KB (*cf.* Equation 3.1) based on the above functions as follows:

$$\begin{gathered}
\phi(r \sqsubseteq s) = \forall x, y(r(x,y) \rightarrow s(x,y)) \\
\phi(C \sqsubseteq D) = \forall x(\phi_C(x) \rightarrow \phi_D(x)) \\
\phi(C \equiv D) = \forall x(\phi_C(x) \leftrightarrow \phi_D(x)) \\
\phi(C(a)) = \phi_C(x)[x/\overline{a}] \\
\phi(r(a,b)) = r(x,y)[x/\overline{a}, y/\overline{b}]
\end{gathered}$$

where $[x_i/\overline{a}]$ denotes the substitution of an occurrence of x for FOL constant $\overline{a}$; and also, $[x/\overline{a}, y/\overline{b}]$ denotes the "simultaneous" substitution of each occurrence of x, y for FOL constant $\overline{a}, \overline{b}$, respectively.

Though DL knowledge base can be translated into FOL, the variable-free syntax of DLs is much more concise [9]. This also lends itself easily to the development of algorithms.

3.3.2 Decidable First Order Fragments

As aforementioned, most of DLs can be seen as "decidable" fragments of FOL *i.e.* preserving the decidability of the satisfiability problem in FOL. Several well-known fragments include two variable logic (denoted by FO^2)[1] [111], the guarded fragment [112], fluted logic [113, 114], and the dual of Maslov's class K [115] (see also [116] for detailed survey of these fragments). Here, we review FO^2, which is often associated with DLs.

Historically, after FOL was independently shown to be undecidable by Church [117] and Turing [118] in 1936 and 1937, respectively, logicians had an ambitious to delineate

[1]This may be sometimes denoted by $\mathcal{L}^2$ in the literature.

the boundary between decidable and undecidable fragments of FOL. One way to classify syntactic fragments is to partition formulae based on the number of variables. This way is known as k-variable logic (denoted by FO^k), *i.e.* relational first-order formulae containing at most k different individual variables, introduced by Henkin [119].

Decidability of FO^2 without equality and with equality were first shown by Scott [120] and Mortimer [111], respectively. As we can observe from Section 3.3.1, $\mathcal{ALCH}$ knowledge base can be expressed by first-order formulae with just two variables *i.e.* it is contained in FO^2. With appropriately reusing variable names, any $\mathcal{ALCH}$ concept can be translated into a FO^2 formula. For example, a direct translation of $\forall r.(\exists r.A)$ yields the formula $\forall x_1(r(x_0, x_1) \rightarrow \exists x_2(r(x_1, x_2) \wedge A(x_2)))$. Since the subformula $\exists x_2(r(x_1, x_2) \wedge A(x_2))$ does not contain x_0, then we can rename the bound variable x_2 to x_0 *i.e.* $\forall x_1(r(x_0, x_1) \rightarrow \exists x_0(r(x_1, x_0) \wedge A(x_0)))$ which uses only two variables. This connection was clearly investigated in [44]. It also shows that any extension of $\mathcal{ALCH}$ by constructors that can be expressed with the help of only two variables and its sublogic such as $\mathcal{FL}_0$ and $\mathcal{ELH}$ yields a decidable DLs. Number restrictions is an example that cannot be expressed within FO^2. Fortunately, this can be expressed by an extension of FO^2 with counting quantifiers (denoted by $\mathcal{C}^2$), which has shown to be decidable in [121].

3.4 Reasoning Services

A DL system goes beyond storing TBox axioms and ABox axioms. Indeed, it is able to perform specific kinds of reasoning services. Typically, reasoning in a DL knowledge base is a process of discovering implicit knowledge entailed by the knowledge base.

Basically, different kinds of reasoning services are defined as logical inferences. In this section, we define reasoning services w.r.t. a knowledge base. Later on, we may consider special cases where an TBox or/and an ABox is empty or where an TBox satisfies additional constraints, such as being unfoldable.

Definition 3.9 (KB Consistency)**.** Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL knowledge base where $\mathcal{T}$ be a TBox and $\mathcal{A}$ be an ABox. $\mathcal{K}$ is called *consistent* if it has a model.

Definition 3.10 (Concept Satisfiability)**.** Let $\mathcal{L}$ be a specific DL, $\mathcal{K}$ be a DL knowledge base, and $C \in \mathsf{Con}(\mathcal{L})$. Then, a concept C is called *satisfiable* w.r.t. $\mathcal{K}$ if there is a model $\mathcal{I}$ of $\mathcal{K}$ with $C^{\mathcal{I}} \neq \emptyset$.

Definition 3.11 (Concept Subsumption)**.** Let $\mathcal{L}$ be a specific DL, $\mathcal{K}$ be a DL knowledge base, and $C, D \in \mathsf{Con}(\mathcal{L})$. Then, a concept D *subsumes* a concept C w.r.t. $\mathcal{K}$ (denoted by $\mathcal{K} \models C \sqsubseteq D$ or $C \sqsubseteq_{\mathcal{K}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{K}$.

Definition 3.12 (Concept Equivalence)**.** Let $\mathcal{L}$ be a specific DL, $\mathcal{K}$ be a DL knowledge base, and $C, D \in \mathsf{Con}(\mathcal{L})$. Then, two concepts C, D are *equivalent* w.r.t. $\mathcal{K}$ (denoted by $\mathcal{K} \models C \equiv D$ or $C \equiv_{\mathcal{K}} D$) if $\mathcal{K} \models C \sqsubseteq D$ and $\mathcal{K} \models D \sqsubseteq C$.

Definition 3.13 (Instance Checking)**.** Let $\mathcal{L}$ be a specific DL, $\mathcal{K}$ be a DL knowledge base, $C \in \mathsf{Con}(\mathcal{L})$, and $a, b \in \mathsf{Ind}$. An individual a and a pair of individuals (a, b) are an

instance of a concept C and an instance of a role r w.r.t. $\mathcal{K}$ (denoted by $\mathcal{K} \models C(a)$ and $\mathcal{K} \models r(a,b)$, respectively) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, respectively, for all models $\mathcal{I}$ of $\mathcal{K}$.

The reasoning services introduced in Definition 3.9 - 3.13 are called *basic reasoning services* and should be supported by most DL systems. There are also additional services which could be implemented by a finite number of calls to the basic services. These are formally defined as follows:

Definition 3.14 (Ontology Classification)**.** Let $\mathcal{K}$ be a DL knowledge base and $\mathsf{CN}(\mathcal{K})$ be a set of concept names occurring in $\mathcal{K}$. Then, an ontology classification of $\mathcal{K}$ is the identification of subsumption between all pairs of concept names in $\mathcal{K}$, *i.e.* for all $A, B \in \mathsf{CN}(\mathcal{K})$, determines whether or not $\mathcal{K} \models A \sqsubseteq B$.

Modern DL systems often represent the result of an ontology classification in the so-called *directed acyclic graph* (DAG), where a directed edge links a concept name to an immediate subsumer. This graph is referred to as the *concept hierarchy*.

In the following, we give another formal definition which is stemmed from the instance checking. This service retrieves all individuals in a DL knowledge base that are instances of a particular concept name.

Definition 3.15 (Instance Retrieval)**.** Let $\mathcal{L}$ be a specific DL, $\mathcal{K}$ be a DL knowledge base, $\mathsf{CN}(\mathcal{K})$ be a set of concept names in $\mathcal{K}$, and $A \in \mathsf{CN}(\mathcal{K})$. Then, an *instance retrieval* for A is the computation of all individuals $a \in \mathsf{Ind}$ such that $\mathcal{K} \models A(a)$.

The above basic reasoning services may depend on each other. If $\mathcal{L}$ is closed under negation, *i.e.* the complement of any $\mathcal{L}$ concept is also an $\mathcal{L}$ concept, then all basic reasoning services can be reduced to knowledge base consistency (*cf.* Definition 3.9) [122]. For example, ones can show $C \sqsubseteq_{\mathcal{K}} D$ by showing that $(\mathcal{T}, (C \sqcap \neg D)(x))$, where x is an arbitrarily chosen individual name, is not consistent. This kind of reduction method is called *refutation*, which gives an advantage to reuse consistency checking algorithms for other reducible problems.

Another technique is eliminating the ABox and the TBox to obtain easier computation procedures. First, eliminating the ABox is based on an important characteristic of DLs. Cconcept satisfiability, concept subsumption, and concept equivalence w.r.t. a DL knowledge base $\mathcal{K}$ are often referred to as *terminological reasoning* or TBox reasoning. Typically, TBox reasoning is not influenced by the ABox $\mathcal{A}$ as long as $\mathcal{A}$ is consistent (*i.e.* it has a model). Thus, concept satisfiability, concept subsumption, and concept equivalence w.r.t. a DL knowledge base $\mathcal{K}$ simply mean concept satisfiability, concept subsumption, and concept equivalence w.r.t. a TBox $\mathcal{T}$. This discussion is formally presented in the following theorem.

Theorem 3.1 ([123])**.** Let $\mathcal{L}$ be a specific DL that does not provide the nominal constructor[1] and $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL knowledge base. Then, for every pair $C, D \in \mathsf{Con}(\mathcal{L})$, we have:

$$(\mathcal{T}, \mathcal{A}) \models C \sqsubseteq D \Longleftrightarrow \mathcal{T} \models C \sqsubseteq D$$

[1]Nominal is the set constructor (*cf.* Table 3.1)

Second, eliminating the TBox can be done through unfolding (*cf.* page 35). This can reduce the reasoning services w.r.t. a TBox to the services for independent concepts. As shown in Table 3.2, reasoning may become conceptually easier by abstracting way from the TBox or assuming that it is empty. The following theorem formally presents the use of unfolding for TBox elimination. When the TBox is eliminated, we omit to denote it.

Theorem 3.2. Let $\mathcal{L}$ be a specific DL and $\mathcal{T}$ be a TBox. Then, for every pair $C, D \in \mathsf{Con}(\mathcal{L})$, we have:

$$\mathcal{T} \models C \sqsubseteq D \Longleftrightarrow \models \mathsf{Unfold}(C, \mathcal{T}) \sqsubseteq \mathsf{Unfold}(D, \mathcal{T})$$

It is worth to note that the procedure of unfolding is only restricted to an unfoldable TBox $\mathcal{T}$ [6]. For example, if $\mathcal{T}$ is not unique, *e.g.* $\{(\mathsf{A} \equiv C), (\mathsf{A} \equiv D)\} \subseteq \mathcal{T}$, then it is not possible to make precisely the substitution for A. If $\mathcal{T}$ contains the cyclic dependency, it could lead to a non-termination problem. If $\mathcal{T}$ contains GCIs, *e.g.* $\exists r.C \sqsubseteq D$, then it could not be guaranteed that an interpretation satisfying an unfolded concepts would also satisfy these axioms.

3.5 Reasoning Algorithms

A variety of reasoning algorithms were introduced for the problems previously discussed in Section 3.4. This section briefly reviews two widely used approaches in DLs, *viz.* tableau-based approaches and structural approaches for sub-Boolean DLs.

Before looking at each algorithmic procedure, let us state the general requirements on the "behaviors" of such procedures [9] as follows:

- The procedure should be a *decision procedure*[1], meaning that it should be:
  1. *sound*, *i.e.* the positive answers should be correct,
  2. *complete*, *i.e.* the negative answers should be correct, and
  3. *terminating*, *i.e.* it should always give an answer in finite time;
- The procedure should be as "efficient" as possible. That is, it should be "optimal" w.r.t. the worst-case complexity of the problem;
- The procedure should be "practical", *i.e.* it should be easy to be implemented, be easy to be optimized, and behave well in applications.

The idea of syntactic translation to FOL (*cf.* Section 3.3) provides an upper bound for the computational complexity of particular logics. However, decision procedures obtained this way could be higher than necessary. For example, the satisfiability problem of FO^2 is NExpTime-complete whereas the satisfiability of $\mathcal{ALC}$, $\mathcal{FL}_0$, and $\mathcal{ELH}$ w.r.t. the empty TBox are PSpace-complete, polynomial, and polynomial, respectively. Hence, instead of

[1]This can be seen as a metaphorical meaning of soundness and completeness in logic (*cf.* page 26).

leveraging existing methods of automated reasoning, DL community flavors on special-purpose techniques. In this section, we discuss two well-known techniques *viz.* tableau-based (*cf.* Subsection 3.5.1) and structural-based (*cf.* Subsection 3.5.2) algorithms. Both techniques are decision procedures. We discuss each procedure in the following.

3.5.1 A Tableau Algorithm for $\mathcal{ALC}$

The tableau-based approach was first introduced in [99]. This approach mainly concentrates on knowledge base consistency since this problem can be reduced to others. For instance, given a DL knowledge $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and $C, D \in \mathsf{CN}$, we know $\mathcal{K} \models C \sqsubseteq D \iff (\mathcal{T}, \mathcal{A} \cup \{(C \sqcap \neg D)(x)\})$ is not consistent, where x is a fresh individual name[1]. The idea behind this approach is that, given a DL knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the approach checks the consistency of $\mathcal{K}$ by constructing a model of $\mathcal{K}$. The approach uses the so-called *expansion rules* to expand the model with constraints and stops when it encounters a clash, *i.e.* $\{A(x), \neg A(x)\} \subseteq \mathcal{A}$ for some individual name x and some concept name A.

As aforementioned, the satisfiability and the consistency problem (without TBox) are PSpace-complete in $\mathcal{ALC}$. The traditional tableau algorithm (as briefly described above) needs exponential space, but it can be modified such that it needs only polynomial space. However, when GCIs are admitted, the tableau-based procedure for the satisfiability and the consistency problem are ExpTime-complete. This means that it can run in worst-case non-deterministic double exponential time w.r.t. the size of an ontology. Fortunately, according to the highly optimized techniques, this approach works well in practice as many modern DL systems such as FaCT [41], FaCT^{++} [52], Racer [42], and Pellet [28] have based their developments on this procedure. Discussing about the development of this procedure is important; however, it is irrelevant to our contributions of this thesis. We refer the readers to [6, 15] for detail.

3.5.2 Structural Approaches

Although the tableau-based approaches are widely employed by modern DL systems, other approaches have been also developed as they might be suitable for some certain needs *e.g.* to analyze the worst-case running-time complexity.

When trying to find a DL with a polynomial subsumption algorithm, it is clear that such a particular DL should not provide all Boolean operators since it will inherit NP-hardness from propositional logic [9]. When ones have to decide to drop an operator, conjunction seems to be indispensable since it is used to state for different properties of a defining concept. Finally, if ones want to call that logic a DL, a constructor using roles is needed. This leads to the consideration of two minimal candidate sub-Boolean DLs, *viz.* $\mathcal{FL}_0$ and $\mathcal{EL}$. As aforementioned in Table 3.2, these DLs exhibit robust behaviors. The following discusses two characterizations for structural subsumption in detail.

[1]That is, x does not occur in $\mathcal{K}$

Subsumption in $\mathcal{FL}_0$

Here, we briefly explain how subsumption in $\mathcal{FL}_0$ can be verified. We discuss the case of subsumption based on two ways, *i.e.* without and with an unfoldable TBox in order.

Structural subsumption is based on the notion of normal forms. Informally, we use the rewrite rule $\forall r.(C \sqcap D) \rightarrow \forall r.C \sqcap \forall r.D$ together with associativity, commutativity, and idempotence of $\sqcap$ to transform any $\mathcal{FL}_0$ concept into an equivalent one of the form $\forall r_1 \ldots \forall r_m.A$ for $m \geq 0$. We also abbreviate $\forall r_1 \ldots \forall r_m.A$ by $\forall r_1 \ldots r_m.A$ where $r_1 \ldots r_m$ is viewed as a word over the alphabet of all role names. If $m = 0$, then this is an empty word, *i.e.* ϵ. In addition, instead of $\forall w_1.A \sqcap \cdots \sqcap \forall w_l.A$ we write $\forall L.A$ where $L \coloneqq \{w_1, \ldots, w_l\}$ is a finite set of words over the alphabet of all role names. Let us note that $\forall \emptyset.A$ is equivalent to $\top$. Hence, this term can be added to a conjunction without changing the meaning of the concept. Using these notions, any pair of $\mathcal{FL}_0$ concepts C, D containing the concept names $A_1, \ldots, A_k$ can be rewritten as:

$$C \equiv \forall U_1.A_1 \sqcap \cdots \sqcap \forall U_k.A_k \text{ and } D \equiv \forall V_1.A_1 \sqcap \cdots \sqcap \forall V_k.A_k \tag{3.2}$$

where U_i, V_i are two finite sets of words over the alphabet of all role names. This normal form provides us with the following characterization of subsumption:

$$C \sqsubseteq D \Longleftrightarrow U_i \supseteq V_i \text{ for all } i, 1 \leq i \leq k \tag{3.3}$$

According to [15], the size of the normal forms is polynomial in the size of the original concepts and the inclusion checking $U_i \supseteq V_i$ can also be computed in polynomial time. Hence, the above procedure can be computed in polynomial time. We illustrate how the subsumption relation between $\mathcal{FL}_0$ concepts can be verified in Example 3.4.

Example 3.4. The concept of Herbivore, whose members are animals which eat only plants, might be defined as Animal $\sqcap$ $\forall$eats.Plant. To show that Animal is subsumed by Herbivore w.r.t. an empty TBox, *i.e.* Animal $\sqsubseteq$ Herbivore, we consider the following steps:

1. Animal is translated as a form: $\forall\{\epsilon\}$.Animal $\sqcap \forall\emptyset$.Plant; and
2. Herbivore is translated as a form: $\forall\{\epsilon\}$.Animal $\sqcap \forall\{$eats$\}$.Plant.

Since $\{\epsilon\} \subseteq \{\epsilon\}$ and $\emptyset \subseteq \{$eats$\}$, we conclude that Herbivore $\sqsubseteq$ Animal. □

This characterization of subsumption via inclusion of finite sets of words can be extended to unfoldable TBox. Formally, let $\mathsf{CN}(\mathcal{T})$, $\mathsf{CN}^{\mathsf{def}}(\mathcal{T})$, and $\mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$ be a set of concept names occurring in $\mathcal{T}$, a set of defined concept names occurring in $\mathcal{T}$, and a set of primitive concept names occurring in $\mathcal{T}$, respectively. A given TBox $\mathcal{T}$ can be translated into a finite automaton $\mathcal{A}_\mathcal{T}$ with word transitions, whose states are $A \in \mathsf{CN}(\mathcal{T})$ and transitions are the corresponding value restrictions. The language $L_{\mathcal{A}_\mathcal{T}}(A, P)$ is the set of all words labeling paths in $\mathcal{A}_\mathcal{T}$ from $A \in \mathsf{CN}^{\mathsf{def}}(\mathcal{T})$ to $P \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, the following is a characterization of subsumption in $\mathcal{FL}_0$ w.r.t. an unfoldable TBox:

$$C \sqsubseteq_\mathcal{T} D \Longleftrightarrow \forall P \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) : (L_{\mathcal{A}_\mathcal{T}}(D, P) \subseteq L_{\mathcal{A}_\mathcal{T}}(C, P)) \tag{3.4}$$

Example 3.5. (Continuation of Example 3.4) We note that an unfoldable TBox $\mathcal{T}$ is defined as:

$$\begin{aligned} \mathsf{Animal}' &\equiv \mathsf{Animal} \sqcap \forall \emptyset.\mathsf{Plant} \\ \mathsf{Herbivore} &\equiv \mathsf{Animal} \sqcap \forall \mathsf{eats}.\mathsf{Plant} \end{aligned}$$

The corresponding acyclic automaton can be depicted on Figure 3.2[1]. It is not difficult to observe that $L_{\mathcal{A}_\mathcal{T}}(\mathsf{A}', \mathsf{A}) = \{\epsilon\} \subseteq L_{\mathcal{A}_\mathcal{T}}(\mathsf{H}, \mathsf{A}) = \{\epsilon\}$ and $L_{\mathcal{A}_\mathcal{T}}(\mathsf{A}', \mathsf{P}) = \{\,\} \subseteq L_{\mathcal{A}_\mathcal{T}}(\mathsf{H}, \mathsf{P}) = \{\mathsf{eats}\}$. Thus, $\mathsf{Herbivore} \sqsubseteq_\mathcal{T} \mathsf{Animal}'$.

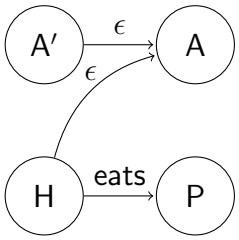


Figure 3.2: The corresponding acyclic automaton

□

We note that the original definitions of these characterizations were developed in [124]. As we shall see soon, our concrete developments of concept similarity (under the agent's preference) in $\mathcal{FL}_0$ are driven by this form of structural characterization.

Subsumption in $\mathcal{EL}$

Suppose that TBox $\mathcal{T}$ is unfoldable and $\mathcal{EL}$ concepts are fully expanded. Let an $\mathcal{EL}$ concept C is of the following form:

$$P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n \tag{3.5}$$

That concept C can be structurally transformed into the corresponding $\mathcal{EL}$ description tree. The root v_0 of the $\mathcal{EL}$ description tree $\mathcal{T}_C$ has $\{P_1, \ldots, P_m\}$ as its label and has n outgoing edges, each labeled r_j to a vertex v_j for $1 \leq j \leq n$. Then, a subtree with the root v_j is defined recursively relative to the concept C_j. In [102, 125], a characterization of subsumption for the DL $\mathcal{EL}$ w.r.t. an unfoldable TBox was proposed. Instead of considering concept descriptions, the so-called $\mathcal{EL}$ description trees corresponding to those concept descriptions are considered. The subsumption is then characterized by an existence of a homomorphism in the reverse direction (*cf.* Definition 3.16 and Theorem 3.3).

Definition 3.16 (Homomorphism [102, 125])**.** An $\mathcal{EL}$ description tree $\mathcal{T}$ is a quintuple (V, E, rt, l, ρ) where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, rt is the root, $l : V \to 2^{\mathsf{CN}^{\mathsf{pri}}}$ is a vertex labeling function, and $\rho : E \to \mathsf{RN}$ is an edge labeling function. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two $\mathcal{EL}$ description trees, $v \in V_1$ and $v_2 \in V_2$. Then, the mapping $h : V_1 \to V_2$ is a *homomorphism* from $\mathcal{T}_1$ to $\mathcal{T}_2$ iff the following conditions are satisfied:

[1]Obvious abbreviation of concept names are used for succinctness.

- For all $v \in V_1$, $l_1(v_1) \subseteq l_2(h(v_1))$; and
- For each successor w_1 of v_1 in $\mathcal{T}_1$, $h(w_1)$ is a successor of $h(v_1)$ with $\rho_1(v_1, w_1) = \rho_2(h(v_1), h(w_1))$.

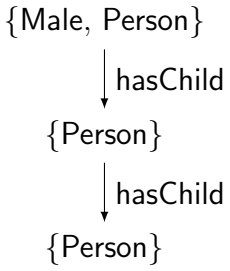
Theorem 3.3 ([102, 125])**.** Let $C, D \in \mathsf{Con}(\mathcal{EL})$ and $\mathcal{T}_C$ and $\mathcal{T}_D$ be the corresponding description trees. Then, $C \sqsubseteq D$ iff there exists a homomorphism (denoted by $h : \mathcal{T}_D \to \mathcal{T}_C$) which maps the root v of $\mathcal{T}_D$ to the root w of $\mathcal{T}_C$.

We illustrate how the subsumption relation between $\mathcal{EL}$ concepts in Example 3.6.

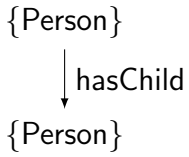
Example 3.6. Let a family TBox is given as follows: $\mathsf{GrandFather} \equiv \mathsf{Man} \sqcap \exists \mathsf{hasChild.Parent}$, $\mathsf{Man} \equiv \mathsf{Male} \sqcap \mathsf{Person}$, and $\mathsf{Parent} \equiv \mathsf{Person} \sqcap \exists \mathsf{hasChild.Person}$. By unfolding, it yields a semantically equivalent TBox $\mathcal{T}'$ as follows:

$$\begin{aligned} \mathsf{GrandFather} &\equiv \mathsf{Male} \sqcap \mathsf{Person} \sqcap \exists \mathsf{hasChild.}(\mathsf{Person} \sqcap \exists \mathsf{hasChild.Person}) \\ \mathsf{Man} &\equiv \mathsf{Male} \sqcap \mathsf{Person} \\ \mathsf{Parent} &\equiv \mathsf{Person} \sqcap \exists \mathsf{hasChild.Person} \end{aligned}$$

To show that $\mathsf{GrandFather} \sqsubseteq \mathsf{Parent}$, we construct the description tree $\mathcal{T}'_{\mathsf{GrandFather}}$ for the concept $\mathsf{GrandFather}$ (*cf.* Figure 3.3a) and the description tree $\mathcal{T}'_{\mathsf{Parent}}$ for the concept Parent (*cf.* Figure 3.3b). Following Definition 3.16, it is not difficult to identify a homomorphism from $\mathcal{T}_{\mathsf{Parent}}$ to $\mathcal{T}_{\mathsf{GrandFather}}$. Thus, $\mathsf{GrandFather} \sqsubseteq \mathsf{Parent}$.



(a) The description tree $\mathcal{T}'_{\mathsf{GrandFather}}$



(b) The description tree $\mathcal{T}'_{\mathsf{Parent}}$

Figure 3.3: The corresponding description trees of concepts $\mathsf{GrandFather}$ and Parent

□

As shown in [102, 125], this form of characterization can be decided in polynomial time (*cf.* Table 3.2). This result is not only of theoretical interest. In fact, well-known medical ontologies such as Gene Ontology [107] and SNOMED CT [105, 106] are expressible with the logic $\mathcal{EL}$. As we shall also see soon, our concrete development for concept similarity under the agent's preferences in $\mathcal{ELH}$[1] is driven by this form of structural subsumption.

[1]Strictly speaking, $\mathcal{ELH}$ extends $\mathcal{EL}$ with the role hierarchy.

3.6 Summary

- One of the most prominent decisions to be made on designing an ontology is a choice of DLs. As presented in this chapter, there are numerous DLs ranging from inexpressive sub-Boolean logics to expressive ones. Usually, the more expressive the logics are, the higher complexity they will have; and
- When a TBox is unfolded, reasoning procedures become simpler and the TBox can be ignored. Due to this advantage, our concrete development in this thesis assumes this characteristic of a knowledge base and the TBox is considered as being empty.

Chapter 4

Concept Similarity Measure in DLs

Concept similarity refers to human judgments of a degree to which a pair of concepts in question is similar. *Concept similarity measures* are computational techniques attempting to imitate the human judgments of concept similarity. Formally, they aims at identifying a degree of commonality of two given concepts and can be regarded as a generalization of the classical reasoning problem of equivalence. Studies have shown that they contribute to many applications. For instance, they were employed in bio-medical ontology-based applications to discover functional similarities of gene [107] and they were often used by ontology alignment algorithms [126].

In this chapter, we review the current state of the art for concept similarity measures. Many ideas have been proposed to automatically calculate the degree of concept similarity. Their advantages and disadvantages are immensely investigated (*cf.* Section 4.1 - 4.2). Finally, we introduce a well-defined general notion of concept similarity measures in DLs (*cf.* Section 4.3) and investigate some of its concrete developments (*cf.* Section 4.4 - 4.5).

4.1 Literature Survey

As aforementioned, many ideas have been proposed to automatically compute the degree of concept similarity. Roughly, they can be classified into five groups:

1. Path Finding;
2. Information Content;
3. Context Vector;
4. Structure Similarity; and
5. Semantic Similarity.

In the following, we discuss their relative advantages and disadvantages in detail.

4.1.1 Path Finding

This method requires to firstly construct the concept hierarchy (*cf.* Definition 3.14). That is, the more general concepts they are, the more they are closer to the root of the hierarchy. Also, the more specific concepts they are, the more they are closer to the leaves of the hierarchy. After the concept hierarchy is constructed, the degree of concept similarity can be computed from paths between concepts. In fact, there are various ways to determine the degree and we discuss each of them in the following.

In the bio-medical domain, [84] proposed a measure based on path lengths between concepts in the Medical Subject Headings (MeSH) ontology[1], which has been distributed by the National Library of Medicine. This measure found a path length between concepts according to successively either more specific concepts or less specific concepts. A similar approach was proposed in [127] in which the measure determined the degree of similarity based on the shortest path between concepts. Ones may also assign different weights to the role depth as done in [128].

In the area of natural language processing, [129] determined the path length from the most specific concept to the root of the hierarchy. The degree of similarity was then scaled by the sum of the distances between concepts in question to their most specific concept. Similarly, a measure in [130] used the shortest path between concepts in question; then, the distance was scaled by the logarithm of twice the maximum depth of the hierarchy. In both approaches, the path length was scaled somehow to avoid strict reliance on the path length. In some other areas, the distance between concepts w.r.t. their least common subsumer could be considered as performed in [131].

It is worth noting that path finding methods are simple to use since they mainly rely on the subset relation. However, these approaches have several obvious disadvantages as follows:

1. They often ignore to consider the constraints explicitly and implicitly defined in an ontology. For example, given $\mathcal{T} = \{X \equiv A \sqcap B, Y \equiv A \sqcap C, Z \equiv A \sqcap D \sqcap E \sqcap F\}$, where concepts $A, \ldots, F$ are primitive, it is reasonable to say that the similarity between X and Y is likely equal to the similarity between X and Z because they share the same parent. Nevertheless, Y may appear to be more similar to X than Z if considering the constraints in $\mathcal{T}$; and

2. When the relationships between concepts cannot be identified, none of similarity information between concepts can be obtained.

4.1.2 Information Content

The limitations addressed in Subsection 4.1.1 shows that using a path between concepts is not enough to identify the degree of similarity between concepts. One possible approach to cope with this situation is to augment concepts with a corpus-based statistics. This is known as an *information content-based* approach.

[1] https://www.nlm.nih.gov/mesh/

In general, the information content of each concept in a hierarchy is calculated based on the frequency of occurrence of that concept in a corpus. The more specific concepts they are, the higher information content values of them will be. Formally, the following equation defines how the information content for a concept C is computed:

$$IC(C) = -\log(\frac{freq(C)}{freq(root)}), \tag{4.1}$$

where $freq(C)$ is the frequency of a concept C and $freq(root)$ is the frequency of the root of the hierarchy. It should be also noted that, in order to count a frequency, each concept in the hierarchy should be mapped to lexical terms in a corpus. Following the above equation, there could be many ways to calculate the degree of similarity. For instance, [132] defined the degree of similarity between concepts as the information content of the least common subsumer of them. Intuitively, this measure was defined to calculate the degree of the shared information between concepts. Let $sim_{res}(C, D)$ denotes the similarity between concepts C and D, as defined in [132]. Its definition was given as follows:

$$sim_{res}(C, D) = IC(LCS(C, D)), \tag{4.2}$$

where $LCS(C, D)$ is the least common subsumer (LCS) between concepts C, D and IC returns the information content of that concept.

On the one hand, this approach may suit for content-based applications since the values of information content are evaluated from a relevant corpus. On the other hand, there could be several drawbacks as follows:

1. It strictly requires on a set of world descriptions such as a text corpus; and
2. Using the least common subsumer as given in Equation 4.2 may be not enough since many concept pairs may share the same least common subsumer.

However, there exists work which addressed the limitation introduced by using LCS such as [133, 134]. Mathematically, the measure in [133] calculated the similarity distance as: $dist_{jcn}(C, D) = IC(C) + IC(D) - 2 \cdot IC(LCS(C, D))$ whereas the measure in [134] computed the degree of similarity as: $sim_{lin}(C, D) = (2 \cdot IC(LCS(C, D)))/(IC(C) + IC(D))$.

4.1.3 Context Vector

On the one hand, the first two approaches utilize the concept hierarchy to compute the degree of similarity. On the other hand, this approach totally relies on the vector representation. Roughly, each concept is represented by a context vector and the cosine of the angle between vectors is used to determine the degree of similarity between related concepts.

In general, this approach starts by creating *word vectors*, which are first-order context vectors for every word in a relevant corpus. More specifically, the vector corresponding to a word w is created as follows:

1. Initialize the first-order context vector to a zero vector $\vec{w}$. The dimensions of this vector are content words derived from text sources;

2. Find every occurrence of each content word in $\vec{w}$ from a given text corpus; and

3. For each occurrence of a content word, we increment the corresponding dimension of the corresponding vector by one.

After that, the degree of similarity between concepts C and D can be computed as the cosine of the angle between their corresponding context vectors as follows:

$$sim_{vector}(C, D) = \frac{\vec{v_C} \cdot \vec{v_D}}{|\vec{v_C}| \cdot |\vec{v_D}|} \tag{4.3}$$

where $\vec{v_C}$ and $\vec{v_D}$ are context vectors corresponding to C and D, respectively, and $|\cdot|$ denotes the absolute value. Work which employs this approach includes [135–138].

The strong points of this approach are that no underlying structure is required and purely based on the empirical knowledge implicit in a given corpus. However, the approach requires computationally intensive and uses short definitions. The latter may result in its inability to investigate further on the fundamental of similarity's characteristics.

4.1.4 Structure Similarity

Similarity can be measured based on the structure or the form tied to particular concepts. This approach has been shown to be useful for many natural language processing tasks, in which sentences or words are represented by particular structures (*e.g.* strings, trees, *etc.*) and structure-based techniques are developed to measure their similarity. We review several of them *viz.* (string) edit distance, tree edit distance, and tree kernel, as follows.

Edit distance is a way of quantifying the degree of dissimilarity from one string to another one by counting the minimum number of operations required to transform one string to the other. This approach has wide range applications such as automatic spelling correction [139] and DNA analysis [140]. Different definitions of an edit distance may use different sets of string operations. For example, the Levenshtein distance uses three operations *viz.* the removal, insertion, and substitution of a character in the string. We give an example of the Levenshtein distance between "kitten" and "sitting" as follows:

1. From "kitten" to "sitten" (by substituting "s" for "k");

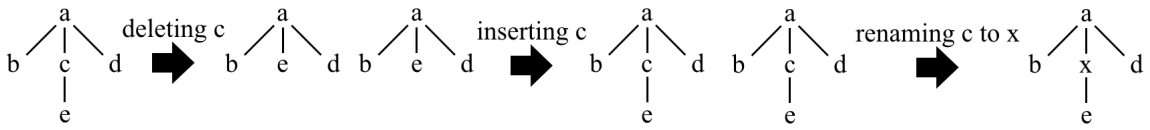
2. From "sitten" to "sittin" (by substituting "i" for "e"); and

3. From "sittin" to "sitting" (by inserting "g" at the end).

This shows that the distance between "kitten" and "sitting" is 3.

The idea of edit distance can be extended to deal with the dissimilarity between two trees. This is known as tree edit distance [141], which is analogously defined as the minimum number of node edit operations for transforming one tree to another. A rooted

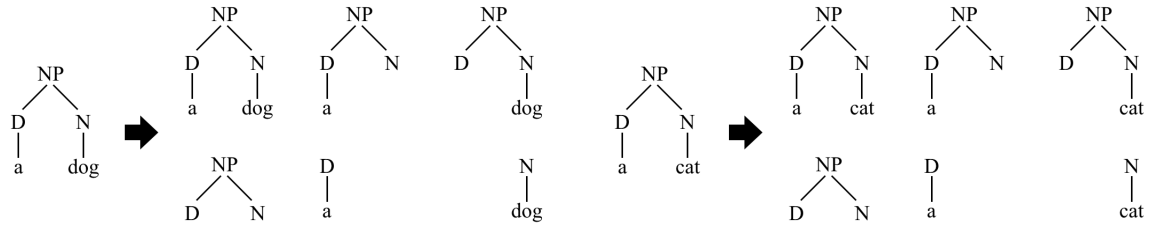
tree T is called a labeled tree if each node is assigned a set of symbols from an alphabet Σ and T is called an ordered tree if left-to-right order among siblings in T is specified. Given an ordered labeled tree T, the tree edit distance uses three operations *viz.* renaming, deletion, and insertion. First, renaming is used to change the label of a node v in T. Second, deletion removes a non-root node v in T with the parent v' and move v's children to be the children of v'. These children are inserted in the place of v so that their relative order is maintained. The last operation inserts a node v as a child of v' in T; thus, makes v to be the parent of a consecutive subsequence of the children of v'. Figure 4.1 illustrates the usage of each tree edit distance operation, where $a, \ldots, e, x$ represent tree's nodes.



(a) Deletion operation (b) Insertion operation (c) Renaming operation

Figure 4.1: Basic operations of tree edit distance

Another tree similarity technique which is often used to measure the similarity between two parse trees is called tree kernel. Informally, tree kernel measures similarity between two syntactic trees in terms of their sub-structures [142]. Suppose we want to measure the similarity between the parse trees of two phrases: "a dog" and "a cat". Figure 4.2 shows all possible sub-trees, where NP, D, and N denotes noun phrase, determinator, and noun, respectively. As 3 structures (out of 5) are completely identical, the similarity between these two phrases are 3.



(a) The parse tree and its subtrees for "a dog" (b) The parse tree and its subtrees for "a cat"

Figure 4.2: Parse trees and their sub-trees

4.1.5 Semantic Similarity

Existing approaches (as discussed above) have been proposed to automatically calculate the degree of similarity between two words or two concepts based on some particular representation of those words or concepts. Some of them may also augment such structural information with corpus based statistics. Though some approaches may satisfy several properties such as ones from metric spaces [87], they are not built to capture human's semantic representation of concepts. As opposed to similarity which calculates based

on structural information, semantic similarity relies on the meaning or semantic content of words or concepts. It is obvious that this way of calculation tends to agree with human perception. For example, most humans would agree that "bird" is more similar to "feather" than it is to "fork" or to "cat". Consequently, it is comfortable to further study on similarity's characteristics (*cf.* Section 4.2) as well as the relationship between similarity measure and the classical notion of concept equivalence.

Conventionally, semantic similarity can be estimated by using ontologies to refer to the semantic contents of concepts in question. Since DLs are the logical underpinning of ontologies, an appropriate way to calculate it should stem from three major ingredients of logics: syntax, semantics (interpretation), and proof theory (proof calculi). Consequently, we categorize existing approaches of semantic similarity in the literature into four groups *viz.* a feature-based approach, a structure-based approach, an interpretation-based approach, and a hybrid approach, as discussed in the following.

Feature-based Approach

In this approach, the notion of concept similarity is built around comprising unique features of a concept pair. Basically, their intuitions rely on the so-called *feature model* proposed by Tversky in [82]. In Tversky's model, an object was considered as a set of features. Hence, the similarity of two objects was measured by the relationship between a number of common features and a number of different features. A simple approach was developed in [143] for the DL $\mathcal{L}_0$ (*i.e.* no use of roles) and was known as *Jaccard Index*. Let $C, D \in \mathsf{Con}(\mathcal{L}_0)$. Its similarity function was mathematically expressed as follows:

$$sim_{Jaccard}(C, D) = \frac{|\widehat{C} \cap \widehat{D}|}{|\widehat{C} \cup \widehat{D}|} \tag{4.4}$$

where $\widehat{\cdot}$ and $|\cdot|$ denotes a set of conjunct concepts and the set cardinality, respectively.

A little more advanced was proposed in [144] where the notion of classical subsumption was used to determine features of each concept. This idea can be seen in Equation 4.5.

$$sim_{feature}(C, D, \mathcal{O}) = \frac{|\{C' \in \mathsf{CN} \mid \mathcal{O} \models C \sqsubseteq C'\} \cap \{D' \in \mathsf{CN} \mid \mathcal{O} \models D \sqsubseteq D'\}|}{|\{C' \in \mathsf{CN} \mid \mathcal{O} \models C \sqsubseteq C'\} \cup \{D' \in \mathsf{CN} \mid \mathcal{O} \models D \sqsubseteq D'\}|} \tag{4.5}$$

where $\mathcal{O}$ and CN represents a DL knowledge base and a set of concept names, respectively. Using subsumption technique for determining features as in [144] is simple to implement as many existing DL systems in the literature which feature subsumption reasoning can be immediately used.

Structure-based Approach

Basically, structure-based measures are defined using the syntactic form of concept descriptions to compute the degree of similarity, These measures often manipulate on a so-called *normal form*. For instance, an extension to Jaccard Index was proposed in [88]. This work also introduced important properties of concept similarity measures and suggested a general framework called *simi* which satisfied most of the properties. In *simi*,

functions and operators such as t-conorm $\bigodot$ and the fuzzy connector $\bowtie$ were to be parameterized and thus left to be specified. The framework also did not contain implementation details. This may cause implementation difficulties since merely promising properties were given and no guideline of how concrete operators could be chosen was provided. Let $C, D \in \mathsf{Con}(\mathcal{EL}\mathcal{H})$. The measure was mathematically expressed as follows:

$$simi(C, D) = simi_d(C, D) \bowtie simi_d(D, C) \tag{4.6}$$

where $\bowtie$ is the fuzzy connector defined as $\bowtie$: $[0, 1]^2 \to [0, 1]$ such that the following properties are satisfied for all $x, y \in [0, 1]$:

- $x \bowtie y = y \bowtie x$ (commutativity),
- $x \bowtie y = 1 \iff x = y = 1$ (equivalence closed),
- $x \leq y \implies 1 \bowtie x \leq 1 \bowtie y$ (weak monotonicity),
- $x \bowtie y = 0 \implies x = 0$ or $y = 0$ (bounded), and
- $0 \bowtie 0 = 0$ (grounded); and

$$simi_d(\top, \top) = simi_d(\top, D) \coloneqq 1 \tag{4.7}$$

$$simi_d(C, \top) = 0 \tag{4.8}$$

$$simi_d(C, D) = \frac{\sum\limits_{C' \in \hat{C}} [g(C') \cdot \bigodot\limits_{D' \in \hat{D}} simi_a(C', D')]}{\sum\limits_{C' \in \hat{C}} g(C')} \tag{4.9}$$

where $\hat{\cdot}$ denotes a set of conjunct concepts, a function $g : N_A \to \mathbb{R}_{>0}$ is called a *weighting function*[1], $\bigodot$ represents a bounded t-conorm, and $simi_a$ is defined as follows:

$$simi_a(A, B) = pm(A, B) \tag{4.10}$$

$$simi_a(\exists r.E, A) = simi_a(A, \exists r.E) = 0 \tag{4.11}$$

$$simi_a(\exists r.E, \exists s.F) = pm(r, s) \cdot [w + (1 - w) simi_d(E, F)] \tag{4.12}$$

where a function $pm : \mathsf{CN}^2 \cup \mathsf{RN}^2 \to [0, 1]$ is called a *primitive measure* function and $0 < w < 1$.

A similar approach which attempts on the DL $\mathcal{EL}$ was proposed in [145] in which the notion of homomorphism degree was developed according to the structural subsumption by means of tree homomorphism. It should be noted that this idea is later extended in the

[1] In [88], N_A denotes a set of atoms, *i.e.* concept names and existential restriction atoms.

thesis at the heart of *concept similarity measure under the agent's preferences* for $\mathcal{ELH}$. Let an $\mathcal{EL}$ concept $C \in \mathsf{Con}(\mathcal{EL})$ is of the following form:

$$P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n \tag{4.13}$$

where $P_i \in \mathsf{CN}^{\mathsf{pri}}$, $r_j \in \mathsf{RN}$, $C_j \in \mathsf{Con}(\mathcal{EL})$ in the same format, $1 \leq i \leq m$, and $1 \leq j \leq n$. The set $P_1, \ldots, P_n$ and the set $\exists r_1.C_1, \ldots, \exists r_n.C_n$ are denoted by $\mathcal{P}_C$ and $\mathcal{E}_C$, respectively, *i.e.* $\mathcal{P}_C = \{P_1, \ldots, P_n\}$ and $\mathcal{E}_C = \{\exists r_1.C_1, \ldots, \exists r_n.C_n\}$. Also, a concept $D \in \mathsf{Con}(\mathcal{EL})$ has the same format as C. Let $\mathcal{T}_C$ and $\mathcal{T}_D$ be the corresponding description tree of concept C and D, respectively. Then, the function sim computes the degree of similarity w.r.t. the structural homomorphism-based subsumption technique:

$$\mathsf{sim}(C, D) = \frac{\mathsf{hd}(\mathcal{T}_C, \mathcal{T}_D) + \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)}{2} \tag{4.14}$$

where the function hd computes the degree of homomorphism mapping from one description tree to one another. This function is mathematically expressed as follows:

$$\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = \mu \cdot \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu) \cdot \mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C), \tag{4.15}$$

where $\mu = |\mathcal{P}_D|/(|\mathcal{P}_D \cup \mathcal{E}_D|)$ and $|\cdot|$ represents the set cardinality;

$$\mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = \begin{cases} 1 & \text{if } \mathcal{P}_D = \emptyset \\ \frac{|\mathcal{P}_D \cap \mathcal{P}_C|}{|\mathcal{P}_D|} & \text{otherwise,} \end{cases} \tag{4.16}$$

$$\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) = \begin{cases} 1 & \text{if } \mathcal{E}_D = \emptyset \\ 0 & \text{if } \mathcal{E}_D \neq \emptyset \text{ and } \mathcal{E}_C = \emptyset \\ \sum\limits_{\epsilon_i \in \mathcal{E}_D} \frac{\max\limits_{\epsilon_j \in \mathcal{E}_C} \{\mathsf{e\text{-}hd}(\epsilon_i, \epsilon_j)\}}{|\mathcal{E}_D|}) & \text{otherwise;} \end{cases} \tag{4.17}$$

with ϵ_i, ϵ_j existential restrictions; and

$$\mathsf{e\text{-}hd}(\exists r.X, \exists s.Y) = \nu + (1 - \nu) \cdot \mathsf{hd}(\mathcal{T}_X, \mathcal{T}_Y) \tag{4.18}$$

where $0 \leq \nu < 1$.

It is worth noting that both Equations 4.6 and 4.14 look similar in the sense that they are recursive definitions for $\mathcal{EL}$ and are derived from the same observation *i.e.* the classical reasoning problem of equivalence (*cf.* Definition 3.12). There were also other approaches which used the syntactical form of concepts in other DLs. For instance, [146], [147], [148], and [149] used the negation normal form for $\mathcal{SHI}$, the $\mathcal{ALCN}$ normal form for $\mathcal{ALC}$, the $\mathcal{ALN}$ normal form for $\mathcal{ALN}$, and the $\mathcal{ELH}$ description tree for $\mathcal{ELH}$, respectively.

Interpretation-based Approach

Interpretation-based approaches are defined using an interpretation for computing the degree of concept similarity. These measures often work on the canonical interpretation

$\mathcal{I}_\mathcal{A}$ (*cf.* Definition 3.7) and the cardinality of a representative set. For example, [150] defined a function $sim_{semantics} : \mathsf{Con}(\mathcal{ALC})^2 \to [0, 1]$ as follows:

$$sim_{semantics}(C, D) = \frac{|I^{\mathcal{I}_\mathcal{A}}|}{|C^{\mathcal{I}_\mathcal{A}}| + |D^{\mathcal{I}_\mathcal{A}}| - |I^{\mathcal{I}_\mathcal{A}}|} \cdot \max(|I^{\mathcal{I}_\mathcal{A}}|/|C^{\mathcal{I}_\mathcal{A}}|, |I^{\mathcal{I}_\mathcal{A}}|/|D^{\mathcal{I}_\mathcal{A}}|) \tag{4.19}$$

where a concept I is constructed as $C \sqcap D$ and $|\cdot|$ represents the set cardinality.

Another approach was presented in [87] where the least common subsumer w.r.t. a TBox (*aka. Good Common Subsumer* or *GCS*) is considered. As aforementioned, let $\mathcal{I}_\mathcal{A}$ be the canonical interpretation. The function $sim_{GCS} : \mathsf{Con}(\mathcal{ALE})^2 \to [0, 1]$ was formally expressed as follows:

$$sim_{GCS}(C, D) = \frac{\min(|C^{\mathcal{I}_\mathcal{A}}|, |D^{\mathcal{I}_\mathcal{A}}|)}{|GCS(C, D)^{\mathcal{I}_\mathcal{A}}|} \cdot (1 - \frac{|GCS(C, D)^{\mathcal{I}_\mathcal{A}}|}{|\Delta^{\mathcal{I}_\mathcal{A}}|} \cdot (1 - \frac{\min(|C^{\mathcal{I}_\mathcal{A}}|, |D^{\mathcal{I}}|)}{|GCS(C, D)^{\mathcal{I}_\mathcal{A}}|})) \tag{4.20}$$

where $|\cdot|$ represents the set cardinality.

Obviously, these measures can give precise outputs if the canonical interpretation is available. Unfortunately, not every ontology is made up with an ABox in practice. For instance, SNOMED CT merely contains a terminology part. Hence, such ontologies are not applicable with the above measures since the notion $\mathcal{I}_\mathcal{A}$ is not constructible.

Hybrid Approach

This approach uses a combination of the above to calculate the degree of similarity between concepts. For example, [147, 148] used the structure-based approach to compute the degree of similarity between defined concept names and used the canonical interpretation to compute the degree of similarity between corresponding primitive concept names.

4.2 Desirable Properties

Various models were proposed to calculate the degree of similarity. One approach to evaluate such models is to investigate their intended behaviors. These are recognized as *desirable properties* [82, 87, 88]. For instance, let δ be a metric distance function that assigns to every pair of points a non-negative number. Then, δ satisfies:

- minimality iff $\delta(a, b) \geq \delta(a, a) = 0$;
- symmetry iff $\delta(a, b) = \delta(b, a)$; and
- triangular inequality iff $\delta(a, b) + \delta(b, c) \geq \delta(a, c)$.

Indeed, the above properties are from *metric space* and δ can be regarded as a measure of dissimilarity. They are very controversial and we discuss their intuitions in the following.

Firstly, the minimality implies that the similarity between an object and itself is the same for all objects. However, this characterization may not hold for all measures. For example, considering clinical measures, it is not constant for all stimuli that identical

stimuli are judged as the same. If we consider this as a required property of measures, then such stimuli violate this property.

Secondly, perceiving similarity judgment as a form of "a is like b" may invoke the feeling of directional similarity. That is, a is a subject and b is a referent. In this sense, the reversion of a statement (*e.g.* "b is like a") may be not equivalent. Humans tend to select the prototype as a referent and the variant as a subject. For example, it is natural saying that "the portrait resembles the person" rather than "the person resembles the portrait". Also, we usually say "the son resembles the father" rather than "the father resembles the son". If we consider this kind of perception as a required property of measures, then it may violate the symmetric property.

Lastly, the triangular inequality asserts that one distance must be smaller than or equal to the sum of the corresponding others. In other words, if "a is quite similar to b" and "b is quite similar to c", then "a and c cannot be very dissimilar from each other". That is, it sets a threshold to the similarity between a and c in terms of the similarity between a and b and between b and c. However, this may be not hold for all cases. For instance, "Russia is similar to China (due to geographical proximity)" and "China is similar to Singapore (due to Chinese population)"; however, "Russia may be not similar to Singapore at all".

Two important points, which are strongly related to the second issue of this thesis, should be mentioned according to the above discussion. First, similarity made by cognitive agents may be "subjective" since each cognitive agent may have different perspectives for his/her similarity judgment. Second, desirable properties of similarity may be controversial. This in fact relates to a point of view when the notion of similarity is investigated. For example, many research on concept similarity in DLs favor on symmetry (*e.g.* [87–89, 92, 147, 148, 150–152]) whereas [146, 153] prefer asymmetry. In fact, the former one agrees on that because reading a pair of concept description from left-to-right or vice versa does not introduce the focus whereas the latter one tends to follow an approach from cognitive science. In addition, we notice that the counterexamples discussed above inherently use preferences in the identification of similarity degree. For example, an uttering agent may have his/her personal perception before speaking that "his son resembles the father". To address this observation, we seek to explicate and theorize the notion of the agent's preferences relevant to concept similarity. Several desirable properties are redefined and introduced to gain the better understanding and easier computation when similarity measure is used w.r.t. the agent's preferences. For instance, instead of saying "his son resembles the father", we rather view "when the particular perception is fixed, his son and the father are similar to each other". We continue this point in Chapter 5.

4.3 Formal Notion of Concept Similarity

We begin to address the first issue of the thesis in this section. As aforementioned, there is substantial work for similarity judgment ranging from a kind of introspective folk psychology (*e.g.* Tversky's model) to numeric computational techniques. In this work, we base our investigation on the definition of logical equivalence. That is, let $\mathcal{T}$ be a TBox

and $C, D \in \mathsf{Con}(\mathcal{L})$ for a particular DL $\mathcal{L}$, then (*cf.* Definition 3.12):

$$C \equiv_{\mathcal{T}} D \Longleftrightarrow C \sqsubseteq_{\mathcal{T}} D \text{ and } D \sqsubseteq_{\mathcal{T}} C \tag{4.21}$$

Intuitively, concept equivalence can be seen as an operation for comparing two concept descriptions. For instance, if two concepts are equivalent, then the concept equivalence yields 1; or yields 0 otherwise. Figure 4.3 illustrates this observation.

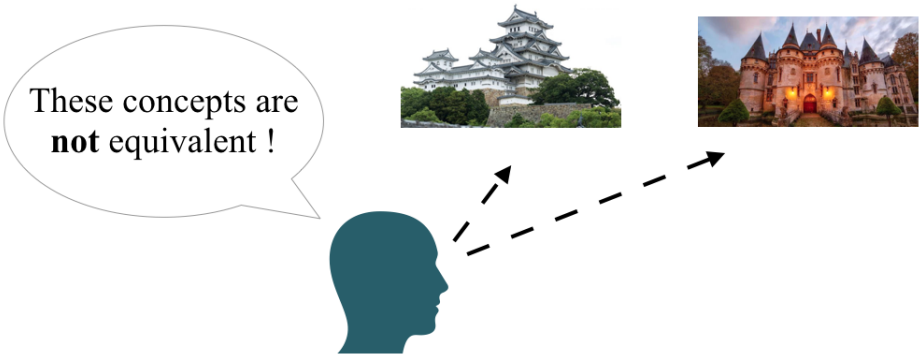


Figure 4.3: Concept equivalence can be seen as an operation for comparing concepts

Unfortunately, concept equivalence cannot give any information to a case of two nonequivalent concepts sharing some commonalities. Regarding this observation, it is very natural to generalize concept equivalence and view this generalization as *concept similarity measure*. That is, we regard two equivalent concepts as being *total similarity* and indicate this relationship with the value 1. On the other hand, two nonequivalent concepts having no commonalities at all are regarded as *total dissimilarity* and are represented by 0. Basically, the higher the value is mapped to, the more likely similarity of them may hold. We give a formal definition for this observation in the following.

Definition 4.1. Given two concept descriptions $C, D \in \mathsf{Con}(\mathcal{L})$, a *concept similarity measure* w.r.t. a TBox $\mathcal{T}$ is a function $\sim_{\mathcal{T}} : \mathsf{Con}(\mathcal{L}) \times \mathsf{Con}(\mathcal{L}) \to [0, 1]$ such that $C \sim_{\mathcal{T}} D = 1$ iff $C \equiv_{\mathcal{T}} D$ (*total similarity*) and $C \sim_{\mathcal{T}} D = 0$ indicates *total dissimilarity* between C and D.

When a TBox $\mathcal{T}$ is clear from the context, we simply write $\sim$. Furthermore, to avoid confusion on the symbols, $\sim_{\mathcal{T}}$ is used when referring to arbitrary measures. An interesting question to this observation is that "how concrete measures for particular DLs can be generalized from the logical notion of equivalence?" This thesis addresses the question by the following analogous characterization:

$$C \sim_{\mathcal{T}} D = 1 \Longleftrightarrow C \rightsquigarrow_{\mathcal{T}} D = 1 \text{ and } D \rightsquigarrow_{\mathcal{T}} C = 1 \tag{4.22}$$

That is, two concepts are totally similar iff the degree of *directional subsumption* from a concept to another one is 1 and vice versa. In Equation 4.22, the notion of directional subsumption degree is denoted by $\rightsquigarrow_{\mathcal{T}}$. It should be noted that the logical conjunction

"and" in the equation should also be generalized in such a way that two numerical values are aggregated and result in a unit interval [0, 1].

As discussed in page 57, similarity may be subjective. Then, the notion of directional subsumption degree should also be generalized in such a way that the agent's preferences can be taken into account. We outline our methodology to develop concrete measures corresponding to our observations as follows:

1. Generalize the notion of concept subsumption to the notion of *subsumption degree* (*cf.* Chapter 4), which contributes to a concrete concept similarity measure as its immediate outcome;

2. Generalize the notion of subsumption degree to the notion of *subsumption degree under the agent's preferences* (*cf.* Chapter 5); and

3. Generalize the logical conjunction (*i.e.* "and") for aggregating two numerical values to result in a unit interval (*cf.* Chapter 4 - 5).

We address each of the above steps in the following subsequent chapters. Basically, classical subsumption reasoning techniques are investigated under scrunity in this thesis. Two logics $\mathcal{FL}_0$ and $\mathcal{ELH}$ are our primary focuses in this work to develop efficient algorithms for measuring concept similarity under the agent's preferences. Taking into account the efficiency, we should not consider the particular DLs which offer all boolean operations since they would inherit NP-hardness from propositional logic.

Two logics $\mathcal{FL}_0$ and $\mathcal{ELH}$ are investigated in this thesis with their goals to find out concept similar measures which can be computed efficiently *i.e.* in polynomial time. Intuitively, computational approaches introduced in this chapter are derived from the scrutiny of structural subsumption approaches in both logics. Section 4.4 and Section 4.5 describe approaches of computing subsumption degree in $\mathcal{FL}_0$ and $\mathcal{ELH}$, respectively.

4.4 From Concept Subsumption to Subsumption Degree in $\mathcal{FL}_0$

For self-containment of this chapter, we summarize basic steps for checking subsumption of $\mathcal{FL}_0$ concepts as follows (*cf.* Subsubsection 3.5.2 for detail):

1. Concepts are fully expanded and are normalized into one of the form $\forall r_1 \dots r_n.A$ where $r_1 \dots r_n$ is seen as a word over the alphabet of all role names and $A \in \mathsf{CN}$;

2. Concepts are grouped into one of the form: $\forall U_1.A_1 \sqcap \dots \sqcap \forall U_k.A_k$ where $U_1, \dots, U_k$ denotes finite sets of words over the alphabet of role names and $\{A_1, \dots, A_k\} \subseteq \mathsf{CN}$. The empty word is also denoted by ϵ; and

3. Given concepts C, D are normalized as $\forall U_1.A_1 \sqcap \dots \sqcap \forall U_k.A_k$ and $\forall V_1.A_1 \sqcap \dots V_k.A_k$, we conclude that $C \sqsubseteq D$ holds iff $U_i \supseteq V_i$ for all i where $1 \leq i \leq k$.

Let us demonstrate how ones can employ the approach to check if subsumption relation holds between $\mathcal{FL}_0$ concepts with the following example and shade some light of the possibility to compute the subsumption degree.

Example 4.1. Consider two concept descriptions

$$\forall r.P \sqcap \forall r.Q \sqcap \forall r.\forall s.P \sqcap \forall s.Q \quad \text{and} \quad \forall r.\forall s.\forall r.P \sqcap \forall s.Q \sqcap \forall r.P$$

Let us denote the former concept by C and the latter concept by D. Following the steps mentioned earlier, we can transform both concepts to ones of the forms: $\forall\{r, rs\}.P \sqcap \forall\{r, s\}.Q$ and $\forall\{r, rsr\}.P \sqcap \forall\{s\}.Q$, respectively.

Then, we can conclude that $C \not\sqsubseteq D$ because $\{r, rsr\} \not\subseteq \{r, rs\}$. Furthermore, we also conclude that $D \not\sqsubseteq C$ because $\{r, rs\} \not\subseteq \{r, rsr\}$. □

Though we conclude that the subsumption relations between two concepts do not hold, we can notice that they have some finite sets of words in common. For instance, considering the primitive concept name Q, it appears that $\{s\} \subseteq \{r, s\}$. This observation leads us to develop approaches for computing the subsumption degree between $\mathcal{FL}_0$ concepts.

An interesting question to this observation could be "how computational procedures should look like?". To counter this question, we develop two computational procedures *viz. skeptical subsumption degree* and *credulous subsumption degree*, which can be used to identify the subsumption degree w.r.t. the structure of normalized concepts. We also formally study the relationship underlying between these two procedures in this chapter.

4.4.1 Skeptical Subsumption Degree

Let us reconsider Example 4.1. The set inclusion relation $\{r, rsr\} \subseteq \{r, rs\}$ does not hold but the relation $\{s\} \subseteq \{r, s\}$ holds. Ones may regard this as *partial subsumption* relation. Basically, the skeptical subsumption degree adopts this viewpoint to develop the computational procedure, *i.e.* set inclusions between set of words.

Definition 4.2 (Skeptical $\mathcal{FL}_0$ Subsumption Degree)**.** Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$ be in their normal forms and $\mathsf{W}(E, A)$ be a set of words w.r.t. the concept E and the primitive A. Then, a *skeptical* $\mathcal{FL}_0$ *degree* from C to D (denoted by $C \rightsquigarrow^s D$) is defined as follows:

$$C \rightsquigarrow^s D = \frac{|\{P \in \mathsf{CN}^{\mathsf{pri}} \mid \mathsf{W}(D, P) \subseteq \mathsf{W}(C, P)\}|}{|\mathsf{CN}^{\mathsf{pri}}|}, \tag{4.23}$$

where $|\cdot|$ denotes the set cardinality.

Example 4.2. (Continuation of Example 4.1) The skeptical $\mathcal{FL}_0$ subsumption degree from C to D can be calculated as follows:

$$C \rightsquigarrow^s D = \frac{|\{Q\}|}{|\{P, Q\}|} = \frac{1}{2} = 0.5$$

Similarly, $D \rightsquigarrow^s C$ can be calculated as follows:

$$D \rightsquigarrow^s C = \frac{|\emptyset|}{|\{P, Q\}|} = \frac{0}{2} = 0$$

□

The example shows that the subsumption degree from concept C to concept D is 0.5 even though concept C is not subsumed by concept D. However, the subsumption degree from concept D to concept C is 0, which means D is unrelated to C at all.

4.4.2 Credulous Subsumption Degree

With the more granularity of degree computing, we may calculate the proportion between sets of words rather than checking the set inclusions between them. To demonstrate this, we reconsider Example 4.2. Considering the set inclusion $\{r, rs\} \subseteq \{r, s\}$, it is clear that the relation does not hold. However, rather than regarding this as 0, we may take into account the proportion between them, *i.e.* $(|\{r, rs\} \cap \{r, s\}|)/(|\{r, s\}|) = 1/2$. We adopt this viewpoint to develop the credulous subsumption degree.

Definition 4.3 (Credulous $\mathcal{FL}_0$ Subsumption Degree)**.** Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$ be in their normal forms and $\mathsf{W}(E, A)$ be a set of words w.r.t. the concept E and the primitive A. Then, a *credulous $\mathcal{FL}_0$ subsumption degree* from C to D (denoted by $C \rightsquigarrow^c D$) is defined as follows:

$$C \rightsquigarrow^c D = \frac{\sum_{P \in \mathsf{CN}^{\mathsf{pri}}} \mu(D, C, P)}{|\mathsf{CN}^{\mathsf{pri}}|}, \tag{4.24}$$

where $|\cdot|$ denotes the set cardinality and $\mu(D, C, P) =$

$$\begin{cases} 1 & \text{if } \mathsf{W}(D, P) = \emptyset \\ \frac{|\mathsf{W}(D,P) \cap \mathsf{W}(C,P)|}{|\mathsf{W}(D,P)|} & \text{otherwise} \end{cases} \tag{4.25}$$

Example 4.3. From Example 4.1, the credulous $\mathcal{FL}_0$ subsumption degree from C to D can be calculated as follows:

$$C \rightsquigarrow^c D = \frac{\frac{|\{rsr, r\} \cap \{r, rs\}|}{|\{rsr, r\}|} + \frac{|\{s\} \cap \{r, s\}|}{|\{s\}|}}{|\{P, Q\}|} = \frac{\frac{1}{2} + \frac{1}{1}}{2} = 0.75$$

Whereas $D \rightsquigarrow^c C$ can be calculated as follows:

$$D \rightsquigarrow^c C = \frac{\frac{|\{rsr, r\} \cap \{r, rs\}|}{|\{r, rs\}|} + \frac{|\{s\} \cap \{r, s\}|}{|\{r, s\}|}}{|\{P, Q\}|} = \frac{\frac{1}{2} + \frac{1}{2}}{2} = 0.5$$

□

According to the example, it speaks out that C is partially subsumed by D in the degree of 0.75 whereas D is partially subsumed by C in the degree of 0.5.

4.4.3 Properties underlying Skeptical Subsumption Degree and Credulous Subsumption Degree

The fact that there exist more than one computational approaches for identifying a degree of subsumption corresponds to different skepticism of a rational agent's judgment. For example, an agent may say that C is partially subsumed by D in the degree of 0.5 whereas another agent may say to the same concepts that their partial subsumption degree is 0.75.

Regarding this point, we are interested to formally investigate the underlying relationship between the functions $\rightsquigarrow^s$ and $\rightsquigarrow^c$. It is worth observing that if $(|\mathsf{W}(D,P) \cap \mathsf{W}(C,P)|)/(|\mathsf{W}(D,P)|) = 1$, then $\mathsf{W}(D,P) \subseteq \mathsf{W}(C,P)$ holds (and vice versa). Using this observation, we can show the following property.

Proposition 4.1. For any $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, it follows that $C \rightsquigarrow^s D \leq C \rightsquigarrow^c D$.

Proof. Fix any $C, D \in \mathsf{Con}(\mathcal{FL}_0)$. We show the following inequality:

$$|\{P \in \mathsf{CN}^{\mathsf{pri}} \mid \mathsf{W}(D,P) \subseteq \mathsf{W}(C,P)\}| \leq \sum_{P \in \mathsf{CN}^{\mathsf{pri}}} \frac{|\mathsf{W}(D,P) \cap \mathsf{W}(C,P)|}{|\mathsf{W}(D,P)|}$$

Fix any $P \in \mathsf{CN}^{\mathsf{pri}}$. We show inequality of the following three cases.

Case 1 (both $\mathsf{W}(D,P)$ and $\mathsf{W}(C,P)$ are identical): Let $\mathsf{W}(D,P) = \{r_1, \ldots, r_n\}$ and $\mathsf{W}(C,P) = \{r_1, \ldots, r_n\}$. Then, we show $(\mathsf{W}(D,P) \subseteq \mathsf{W}(C,P)) \leq (|\mathsf{W}(D,P) \cap \mathsf{W}(C,P)|)/(|\mathsf{W}(D,P)|) \Longleftrightarrow 1 \leq 1$.

Case 2 (both $\mathsf{W}(D,P)$ and $\mathsf{W}(C,P)$ share some commonalities): Let $\mathsf{W}(D,P) = \{r_1, \ldots, r_n, s_1, \ldots, s_m\}$ and $\mathsf{W}(C,P) = \{r_1, \ldots, r_n, t_1, \ldots, t_o\}$. Then, we show $(\mathsf{W}(D,P) \subseteq \mathsf{W}(C,P)) \leq (|\mathsf{W}(D,P) \cap \mathsf{W}(C,P)|)/(|\mathsf{W}(D,P)|) \Longleftrightarrow 0 \leq (n)/(n+m)$.

Case 3 (both $\mathsf{W}(D,P)$ and $\mathsf{W}(C,P)$ do not share any commonalities) Let $\mathsf{W}(D,P) = \{s_1, \ldots, s_m\}$ and $\mathsf{W}(C,P) = \{t_1, \ldots, t_o\}$. Then, we show $(\mathsf{W}(D,P) \subseteq \mathsf{W}(C,P)) \leq (|\mathsf{W}(D,P) \cap \mathsf{W}(C,P)|)/(|\mathsf{W}(D,P)|) \Longleftrightarrow 0 \leq (0/m)$. ❑

Definition 4.4 (Ordering of Functions)**.** Let α and β be different functions. Then, α is more *skeptical than or equal to* β (denoted by $\alpha \preceq \beta$) if $(C\ \alpha\ D) \leq (C\ \beta\ D)$ for all concepts $C, D \in \mathsf{Con}(\mathcal{L})$.

Proposition 4.2. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$. Then, the following ordering holds:

$$\sqsubseteq \preceq \rightsquigarrow^s \preceq \rightsquigarrow^c$$

Proof. Let us view $\sqsubseteq$ as a function which returns 1 if $C \sqsubseteq D$ holds for any C, D or 0 otherwise. If $C \sqsubseteq D$ holds, then $C \rightsquigarrow^s D = C \rightsquigarrow^c D = 1$. Otherwise, it immediately follows from Proposition 4.1 that $C \sqsubseteq D \leq C \rightsquigarrow^s D \leq C \rightsquigarrow^c D$ together with considering $C \not\sqsubseteq D$ as the value 0. ❑

Theorem 4.1. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$. Then, the following are equivalent:

1. $C \sqsubseteq D$;
2. $C \rightsquigarrow^s D = 1$; and
3. $C \rightsquigarrow^c D = 1$.

Proof. Let $C \coloneqq \forall L_1.A_1 \sqcap \cdots \sqcap \forall L_n.A_n$ and $D \coloneqq \forall M_1.A_1 \sqcap \cdots \sqcap \forall M_n.A_n$. We need to show $C \sqsubseteq D \Longleftrightarrow C \rightsquigarrow^s D = 1$ and $C \rightsquigarrow^s D = 1 \Longleftrightarrow C \rightsquigarrow^c D = 1$.
($1 \Longrightarrow 2$) Assume $C \sqsubseteq D$ *i.e.* $M_i \subseteq L_i$ for $i = 1, \ldots, n$. Then, we have $C \rightsquigarrow^s D = 1$.
($2 \Longrightarrow 1$) Assume $C \rightsquigarrow^s D = 1$. This implies that $M_i \subseteq L_i$ for $i = 1, \ldots, n$. Thus, we conclude $C \sqsubseteq D$.
($2 \Longrightarrow 3$) Assume $C \rightsquigarrow^s D = 1$. We have $C \rightsquigarrow^c D = 1$ (by Proposition 4.1).
($3 \Longrightarrow 2$) Assume $C \rightsquigarrow^c D = 1$. This implies that $M_i \subseteq L_i$ for $i = 1, \ldots, n$. Thus, we conclude $C \rightsquigarrow^s D = 1$. ❏

In the following, we show that both $\rightsquigarrow^s$ and $\rightsquigarrow^c$ can be computed in polynomial time.

Theorem 4.2. Let L, M be sets of words over the alphabet of role names corresponding to concepts C, D, respectively. The computational complexity of both $\rightsquigarrow^s$ and $\rightsquigarrow^c$ is $\mathcal{O}(n|M||L|)$, where n is the size of concepts C, D.

Proof. Let $C \coloneqq \forall L_1.A_1 \sqcap \cdots \sqcap \forall L_n.A_n$ and $D \coloneqq \forall M_1.A_1 \sqcap \cdots \sqcap \forall M_n.A_n$, where L_i, M_i ($1 \leq i \leq n$) are sets of words over the alphabet of role names. Checking the inclusion of finite languages (*cf.* Definition 4.2) and the proportion of finite languages (*cf.* Definition 4.3) can be done in polynomial time, *i.e.* in the worst case we have to check for all words $w \in M_i$ and $v \in L_i$ whether $w = v$. Each equality checking can be done in $\min(|w|, |v|)$ and such tests have to be done for $|M_i| \cdot |L_i|$. Assume in the worst case that each M_i and L_i are identical *i.e.* $|M_i| = |M|$ and $|L_i| = |L|$ for every i. Therefore, we have shown that both measures are bound by $\mathcal{O}(n|M||L|)$. ❏

4.5 From Concept Subsumption to Subsumption Degree in $\mathcal{ELH}$

Now, we concentrate on a computational approach for identifying the subsumption degree between $\mathcal{ELH}$ concepts. We have reviewed the basis of structural subsumption for $\mathcal{EL}$ in Subsubsection 3.5.2. Since $\mathcal{ELH}$ is a superlogic of $\mathcal{EL}$, its structural subsumption procedure can be slightly modified from $\mathcal{EL}$ as follows:

1. Concepts are fully expanded to the form $P_1 \sqcap \ldots P_m \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n$;
2. Fully expanded concepts are structurally transformed into the corresponding description trees, where its root has $\{P_1, \ldots, P_m\}$ as its label, has n outgoing edges, each labeled by the set $\mathcal{R}_{r_j}$ of all r_j's super roles to a vertex v_j for $1 \leq j \leq n$.

Formally, $\mathcal{R}_r = \{s \mid r \sqsubseteq^* s\}$ and $r \sqsubseteq^* s$ if $r = s$ or $r_i \sqsubseteq r_{i+1} \in \mathcal{T}$ where $1 \leq i \leq n$, $r_1 = r$, $r_n = s$. That is, $\sqsubseteq^*$ denotes a transitive closure of $\sqsubseteq$ between roles. Then, a subtree with the root v_j is defined recursively relative to the concept C_j; and

3. Given two description trees $\mathcal{T}_C, \mathcal{T}_D$, we conclude that $C \sqsubseteq D$ holds iff there exists a homomorphism from $\mathcal{T}_D$ to $\mathcal{T}_C$ according to the following definition and theorem.

Definition 4.5 (Homomorphism [102, 125])**.** An $\mathcal{ELH}$ description tree $\mathcal{T}$ is a quintuple (V, E, rt, l, ρ) where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, rt is the root, $l : V \to 2^{\mathsf{CN}^{\mathsf{pri}}}$ is a vertex labeling function, and $\rho : E \to 2^{\mathsf{RN}}$ is an edge labeling function. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two $\mathcal{ELH}$ description trees, $v \in V_1$ and $v_2 \in V_2$. Then, the mapping $h : V_1 \to V_2$ is a *homomorphism* from $\mathcal{T}_1$ to $\mathcal{T}_2$ iff the following conditions are satisfied:

- For all $v \in V_1$, $l_1(v_1) \subseteq l_2(h(v_1))$; and
- For each successor w_1 of v_1 in $\mathcal{T}_1$, $h(w_1)$ is a successor of $h(v_1)$ with $\rho_1(v_1, w_1) \subseteq \rho_2(h(v_1), h(w_1))$.

Theorem 4.3. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and $\mathcal{T}_C$ and $\mathcal{T}_D$ be the corresponding description trees. Then, $C \sqsubseteq D$ iff there exists a homomorphism (denoted by $h : \mathcal{T}_D \to \mathcal{T}_C$) which maps the root v of $\mathcal{T}_D$ to the root w of $\mathcal{T}_C$.

Let us demonstrate how ones can employ the approach to check if subsumption relation holds between $\mathcal{ELH}$ concepts with the following example and shade some light of the possibility to compute the subsumption degree.

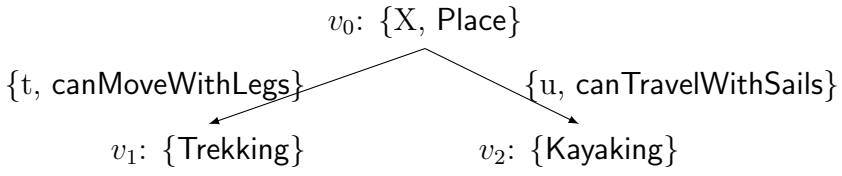
Example 4.4. An agent A wants to visit a place for doing some physical activities (*i.e.* ActivePlace). Suppose that a place ontology is modeled as follows. The classical reasoning of subsumption may be used to find out a concept subsumed by ActivePlace.

$$\begin{aligned}
\mathsf{ActivePlace} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking} \sqcap \exists \mathsf{canSail.Kayaking} \\
\mathsf{Mangrove} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking} \\
\mathsf{Beach} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canSail.Kayaking} \\
\mathsf{canWalk} &\sqsubseteq \mathsf{canMoveWithLegs} \\
\mathsf{canSail} &\sqsubseteq \mathsf{canTravelWithSails}
\end{aligned}$$

Following the above steps, each primitive definition is transformed to a corresponding equivalent full definition and the corresponding description tree is constructed accordingly.

$$\begin{aligned}
\mathsf{ActivePlace} &\equiv X \sqcap \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking} \sqcap \exists \mathsf{canSail.Kayaking} \\
\mathsf{Mangrove} &\equiv Y \sqcap \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking} \\
\mathsf{Beach} &\equiv Z \sqcap \mathsf{Place} \sqcap \exists \mathsf{canSail.Kayaking}
\end{aligned}$$

where X, Y, and Z are fresh primitive concept names. $\mathsf{canWalk} \equiv t \sqcap \mathsf{canMoveWithLegs}$ and $\mathsf{canSail} \equiv u \sqcap \mathsf{canTravelWithSails}$, where t and u are fresh primitive role names. In other words, $\mathcal{R}_{\mathsf{canWalk}} = \{t, \mathsf{canMoveWithLegs}\}$ and $\mathcal{R}_{\mathsf{canSail}} = \{u, \mathsf{canTravelWithSails}\}$. Figure 4.4a - 4.4c depict $\mathcal{T}_{\mathsf{ActivePlace}}$, $\mathcal{T}_{\mathsf{Mangrove}}$, and $\mathcal{T}_{\mathsf{Beach}}$, respectively.



(a) The description tree of concept ActivePlace

(b) The description tree of concept Mangrove

(c) The description tree of concept Beach

Figure 4.4: The description tress of concepts $\mathcal{T}_{\mathsf{ActivePlace}}$, $\mathcal{T}_{\mathsf{Mangrove}}$, and $\mathcal{T}_{\mathsf{Beach}}$

It is not difficult to find a failed attempt of identifying a homomorphism mapping the the root of $\mathcal{T}_{\mathsf{ActivePlace}}$ to the root of $\mathcal{T}_{\mathsf{Mangrove}}$, *i.e.* $h : \mathcal{T}_{\mathsf{ActivePlace}} \not\rightarrow \mathcal{T}_{\mathsf{Mangrove}}$. Hence, this infers Mangrove $\not\sqsubseteq$ ActivePlace. Similarly, we can conclude that Beach $\not\sqsubseteq$ ActivePlace.

□

Though we conclude that subsumption relations from ActivePlace to Mangrove and from ActivePlace to Beach do not hold, we can notice that they have some commonalities among their structures. For instance, considering the roots of $\mathcal{T}_{\mathsf{ActivePlace}}$ and $\mathcal{T}_{\mathsf{Beach}}$, it appears that Place is belonged to both $\{X, \mathsf{Place}\}$ and $\{Y, \mathsf{Place}\}$. This observation leads us to develop approaches for computing the subsumption degree between $\mathcal{ELH}$ concepts. In the next subsection, a homomorphism-based structural subsumption degree function is discussed. And, its properties are investigated accordingly.

4.5.1 Homomorphism Degree

Let us reconsider Example 4.4. It is obvious that $h : \mathcal{T}_{\mathsf{ActivePlace}} \not\rightarrow \mathcal{T}_{\mathsf{Mangrove}}$ holds due to $\{X, \mathsf{Place}\} \not\subseteq \{Y, \mathsf{Place}\}$. However, Place appears to be in common on both sets. Ones may regard this as *partial mapping* from $\mathcal{T}_{\mathsf{ActivePlace}}$ to $\mathcal{T}_{\mathsf{Mangrove}}$. Intuitively, the homomorphism degree function adopts this viewpoint to develop the computational procedure.

Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ be fully expanded concept of the form: $P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n$. We denote the set $P_1, \ldots, P_m$ of the concepts C, D and the set $\exists r_1.C_1, \ldots, \exists r_n.C_n$ of the concepts C, D by $\mathcal{P}_C, \mathcal{P}_D$ and $\mathcal{E}_C, \mathcal{E}_D$, respectively. The super roles $\mathcal{R}_r, \mathcal{R}_s$ are as defined on page 64. The following definition extends Theorem 4.3 to the case where no such homomorphism exists but there is some commonality.

Definition 4.6 (Homomorphism Degree [151])**.** Let $\mathbf{T}^{\mathcal{ELH}}$ be a set of all $\mathcal{ELH}$ description trees and $\mathcal{T}_C, \mathcal{T}_D \in \mathbf{T}^{\mathcal{ELH}}$ correspond to two $\mathcal{ELH}$ concept names C and D, respectively. The *homomorphism degree* function $\mathsf{hd} : \mathbf{T}^{\mathcal{ELH}} \times \mathbf{T}^{\mathcal{ELH}} \rightarrow [0, 1]$ is inductively defined as follows:

$$\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = \mu \cdot \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu) \cdot \mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C), \tag{4.26}$$

where $\mu = |\mathcal{P}_D|/(|\mathcal{P}_D \cup \mathcal{E}_D|)$ and $|\cdot|$ represents the set cardinality;

$$\mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = \begin{cases} 1 & \text{if } \mathcal{P}_D = \emptyset \\ \frac{|\mathcal{P}_D \cap \mathcal{P}_C|}{|\mathcal{P}_D|} & \text{otherwise,} \end{cases} \tag{4.27}$$

$$\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) = \begin{cases} 1 & \text{if } \mathcal{E}_D = \emptyset \\ 0 & \text{if } \mathcal{E}_D \neq \emptyset \text{ and } \mathcal{E}_C = \emptyset \\ \sum\limits_{\epsilon_i \in \mathcal{E}_D} \frac{\max\limits_{\epsilon_j \in \mathcal{E}_C} \{\mathsf{e\text{-}hd}(\epsilon_i, \epsilon_j)\}}{|\mathcal{E}_D|}) & \text{otherwise;} \end{cases} \tag{4.28}$$

with ϵ_i, ϵ_j existential restrictions; and

$$\mathsf{e\text{-}hd}(\exists r.X, \exists s.Y) = \gamma(\nu + (1-\nu) \cdot \mathsf{hd}(\mathcal{T}_X, \mathcal{T}_Y)) \tag{4.29}$$

where $\gamma = (|\mathcal{R}_r \cap \mathcal{R}_s|)/|\mathcal{R}_r|$ and $0 \leq \nu < 1$.

The value of ν determines how important the roles are to be considered for similarity between two existential restriction information. For instance, $\exists$canWalk.Trekking and $\exists$canWalk.Parading for dissimilar nested concepts Trekking and Parading should not be regarded as entirely dissimilar themselves. If ν is assigned the values 0.3, 0.4, and 0.5, then e-hd($\exists$canWalk.Trekking, $\exists$canWalk.Parading) is 0.3, 0.4, and 0.5, respectively. This value may vary among applications. In this work, ν is set to 0.4 (if it is not explicitly defined) for exemplifying the calculation of hd.

Example 4.5. (Continuation of Example 4.4)
For brevity, let ActivePlace, Mangrove, Beach, Place, Trekking, Kayaking, canWalk, and canSail be abbreviated as AP, M, B, P, T, K, cW, and cS, respectively. Using Definition 4.6, the homomorphism degree from $\mathcal{T}_{\mathsf{AP}}$ to $\mathcal{T}_{\mathsf{M}}$, or

$$\begin{aligned} \mathsf{hd}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{M}}) &= (\tfrac{2}{4})(\tfrac{1}{2}) + (\tfrac{2}{4})(\tfrac{\max\{\mathsf{e\text{-}hd}(\exists \mathsf{cW.T}, \exists \mathsf{cW.T})\}}{2} + \tfrac{\max\{\mathsf{e\text{-}hd}(\exists \mathsf{cS.K}, \exists \mathsf{cW.T})\}}{2}) \\ &= (\tfrac{2}{4})(\tfrac{1}{2}) + (\tfrac{2}{4})(\tfrac{1+0}{2}) = 0.5 \end{aligned}$$

Similarly, $\mathsf{hd}(\mathcal{T}_{\mathsf{M}}, \mathcal{T}_{\mathsf{AP}}) = 0.67$, $\mathsf{hd}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{B}}) = 0.5$, and $\mathsf{hd}(\mathcal{T}_{\mathsf{B}}, \mathcal{T}_{\mathsf{AP}}) = 0.67$. □

The example shows that the homomorphism degree from $\mathcal{T}_{\mathsf{AP}}$ to $\mathcal{T}_{\mathsf{M}}$ is 0.5 even though M is not subsumed by AP. Similar interpretations can be applied for the other results.

4.5.2 Properties underlying Homomorphism Degree

Theorem 4.4. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and $\mathcal{T}_C, \mathcal{T}_D$ be their corresponding description tree, respectively. Then, the following are equivalent:

1. $C \sqsubseteq D$; and
2. $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$.

Proof. ($\Longrightarrow$) Assume $C \sqsubseteq D$ *i.e.* there exists a homomorphism h which maps the root of $\mathcal{T}_D$ to the root of $\mathcal{T}_C \Longleftrightarrow l_D(v_D) \subseteq l_C(h(v_D))$ for each $v_D \in V_D$ and $\rho_D(v_D, w_D) \subseteq \rho_C(h(v_C), h(w_C))$ for each successor w_D of v_D. We show $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$ by cases.

- When $|V_D| = 1$ *i.e.* $\mathcal{T}_D$ contains only one node, then we show $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longleftrightarrow \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = 1$ (by Definition 4.6). This is obvious since $\mathcal{P}_D \subseteq \mathcal{P}_C$.
- When $|V_D| > 1$, then we need to show $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longleftrightarrow \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = 1$ and $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) = 1$ (by Definition 4.6). Since $l_D(rt_D) \subseteq l_C(h(rt_D))$ (by assumption), then $\mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = |\mathcal{P}_D|/|\mathcal{P}_D| = 1$. To show $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) = 1$, we need to show that $\rho_D(rt_D, w) \subseteq \rho_C(rt_C, h(w))$ for each successor w of rt_D (in order to have $\gamma = 1$) and there exists a homomorphism h' which maps w of its subtree $\mathcal{T}_{D_i}$ to $h'(h(w))$ of another subtree $\mathcal{T}_{C_j}$ (in order to have $\mathsf{hd}(\mathcal{T}_{D_i}, \mathcal{T}_{C_j}) = 1$). The former is obvious by assumption. Since $\mathcal{T}_{D_i}$ is part of $\mathcal{T}_D$ and $\mathcal{T}_{C_j}$ is also part of $\mathcal{T}_C$, then such h' also exists by assumption. Thus, we have $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$.

($\Longleftarrow$) Assume $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$ *i.e.* $\mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = 1$ and $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) = 1$ (by Definition 4.6). Then, we need to show $C \sqsubseteq D$ *i.e.* there exists a homomorphism which maps the root of $\mathcal{T}_D$ to the root of $\mathcal{T}_C$. By Definition 4.6, $\mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = 1$ implies that $\mathcal{P}_D \subseteq \mathcal{P}_C$. Also, $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) = 1 \Longleftrightarrow \gamma = 1$ and $\mathsf{hd}(\mathcal{T}_{D_i}, \mathcal{T}_{C_j}) = 1$ for each depth of the tree $\mathcal{T}_D \Longleftrightarrow l_D(v_D) \subseteq l_C(h(v_D))$ for each $v_D \in V_D$ and $\rho_D(v_D, w_D) \subseteq \rho_C(h(v_C), h(w_C))$ for each successor w_D of v_D. Therefore, we conclude that $C \sqsubseteq D$. ❏

Theorem 4.4 describes a property of concept subsumption, *i.e.* C is a sub-concept of D if the homomorphism degree of the corresponding description tree $\mathcal{T}_D$ to $\mathcal{T}_C$ is equal to 1, and vice versa. In other words, the more value of $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$ is closer to 1, the more likely the subsumption of C and D may hold.

In the following, we show that hd can be computed in polynomial time.

Theorem 4.5. Let V_1, V_2 be sets of vertices corresponding to $\mathcal{T}_1, \mathcal{T}_2$, respectively. The computational complexity of hd is $\mathcal{O}(|V_1| \cdot |V_2|)$.

Proof. Let $C \coloneqq P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n$, $D \coloneqq Q_1 \sqcap \cdots \sqcap Q_l \sqcap \exists s_1.D_1 \sqcap \cdots \sqcap \exists r_o.D_o$, and $\mathcal{T}_C, \mathcal{T}_D$ be the corresponding description trees. We need to show μ, γ, $\mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C)$, and $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C)$ are bounded by $\mathcal{O}(|V_1||V_2|)$.

Since the set union, the intersection, and the set cardinality $|\cdot|$ can be computed in polynomial time in the worst case, then μ, $\mathsf{p\text{-}hd}(\mathcal{P}_C, \mathcal{P}_D)$, and γ are bounded by $\mathcal{O}(|V_1||V_2|)$.

Computing $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C)$ requires to call $\mathsf{e\text{-}hd}$ for $|\mathcal{E}_D| \cdot |\mathcal{E}_C|$ times. Each call of $\mathsf{e\text{-}hd}$ will make a recursive call to hd and its number of calls is bounded by the height of $\mathcal{T}_D$ and $\mathcal{T}_C$. Hence, $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C)$ are bounded by $\mathcal{O}(|V_1||V_2|)$. ❏

4.6 From Subsumption Degree to Concept Similarity

We recall from Equation 4.22 that the degree of concept similarity can be determined from the two directional subsumption degree of each direction. Mathematically, such

aggregation can be defined as any binary operators accepting the unit interval, *e.g.* the average, the multiplication, and the root mean square. In the following, two concrete measures in $\mathcal{FL}_0$, *viz.* the skeptical measure $\sim^s$ [89, 90] and the credulous measure $\sim^c$ [89, 90], and one measure in $\mathcal{ELH}$, *viz.* the measure sim [145, 151], have defined to base the computation on the average. Their formal definitions are precisely defined in order and arguments supporting our selection *i.e.* the average are discussed.

Definition 4.7 (Skeptical $\mathcal{FL}_0$ Similarity Degree)**.** Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$. The *skeptical* $\mathcal{FL}_0$ *similarity degree* between C and D (denoted by $C \sim^s D$), is defined as follows:

$$C \sim^s D = \frac{(C \rightsquigarrow^s D) + (D \rightsquigarrow^s C)}{2} \tag{4.30}$$

Example 4.6. (Continuation of Example 4.2) The skeptical $\mathcal{FL}_0$ similarity degree between C and D can be calculated as follows:

$$C \sim^s D = \frac{C \rightsquigarrow^c D + \mathsf{D} \rightsquigarrow^s C}{2} = \frac{0.5 + 0}{2} = 0.25$$

□

Definition 4.8 (Credulous $\mathcal{FL}_0$ Similarity Degree)**.** Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$. The *credulous* $\mathcal{FL}_0$ *similarity degree* between C and D (denoted by $C \sim^s D$), is defined as follows:

$$C \sim^c D = \frac{(C \rightsquigarrow^c D) + (D \rightsquigarrow^c C)}{2} \tag{4.31}$$

Example 4.7. (Continuation of Example 4.3) The credulous $\mathcal{FL}_0$ similarity degree between C and D can be calculated as follows:

$$C \sim^c D = \frac{C \rightsquigarrow^c D + D \rightsquigarrow^c C}{2} = \frac{0.75 + 0.5}{2} = 0.625$$

□

It is worth mentioning that other choices of the operator may be used as discussed earlier. However, redefining the aggregation operator may produce a different behavior. We continue the discussion later on page 71 in this chapter.

The following propositions discuss about some inherited properties of the two measures for $\mathcal{FL}_0$ concepts. That is, they are symmetric measures and preserve ordering in the viewpoint of skepticism between relations.

Proposition 4.3 (Symmetry)**.** Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$. The following holds:

1. $C \sim^s D = D \sim^s C$; and
2. $C \sim^c D = D \sim^c C$.

Proof. These are obvious by the average. ❏

Proposition 4.4. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$. Then, the following ordering holds[1]

$$\equiv \quad \preceq \quad \sim^s \quad \preceq \quad \sim^c$$

Proof. By average, it suffices to show $\sqsubseteq \preceq \rightsquigarrow^s \preceq \rightsquigarrow^c$. This has already been proven by Proposition 4.2. ❑

Intuitively, the above property spells out that, for any $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, we have $(C \equiv D) \leq (C \sim^s D) \leq (C \sim^c D)$. In particular, if $C \equiv D$, then $(C \sim^s D) = (C \sim^c D) = 1$. It also tells us that both $\sim^s$ and $\sim^c$ can be used to identify the equivalent degree between concepts when the equivalent relation between them does not hold. In other words, they are the more elastic notions of the concept equivalence.

Theorem 4.6. Let L, M be sets of words over the alphabet of role names corresponding to concepts C, D, respectively. The computational complexity of both $\sim^s$ and $\sim^c$ is $\mathcal{O}(n|M||L|)$, where n is the size of concepts C, D.

Proof. This is immediately followed from Theorem 4.2 and the average. ❑

Now, we show that both $\sim^s$ and $\sim^c$ are procedures which ensures termination and can be used as an indicator for the degree of commonalities between $\mathcal{FL}_0$ concepts. Intuitively, Lemma 4.1 ensures that the positive results are correct and Lemma 4.2 ensures that the negative results are also correct. Termination guarantees that they always provide an answer in finite time.

Lemma 4.1. Let C, D be $\mathcal{FL}_0$ concepts. It follows that:

- if $C \sim^s D \in (0, 1]$, then both C and D share commonalities among each other;
- if $C \sim^c D \in (0, 1]$, then both C and D share commonalities among each other.

Proof. Let C, D be any $\mathcal{FL}_0$ concepts. By the average, it suffices to show:

- $C \rightsquigarrow^s \in (0, 1]$, then C is "partially subsumed" by D;
- $C \rightsquigarrow^c \in (0, 1]$, then C is "partially subsumed" by D.

To show the first point, we assume that $C \rightsquigarrow^s \in (0, 1]$. By assumption, Theorem 4.1, and Proposition 4.2, we know that C is partially subsumed by D (based on the characterization of language inclusion).

We can also show the second point in an analogous manner. ❑

[1] See Definition 4.4 for the meaning of $\preceq$

Lemma 4.2. Let C, D be any $\mathcal{FL}_0$ concepts. It follows that:

- if both C and D share commonalities among each other, then $C \sim^s D \in (0, 1]$;
- if both C and D share commonalities among each other, then $C \sim^c D \in (0, 1]$.

Proof. Let C, D be any $\mathcal{FL}_0$ concepts. We show their contraposition *i.e.*

- if $C \sim^s D = 0$, then both C and D do not share commonalities to each other;
- if $C \sim^c D = 0$, then both C and D do not share commonalities to each other.

To show the first point, we assume that $C \sim^s D = 0$. By assumption and the average, we know that $C \rightsquigarrow^s D = 0$ and $D \rightsquigarrow^s C = 0$. This means that both C and D do not share any commonalities to each other.

We can also show the second point in analogous manner. ❑

Theorem 4.7. Both $\sim^s$ and $\sim^c$ are guaranteed for termination and fulfill the conditions:

- $C \sim^s D \in (0, 1]$ iff both C and D share commonalities among each other;
- $C \sim^c D \in (0, 1]$ iff both C and D share commonalities among each other.

Proof. These are obvious by Lemma 4.1, Lemma 4.2, and Theorem 4.6. ❑

The measure sim for the DL $\mathcal{ELH}$ can be developed in the similar fashion as in $\mathcal{FL}_0$.

Definition 4.9 ($\mathcal{ELH}$ Similarity Degree)**.** Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and $\mathcal{T}_C, \mathcal{T}_D$ be the corresponding description trees. Then, the $\mathcal{ELH}$ *similarity degree* between C and D (denoted by $\mathsf{sim}(C, D)$) is defined as follows:

$$\mathsf{sim}(C, D) = \frac{\mathsf{hd}(\mathcal{T}_C, \mathcal{T}_D) + \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)}{2} \tag{4.32}$$

Example 4.8. (Continuation of Example 4.5) The $\mathcal{ELH}$ similarity degree between AP and M can be calculated as follows:

$$\mathsf{sim}(\mathsf{AP}, \mathsf{M}) = \frac{\mathsf{hd}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{M}}) + \mathsf{hd}(\mathcal{T}_{\mathsf{M}}, \mathcal{T}_{\mathsf{AP}})}{2} = \frac{0.5 + 0.67}{2} = 0.585$$

Similarly, $\mathsf{sim}(\mathsf{AP}, \mathsf{B}) = 0.585$. □

The following propositions discuss about some inherited properties of the above measure for $\mathcal{ELH}$ concepts *i.e.* it is symmetric and is less skeptical than the concept equivalence.

Proposition 4.5. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$. Then, the following properties hold:

1. $\mathsf{sim}(C, D) = \mathsf{sim}(D, C)$; and
2. $\equiv \preceq \mathsf{sim}$.

Proof. (1) This is obvious by the average.
(2) This is immediately followed from Theorem 4.4 and the average. ❑

Theorem 4.8. Let V_1, V_2 be sets of vertices corresponding to $\mathcal{T}_1, \mathcal{T}_2$, respectively. The computational complexity of sim is $\mathcal{O}(|V_1| \cdot |V_2|)$.

Proof. This is immediately followed from Theorem 4.5 and the average. ❑

We can show that sim is also a procedure which ensures termination and can be used as an indicator for the degree of commonalities between $\mathcal{ELH}$ concepts. Intuitively, we ensure that the correct results are correct (*cf.* Lemma 4.3) and the negative results are also correct (*cf.* Lemma 4.4). Termination guarantees to provide an answer in finite time.

Lemma 4.3. Let C, D be any $\mathcal{ELH}$ concepts and $\nu \in (0, 1)$. Then, $\mathsf{sim}(C, D) \in (0, 1]$ implies that C and D share commonalities among each other.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathcal{T}_C, \mathcal{T}_D$ be their corresponding trees, and $\nu \in (0, 1)$. With Theorem 4.4 and the average, it suffices to show that $\mathsf{hd}(\mathcal{T}_C, \mathcal{T}_D) \in (0, 1]$ implies $\mathsf{p\text{-}hd}(\mathcal{P}_C, \mathcal{P}_D) > 0$ or $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_D) > 0$. We show these cases as follows:

- If there exists $v \in V_C$ such that $l_C(v) \cap l_D(h(v)) \neq \emptyset$, then we show $\mathsf{hd}(\mathcal{T}_C, \mathcal{T}_D) \in (0, 1]$. Since $|l_C(v) \cap l_D(h(v))| > 0$, then we know $\mu > 0$ and $\mathsf{p\text{-}hd}(\mathcal{P}_v, \mathcal{P}_{h(v)}) > 0$. That is, $\mathsf{p\text{-}hd}(\mathcal{P}_C, \mathcal{P}_D) > 0$.
- If there exist $v, w \in V_C$ such that $\rho_C(v, w) \cap \rho_D(h(v), h(w)) \neq 0$, then we show $\mathsf{hd}(\mathcal{T}_C, \mathcal{T}_D) \in (0, 1]$. Since $|\rho_C(v, w) \cap \rho_D(h(v), h(w))| > 0$, then we know $\gamma > 0$. Since hd cannot be decreased, we know $\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_D) > 0$.

❑

Lemma 4.4. Let C, D be $\mathcal{ELH}$ concepts and $\nu \in$. Then, if C and D share commonalities among each other, then $\mathsf{sim}(C, D) \in (0, 1]$.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathcal{T}_C, \mathcal{T}_D$ be their corresponding trees, and $\nu \in (0, 1)$. We show their contraposition *i.e.* $\mathsf{sim}(C, D) = 0$ implies that C and D do not share commonalities to each other.

By the average, we know that $\mathsf{hd}(\mathcal{T}_C, \mathcal{T}_D) = 0$ and $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 0$. This means that both C and D do not share any commonalities to each other. ❑

Theorem 4.9. The measure sim is guaranteed for termination and fulfills the condition:

$$\mathsf{sim}(C, D) \in (0, 1] \text{ iff both } C \text{ and } D \text{ share commonalities among each other.}$$

Proof. This is obvious by Lemma 4.3, Lemma 4.4, and Theorem 4.8. ❑

Discussing about the choice of an aggregating operator, ones may argue to base the definitions on other methods. However, those may create unsatisfactory results for the extreme cases. To illustrate this, we define the functions $\sim_{\times}$ and $\sim_{\text{rms}}$ as follows.

$$C \sim_{\times} D = (C \rightsquigarrow^{s} D) \times (D \rightsquigarrow^{s} D) \tag{4.33}$$

$$C \sim_{\text{rms}} D = \sqrt{\frac{(C \rightsquigarrow^{s} D)^{s} + (D \rightsquigarrow^{s} C)^{2}}{2}} \tag{4.34}$$

Then, we have $A \sim_{\times} \top = 0 \times 1 = 0$ and $A \sim_{\text{rms}} \top = \sqrt{(0^2 + 1^2)/2} = 0.707$, whereas $A \sim^{s} \top = (0 + 1)/2 = 0.5$. Since $C \sim_{\times} D \leq C \sim^{s} D \leq C \sim_{\text{rms}} D$ for any concepts C and D, we agree with [93, 145] that the average-based definition as given above is the most appropriate method.[1]

Figure 4.5: Different types of agents

Proposition 4.4 and Example 4.6 - 4.7 exhibit that there is no the unique concept similarity measure for similarity-based applications. Which measure should be used depends on concrete applications, especially the type of an agent. For example, when employing the notion $\sim$ to a query answering system, a credulous agent may want to see answers as much as possible; hence, the measure $\sim^{c}$ is employed. On the other hand, a skeptical agent would like to see sufficient relevant answers; hence, the measure $\sim^{s}$ is employed. Figure 4.5 exemplifies a similar situation where two different types of agents are looking at two different kinds of castles. Indeed, we could simulate this situation by using $\sim^{s}$ for the skeptical agent and $\sim^{c}$ for the credulous agent. This fact of having multiple measures also corresponds to an experiment in [154]. That is, similarity measures might depend on target applications (*e.g.* target ontologies) and applicable similarity measures should be personalized to the agent's similarity judgment style. We shall further investigate this point on the notion of concept similarity under preferences in the next chapter.

4.7 Summary

- Most of current approaches for concept similarity do not consider the ontological constraints and definitions defined in an ontology. This may result in counter-intuitive results when a thorough consideration on the similarity of concepts is involved. This chapter has improved this problem by redefining the problem of concept similarity as a generalization of concept equivalence in DLs, *i.e.* two equivalent concepts are considered to be totally similar and are assigned to 1 whereas total dissimilar concepts are assigned to 0;

[1] Though we recommend to use the average, its choice of operators may be changed and it may produce a different behavior as discussed.

- Subsumption degree approaches for $\mathcal{FL}_0$ and $\mathcal{ELH}$ were developed based on their structural subsumption techniques; and also, their properties *e.g.* the relationship between subsumption degree and subsumption relation were explored; and
- Concept similarity measures for $\mathcal{FL}_0$ and $\mathcal{ELH}$ were developed based on the corresponding notions of subsumption degree and their properties were investigated. Finally, they were shown to be procedures which ensure termination and can be used as indicators for similarity between concepts in an ontology.

Chapter 5

Personalization of Concept Similarity Measure

The approaches for identifying subsumption degree and concept similarity degree can be used to extract information about the commonalities and the discrepancies of the subset relation and equivalence relation, respectively, between concepts. Unfortunately, when they are employed by an agent-related application (or other domains similar to this), counter-intuitive results may obtain. We illustrate this situation in the following example (by slightly modified from Example 4.4) to emphasize the need of considering the agent's preference in practice. We note that this is related to the second issue of the thesis.

Example 5.1. An agent A wants to visit a place for doing some physical activities (*i.e.* ActivePlace). At that moment, he would like to enjoy walking. Suppose that a place ontology has been modeled as follows:

$$\begin{aligned}
\mathsf{ActivePlace} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking} \sqcap \exists \mathsf{canSail.Kayaking} \\
\mathsf{Mangrove} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking} \\
\mathsf{Beach} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canSail.Kayaking} \\
\mathsf{canWalk} &\sqsubseteq \mathsf{canMoveWithLegs} \\
\mathsf{canSail} &\sqsubseteq \mathsf{canTravelWithSails}
\end{aligned}$$

Since the above ontology is expressed in $\mathcal{ELH}$, we may use the measure sim to query the similarity degree between ActivePlace and Beach *i.e.* $\mathsf{sim}(\mathsf{AP}, \mathsf{B})$, and also, between ActivePlace and Mangrove *i.e.* $\mathsf{sim}(\mathsf{AP}, \mathsf{M})$. As shown in Example 4.8, $\mathsf{sim}(\mathsf{AP}, \mathsf{B}) = \mathsf{sim}(\mathsf{AP}, \mathsf{M}) = 0.585$. These information shows that both Mangrove and Beach are equally similar to ActivePlace. We note that sim was developed based on the structural subsumption algorithm; thus, it merely considers the objective aspects. Taking into account also the agent's preferences, Mangrove may appear to be more suitable for his perception of ActivePlace at that moment. In other words, he will not be happy if an intelligent system happens to recommend him to go for a Beach. □

To address this issue, a new formalism for expressing preferential aspects of a context in consideration (*e.g.* the agent) has to be developed. Since DL concepts are constructed

inductively from the set of concept names and the set of role names, such formalism should offer ways to express the preferences w.r.t. concept names and role names. As guided in Chapter 4 (on page 59), the previously developed subsumption degree techniques should be generalized w.r.t. those preferential aspects and such generalization will be used to derive a new notion for concept similarity measure under the agent's preferences.

5.1 Preference Profile

We first introduced *preference profile* (denoted by π) in [91] as a collection of preferential elements in which the development of similarity measure of concepts for a particular cognitive agent should consider. Its first intuition is to model different forms of preferences (of an agent) based on concept names and role names. Similarity measure which adopts this notion is flexible to be tuned by an agent and can determine the similarity conformable to that agent's perception.

The syntax and semantics of each form are given in term of "partial" functions because agents may not have preferences over all concept names and role names. We recommend to devise similarity measures with considerations on preference profile if we aim at developing concept similarity measure for general purposes – a measure based on both subjective and objective factors. Mathematical definitions for each form of preferences are formally defined as follows.

Definition 5.1 (Primitive Concept Importance)**.** Let $\mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$ be a set of primitive concept names occurring in a TBox $\mathcal{T}$. Then, a *primitive concept importance* is a "partial" function $\mathfrak{i}^{\mathfrak{c}} : \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) \to [0, 2]$[1].

For any $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{c}}(A) = 1$ captures an expression of normal importance for A, $\mathfrak{i}^{\mathfrak{c}}(A) > 1$ (and $\mathfrak{i}^{\mathfrak{c}}(A) < 1$) indicates that A has higher (and lower, respectively) importance, and $\mathfrak{i}^{\mathfrak{c}}(A) = 0$ indicates that A is of no importance to the agent.

Example 5.2. (Continuation of Example 5.1) Suppose that an agent A is using a similarity measure for querying some names similar to $\mathsf{ActivePlace}$. He concerns that those names will be similar to $\mathsf{ActivePlace}$ if they are "places". Thus, the agent can express this preference as $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Place}) = 2$, *i.e.* values should be higher than 1.

On the other hand, suppose he "does not care" if those are places or not, he may express this preference as $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Place}) = 0$, *i.e.* values must be equal to 0. □

Definition 5.2 (Role Importance)**.** Let $\mathsf{RN}(\mathcal{T})$ be a set of role names occurring in $\mathcal{T}$. Then, a *role importance* is a "partial" function $\mathfrak{i}^{\mathfrak{r}} : \mathsf{RN}(\mathcal{T}) \to [0, 2]$.

For any $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{r}}(r) = 1$ captures an expression of normal importance for r, $\mathfrak{i}^{\mathfrak{r}}(r) > 1$ (and $\mathfrak{i}^{\mathfrak{r}}(r) < 1$) indicates that r has higher (and lower, respectively) importance, and $\mathfrak{i}^{\mathfrak{r}}(r) = 0$ indicates that r is of no importance to the agent.

[1] In the original definition of preference profile [91], elements in the domains of both $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{i}^{\mathfrak{r}}$ are mapped to $\mathbb{R}_{\geq 0}$, which is a minor error.

Example 5.3. (Continuation of Example 5.1) Suppose that the agent A wants to enjoy "walking". He may express this preference as $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 2$, *i.e.* values should be higher than 1. □

Definition 5.3 (Primitive Concepts Similarity)**.** Let $\mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$ be a set of primitive concept names occurring in $\mathcal{T}$. For $A, B \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, a *primitive concepts similarity* is a "partial" function $\mathfrak{s}^{\mathfrak{c}} : \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) \times \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) \to [0, 1]$ such that $\mathfrak{s}^{\mathfrak{c}}(A, B) = \mathfrak{s}^{\mathfrak{c}}(B, A)$ and $\mathfrak{s}^{\mathfrak{c}}(A, A) = 1$.

For $A, B \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{s}^{\mathfrak{c}}(A, B) = 1$ captures an expression of total similarity between A and B and $\mathfrak{s}^{\mathfrak{c}}(A, B) = 0$ captures an expression of their total dissimilarity.

Example 5.4. (Continuation of Example 5.1) Suppose that the agent A believes that "trekking" and "kayaking" invoke similar feeling. Thus, he can express $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Trekking}, \mathsf{Kayaking}) = 0.1$, *i.e.* values should be higher than 0. □

Another example is the similarity of concepts Pet_1 and Pet_2, in which both are defined as follows: $\mathsf{Pet}_1 \sqsubseteq \mathsf{Dog} \sqcap \exists \mathsf{hasOwned.Human}$; $\mathsf{Pet}_2 \sqsubseteq \mathsf{Cat} \sqcap \exists \mathsf{hasOwned.Human}$. Here, Dog and Cat are both primitive concept names. Intuitively, Dog and Cat are similar, then we may attach this knowledge in form of $\mathfrak{s}^{\mathfrak{c}}$ in order to yield more accuracy on the measure.

Definition 5.4 (Primitive Roles Similarity)**.** Let $\mathsf{RN}^{\mathsf{pri}}(\mathcal{T})$ be a set of primitive role names occurring in $\mathcal{T}$. For $r, s \in \mathsf{RN}^{\mathsf{pri}}(\mathcal{T})$, a *primitive roles similarity* is a "partial" function $\mathfrak{s}^{\mathfrak{r}} : \mathsf{RN}^{\mathsf{pri}}(\mathcal{T}) \times \mathsf{RN}^{\mathsf{pri}}(\mathcal{T}) \to [0, 1]$ such that $\mathfrak{s}^{\mathfrak{r}}(r, s) = \mathfrak{s}^{\mathfrak{r}}(s, r)$ and $\mathfrak{s}^{\mathfrak{r}}(r, r) = 1$.

For $r, s \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{s}^{\mathfrak{r}}(r, s) = 1$ captures an expression of total similarity between r and s and $\mathfrak{s}^{\mathfrak{r}}(r, s) = 0$ captures an expression of their total dissimilarity.

Example 5.5. (Continuation of Example 5.1) Suppose that the agent A believes that "moving with legs" and "traveling with sails" invoke similar feeling. He may express $\mathfrak{s}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}, \mathsf{canTravelWithSails}) = 0.1$, *i.e.* values should be higher than 0. □

Basically, our motivations of both functions $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$ are the same, *i.e.* we aim at attaching subjective feeling of proximity (about primitive concept names and primitive role names) into a measure. In DLs, different primitive concept names (and also primitive role names) are considered to be total dissimilarity even though they may be recognized as being similar in real-world domains.

Definition 5.5 (Role Discount Factor)**.** Let $\mathsf{RN}(\mathcal{T})$ be a set of role names occurring in $\mathcal{T}$. Then, a *role discount factor* is a "partial" function $\mathfrak{d} : \mathsf{RN}(\mathcal{T}) \to [0, 1]$.

Intuitively, role discount factor means a factor that discounts an important contribution of a role. This aspect plays a part when comparing two existential restrictions or two value restrictions, *i.e.* concepts of the form $\exists r.C$ or concepts of the form $\forall r.C$, respectively, are being compared. For example, comparing $\exists r_1.(\exists r_2.C_1)$ and $\exists r_3.(\exists r_4.C_2)$ involves checking the commonality of r_1, r_3 and the commonality of $\exists r_2.C_1, \exists r_4.C_2$. Depending on a context of consideration, the commonality appeared in r_1 may have more/less importance than the commonality appeared in its nested concept part *i.e.* $\exists r_2.C_1$.

More formally, for any $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{d}(r) = 1$ captures an expression of total importance on the role (beyond a corresponding nested concept) and $\mathfrak{d}(r) = 0$ captures an expression of total importance on a nested concept (beyond the correspondent role r).

Example 5.6. (Continuation of Example 5.1) Suppose that the agent A does not concern much if places permit to either walk or to sail. He would rather consider on actual activities which he can perform. Thus, he may express $\mathfrak{d}(\mathsf{canWalk}) = 0.3$ and $\mathfrak{d}(\mathsf{canSail}) = 0.3$, *i.e.* values should be close to 0. □

Definition 5.6 (Preference Profile)**.** A *preference profile*, in symbol π, is a quintuple $\langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ where $\mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}$, and $\mathfrak{d}$ are as defined above and the *default preference profile*, in symbol π_0, is the quintuple $\langle \mathfrak{i}^{\mathfrak{c}}_0, \mathfrak{i}^{\mathfrak{r}}_0, \mathfrak{s}^{\mathfrak{c}}_0, \mathfrak{s}^{\mathfrak{r}}_0, \mathfrak{d}_0 \rangle$ where

$$\begin{aligned}
\mathfrak{i}^{\mathfrak{c}}_0(A) &= 1 \text{ for all } A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}),\\
\mathfrak{i}^{\mathfrak{r}}_0(r) &= 1 \text{ for all } r \in \mathsf{RN}(\mathcal{T}),\\
\mathfrak{s}^{\mathfrak{c}}_0(A, B) &= 0 \text{ for all } (A, B) \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) \times \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}),\\
\mathfrak{s}^{\mathfrak{r}}_0(r, s) &= 0 \text{ for all } (r, s) \in \mathsf{RN}^{\mathsf{pri}}(\mathcal{T}) \times \mathsf{RN}^{\mathsf{pri}}(\mathcal{T}), \text{ and}\\
\mathfrak{d}_0(r) &= 0.4 \text{ for all } r \in \mathsf{RN}(\mathcal{T}),
\end{aligned}$$

Intuitively, the default preference profile π_0 represents the agent's preference in the default manner, *i.e.* when preferences are not given. That is, every $A \in \mathsf{CN}^{\mathsf{pri}}$ has normal importance and so does every $r \in \mathsf{RN}$. Also, every $(A, B) \in \mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}}$ is totally different and so does every $(r, s) \in \mathsf{RN}^{\mathsf{pri}} \times \mathsf{RN}^{\mathsf{pri}}$. Lastly, every $r \in \mathsf{RN}$ is considered 0.4 importance for the similarity of two existential restriction information (or two value restriction information). It is interesting to note that changes in the definition of the default preference profile yield different interpretations of the default preference and thereby may produce a different degree of similarity under the default manner. As for its exemplification, the value 0.4 is used by $\mathfrak{d}_0$ to conform with the value of ν used by sim in Chapter 4.

In this work, a preference profile of an agent is denoted by subscribing that agent below π, *e.g.* π_A represents a preference profile of the agent A.

5.2 From Subsumption Degree to Subsumption Degree under Preferences in $\mathcal{FL}_0$

Now, we are ready to exemplify how the notion of preference profile can be adopted toward the development of subsumption degree under preferences in $\mathcal{FL}_0$. To exemplify a development, we generalize the function $\rightsquigarrow^s$ to expose preferential elements of preference profile. As a result, the new function $\overset{\pi}{\rightsquigarrow} s$ is also driven by the structural subsumption characterization by means of language inclusion in $\mathcal{FL}_0$. It is worth noticing that role names appearing in $\mathcal{FL}_0$ are always primitive. This suggests that both $\mathsf{RN}^{\mathsf{pri}}(\mathcal{T})$ and $\mathsf{RN}(\mathcal{T})$ can be considered identically in Definition 5.6. Furthermore, due to the employed characterization, the notions $\mathfrak{i}^{\mathfrak{r}}$, $\mathfrak{s}^{\mathfrak{r}}$, and $\mathfrak{d}$ are not used by this generalization in $\mathcal{FL}_0$.

We start by presenting a relevant aspect of preference profile in terms of "total" functions in order to avoid computing on null values. A *total concept similarity* function is also presented as $\hat{\mathfrak{s}} : \mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}} \to [0, 1]$ as follows:

$$\hat{\mathfrak{s}}(x, y) = \begin{cases} 1 & \text{if } x = y \\ \mathfrak{s}^{\mathfrak{c}}(x, y) & \text{if } (x, y) \in \mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}} \\ & \text{and } \mathfrak{s}^{\mathfrak{c}} \text{ is defined on } (x, y) \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Intuitively, identical concepts are considered totally similar, *i.e.* they are set to 1. Otherwise, in case that they are not defined, different concepts are considered totally dissimilar by default.

The next step is to generalize the function $\rightsquigarrow^s$. We rewrite the numerator of $\rightsquigarrow^s$ to:

$$\sum_{P \in \mathsf{CN}^{\mathsf{pri}}} \max_{Q \in \mathsf{CN}^{\mathsf{pri}}} \{\hat{\mathfrak{s}}(P, Q) \mid \mathsf{W}(D, P) \subseteq \mathsf{W}(C, Q)\} \tag{5.2}$$

Basically, Equation 5.2 combines value of each maximal primitive concepts similarity element. Its objective is to also take into account the value of each similar concept pair, if this value is defined.

We may also put the notion of concept importance into our computational procedure. As suggested in Section 5.1, this results in the flexibility for weighting on primitive concepts (ranging from having no importance to having the maximum importance).

To achieve this, we continue with a similar attempt. That is, a *total concept importance* function is introduced as $\hat{\mathfrak{i}} : \mathsf{CN}^{\mathsf{pri}} \to [0, 2]$ as follows:

$$\hat{\mathfrak{i}}(x) = \begin{cases} \mathfrak{i}^{\mathfrak{c}}(x) & \text{if } x \in \mathsf{CN}^{\mathsf{pri}} \text{and } \mathfrak{i}^{\mathfrak{c}} \text{ is defined on } x \\ 1 & \text{otherwise} \end{cases} \tag{5.3}$$

Basically, the above equation says that each concept has normal importance by default, if it is not defined.

To take these matters into account, we should rewrite both the numerator and the denominator such that they expose some rooms for tuning with the concept importance. Thus, we rewrite each part, respectively, as follows:

$$\sum_{P \in \mathsf{CN}^{\mathsf{pri}}} \hat{\mathfrak{i}}(P) \cdot \max_{Q \in \mathsf{CN}^{\mathsf{pri}}} \{\hat{\mathfrak{s}}(P, Q) \mid \mathsf{W}(D, P) \subseteq \mathsf{W}(C, Q)\} \tag{5.4}$$

$$\sum_{P \in \mathsf{CN}^{\mathsf{pri}}} \hat{\mathfrak{i}}(P) \tag{5.5}$$

Finally, putting each rewritten part together yields a concrete function for concept subsumption degree under preference profile. We denote this new function by $\overset{\pi}{\rightsquigarrow} s$ (*cf.* Definition 5.7) as it presents a generalization of $\rightsquigarrow^s$ w.r.t. preference profile.

Definition 5.7 (Skeptical $\mathcal{FL}_0$ Subsumption Degree under π)**.** Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$ be in their normal forms and $\mathsf{W}(E, A)$ be a set of words w.r.t. the concept E and the primitive A. Then, a *skeptical $\mathcal{FL}_0$ subsumption degree under π* from C to D (denoted by $C \overset{\pi}{\rightsquigarrow} s\ D$) is defined as follows:

$$C \overset{\pi}{\rightsquigarrow} s\ D = \frac{\sum\limits_{P \in \mathsf{CN}^{\mathsf{pri}}} \hat{\mathfrak{i}}(P) \cdot \max\limits_{Q \in \mathsf{CN}^{\mathsf{pri}}} \{\hat{\mathfrak{s}}(P, Q) | \mathsf{W}(D, P) \subseteq \mathsf{W}(C, Q)\}}{\sum\limits_{P \in \mathsf{CN}^{\mathsf{pri}}} \hat{\mathfrak{i}}(P)} \tag{5.6}$$

Example 5.7. An agent A is searching for a hotel room during his vacation. At that moment, he prefers to stay in a Japanese-style room or something similar. In the following, his desired room may be expressed as the concept DesiredRoom. Suppose RoomA and RoomB are concepts in a room ontology as follows:

$$\begin{aligned} \mathsf{DesiredRoom} &\sqsubseteq \mathsf{Room} \sqcap \forall \mathsf{floor.Tatami} \\ \mathsf{RoomA} &\sqsubseteq \mathsf{Room} \sqcap \forall \mathsf{floor.Bamboo} \\ \mathsf{RoomB} &\sqsubseteq \mathsf{Room} \sqcap \forall \mathsf{floor.Marble} \end{aligned}$$

To express the agent's preferences that Bamboo is quite similar to Tatami, we may express the agent A's preferences as: $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Bamboo}, \mathsf{Tatami}) = 0.8$. Following Definition 5.7, it yields that

$$\mathsf{DR} \overset{\pi}{\rightsquigarrow} s\ \mathsf{RA} = \frac{1 + 0 + 1 + 1 + 1 + 0.8 + 1}{|X, Y, Z, R, T, B, M|} = \frac{5.8}{7}$$

and

$$\mathsf{RA} \overset{\pi}{\rightsquigarrow} s\ \mathsf{DR} = \frac{0 + 1 + 1 + 1 + 0.8 + 1 + 1}{7} = \frac{5.8}{7},$$

Similarly, it yields $\mathsf{DR} \overset{\pi}{\rightsquigarrow} s\ \mathsf{RB} = \mathsf{RB} \overset{\pi}{\rightsquigarrow} s\ \mathsf{DR} = \frac{5}{7}$. □

It is worth noticing that subsumption degree under preferences for the credulous subsumption degree can also be developed by incorporating with the notions role importance ($\mathfrak{i}^{\mathfrak{r}}$) and primitive roles similarity ($\mathfrak{s}^{\mathfrak{r}}$) (*cf.* Definition 5.6). However, it requires us to well investigate how those elements should be incorporated and we leave this as a future task.

Under a special setting of preference profile *i.e.* the default preference profile, the function $\overset{\pi}{\rightsquigarrow} s$ can be reduced backward to $\rightsquigarrow^s$. This means that $\overset{\pi}{\rightsquigarrow} s$ can be also used for a situation when preferences are not given. As for its syntactic sugar, let us denote a setting on $\overset{\pi}{\rightsquigarrow} s$ by replacing the setting with π. For instance, we may write the setting with π_0 as $\overset{\pi_0}{\rightsquigarrow} s$. Next, we show that, under this special setting on $\overset{\pi}{\rightsquigarrow} s$, the computation produces the same outcome as $\rightsquigarrow^s$.

Proposition 5.1. For any $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, $C \overset{\pi_0}{\rightsquigarrow} s\ D = C \rightsquigarrow^s D$.

Proof. Recall by Definition 5.6 that default preference profile π_0 is the quintuple $\langle \mathfrak{i}^{\mathfrak{c}}_0, \mathfrak{i}^{\mathfrak{r}}_0, \mathfrak{s}^{\mathfrak{c}}_0, \mathfrak{s}^{\mathfrak{r}}_0, \mathfrak{d}_0 \rangle$. We notice that only $\mathfrak{i}^{\mathfrak{c}}_0$ and $\mathfrak{s}^{\mathfrak{c}}_0$ are relevant to $\overset{\pi}{\rightsquigarrow}$. Fix any $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, we show that, under this special setting, $C \overset{\pi_0}{\rightsquigarrow} s\ D = C \rightsquigarrow^s D$ as follows:

$$C \overset{\pi_0}{\rightsquigarrow} s\ D = \frac{\sum\limits_{P \in \mathsf{CN}^{\mathsf{pri}}} 1 \cdot \max\limits_{Q \in \mathsf{CN}^{\mathsf{pri}}} \{\hat{\mathfrak{s}}(P,Q) | \mathsf{W}(D,P) \subseteq \mathsf{W}(C,Q)\}}{\sum\limits_{P \in \mathsf{CN}^{\mathsf{pri}}} 1}$$
$$= \frac{1 \cdot \sum\limits_{P \in \mathsf{CN}^{\mathsf{pri}}} \max\limits_{Q \in \mathsf{CN}^{\mathsf{pri}}} \{\hat{\mathfrak{s}}(P,Q) | \mathsf{W}(D,P) \subseteq \mathsf{W}(C,Q)\}}{|\mathsf{CN}^{\mathsf{pri}}|}$$

Since $\mathfrak{s}_0^{\mathfrak{c}}$ maps identity to 1 and else to 0, $\sum\limits_{P \in \mathsf{CN}^{\mathsf{pri}}} \max\limits_{Q \in \mathsf{CN}^{\mathsf{pri}}} \{\hat{\mathfrak{s}}(P,Q) | \mathsf{W}(D,P) \subseteq \mathsf{W}(C,Q)\} = |\{P \in \mathsf{CN}^{\mathsf{pri}} \mid \mathsf{W}(D,P) \subseteq \mathsf{W}(C,P)\}|$. We have shown that $C \overset{\pi_0}{\rightsquigarrow} s\ D = C \rightsquigarrow^s D$. ❑

5.3 From Subsumption Degree to Subsumption Degree under Preferences in $\mathcal{ELH}$

Generalizing subsumption degree to subsumption degree under preference profile in $\mathcal{ELH}$ is similar to what we have just done in $\mathcal{FL}_0$. The main difference, in which we shall see, is the utilization of every aspect in preference profile. Therefore, the previous total functions are augmented and the missing total function for the role discount factor is added.

As mentioned, we present each aspect of preference profile in term of "total" functions in order to avoid computing on null values. A *total importance* function is firstly introduced as $\hat{\mathfrak{i}} : \mathsf{CN}^{\mathsf{pri}} \cup \mathsf{RN} \to [0,2]$ based on the primitive concept importance and the role importance.

$$\hat{\mathfrak{i}}(x) = \begin{cases} \mathfrak{i}^{\mathfrak{c}}(x) & \text{if } x \in \mathsf{CN}^{\mathsf{pri}} \text{and } \mathfrak{i}^{\mathfrak{c}} \text{ is defined on } x \\ \mathfrak{i}^{\mathfrak{r}}(x) & \text{if } x \in \mathsf{RN} \text{ and } \mathfrak{i}^{\mathfrak{r}} \text{ is defined on } x \\ 1 & \text{otherwise} \end{cases} \tag{5.7}$$

A *total similarity* function is also presented as $\hat{\mathfrak{s}} : (\mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}}) \cup (\mathsf{RN}^{\mathsf{pri}} \times \mathsf{RN}^{\mathsf{pri}}) \to [0,1]$ using the primitive concepts similarity and the primitive roles similarity.

$$\hat{\mathfrak{s}}(x,y) = \begin{cases} 1 & \text{if } x = y \\ \mathfrak{s}^{\mathfrak{c}}(x,y) & \text{if } (x,y) \in \mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}} \\ & \text{and } \mathfrak{s}^{\mathfrak{c}} \text{ is defined on } (x,y) \\ \mathfrak{s}^{\mathfrak{r}}(x,y) & \text{if } (x,y) \in \mathsf{RN}^{\mathsf{pri}} \times \mathsf{RN}^{\mathsf{pri}} \\ & \text{and } \mathfrak{s}^{\mathfrak{r}} \text{ is defined on } (x,y) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

Similarly, a *total role discount factor* function[1] is presented in the following in term of a function $\hat{\mathfrak{d}} : \mathsf{RN} \to [0,1]$ based on the role discount factor.

$$\hat{\mathfrak{d}}(x) = \begin{cases} \mathfrak{d}(x) & \text{if } \mathfrak{d} \text{ is defined on } x \\ 0.4 & \text{otherwise} \end{cases} \tag{5.9}$$

[1]We set the default value to 0.4 to comply with the default value of π_0.

The next step is to generalize the notion of homomorphism degree hd (*cf.* Definition 4.6). Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and $r, s \in \mathsf{RN}$. Also, let $\mathcal{T}_C, \mathcal{T}_D, \mathcal{P}_C$, $\mathcal{P}_D$, $\mathcal{E}_C$, $\mathcal{E}_D$, $\mathcal{R}_r$, and $\mathcal{R}_s$ be as defined in Subsection 4.5.1. The homomorphism degree under preference profile π from $\mathcal{T}_D$ to $\mathcal{T}_C$ can be formally defined in Definition 5.8.

Definition 5.8. Let $\mathbf{T}^{\mathcal{ELH}}$ be a set of all $\mathcal{ELH}$ description trees, and $\pi = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be a preference profile. The *homomorphism degree under preference profile* π is a function $\mathsf{hd}^{\pi} : \mathbf{T}^{\mathcal{ELH}} \times \mathbf{T}^{\mathcal{ELH}} \rightarrow [0, 1]$ defined inductively as follows:

$$\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) = \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D) \cdot \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D)) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C), \quad (5.10)$$

$$\text{where } \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D) = \begin{cases} 1 & \text{if } \sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) + \sum\limits_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) = 0 \\ \dfrac{\sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A)}{\sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) + \sum\limits_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r)} & \text{otherwise;} \end{cases} \quad (5.11)$$

$$\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) = \begin{cases} 1 & \text{if } \sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) = 0 \\ 0 & \text{if } \sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) \neq 0 \text{ and } \sum\limits_{B \in \mathcal{P}_C} \hat{\mathfrak{i}}(B) = 0 \\ \dfrac{\sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) \cdot \max\limits_{B \in \mathcal{P}_C} \{\hat{\mathfrak{s}}(A,B)\}}{\sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A)} & \text{otherwise;} \end{cases} \quad (5.12)$$

$$\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C) = \begin{cases} 1 & \text{if } \sum\limits_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) = 0 \\ 0 & \text{if } \sum\limits_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) \neq 0 \text{ and } \sum\limits_{\exists s.Y \in \mathcal{E}_C} \hat{\mathfrak{i}}(s) = 0 \\ \dfrac{\sum\limits_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) \cdot \max\limits_{\epsilon_j \in \mathcal{E}_C} \{\mathsf{e\text{-}hd}^{\pi}(\exists r.X, \epsilon_j)\}}{\sum\limits_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r)} & \text{otherwise;} \end{cases} \quad (5.13)$$

where ϵ_j is an existential restriction; and

$$\mathsf{e\text{-}hd}^{\pi}(\exists r.X, \exists s.Y) = \gamma^{\pi}(r, s) \cdot (\hat{\mathfrak{d}}(r) + (1 - \hat{\mathfrak{d}}(r)) \cdot \mathsf{hd}^{\pi}(\mathcal{T}_X, \mathcal{T}_Y)) \quad (5.14)$$

$$\text{where } \gamma^{\pi}(r, s) = \begin{cases} 1 & \text{if } \sum\limits_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r') = 0 \\ \dfrac{\sum\limits_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r') \cdot \max\limits_{s' \in \mathcal{R}_s} \{\hat{\mathfrak{s}}(r', s')\}}{\sum\limits_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r')} & \text{otherwise.} \end{cases} \quad (5.15)$$

Intuitively, the function hd^{π} (Equation 5.10) is defined as the weighted sum of the degree under preferences of the vertex set commonalities ($\mathsf{p\text{-}hd}^{\pi}$) and the degree under preferences of edge condition matching ($\mathsf{e\text{-}set\text{-}hd}^{\pi}$). Equation 5.12 calculates the average of the best matching under preferences of primitive concepts in $\mathcal{P}_D$. Equation 5.14 calculates the degree under preferences of a potential homomorphism of a matching edge. If edge labels share some commonalities under preferences (Equation 5.15), *i.e.* $0 < \gamma^{\pi} \leq 1$, then part of the edge matching is satisfied; but the successors' labels and structures have yet to be checked. This is defined recursively as $\mathsf{hd}^{\pi}(\mathcal{T}_X, \mathcal{T}_Y)$ in Equation 5.14. Equation 5.13 calculates the best possible edge matching under preferences of each edge in $\mathcal{E}_D$ and returns the average thereof.

The weight μ^{π} in Equation 5.10 determines how important the primitive concept names are to be considered for preference-based similarity. For the special case where $D = \top$, *i.e.* $\mathcal{P}_D = \mathcal{E}_D = \emptyset$, μ^{π} is irrelevant as $\mathcal{T}_{\top}$ is the smallest $\mathcal{ELH}$ description tree and $\mathsf{hd}^{\pi}(\mathcal{T}_{\top}, \mathcal{T}_C) = 1$ for all concepts C.

It is to be mentioned that the function hd^{π} may look similar to $simi_d$ as both are recursive definitions for the same DL $\mathcal{ELH}$. However, they are obviously different caused by the distinction of their inspirations and their viewpoints of the development. While hd^{π} is inspired by the homomorphism-based structural subsumption characterization, $simi_d$ is inspired by the Jaccard Index [143]. Technically speaking, $simi_d$ employs t-conorm instead of fixing an operator. However, unlike $simi_d$, the use of μ^{π} for determining how primitive concepts are weighted and the use of γ^{π} for determining the proportion of shared super roles are employed. Furthermore, $simi_d$ is originated from the viewpoint of ordinary concept similarity measure, thus some aspects of preference profile are missed; though some may exist. We continue the discussion in Section 5.8.

The function hd^{π} yields a numerical value that represents structural similarity w.r.t. a particular profile π of a concept against another concept. We present an example about the calculation of hd^{π} in the following.

Example 5.8. (Continuation of Example 5.1) Let enrich the example. Assume the agent A's preference profile is defined as follows: (i) $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Place}) = 2$; (ii) $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 2$; (iii) $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Trekking}, \mathsf{Kayaking}) = 0.1$; (iv) $\mathfrak{s}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}, \mathsf{canTravelWithSails}) = 0.1$; (v) $\mathfrak{d}(\mathsf{canWalk}) = 0.3$ and $\mathfrak{d}(\mathsf{canSail}) = 0.3$. Let ActivePlace, Mangrove, Beach, Place, Trekking, Kayaking, canWalk, and canSail are rewritten shortly as AP, M, B, P, T, K, cW, and cS, respectively. Using Definition 5.8, $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{M}})$

$$
\begin{aligned}
&= (\tfrac{3}{6}) \cdot \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_{\mathsf{AP}}, \mathcal{P}_{\mathsf{M}}) + (\tfrac{3}{6}) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= (\tfrac{3}{6}) \cdot (\tfrac{\mathsf{i}(X) \cdot \max\{\mathfrak{s}(X,Y), \mathfrak{s}(X,\mathsf{P})\} + \mathsf{i}(\mathsf{P}) \cdot \max\{\mathfrak{s}(\mathsf{P},\mathsf{Y}), \mathfrak{s}(\mathsf{P},\mathsf{P})\}}{\mathsf{i}(X) + \mathsf{i}(\mathsf{P})}) \\
&+ (\tfrac{3}{6}) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= (\tfrac{3}{6})(\tfrac{1 \cdot \max\{0,0\} + 2 \cdot \max\{0,1\}}{1+2}) + (\tfrac{3}{6}) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= (\tfrac{3}{6})(\tfrac{2}{3}) + (\tfrac{3}{6}) \left[\tfrac{\mathsf{i}(\mathsf{cW}) \cdot \max\{\mathsf{e\text{-}hd}^{\pi}(\exists \mathsf{cW}.\mathsf{T}, \exists \mathsf{cW}.\mathsf{T})\} + 1 \cdot \max\{0.019\}}{\mathsf{i}(\mathsf{cW}) + \mathsf{i}(\mathsf{cS})} \right] \\
&= (\tfrac{3}{6})(\tfrac{2}{3}) + (\tfrac{3}{6}) \left[\tfrac{2 \cdot \max\{(1)(0.3 + 0.7(1))\} + 1 \cdot \max\{0.019\}}{\mathsf{i}(\mathsf{cW}) + \mathsf{i}(\mathsf{cS})} \right] \\
&= (\tfrac{3}{6})(\tfrac{2}{3}) + (\tfrac{3}{6}) \left[\tfrac{(2)(1) + (1)(0.019)}{2+1} \right] \approx 0.67
\end{aligned}
$$

Similarly, we obtain $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{M}}, \mathcal{T}_{\mathsf{AP}}) = 0.80$. Furthermore, using Definition 5.8, $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{B}}) \approx 0.51$ and $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{B}}, \mathcal{T}_{\mathsf{AP}}) = 0.75$. □

Similar to Proposition 5.1, the function hd^{π} can be used when preferences of the agent are not given. That is, we tune the function according to the default preference profile *i.e.* hd^{π_0}. We state this property in the following proposition.

Proposition 5.2. For $\mathcal{T}_D, \mathcal{T}_C \in \mathbf{T}^{\mathcal{ELH}}$, $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$.

Proof. Recall by Definition 5.6 that the default preference profile π_0 is the quintuple $\langle \mathfrak{i}_0^{\mathfrak{c}}, \mathfrak{i}_0^{\mathfrak{r}}, \mathfrak{s}_0^{\mathfrak{c}}, \mathfrak{s}_0^{\mathfrak{r}}, \mathfrak{d}_0 \rangle$. Also, suppose a concept name D is of the form: $P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_n.D_n$, where $P_i \in \mathsf{CN}^{\mathsf{pri}}$, $r_j \in \mathsf{CN}$, $D_j \in \mathsf{Con}(\mathcal{ELH})$, $1 \leq i \leq m$, $1 \leq j \leq n$, $P_1 \sqcap \cdots \sqcap P_m$ is denoted by $\mathcal{P}_D$, and $\exists r_1.D_1 \sqcap \cdots \sqcap \exists r_n.D_n$ is denoted by $\mathcal{E}_D$. Let d be the depth of $\mathcal{T}_D$. We prove that, for any $d \in \mathbb{N}$, $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$ by induction on d.

When $d = 0$, we know that $D = P_1 \sqcap \cdots \sqcap P_m$. To show that $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$, we need to show that $\mu^{\pi_0} = \mu$ and $\mathsf{p\text{-}hd}^{\pi_0}(\mathcal{P}_D, \mathcal{P}_C) = \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C)$. Let us derive as follows:

$$\mu^{\pi_0} = \frac{\sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A)}{\sum\limits_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) + \sum\limits_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r)} = \frac{\sum\limits_{i=1}^{m} 1}{\sum\limits_{i=1}^{m} 1 + 0} = \frac{m}{m+0} = \mu.$$

Furthermore, we only need to show $\sum\limits_{A \in \mathcal{P}_D} \max\{\hat{\mathfrak{s}}(A, B) : B \in \mathcal{P}_C\} = |\mathcal{P}_D \cap \mathcal{P}_C|$ in order to show $\mathsf{p\text{-}hd}^{\pi_0}(\mathcal{P}_D, \mathcal{P}_C) = \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C)$. We know that $\mathfrak{s}_0^{\mathfrak{c}}$ maps name identity to 1 and otherwise to 0. Thus, $\sum\limits_{A \in \mathcal{P}_D} \max\{\hat{\mathfrak{s}}(A, B) : B \in \mathcal{P}_C\} = |\{x : x \in \mathcal{P}_D \text{ and } x \in \mathcal{P}_C\}| = |\mathcal{P}_D \cap \mathcal{P}_C|$.

We must now prove that if $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$ holds for $d = h - 1$ where $h > 1$ and $D = P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_n.D_n$ then $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$ also holds for $d = h$. To do that, we have to show $\mathsf{e\text{-}set\text{-}hd}^{\pi_0}(\mathcal{E}_D, \mathcal{E}_C) = \mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C)$. This can be done by showing in the similar manner that $\gamma^{\pi_0} = \gamma$ and $\mathsf{hd}^{\pi_0}(\mathcal{T}_X, \mathcal{T}_Y) = \mathsf{hd}(\mathcal{T}_X, \mathcal{T}_Y)$ from $\mathsf{e\text{-}hd}^{\pi_0}(\exists r.X, \exists s.Y) = \mathsf{e\text{-}hd}(\exists r.X, \exists s.Y)$, where $\exists r.X \in \mathcal{E}_D$ and $\exists s.Y \in \mathcal{E}_C$. Consequently, it follows by induction that, for $\mathcal{T}_D, \mathcal{T}_C \in \mathbf{T}^{\mathcal{ELH}}$, $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$. ❑

5.4 Concept Similarity under Preference Profile

In this section, we present an "abstract" notion of *concept similarity measure under the agent's preferences* [92, 93] and its desirable properties. As we shall see, the previous developments on subsumption degree under preference profile can be utilized to develop concrete measures of this abstract notion. Two "concrete" measures *viz.* $\overset{\pi}{\sim}s$ and sim^{π} are introduced by utilizing the functions $\overset{\pi}{\rightsquigarrow}$ and hd^{π}, respectively. Our first intuition is to exemplify the applicability of preference profile onto an arbitrary existing measure of concept similarity. This shows that our proposed notion of preference profile can be considered as a collection of noteworthy aspects for the development of concept similarity measure under the agent's preferences.

Definition 5.9. Given a preference profile π, two concepts $C, D \in \mathsf{Con}(\mathcal{L})$, and a TBox $\mathcal{T}$, a *concept similarity measure under preference profile* w.r.t. a TBox $\mathcal{T}$ is a function $\overset{\pi}{\sim}_{\mathcal{T}} : \mathsf{Con}(\mathcal{L}) \times \mathsf{Con}(\mathcal{L}) \to [0, 1]$.

When a TBox $\mathcal{T}$ is clear from the context, we simply write $\overset{\pi}{\sim}$. Furthermore, to avoid confusion on the symbols, $\overset{\pi}{\sim}_{\mathcal{T}}$ is used when referring to arbitrary measures.

The notion $\overset{\pi}{\sim}$ may be informally read as "the computation of $\sim$ is influenced by π". That informal interpretation shapes our intuition to consider this kind as a more generalized concept similarity *i.e.* not only objective factors but also subjective factors are considered in the identification of the degree of similarity. With adopting of this viewpoint of the interpretation, we can agree that $\overset{\pi}{\sim} s$ and sim^{π} (*cf.* Subsection 5.4.1) are informally interpreted as "we compute $\sim^{s}$ and sim (*cf.* Definition 4.7 and Definition 4.9, respectively) under an existence of a given preference profile π".

Basically, the notion $\overset{\pi}{\sim}$ is a function mapping a pair of two concept descriptions w.r.t. a particular π to a unit interval. We have identified a property called *preference invariance w.r.t. equivalence* in our preliminary study [92]. To identify more important properties of $\overset{\pi}{\sim}$, we started by investigating important properties of concept similarity measure existing in the literature (*e.g.* [87, 88]). Our primary motivation is to identify the properties of concept similarity measure which are also reasonable for $\overset{\pi}{\sim}$. The following collects fundamental properties for the introduced concept similarity measure under preference profile. They can be used to answer the question "What could be good preference-based similarity measures?". In other words, any preference-based measures satisfying the fundamental properties are considered to be good ones.

Definition 5.10. Let $C, D, E \in \mathsf{Con}(\mathcal{L})$ and Π be a countably infinite set of preference profile. Then, we call a concept similarity measure under preference profile $\overset{\pi}{\sim}$ is:

1. *symmetric* iff $\forall \pi' \in \Pi : (C \overset{\pi'}{\sim} D = D \overset{\pi'}{\sim} C)$;
2. *equivalence invariant* iff $C \equiv D \Longrightarrow \forall \pi' \in \Pi : (C \overset{\pi'}{\sim} E = D \overset{\pi'}{\sim} E)$;
3. *structurally dependent* iff for any finite sets of concepts C_1 and C_2 with the following conditions:
   - $\mathsf{C}_1 \subseteq \mathsf{C}_2$,
   - concepts $A, B \notin \mathsf{C}_2$,
   - $\mathfrak{i}^{\mathfrak{c}}(\Phi) > 0$ if Φ is primitive and $\Phi \in \mathsf{C}_2$, and
   - $\mathfrak{i}^{\mathfrak{r}}(\varphi) > 0$ if Φ is existential, *i.e.* $\Phi \coloneqq \exists \varphi.\Psi$, and $\Phi \in \mathsf{C}_2$,

   the concepts $C \coloneqq \bigsqcap(\mathsf{C}_1 \cup \{A\})$, $D \coloneqq \bigsqcap(\mathsf{C}_1 \cup \{B\})$, $E \coloneqq \bigsqcap(\mathsf{C}_2 \cup \{A\})$ and $F \coloneqq \bigsqcap(\mathsf{C}_2 \cup \{B\})$ fulfill the condition $\forall \pi' \in \Pi : (C \overset{\pi'}{\sim} D \leq E \overset{\pi'}{\sim} F)$; and
4. *preference invariant w.r.t. equivalence* iff $C \equiv D \Longleftrightarrow \forall \pi' \in \Pi : C \overset{\pi'}{\sim} D = 1$.

Next, we discuss the underlying intuitions of each property subsequently. We note that the properties 1 to 3 are adopted from [87, 88]. However, to the best of our knowledge, the property 4 is first introduced for *concept similarity measure under preference profile* in this work (originally introduced in [92]).

Let Π be a countably infinite set of preference profile. In the following, we discuss the intuitive interpretation of each property. Firstly, *symmetry* states that an order of concepts in question does not influence the notion $\overset{\pi'}{\sim}$ for any $\pi' \in \Pi$. For instance, $\mathsf{Mangrove} \overset{\pi'}{\sim} \mathsf{Beach} = \mathsf{Beach} \overset{\pi'}{\sim} \mathsf{Mangrove}$ w.r.t. any particular context π'. This property is controversial since cognitive science believes that similarity is asymmetric. An example given in [82] is as follows: People usually speak "the son resembles the father" rather than "the father resembles the son". Some work in DLs also prefer asymmetry such as [146,153]. It is worth observing that such a statement is made w.r.t. some particular contexts. Thus, this work favors on symmetry as it appears more natural to use and gives more intuitive computational understanding. For example, rather than viewing like "the son resembles the father", we would view like "if certain contexts are fixed, then the son and the father are similar to each other" (*cf.* [89, 92, 93]). Furthermore, we agree on the symmetry because axiomatic information in TBox is not dynamically changed; and also, the notion of preference profile studied in this work is static, *i.e.* it can be changed merely by tuning. Some work in DLs which favors on symmetry includes [87–89, 92, 147, 148, 150–152].

Secondly, *equivalence invariance* (alternatively called *equivalence soundness* [87] in the context of dissimilarity measure) states that if two concepts C and D are logically equivalent, then measuring the similarity of each toward the third concept E w.r.t. any $\pi' \in \Pi$ must be the same. This property is inspired from a characteristics of synonym concepts, *i.e.* concepts that means exactly the same. For instance, let $C \equiv \exists \mathsf{canWalk.Trekking}$ and $D \equiv \exists \mathsf{canWalk.Trekking}$. It is clear that C and D are logically equivalent. Therefore, let $E \in \mathsf{Con}(\mathcal{L})$, $C \overset{\pi'}{\sim} E = D \overset{\pi'}{\sim} E$ for any $\pi' \in \Pi$.

Thirdly, the notion of *structural dependence* was originally introduced by Tversky in [82]. Later, the authors of [88] has collected it as another important properties for concept similarity measure in their work. Basically, in Tversky's model, an object was considered as a set of features. Then, the similarity of two objects was measured by the relationship between a number of common features and a number of different features. Extending this idea to $\overset{\pi}{\sim}$ gives the meaning that the similarity of two concepts C, D increases if a more number of "equivalent" concepts is shared and each is considered "important".

Lastly, *preference invariance w.r.t. equivalence* states that if two concepts are logically equivalent, then the similarity degree of two concepts under preference profile π is always 1 for every $\pi \in \Pi$, and vice versa. Taking the negation both sides, this means $C \not\equiv D \iff \exists \pi' \in \Pi : C \overset{\pi'}{\sim} D \neq 1$. For instance, let $C \equiv \exists \mathsf{canWalk.Trekking}$ and $D \equiv \exists \mathsf{canWalk.Parading}$. It is clear that C and D are not logically equivalent, then taking $\pi = \pi_0$ obtains $C \overset{\pi_0}{\sim} D \neq 1$; though, taking $\pi = \pi_1$ where $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Trekking}, \mathsf{Parading}) = 1$ is defined in π_1 yields $C \overset{\pi_1}{\sim} D = 1$.

There are several properties which are not considered as fundamental properties of concept similarity measure under preference profile because the behaviors may not obey their properties when used under "non-default" preference profiles, *e.g. reverse subsumption*

preserving. According to [88], a concrete measure $\sim$ satisfies the *reverse subsumption preserving* iff, for any concepts C, D, and E, $C \sqsubseteq D \sqsubseteq E \Longrightarrow C \sim E \leq D \sim E$. The property states that the similarity of D and E is higher than the one of C and E because E is closer to D than C. To refute it, we need only one preference profile π such that the implication does not hold (*cf.* Example 5.9), *i.e.* to show that $(C \sqsubseteq D \sqsubseteq E)$ and $\exists \pi' \in \Pi : (C \overset{\pi'}{\sim} E > D \overset{\pi'}{\sim} E)$.

Example 5.9. Suppose concepts $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3$, and A_4 are primitive. Query describes features of an item that an agent is searching for. Item_1 and Item_2 are items, which compose of features $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3$ and $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3, \mathsf{A}_4$, respectively.

$$\begin{aligned} \mathsf{Query} &\equiv \mathsf{A}_1 \sqcap \mathsf{A}_2 \\ \mathsf{Item}_1 &\equiv \mathsf{A}_1 \sqcap \mathsf{A}_2 \sqcap \mathsf{A}_3 \\ \mathsf{Item}_2 &\equiv \mathsf{A}_1 \sqcap \mathsf{A}_2 \sqcap \mathsf{A}_3 \sqcap \mathsf{A}_4 \end{aligned}$$

The ontology shows the hierarchy: $\mathsf{Item}_2 \sqsubseteq \mathsf{Item}_1 \sqsubseteq \mathsf{Query}$. By taking $\mathfrak{s}^{\mathfrak{c}}(\mathsf{A}_2, \mathsf{A}_4) = 1$, it is reasonable to conclude that $\mathsf{Item}_2 \overset{\pi}{\sim} \mathsf{Query} > \mathsf{Item}_1 \overset{\pi}{\sim} \mathsf{Query}$ due to an increased number of totally similar concepts. □

Similarity measures can inherently have different skepticism. We have shown this evidence by introducing two measures in $\mathcal{FL}_0$, *viz.* the skeptical measure $\sim^s$ and the credulous measure $\sim^c$. Understanding the relationship between these two measures provide a way to personify similarity-based applications w.r.t. the agent's style. A similar experiment was done in [154] where different measures were used in target ontologies and obtained the better results than just using a single measure. The following definition offers another way of personalizing similarity-based applications. That is, different personalization of $\overset{\pi}{\sim}$ may contribute different skepticism of an agent.

Definition 5.11. Let Π be a countably infinite set of preference profile and $\pi_1, \pi_2 \in \Pi$. For any fixed measure $\overset{\pi}{\sim}$, the concept similarity measure under π_1 is *more skeptical than* π_2 (denoted by $\overset{\pi_1}{\sim} \preceq \overset{\pi_2}{\sim}$) if $C \overset{\pi_1}{\sim} D \leq C \overset{\pi_2}{\sim} D$ for all $C, D \in \mathsf{Con}(\mathcal{L})$.

Intuitively, if an arbitrary concept similarity measure under preference profile $\overset{\pi}{\sim}$ is fixed, measuring the similarity of two concepts under different preference profiles may yield different values.

5.4.1 From Subsumption Degree under Preferences to Concept Similarity under Preferences

The idea of developing "concrete" concept similarity measures under preference profile can be analogously brought from concept similarity. Indeed, we have pointed out this in Section 4.3, *i.e.* the second and the third step of our outlined methodology (*cf.* page 59). We formally recast this in the following.

$$C \overset{\pi}{\sim}_{\mathcal{T}} D = 1 \Longleftrightarrow C \overset{\pi}{\rightsquigarrow}_{\mathcal{T}} D = 1 \text{ and } D \overset{\pi}{\rightsquigarrow}_{\mathcal{T}} C = 1 \tag{5.16}$$

where the notion of directional subsumption degree under preference profile is denoted by $\overset{\pi}{\rightsquigarrow}_{\mathcal{T}}$ and the binary operator "and" should be generalized to aggregate two unit intervals. In the following, we show how ones can employ this idea to develop concrete concept similarity measure under preference profile for the DLs $\mathcal{FL}_0$ and $\mathcal{ELH}$ in this subsection.

The function $\overset{\pi}{\rightsquigarrow} s$ yields a numerical value that represents structural similarity w.r.t. a particular profile π of a $\mathcal{FL}_0$ concept against another $\mathcal{FL}_0$ concept. This knowledge can be used to develop a concrete measure of $\mathcal{FL}_0$ concept similarity under preference profile.

Definition 5.12. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$ be in their normal forms and $\pi = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be a preference profile. Then, the *skeptical* $\mathcal{FL}_0$ *similarity measure under preference profile* π between C and D (denoted by $C \overset{\pi}{\sim}_s D$) is defined as follows:

$$C \overset{\pi}{\sim}_s D = \frac{C \overset{\pi}{\rightsquigarrow} s\ D + D \overset{\pi}{\rightsquigarrow} s\ C}{2} \tag{5.17}$$

Example 5.10. (Continuation of Example 5.7) Using Definition 5.12, it yields that

$$\mathsf{DR} \overset{\pi}{\sim}_s \mathsf{RA} = \frac{\frac{5.8}{7} + \frac{5.8}{7}}{2} = \frac{5.8}{7}$$

Similarly, it yields that $\mathsf{DR} \overset{\pi}{\sim}_s \mathsf{RA} = \frac{5}{7}$. Since $\mathsf{DR} \overset{\pi}{\sim}_s \mathsf{RA} > \mathsf{DR} \overset{\pi}{\sim}_s \mathsf{RB}$, it corresponds to the agent A's perception that he may decide to stay in RoomA when his $\mathsf{DesiredRoom}$ is not available. □

The function hd^{π} yields a numerical value that represents structural similarity w.r.t. a particular profile π of a concept against another concept. We can use this knowledge to develop a concrete measure of $\mathcal{ELH}$ concepts as follows.

Definition 5.13. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathcal{T}_C$ and $\mathcal{T}_D$ be the corresponding description trees, and $\pi = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be a preference profile. Then, the $\mathcal{ELH}$ *similarity measure under preference profile* π between C and D (denoted by $\mathsf{sim}^{\pi}(C, D)$) is defined as follows:

$$\mathsf{sim}^{\pi}(C, D) = \frac{\mathsf{hd}^{\pi}(\mathcal{T}_C, \mathcal{T}_D) + \mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C)}{2} \tag{5.18}$$

Example 5.11. (Continuation of Example 5.8) Using Definition 5.13, it yields that

$$\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) = \frac{0.67 + 0.80}{2} \approx 0.74$$

Similarly, $\mathsf{sim}^{\pi}(\mathsf{B}, \mathsf{AP}) \approx 0.63$. The fact that $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) > \mathsf{sim}^{\pi}(\mathsf{B}, \mathsf{AP})$ corresponds with the agent A's needs and preferences. □

The above definitions use the average to aggregate two corresponding unit intervals. We may also argue to aggregate both values based on alternative operators accepting unit intervals *e.g.* the multiplication or the root mean square of both values. Unfortunately, those give unsatisfactory values for the extreme cases. Similar arguments about this point has been discussed on page 71. Hence, we believe that the average-based definition given

above is the most appropriate method for aggregating two values of subsumption degree under preference profile. Based on this form, $\overset{\pi}{\sim}s$ and sim^{π} are basically considered as a generalization of $\sim^s$ and sim, respectively, which determines similarity under preference profile, *i.e.* behavioral expectation of the measure will conform to the agent's perception. We note that, though we recommend to use the average, its choice of operators may be changed and it may produce a different behavior.

The following discusses some inherited properties of the measures $\overset{\pi}{\sim}s$ and sim^{π}.

First, both measure $\overset{\pi}{\sim}s$ and sim^{π} can be used in the case that a preference profile is not defined by the agent. In such a case, we tune the profile setting to π_0. That is, computing $\overset{\pi_0}{\sim}s$ and sim^{π_0} yields the degree of concept similarity measure merely w.r.t. the structure of concept descriptions in question.

Theorem 5.1. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, $C \overset{\pi_0}{\sim}s\ D = C \sim^s D$.

Proof. It immediately follows from Lemma 5.1, Definition 4.7, and Definition 5.12. ❑

Theorem 5.1 tells us that $\overset{\pi}{\sim}s$ is backward compatible in the sense that using $\sim^s$ with $\pi = \pi_0$, *i.e.* $\overset{\pi_0}{\sim}s$, coincides with $\sim^s$. Technically speaking, $\overset{\pi_0}{\sim}s$ can be used to handle the case of similar concepts regardless of the agent's preferences.

Theorem 5.2. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathsf{sim}^{\pi_0}(C, D) = \mathsf{sim}(C, D)$.

Proof. It immediately follows from Lemma 5.2, Definition 4.9, and Definition 5.13. ❑

Like in the case of $\overset{\pi_0}{\sim}s$, the above theorem shows that sim^{π} is also backward compatible in the sense that using sim^{π} with $\pi = \pi_0$, *i.e.* sim^{π_0}, coincides with sim.

Next, we show that $\overset{\pi}{\sim}s$ is a symmetric measure and can be computed in polynomial time.

Theorem 5.3. $\overset{\pi}{\sim}s$ is *symmetric*.

Proof. Let Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$ and $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, we have $C \overset{\pi}{\sim}s\ D = D \overset{\pi}{\sim}s\ C$ by Definition 5.12 and the average. ❑

Theorem 5.4. Assume that a value from any preference functions is retrieved in $\mathcal{O}(1)$. Let L, M be sets of words over the alphabet of role names corresponding to C, D, respectively. Then, $C \overset{\pi}{\sim}s\ D \in \mathcal{O}(|\mathsf{CN}^{\mathsf{pri}}|^2|L||M|)$.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$, π be any preference profile; and, let L, M be sets of words over the alphabet of role names corresponding to C, D, respectively. By Definition 5.12, we need to show that $C \overset{\pi}{\rightsquigarrow}s\ D \in \mathcal{O}(|\mathsf{CN}^{\mathsf{pri}}|^2|L||M|)$ and $D \overset{\pi}{\rightsquigarrow}s\ C \in \mathcal{O}(|\mathsf{CN}^{\mathsf{pri}}|^2|L||M|)$. By the average, it suffices to show $C \overset{\pi}{\rightsquigarrow}s\ D \in \mathcal{O}(|\mathsf{CN}^{\mathsf{pri}}|^2|L||M|)$ as follows.

Checking the inclusion of finite similar languages can be done in polynomial time *i.e.* to decide $\sum_{P \in \mathsf{CN}^{\mathsf{pri}}} \hat{\mathfrak{i}}(P) \cdot \max_{Q \in \mathsf{CN}^{\mathsf{pri}}} \{\hat{\mathfrak{s}}(P, Q) \mid \mathsf{W}(D, P) \subseteq \mathsf{W}(C, Q)\}$, in the worst case we have to check for all possible pairs $P \in \mathsf{CN}^{\mathsf{pri}}$ and $Q \in \mathsf{CN}^{\mathsf{pri}}$. Such test can be done in time $|\mathsf{CN}^{\mathsf{pri}}||\mathsf{CN}^{\mathsf{pri}}|$. To decide $\mathsf{W}(D, P) \subseteq \mathsf{W}(C, Q)$ in the inner loop, another polynomial

time operation is also required *i.e.* we have to check whether, for all words $w \in \mathsf{W}(D, P)$, $w \in \mathsf{W}(C, Q)$ for checking the set inclusion. This requires $|L||M|$ numbers of the operation. The summation (*cf.* the denominator of Definition 5.7) requires linear time *i.e.* in the size of $\mathsf{CN}^{\mathsf{pri}}$. ❏

The above theorem spells out that the measure $\overset{\pi}{\sim}s$ can be computed in polynomial time. In the following, we show that the measure sim^{π} can also be computed in polynomial time *i.e.* there exists an algorithmic procedure whose execution time is upper bounded by a polynomial expression in the size of the description trees

Theorem 5.5. Assume that a value from any preference functions is retrieved in $\mathcal{O}(1)$. Given $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathsf{sim}^{\pi}(C, D) \in \mathcal{O}(|V_C| \cdot |V_D|)$ where V_C and V_D are set of vertices of the description trees $\mathcal{T}_C$ and $\mathcal{T}_D$, respectively.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $C \coloneqq P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n$, $D \coloneqq Q_1 \sqcap \cdots \sqcap Q_l \sqcap \exists s_1.D_1 \sqcap \cdots \sqcap \exists r_o.D_o$, π be any preference profile, and $\mathcal{T}_C, \mathcal{T}_D$ be corresponding description trees. By Definition 5.13, we show $\mathsf{hd}^{\pi}(\mathcal{T}_C, \mathcal{T}_D) \in \mathcal{O}(|V_C| \cdot |V_D|)$ and $\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) \in \mathcal{O}(|V_D| \cdot |V_C|)$. Without loss of generality, it suffices to show merely $\mathsf{hd}^{\pi}(\mathcal{T}_C, \mathcal{T}_D) \in \mathcal{O}(|V_C| \cdot |V_D|)$. That is, we need to show μ^{π}, γ^{π}, $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C)$, and $\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C)$ are bounded by $\mathcal{O}(|V_C| \cdot |V_D|)$.

Since the summation, the maximal matching between $\mathcal{P}_D$ and $\mathcal{P}_C$, and the maximal matching between $\mathcal{R}_r$ and $\mathcal{R}_s$ can be computed in polynomial time in the worst case, the functions μ^{π}, γ^{π}, and $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C)$ are bounded by $\mathcal{O}(|V_C| \cdot |V_D|)$.

Computing $\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C)$ requires to call $\mathsf{e\text{-}hd}^{\pi}$ for $|\mathcal{E}_D| \cdot |\mathcal{E}_C|$ times. Each call of $\mathsf{e\text{-}hd}^{\pi}$ will make a recursive call to hd^{π} and its number of calls is bounded by the height of $\mathcal{T}_D$ and $\mathcal{T}_C$. Hence, $\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C)$ are bounded by $\mathcal{O}(|V_C| \cdot |V_D|)$. ❏

In the following, we show $\overset{\pi}{\sim}s$ is a procedure which ensures termination and can be used as an indicator for the degree of commonalities under preference profile π between $\mathcal{FL}_0$ concepts. Intuitively, Lemma 5.1 ensures that the positive results are correct and Lemma 5.2 ensures that the negative results are also correct. Termination ensures to provide an answer in finite time.

Lemma 5.1. Let C, D be $\mathcal{FL}_0$ concepts and $\pi' = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be any preference profile, where $\mathfrak{i}^{\mathfrak{c}}(A) \in (0, 2]$ for all $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$. Then, $C \overset{\pi'}{\sim}s\ D \in (0, 1]$ implies both C and D share commonalities under π' among each other.

Proof. Let C, D be $\mathcal{FL}_0$ concepts and π' be any preference profile. By the average, it suffices to show that if $C \overset{\pi'}{\rightsquigarrow}s\ D \in (0, 1]$, then C is "partially subsumed" under π' by D.

Assume $C \overset{\pi'}{\rightsquigarrow}s\ D \in (0, 1]$. By assumption, we know that (*cf.* Definition 5.7), for some $P \in \mathsf{CN}^{\mathsf{pri}}$, for some $Q \in \mathsf{CN}^{\mathsf{pri}}$, it holds that $\hat{\mathfrak{i}}(P) > 0$, $\hat{\mathfrak{s}}(P, Q) > 0$, and $\mathsf{W}(D, P) \subseteq \mathsf{W}(C, Q)$. This shows that C is partially subsumed under π' by D based on the characterization of language inclusion. ❏

Lemma 5.2. Let C, D be $\mathcal{FL}_0$ concepts and $\pi' = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be any preference profile, where $\mathfrak{i}^{\mathfrak{c}}(A) \in (0, 2]$ for all $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$. Then, both C and D share commonalities under π' implies that $C \overset{\pi'}{\sim} s\ D \in (0, 1]$.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{FL}_0)$ and π' be any preference profile. We show its contraposition *i.e.* if $C \overset{\pi'}{\sim} s\ D = 0$, then both C and D do not share commonalities under π' to each other.

Assume $C \overset{\pi'}{\sim} s\ D = 0$. By assumption and the average, we know that $C \overset{\pi'}{\rightsquigarrow} s\ D = 0$ and $D \overset{\pi'}{\rightsquigarrow} s\ C = 0$. This means that (*cf.* Definition 5.7), for any $P \in \mathsf{CN}^{\mathsf{pri}}$, for any $Q \in \mathsf{CN}^{\mathsf{pri}}$, it does not hold that $\hat{\mathfrak{i}}(P) > 0$, $\hat{\mathfrak{s}}(P, Q) > 0$, and $\mathsf{W}(D, P) \subseteq \mathsf{W}(C, Q)$. Also, for any $P \in \mathsf{CN}^{\mathsf{pri}}$, for any $Q \in \mathsf{CN}^{\mathsf{pri}}$, it does not hold that $\hat{\mathfrak{i}}(P) > 0$, $\hat{\mathfrak{s}}(P, Q) > 0$, and $\mathsf{W}(C, P) \subseteq \mathsf{W}(D, Q)$. This means that both C and D do not share any commonalities under π' with each other. Hence, both C and D do not share commonalities under π' to each other. ❑

Theorem 5.6. The measure $\overset{\pi}{\sim} s$ is guaranteed for termination and fulfills the condition:

$C \overset{\pi'}{\sim} s\ D \in (0, 1]$ iff both C and D share commonalities under π' among each other.

Proof. This is obvious by Lemma 5.1, Lemma 5.2, and Theorem 5.4. ❑

We can also show sim^{π} is a procedure which ensures termination and can be used as an indicator for the degree of commonalities under preference profile π between $\mathcal{ELH}$ concepts. That is, we ensure that the correct results are corrects (*cf.* Lemma 5.3) and the negative results are also correct (*cf.* Lemma 5.4). Termination ensures to provide an answer in finite time.

Lemma 5.3. Let C, D be $\mathcal{ELH}$ concepts and $\pi' = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be any preference profile, where $\mathfrak{i}^{\mathfrak{c}}(A) \in (0, 2]$ for all $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{r}}(r) \in (0, 2]$ for all $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{d}(r) \in (0, 1]$ for all $r \in \mathsf{RN}(\mathcal{T})$. Then, $\mathsf{sim}^{\pi'}(C, D) \in (0, 1]$ implies that both C and D share commonalities under π' among each other.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathcal{T}_C, \mathcal{T}_D$ be their corresponding trees, and π' be any preference profile where $\mathfrak{i}^{\mathfrak{c}}(A) \in (0, 2]$ for all $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{r}}(r) \in (0, 2]$ for all $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{d}(r) \in (0, 1]$ for all $r \in \mathsf{RN}(\mathcal{T})$. With Lemma 4.3, Theorem 5.2, and the average, it suffices to show that $\mathsf{hd}^{\pi'}(\mathcal{T}_C, \mathcal{T}_D) \in (0, 1]$ implies the partial subsumption under π' from D to C based on the characterization of homomorphism structural subsumption $\Longleftrightarrow$ $\mathsf{hd}^{\pi'}(\mathcal{T}_C, \mathcal{T}_D) \in (0, 1]$ implies $\mathsf{p\text{-}hd}^{\pi'}(\mathcal{P}_C, \mathcal{P}_D) > 0$ or $\mathsf{e\text{-}set\text{-}hd}^{\pi'}(\mathcal{E}_C, \mathcal{E}_D) > 0$. We show these cases as follows:

- For any $v \in V_C$, for any $h(v) \in V_D$, if there exists $A \in l_C(v)$ and $B \in l_D(h(v))$ such that $\hat{\mathfrak{s}}(A, B) > 0$, then we show that $\mathsf{p\text{-}hd}^{\pi'}(\mathcal{P}_C, \mathcal{P}_D) > 0$. To show this, we fix any $v' \in V_C$, any $h(v') \in V_D$; and assume $A \in l_C(v')$, $B \in l_D(h(v'))$, and $\hat{\mathfrak{s}}(A, B) > 0$. By Definition 5.13, we know $\mathsf{p\text{-}hd}^{\pi'}(\mathcal{P}_C, \mathcal{P}_D) > 0$.

- For any $v, w \in V_C$, for any $h(v), h(w) \in V_D$, if there exists $r \in \rho_C(v, w)$ and $s \in \rho_D(h(v), h(w))$ such that $\hat{\mathfrak{s}}(r, s) > 0$, then we show that $\mathsf{e\text{-}set\text{-}hd}^{\pi'}(\mathcal{E}_C, \mathcal{E}_D) > 0$. To show this, we fix any $v', w' \in V_C$, any $h(v'), h(w') \in V_D$; and assume $r \in \rho_C(v', w')$, $s \in \rho_D(h(v'), h(w'))$, and $\hat{\mathfrak{s}}(r, s) > 0$. By assumptions, we know $\gamma^{\pi'}(r, s) > 0$. Since $\mathsf{hd}^{\pi'}$ cannot be decreased according to Definition 5.13, we conclude that $\mathsf{e\text{-}set\text{-}hd}^{\pi'}(\mathcal{E}_C, \mathcal{E}_D) > 0$.

❑

Lemma 5.4. Let C, D be any $\mathcal{ELH}$ concepts and $\pi' = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be any preference profile where $\mathfrak{i}^{\mathfrak{c}}(A) \in (0, 2]$ for all $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{r}}(r) \in (0, 2]$ for all $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{d}(r) \in (0, 1]$ for all $r \in \mathsf{RN}(\mathcal{T})$. Then, if both C and D share commonalities under π' among each other, then $\mathsf{sim}^{\pi'}(C, D) \in (0, 1]$.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathcal{T}_C, \mathcal{T}_D$ be their corresponding trees, and π' be any preference profile where $\mathfrak{i}^{\mathfrak{c}}(A) \in (0, 2]$ for all $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{r}}(r) \in (0, 2]$ for all $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{d}(r) \in (0, 1]$ for all $r \in \mathsf{RN}(\mathcal{T})$. We show its contraposition *i.e.* $\mathsf{sim}^{\pi'}(C, D) = 0$ implies that C and D do not share commonalities under π' to each other.

By the average, we know that $\mathsf{hd}^{\pi'}(\mathcal{T}_C, \mathcal{T}_D) = 0$ and $\mathsf{hd}^{\pi'}(\mathcal{T}_D, \mathcal{T}_C) = 0$. This means that both C and D do not share any commonalities under π' to each other. ❑

Theorem 5.7. The measure sim^{π} is guaranteed for termination and fulfills the condition:

$\mathsf{sim}^{\pi'}(C, D) \in (0, 1]$ iff both C and D share commonalities under π' among each other.

Proof. This is obvious by Lemma 5.3, Lemma 5.4, Theorem 5.5. ❑

5.4.2 Desirable Properties of sim^{π}

Previously, we theorize a set of desirable properties that a concept similarity measure under preference profile should satisfy and systematically introduce the measures $\overset{\pi}{\sim}s$ and sim^{π}. Due to the time constraint, we only provide mathematical proofs for the desirable properties of sim^{π} in this thesis. Understanding the properties gives many benefits to the users of sim^{π} since they can predict its expected behaviors.

Theorem 5.8. sim^{π} is *symmetric*.

Proof. Let Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$ and $C, D \in \mathsf{Con}(\mathcal{ELH})$, we have $\mathsf{sim}^{\pi}(C, D) = \mathsf{sim}^{\pi}(D, C)$ by Definition 5.13. ❑

Theorem 5.9. sim^{π} is *equivalence invariant*.

Proof. Let Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$ and $C, D, E \in \mathsf{Con}(\mathcal{ELH})$, we show $C \equiv D \Longrightarrow \mathsf{sim}^{\pi}(C, E) = \mathsf{sim}^{\pi}(D, E)$.

Suppose $C \equiv D$, *i.e.* $C \sqsubseteq D$ and $D \sqsubseteq C$, then we know there exists a homomorphism $h_1 : \mathcal{T}_D \to \mathcal{T}_C$ which maps the root of $\mathcal{T}_D$ to the root of $\mathcal{T}_C$ and $h_2 : \mathcal{T}_C \to \mathcal{T}_D$ which maps the root of $\mathcal{T}_C$ to the root of $\mathcal{T}_D$, respectively, by Theorem 4.3. This means $\mathcal{T}_C = \mathcal{T}_D$. Thus, $\mathsf{sim}^{\pi}(C, E) = \mathsf{sim}^{\pi}(D, E)$. ❑

Theorem 5.10. sim^{π} is *structurally dependent*.

Proof. Let Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$ and any finite sets of concepts C_1 and C_2 with the following conditions: **(1)** $\mathsf{C}_1 \subseteq \mathsf{C}_2$; **(2)** concepts $A, B \notin \mathsf{C}_2$; **(3)** $\mathfrak{i}^{\mathfrak{c}}(\Phi) > 0$ if primitive $\Phi \in \mathsf{C}_2$; **(4)** $\mathfrak{i}^{\mathfrak{r}}(\varphi) > 0$ if existential $\exists\varphi.\Psi \in \mathsf{C}_2$. Suppose $C \coloneqq \bigsqcap(\mathsf{C}_1 \cup \{A\})$, $D \coloneqq \bigsqcap(\mathsf{C}_1 \cup \{B\})$, $E \coloneqq \bigsqcap(\mathsf{C}_2 \cup \{A\})$ and $F \coloneqq \bigsqcap(\mathsf{C}_2 \cup \{B\})$ where $\mathsf{C}_1 = \{P_1, \ldots, P_m, \exists r_1.P'_1, \ldots, \exists r_n.P'_n\}$ and $\mathsf{C}_2 = \{P_1, \ldots, P_i, \exists r_1.P'_1, \ldots, \exists r_j.P'_j\}$, w.l.o.g. we show $\mathsf{sim}^{\pi}(C, D) \leq \mathsf{sim}^{\pi}(E, F)$ by following two cases.

Suppose $m \leq i$, $n = j$ and A, B be primitives, we have $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_C, \mathcal{P}_D) = \frac{\sum_{P \in \mathcal{P}_C} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_C} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(A)}$, $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) = \frac{\sum_{P \in \mathcal{P}_D} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_D} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(B)}$, $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_E, \mathcal{P}_F) = \frac{\sum_{P \in \mathcal{P}_E} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_E} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(A)}$, and $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_F, \mathcal{P}_E) = \frac{\sum_{P \in \mathcal{P}_F} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_F} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(B)}$.
Since $m \leq i$, we know $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_C, \mathcal{P}_D) \leq \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_E, \mathcal{P}_F)$ and $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) \leq \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_F, \mathcal{P}_E)$. This infers $\mathsf{sim}^{\pi}(C, D) \leq \mathsf{sim}^{\pi}(E, F)$.

Suppose $m = i$, $n \leq j$ and A, B be existentials, then with the similar manner, we can show $\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_C, \mathcal{E}_D) \leq \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_E, \mathcal{E}_F)$, $\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C) \leq \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_F, \mathcal{E}_E)$. This also infers $\mathsf{sim}^{\pi}(C, D) \leq \mathsf{sim}^{\pi}(E, F)$.

Therefore, we have shown $\mathsf{sim}^{\pi}(C, D) \leq \mathsf{sim}^{\pi}(E, F)$. ❏

Lemma 5.5. Let $\mathcal{T}_D, \mathcal{T}_C \in \mathbf{T}^{\mathcal{ELH}}$ and Π be a countably infinite set of preference profile. Then, $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longleftrightarrow \forall \pi \in \Pi : \mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) = 1$.

Proof. Let Π be a countably infinite set of preference profile and π_0 be the default preference profile. Fix any $\pi \in \Pi$, we show $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longleftrightarrow \mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) = 1$.

($\Longrightarrow$) $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$ implies that there exists a homomorphism $h : \mathcal{T}_D \to \mathcal{T}_C$ which maps the root of $\mathcal{T}_D$ to the root of $\mathcal{T}_C$. Consequently, any setting on π does not influence the calculation on $\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C)$.

($\Longleftarrow$) In particular, it suffices to show $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longrightarrow \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$. By Lemma 5.2, it is the case that $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$. ❏

Theorem 5.11. sim^{π} is *preference invariant w.r.t. equivalence*.

Proof. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$, we show $C \equiv D \Longleftrightarrow \mathsf{sim}^{\pi}(C, D) = 1$.

($\Longrightarrow$) Assume $C \equiv D$, we need to show $\mathsf{sim}^{\pi}(C, D) = 1$. By Theorem 4.4, we know $C \equiv D \Longleftrightarrow \mathsf{sim}(C, D) = 1$. With the usage of Lemma 5.5, Definition 4.9, and Definition 5.13, we can derive $\mathsf{sim}^{\pi}(C, D) = 1$.

($\Longleftarrow$) This can be shown similarly as in the forward direction. ❏

Theorem 5.8 to 5.11 spells out that sim^{π} satisfies all fundamental properties of concept similarity measure under preference profile.

Definition 5.11 suggests that different preference profile settings represent different types of a rational agent. An easy characterization is observed from the aspect of role

discount factor ($\mathfrak{d}$). Intuitively, when the settings $\mathfrak{i}^{\mathfrak{c}}$, $\mathfrak{i}^{\mathfrak{r}}$, $\mathfrak{s}^{\mathfrak{c}}$, and $\mathfrak{s}^{\mathfrak{r}}$ defined by two rational agents A, B are the same, the agent which defines the lower $\mathfrak{d}$ on every $r \in \mathsf{RN}$ is always more skeptical. For instance, if $\mathfrak{d}_A(\mathsf{canWalk}) = 0.3$ and $\mathfrak{d}_B(\mathsf{canWalk}) = 0.4$, then $\mathsf{sim}^{\pi_A}(\exists \mathsf{canWalk}.\mathsf{Trekking}, \exists \mathsf{canWalk}.\mathsf{Parading}) = 0.3$ and $\mathsf{sim}^{\pi_B}(\exists \mathsf{canWalk}.\mathsf{Trekking}, \exists \mathsf{canWalk}.\mathsf{Parading}) = 0.4$. This is clear that the agent A is more skeptical than the agent B.

Proposition 5.3. Let Π be a countably infinite set of preference profile and $\pi_1, \pi_2 \in \Pi$ such that $\pi_1 = \langle \mathfrak{i}_1^{\mathfrak{c}}, \mathfrak{i}_1^{\mathfrak{r}}, \mathfrak{s}_1^{\mathfrak{c}}, \mathfrak{s}_1^{\mathfrak{r}}, \mathfrak{d}_1 \rangle$, $\pi_2 = \langle \mathfrak{i}_2^{\mathfrak{c}}, \mathfrak{i}_2^{\mathfrak{r}}, \mathfrak{s}_2^{\mathfrak{c}}, \mathfrak{s}_2^{\mathfrak{r}}, \mathfrak{d}_2 \rangle$, and RN be a set of role names. The following holds:[1]

$$\forall r \in \mathsf{RN} : (\mathfrak{d}_1(r) \leq \mathfrak{d}_2(r)) \Longrightarrow \equiv \preceq \ \mathsf{sim}^{\pi_1} \ \preceq \ \mathsf{sim}^{\pi_2}$$

for fixed functions $\mathfrak{i}_1^{\mathfrak{c}} = \mathfrak{i}_2^{\mathfrak{c}}$, $\mathfrak{i}_1^{\mathfrak{r}} = \mathfrak{i}_2^{\mathfrak{r}}$, $\mathfrak{s}_1^{\mathfrak{c}} = \mathfrak{s}_2^{\mathfrak{c}}$, and $\mathfrak{s}_1^{\mathfrak{r}} = \mathfrak{s}_2^{\mathfrak{r}}$.

5.5 Finding Suitable Values for Preference Profile

To identify suitable values for preference profile, we have proposed strategies to obtain such values. With the proposed strategies, we can start with the values obtained from applying each of them and re-adjust each value where refinement is required.

Intuitively, they can be classified into two underlying basic ideas. First, sub-concepts and sub-roles can inherit their importance from their super-concepts and super-roles, by default, respectively. Second, when an ABox is presented, we can use the canonical interpretation (*cf.* Definition 3.7 on page 37) to calculate the initial values for similarity of primitive concept names and similarity of primitive role names.

5.5.1 Tuning $\mathfrak{i}^{\mathfrak{c}}$

This subsection exhibits a strategy for tuning primitive concept importance $\mathfrak{i}^{\mathfrak{c}}$ in practice. Realistic ontologies are generally complex – consisting in plenty of concept names. Hence, having some strategies of tuning is useful since it helps to pave the way for a more convenient use of preference profile.

As a starting point, we seek to observe characteristics of realistic ontologies whose TBox is *unfoldable*[2], *e.g.* a popular medical ontology SNOMED CT, denoted by $\mathcal{O}_{\mathsf{med}}$, [105]. Figure 5.1 gives an example of concept definitions in $\mathcal{O}_{\mathsf{med}}$ and Figure 5.2 shows the concept hierarchy w.r.t. $\mathcal{O}_{\mathsf{med}}$.

According to the above figures, it is intuitive to express primitive concept importance through the concept hierarchy. For instance, an agent may say "my concept importance goes through $\mathsf{Disease}$"; or "my concept importance goes through $\mathsf{HeartDisease}$". Informally investigating, let $\mathsf{CN}^{\mathsf{pri}}(\mathcal{O}_{\mathsf{med}})$ be a set of primitive concept names occurring in $\mathcal{O}_{\mathsf{med}}$. Since $\mathsf{Disease} \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{O}_{\mathsf{med}})$, the former case is simple, *e.g.* an agent may mean $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2$. The latter case is a bit complicated since $\mathsf{HeartDisease} \notin \mathsf{CN}^{\mathsf{pri}}(\mathcal{O}_{\mathsf{med}})$. However, the agent's

[1] See Definition 5.11 for the meaning of $\preceq$

[2] According to this investigation, we assume an ontology $\mathcal{O}$ has an unfoldable TBox $\mathcal{T}$ in this section.

$$
\begin{array}{rcl}
\mathsf{Pericardium} & \sqsubseteq & \mathsf{Tissue} \sqcap \exists \mathsf{partOf}.\mathsf{Heart} \\
\mathsf{Endocardium} & \sqsubseteq & \mathsf{Tissue} \sqcap \exists \mathsf{partOf}.\mathsf{Heart} \\
\mathsf{Appendicitis} & \equiv & \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Appendix} \\
\mathsf{Pericarditis} & \equiv & \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Pericardium} \\
\mathsf{Endocarditis} & \equiv & \mathsf{Inflammation} \sqcap \exists \mathsf{hasLocation}.\mathsf{Endocardium} \\
\mathsf{Inflammation} & \sqsubseteq & \mathsf{Disease} \\
\mathsf{HeartDisease} & \equiv & \mathsf{Disease} \sqcap \exists \mathsf{hasLocation}.\exists \mathsf{partOf}.\mathsf{Heart}
\end{array}
$$

Figure 5.1: Example of concept definitions in $\mathcal{O}_{\mathsf{med}}$.

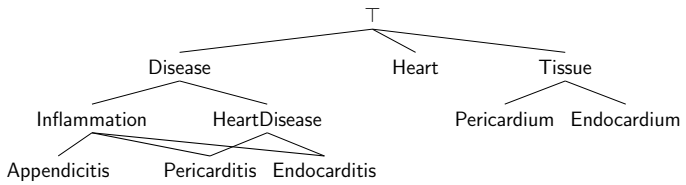


Figure 5.2: The concept hierarchy of $\mathcal{O}_{\mathsf{med}}$.

intention may mean $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2$ and $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Heart}) = 1.2$. This informal investigation shapes the development as follows:

Definition 5.14. Let $\mathsf{CN}(\mathcal{T})$ ($\mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$ and $\mathsf{CN}^{\mathsf{def}}(\mathcal{T})$) be a set of concept names (primitive concept names and defined concept names, respectively) occurring in $\mathcal{T}$. Then, a *propagation for primitive concept importance* is a "partial" function $\mathfrak{I}^{\mathfrak{c}} : \mathsf{CN}(\mathcal{T}) \cup \{\top\} \to [0, 2]$ such that a mapping n of $\mathfrak{I}^{\mathfrak{c}}$ on X (*i.e.* $\mathfrak{I}^{\mathfrak{c}}(X) = n$) is defined inductively as follows:[1].

1. $X \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) \Longrightarrow \mathfrak{i}^{\mathfrak{c}}(X) = n$;
2. $X \coloneqq \top \Longrightarrow \forall x \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) : \mathfrak{i}^{\mathfrak{c}}(x) = n$; and
3. $X \in \mathsf{CN}^{\mathsf{def}}(\mathcal{T}) \Longrightarrow \forall x \in \mathsf{RHS}(X) : \mathfrak{I}^{\mathfrak{c}}(x) = n$.

where $\mathsf{RHS}(X)$ is a set of concept names appearing on the right-hand side of X.

Its interpretation is defined in a usual way. That is, for any $A \in \mathsf{CN}(\mathcal{T})$, $\mathfrak{I}^{\mathfrak{c}}(A) = 1$ captures an expression of normal importance on A, $\mathfrak{I}^{\mathfrak{c}}(A) > 1$ (and $\mathfrak{I}^{\mathfrak{c}}(A) < 1$) indicates that A has higher (and lower, respectively) importance, $\mathfrak{I}^{\mathfrak{c}}(A) = 0$ indicates that A has no importance to the agent. For the special case, $\mathfrak{I}^{\mathfrak{c}}(\top) = n$ indicates that every primitive concept name occurring on $\mathcal{T}$ is of equal importance at n.

Example 5.12. From Figure 5.1, suppose that an agent A is using a concept similarity measure under preference profile for querying some names that expose the similar characteristics to $\mathsf{HeartDisease}$. Thus, the agent can express a preference $\mathfrak{I}^{\mathfrak{c}}(\mathsf{HeartDisease}) = 1.2$ instead of individually specifying $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2$ and $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Heart}) = 1.2$. □

[1]Later, we discuss some restrictions the readers should take into account when the notion $\mathfrak{I}^{\mathfrak{c}}$ is employed.

There are a few concerns that we should take into account, *i.e.* **(1)** inconsistent preferences of concepts occurring on the same branch of the concept hierarchy; **(2)** inconsistent preferences of defined concepts occurring on different branches of the concept hierarchy; and **(3)** expressing preferences when a TBox $\mathcal{T}$ contains equivalent defined concepts.

Let us take a look on our first concern through the following preference expression: "my concept importance goes through Disease, especially HeartDisease". In this example, we may take $\mathfrak{I}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2$ and $\mathfrak{I}^{\mathfrak{c}}(\mathsf{HeartDisease}) = 1.3$. Here, Disease is redefined on $\mathfrak{i}^{\mathfrak{c}}$ twice. This kind of scenarios is possible to happen because we are extending the aspect of primitive concept importance toward both types of concept names. There are many ways to handle this with the use of operators $\oplus : [0,2]^2 \to [0,2]$.

Definition 5.15. Let $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$ be a set of primitive concept names occurring in $\mathcal{T}$ and $x_0, x_1 \in [0,2]$. Also, let $\mathfrak{i}^{\mathfrak{c}}(A) = x_0$ be the previous mapping on A. We compute a new mapping $\mathfrak{i}^{\mathfrak{c}}(A) = x_1$ as follows:

$$\mathfrak{i}^{\mathfrak{c}}(A) = \begin{cases} x_1 & \text{if } \mathfrak{i}^{\mathfrak{c}} \text{ is not defined on } A \\ x_0 \oplus x_1 & \text{otherwise} \end{cases} \tag{5.19}$$

The notion of the operator remains abstract here as its concrete operators may vary on the context of use. In the following, we establish some of the abstract notion $\oplus$, *i.e.* $\oplus_{max}$, $\oplus_{first}$, and $\oplus_{last}$. Let two real numbers $x_1, x_0 \in [0,2]$. Then,

$$x_0 \oplus_{max} x_1 = \max\{x_0, x_1\} \tag{5.20}$$

$$x_0 \oplus_{first} x_1 = x_0 \tag{5.21}$$

$$x_0 \oplus_{last} x_1 = x_1 \tag{5.22}$$

Example 5.13. From Figure 5.1, an agent might say "My interest is Disease except HeartDisease". That is, we may take $\mathfrak{I}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2$ (*i.e.* $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2$) and $\mathfrak{I}^{\mathfrak{c}}(\mathsf{HeartDisease}) = 0$ (*i.e.* $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2 \oplus 0$ and $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Heart}) = 0$). Taking $\oplus$ as $\oplus_{max}$ yields $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 1.2$ and $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Heart}) = 0$. It also yields the same results by taking $\oplus$ as $\oplus_{first}$.

□

Example 5.14. From Figure 5.1, an agent might say "My concern is nothing except HeartDisease". That is, we may take $\mathfrak{I}^{\mathfrak{c}}(\top) = 0$ (*i.e.* $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 0$, $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Tissue}) = 0$, and $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Heart}) = 0$) and $\mathfrak{I}^{\mathfrak{c}}(\mathsf{HeartDisease}) = 1$ (*i.e.* $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 0 \oplus 1$ and $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Heart}) = 0 \oplus 1$). Taking $\oplus$ as $\oplus_{max}$ yields $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Disease}) = 1$, $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Tissue}) = 0$, and $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Heart}) = 1$. It also yields the same results by taking $\oplus$ as $\oplus_{last}$.

□

Now, we discuss our second concern on the application of $\mathfrak{I}^{\mathfrak{c}}$. Let us consider the following preference expression of an agent: $\mathfrak{I}^{\mathfrak{c}}(\mathsf{Pericarditis}) = 1.2$ and $\mathfrak{I}^{\mathfrak{c}}(\mathsf{Endocarditis}) =$

1.6. One may notice that both use the primitive concept name Disease in common. Now, a natural question to ask is "how a concept importance value should be propagated" since a propagation may cause an inconsistency of preference values for a primitive concept name, such as Disease in this example. This requires further work to study. However, one simple way for handling this problem is to prevent a mapping leading to this situation. In other words, an agent has to tune primitive concept names via the primitive concept importance $\mathfrak{i}^{\mathfrak{c}}$ individually.

Lastly, we contemplate our third concern. That is, "what happens if a defined concept name C_1 is defined on $\mathfrak{I}^{\mathfrak{c}}$ and there exists another defined concept name C_2 such that $C_1 \equiv_{\mathcal{T}} C_2$?" A natural way for handling this problem is to treat C_2 in the same way as C_1 (because they are equivalent). In particular, $\forall C_1, C_2 \in \mathsf{CN}^{\mathsf{def}}(\mathcal{T}) : C_1 \equiv_{\mathcal{T}} C_2 \Longrightarrow \mathfrak{I}^{\mathfrak{c}}(C_1) = \mathfrak{I}^{\mathfrak{c}}(C_2)$. Nevertheless, this also requires further work to explore other possibilities for coping with this problem and investigate desired properties the notion $\overset{\pi}{\sim}_{\mathcal{T}}$ should hold when it is used with $\mathfrak{I}^{\mathfrak{c}}$.

5.5.2 Tuning $\mathfrak{i}^{\mathfrak{r}}$

Let us remind that $\mathfrak{i}^{\mathfrak{r}}$ is a function which maps a role name $r \in \mathsf{RN}$ to a value $x \in [0, 2]$. Its primary motivation is to define a user-identified importance value for an individual role name. A distinguished characteristic of $\mathfrak{i}^{\mathfrak{r}}$ to $\mathfrak{i}^{\mathfrak{c}}$ is that, not only restricted to primitive ones, ones may also define an importance on defined role names. We bear this understanding on the development of a strategy to tune $\mathfrak{i}^{\mathfrak{r}}$ as follows.

Our primary motivation of providing a strategy to help tuning $\mathfrak{i}^{\mathfrak{r}}$ is similar to that one of $\mathfrak{i}^{\mathfrak{c}}$. That is, realistic ontologies are complex – consisting in plenty of role names. An intuitive way to simplify the task of tuning $\mathfrak{i}^{\mathfrak{r}}$ is to proceed on a more general role name. We note that, suppose $r \sqsubseteq s \in \mathcal{T}$, a role s is said to be more general than a role r. This intuition shapes our development as follows:

Definition 5.16. Let $\mathsf{RN}(\mathcal{T})$ be a set of role names occurring in $\mathcal{T}$. Then, a *propagation for role importance* is a "partial" function $\mathfrak{I}^{\mathfrak{r}} : \mathsf{RN}(\mathcal{T}) \to [0, 2]$ such that a mapping n of $\mathfrak{I}^{\mathfrak{r}}$ on X (*i.e.* $\mathfrak{I}^{\mathfrak{r}}(X) = n$) is inductively defined as follows:[1].

1. $\forall X' \in \mathsf{RN}(\mathcal{T})$: $(X' \sqsubseteq X \in \mathcal{T} \Longrightarrow \mathfrak{i}^{\mathfrak{r}}(X) = n$ and $\mathfrak{I}^{\mathfrak{r}}(X') = n)$; and
2. $\forall X' \in \mathsf{RN}(\mathcal{T})$: $(X' \sqsubseteq X \notin \mathcal{T} \Longrightarrow \mathfrak{i}^{\mathfrak{r}}(X) = n)$.

There are a few concerns that we should take into account, *i.e.* **(1)** inconsistent preferences of roles occurring on the same branch of the role hierarchy; and **(2)** expressing preferences when a TBox $\mathcal{T}$ contains equivalent role names. We discuss these in order.

Let us take a look on our first concern through the following preference expression (according to an ontology given in Example 5.1): "my role importance goes through canMoveWithLegs, especially canWalk". Suppose we take $\mathfrak{I}^{\mathfrak{c}}(\mathsf{canMoveWithLegs}) = 1.2$ and $\mathfrak{I}^{\mathfrak{c}}(\mathsf{canWalk}) = 1.3$. Here, canWalk is redefined on $\mathfrak{i}^{\mathfrak{r}}$ twice. Similar to $\mathfrak{I}^{\mathfrak{c}}$, we handle this problem with the use of operators $\oplus : [0, 2]^2 \to [0, 2]$.

[1]Later, we discuss some restrictions the readers should take into account when the notion $\mathfrak{I}^{\mathfrak{r}}$ is employed.

Definition 5.17. Let $r \in \mathsf{RN}(\mathcal{T})$ be a set of role names occurring in $\mathcal{T}$ and $x_0, x_1 \in [0, 2]$. Also, let $\mathfrak{i}^{\mathfrak{r}}(r) = x_0$ be the previous mapping on r. We compute a new mapping $\mathfrak{i}^{\mathfrak{r}}(r) = x_1$ as follows:

$$\mathfrak{i}^{\mathfrak{r}}(r) = \begin{cases} x_1 & \text{if } \mathfrak{i}^{\mathfrak{r}} \text{ is not defined on } r \\ x_0 \oplus x_1 & \text{otherwise} \end{cases} \tag{5.23}$$

The operator remains abstract here as its concrete operators may vary on the context of use and may be defined in the same sense as $\mathfrak{I}^{\mathfrak{c}}$ (*e.g.* Equation 5.20 to 5.22). For example, an agent may prefer to take the last mapping when that agent says exceptional cases (*e.g.* r except s where $r \in \mathcal{R}_s$) in order to suppress the previously propagated value. Also, an agent may prefer to take the last mapping when that agent would like to emphasize some special circumstances (*e.g.* r especially s where $r \in \mathcal{R}_s$) in order to suppress the previously propagated value.

Example 5.15. From Example 5.1, an agent might say "My interest is $\mathsf{canMoveWithLegs}$ except $\mathsf{canWalk}$". Let us take $\mathfrak{I}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}) = 1.2$ (*i.e.* $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}) = 1.2$ and $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 1.2$) and $\mathfrak{I}^{\mathfrak{r}}(\mathsf{canWalk}) = 0$ (*i.e.* $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 1.2 \oplus 2$). Using $\oplus_{first}$ for $\oplus$ yields $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}) = 1.2$ and $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 0$.

□

Example 5.16. From Example 5.1, an agent might say "My interest is $\mathsf{canMoveWithLegs}$, especially $\mathsf{canWalk}$". Let us take $\mathfrak{I}^{\mathfrak{c}}(\mathsf{canMoveWithLegs}) = 1.2$ (*i.e.* $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}) = 1.2$ and $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 1.2$) and $\mathfrak{I}^{\mathfrak{c}}(\mathsf{canWalk}) = 1.3$ (*i.e.* $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 1.2 \oplus 1.3$). Using $\oplus_{last}$ for $\oplus$ yields $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}) = 1.2$ and $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 1.3$.

□

Lastly, we discuss the second concern. That is, "what happens if a defined role name r_1 is defined on $\mathfrak{I}^{\mathfrak{r}}$ and there exists another defined role name r_2 such that $r_1 \sqsubseteq_{\mathcal{T}} r_2$ and $r_2 \sqsubseteq_{\mathcal{T}} r_1$?" Similar to our basic handling of this case in $\mathfrak{I}^{\mathfrak{c}}$, we recommend to treat r_2 in the same way as r_1 (because they are equivalent). Nevertheless, this also requires further work to explore other possibilities for coping with this problem and investigate desired properties the notion $\overset{\pi}{\sim}_{\mathcal{T}}$ should hold when it is used with $\mathfrak{I}^{\mathfrak{r}}$.

5.5.3 Tuning $\mathfrak{s}^{\mathfrak{c}}$

In this subsection, we present a strategy for tuning primitive concepts similarity. If an ABox $\mathcal{A}$ is presented, then we can induce the canonical interpretation $\mathcal{I}_{\mathcal{A}}$ (*cf.* Definition 3.7) from $\mathcal{A}$ to calculate primitive concepts similarity for all possible primitive concept pairs. Suppose that $\mathcal{I}_{\mathcal{A}}$ is constructed and let $A, B \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, we establish the following calculation for the function $\mathfrak{s}^{\mathfrak{c}}$.

$$\mathfrak{s}^{\mathfrak{c}}(A, B) = \begin{cases} 1 & \text{if } A^{\mathcal{I}_{\mathcal{A}}} = B^{\mathcal{I}_{\mathcal{A}}} = \emptyset \\ \frac{|A^{\mathcal{I}_{\mathcal{A}}} \cap B^{\mathcal{I}_{\mathcal{A}}}|}{|A^{\mathcal{I}_{\mathcal{A}}} \cup B^{\mathcal{I}_{\mathcal{A}}}|} & \text{otherwise} \end{cases} \tag{5.24}$$

where $|\cdot|$ represents the set cardinality.

Intuitively, Equation 5.24 computes the commonality of both primitive concept names. Since $\mathcal{O}_{\mathsf{med}}$ does not contain an ABox $\mathcal{A}$, let us use a handcraft ontology to exemplify the calculation.

Example 5.17. Let a family ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ in which $\mathcal{T}$ is defined as follows:

$$\begin{aligned} \mathsf{Grandfather} &\equiv \mathsf{Man} \sqcap \exists \mathsf{child}.\mathsf{Parent} \\ \mathsf{Parent} &\equiv \mathsf{Person} \sqcap \exists \mathsf{child}.\mathsf{Person} \\ \mathsf{Man} &\equiv \mathsf{Male} \sqcap \mathsf{Person} \end{aligned}$$

Let an ABox $\mathcal{A}$ is defined as follows:

$$\begin{array}{rl} \mathsf{child}(\mathit{john}, \mathit{elise}) & \mathsf{child}(\mathit{emma}, \mathit{watson}) \\ \mathsf{Person}(\mathit{john}) & \mathsf{Person}(\mathit{elise}) \\ \mathsf{Person}(\mathit{emma}) & \mathsf{Person}(\mathit{watson}) \\ \mathsf{Male}(\mathit{john}) & \mathsf{Male}(\mathit{watson}) \end{array}$$

Thus, $\Delta^{\mathcal{I}_\mathcal{A}} = \{\mathit{elise}, \mathit{john}, \mathit{emma}, \mathit{watson}\}$, $\mathsf{Person}^{\mathcal{I}_\mathcal{A}} = \{\mathit{john}, \mathit{elise}, \mathit{emma}, \mathit{watson}\}$, $\mathsf{Male}^{\mathcal{I}_\mathcal{A}} = \{\mathit{john}, \mathit{watson}\}$, and $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Person}, \mathsf{Male}) = \frac{|\mathit{john}, \mathit{watson}|}{|\mathit{john}, \mathit{elise}, \mathit{emma}, \mathit{watson}|} = \frac{1}{2} = 0.5$. □

5.5.4 Tuning $\mathfrak{s}^{\mathfrak{r}}$

This subsection presents a strategy for tuning primitive roles similarity. Indeed, we attempt in the similar fashion as what we do for $\mathfrak{s}^{\mathfrak{c}}$. That is, we use the canonical interpretation $\mathcal{I}_\mathcal{A}$ to obtain primitive roles similarity. Let $r, s \in \mathsf{RN}^{\mathsf{pri}}(\mathcal{T})$ and define operators $\cdot^f$ and $\cdot^s$ for any primitive role r as $r^f = \{x \mid (x, y) \in r^{\mathcal{I}_\mathcal{A}}\}$ and $r^s = \{y \mid (x, y) \in r^{\mathcal{I}_\mathcal{A}}\}$, respectively, then:

$$\mathfrak{s}^{\mathfrak{r}}(r, s) = \begin{cases} 1 & \text{if } r^{\mathcal{I}_\mathcal{A}} = s^{\mathcal{I}_\mathcal{A}} = \emptyset \\ \lambda \cdot \frac{|r^f \cap s^f|}{|r^f \cup s^f|} + (1 - \lambda) \cdot \frac{|r^s \cap s^s|}{|r^s \cup s^s|} & \text{otherwise} \end{cases} \tag{5.25}$$

where $0 < \lambda < 1$ and $|\cdot|$ represents the set cardinality.

Intuitively, Equation 5.25 is defined as the weighted sum of the commonality on the first arguments of roles and the commonality on the second arguments of roles. It is recommended to set the weight $\lambda = (|r^f \cup s^f|)/(|r^f \cup s^f| + |r^s \cup s^s|)$, *i.e.* the proportion of all individuals appearing on the first arguments to all individuals appearing on both arguments. The following example exemplifies the calculation.

Example 5.18. Let a family ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ in which $\mathcal{T}$ is defined as follows:

$$\begin{aligned} \mathsf{Parent} &\equiv \mathsf{Person} \sqcap \exists \mathsf{child}.\mathsf{Person} \\ \mathsf{BrotherSister} &\equiv \mathsf{Person} \sqcap \exists \mathsf{sibling}.\mathsf{Person} \end{aligned}$$

Let an ABox $\mathcal{A}$ is defined as follows:

$$\begin{array}{rl} \mathsf{sibling}(john, max) & \mathsf{sibling}(yok, watson) \\ \mathsf{child}(emma, yok) & \mathsf{child}(john, elise) \\ \mathsf{child}(emma, watson) & \mathsf{Person}(john) \\ \mathsf{Person}(elise) & \mathsf{Person}(emma) \\ \mathsf{Person}(watson) & \mathsf{Person}(max) \\ \mathsf{Person}(yok) & \end{array}$$

Thus, $\Delta^{\mathcal{I}_\mathcal{A}} = \{elise, john, emma, watson, max, yok\}$, $\mathsf{child}^f = \{emma, john\}$, $\mathsf{sibling}^f = \{yok, john\}$, $\mathsf{child}^s = \{yok, watson, elise\}$, $\mathsf{sibling}^s = \{watson, max\}$, and $\mathfrak{s}^\mathfrak{r}(\mathsf{child}, \mathsf{sibling}) = \frac{3}{7} \cdot \frac{1}{3} + \frac{4}{7} \cdot \frac{1}{4} \approx 0.48$.

□

5.5.5 Tuning $\mathfrak{d}$

The primary motivation of this aspect is to capture an expression of total expression on a role beyond a corresponding nested concept [91]. Hence, tuning this aspect may requires skilled domain expertise. For example, SNOMED CT ontology engineers realize that $\mathsf{roleGroup}$ is used to nestedly group existential restrictions; hence, it can unintentionally increase the degree of similarity due to role commonality. Considering this fact, they may set $\mathfrak{d}(\mathsf{roleGroup}) = 0$. This shows that role discount factor of different role names may be independent. However, the same strategy as $\mathfrak{I}^\mathfrak{r}$ can be employed to comfort on configuring this aspect, *i.e.* a propagation for role discount factor via a more general role name.

5.5.6 Example: Using The Strategies

Now, we are ready to discuss a simple methodology for reconciling the previously proposed strategies with the measure sim^π (Definition 5.13). Section 5.5 shows that each aspect of preference profile may need different strategies for tuning. For example, $\mathfrak{i}^\mathfrak{c}$ may be tuned via a defined concept name using a propagation for primitive concept importance $\mathfrak{I}^\mathfrak{c}$ whereas $\mathfrak{i}^\mathfrak{r}$ may be tuned via a more general role name using a propagation for role importance $\mathfrak{I}^\mathfrak{r}$. Also, both $\mathfrak{s}^\mathfrak{c}$ and $\mathfrak{s}^\mathfrak{r}$ may employ the canonical interpretation $\mathcal{I}_\mathcal{A}$ to initialize values. In the following, we explain procedural steps that the readers may follow to tune preference profile for their use. These steps also hint a system flow of similarity-based under the agent's profile applications, such as the best matching concept under the agent's profile application.

1. An agent may start with tuning each aspect of preference profile individually;

2. To help tuning $\mathfrak{i}^\mathfrak{c}$, a system may present the concept hierarchy w.r.t. an ontology. Then, an agent indirectly specifies primitive concept names via a defined concept name depicted on the hierarchy with the notion $\mathfrak{I}^\mathfrak{c}$ (*cf.* Definition 5.14 and Definition 5.15). Some patterns of an agent's utterance may be associated with certain operators, *e.g.*

(a) We may associate "A especially B" with $\oplus_{first}$, where $depth(A) < depth(B)$ and $depth(X)$ is the depth of X on the concept hierarchy;

(b) We may associate "A except B" with $\oplus_{first}$, where $depth(A) < depth(B)$ and $C_1 \neq \top$;

(c) We may associate "$\top$ except B" with $\oplus_{last}$; and

(d) Otherwise, the agent-defined default concrete operator is used;

3. To help tuning $\mathfrak{i}^{\mathfrak{r}}$, a system may present the role hierarchy w.r.t. an ontology. Then, an agent indirectly specifies role names via a more general role name name depicted on the hierarchy with the noion $\mathfrak{I}^{\mathfrak{r}}$ (*cf.* Definition 5.16 and Definition 5.17). Similarly, some patterns of an agent's utterance may be associated with certain operators, *e.g.*

(a) We may associate "r especially s" with $\oplus_{last}$, where $r \in \mathcal{R}_s$;

(b) We may associate "r except s" with $\oplus_{last}$, where $r \in \mathcal{R}_s$; and

(c) Otherwise, the agent-defined default concrete operator is used;

4. To help tuning $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$, a system may construct the canonical interpretation, which is induced from $\mathcal{A}$. Then, each initial value for all possible primitive concept pairs and primitive role pairs is calculated according to Equation 5.24 and Equation 5.25, respectively;

5. An agent may refine the agent's preference profile if that agent wishes.

We exemplify the methodology in its applicable use cases, such as trip planning (Example 5.19).

Example 5.19. (Continuation from Example 5.1) We expand each definition in $\mathcal{T}$ as follows:

$$\begin{aligned} \mathsf{ActivePlace} &\equiv X \sqcap \mathsf{Place} \sqcap \exists \mathsf{canWalk}.\mathsf{Trekking} \sqcap \exists \mathsf{canSail}.\mathsf{Kayaking} \\ \mathsf{Mangrove} &\equiv Y \sqcap \mathsf{Place} \sqcap \exists \mathsf{canWalk}.\mathsf{Trekking} \\ \mathsf{Beach} &\equiv Z \sqcap \mathsf{Place} \sqcap \exists \mathsf{canSail}.\mathsf{Kayaking} \end{aligned}$$

where X, Y, and Z are fresh primitive concept names. Furthermore, $\mathcal{R}_{\mathsf{canWalk}} = \{t, \mathsf{cMWL}\}$[1] and $\mathcal{R}_{\mathsf{canSail}} = \{t, \mathsf{cTWS}\}$ where t and u are also fresh primitive role names.

Let an ABox $\mathcal{A}$ be defined as follows:

$$\begin{array}{rl} \mathsf{cMWL}(p_2, t_1) & \mathsf{cTWS}(p_3, k_1) \\ \mathsf{canWalk}(p_2, t_1) & \mathsf{canSail}(p_3, k_1) \\ \mathsf{Trekking}(t_1) & \mathsf{Kayaking}(k_1) \\ \mathsf{Place}(p_1) & \mathsf{Place}(p_2) \\ \mathsf{Place}(p_3) & \mathsf{Mangrove}(p_2) \\ \mathsf{Beach}(p_3) & \end{array}$$

[1]Obvious abbreviations are used here for the sake of succinctness.

To query for a desired place, an agent needs to express his preferences. Suppose the agent says "My interest is a place where I can travel with by feet, especially walking", *i.e.* $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Place}) = 1.5$, $\mathfrak{I}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}) = 1.5$, and $\mathfrak{I}^{\mathfrak{r}}(\mathsf{canWalk}) = 1.8$. Also, it yields $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}) = 1.5$ and $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 1.5 \oplus_{last} 1.8 = 1.8$.

Constructing the canonical interpretation from $\mathcal{A}$, we obtain $\Delta^{\mathcal{I}_{\mathcal{A}}} = \{p_1, p_2, p_3, t_1, k_1\}$, $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Trekking}, \mathsf{Kayaking}) = 0$, and $\mathfrak{s}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}, \mathsf{canTravelWithSails}) = 0$.

Let ActivePlace, Mangrove, Place, Trekking, Kayaking, canWalk, and canSail are rewritten shortly as AP, M, P, T, K, cW, and cS, respectively. Using Definition 5.8, $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{M}})$

$$
\begin{aligned}
&= (\tfrac{2.5}{5.3}) \cdot \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_{\mathsf{AP}}, \mathcal{P}_{\mathsf{M}}) + (\tfrac{2.8}{5.3}) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= (\tfrac{2.5}{5.3}) \cdot (\tfrac{\mathfrak{i}(X)\cdot\max\{\mathfrak{s}(X,Y),\mathfrak{s}(X,\mathsf{P})\}+\mathfrak{i}(\mathsf{P})\cdot\max\{\mathfrak{s}(\mathsf{P},\mathsf{Y}),\mathfrak{s}(\mathsf{P},\mathsf{P})\}}{\mathfrak{i}(X)+\mathfrak{i}(\mathsf{P})}) \\
&+(\tfrac{2.8}{5.3}) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= (\tfrac{2.5}{5.3})(\tfrac{1\cdot\max\{0,0\}+1.5\cdot\max\{0,1\}}{1+1.5}) \\
&+(\tfrac{2.8}{5.3}) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= (\tfrac{2.5}{5.3})(\tfrac{1.5}{2.5}) + (\tfrac{2.8}{5.3}) \left[\tfrac{\mathfrak{i}(\mathsf{cW})\cdot\max\{\mathsf{e\text{-}hd}^{\pi}(\exists\mathsf{cW}.\mathsf{T},\exists\mathsf{cW}.\mathsf{T})\}+1\cdot 0}{\mathfrak{i}(\mathsf{cW})+\mathfrak{i}(\mathsf{cS})}\right] \\
&= (\tfrac{2.5}{5.3})(\tfrac{1.5}{5.3}) + (\tfrac{2.8}{5.3}) \left[\tfrac{1.8\cdot 1+1\cdot 0}{1+1.8}\right] \approx 0.623
\end{aligned}
$$

Following the same step, we obtain $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{M}}, \mathcal{T}_{\mathsf{AP}}) \approx 0.767$. Hence, $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) \approx 0.695$ by using Definition 5.13. Also, we obtain $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{B}}) \approx 0.472$ and $\mathsf{hd}^{\pi}(\mathcal{T}_{\mathsf{B}}, \mathcal{T}_{\mathsf{AP}}) \approx 0.714$. Hence, $\mathsf{sim}^{\pi}(\mathsf{B}, \mathsf{AP}) \approx 0.593$.

The fact that $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) > \mathsf{sim}^{\pi}(\mathsf{B}, \mathsf{AP})$ corresponds to the agent's perception. □

5.5.7 Relationship to Learning-based Approach

Our proposed development uses the canonical interpretation $\mathcal{I}_{\mathcal{A}}$ to compute numerical values for mappings on $\mathfrak{s}^{\mathfrak{c}}$ (*cf.* Subsection 5.5.3) and $\mathfrak{s}^{\mathfrak{r}}$ (*cf.* Subsection 5.5.4). Its drawback is that an existence of the canonical interpretation $\mathcal{I}_{\mathcal{A}}$ is required. This section rather discusses an alternative approach to obtain values for mappings on $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$.

In addition to our proposed logic-based approach, another natural way to configure both $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$ is to employ existing machine learning techniques on a large corpus. For example, one may use Word2vec [155] with a large corpus of text to produce a vector space. Each word in the corpus will be assigned by a corresponding vector in the space. Word vectors are positioned in the vector space such that words sharing common features in the corpus are located in close proximity to one another in the space. This characterization can later be converted into elements of the mapping $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$. Reconciling an ontology with machine learning techniques to improve an application of $\overset{\pi}{\sim}_{\mathcal{T}}$ is interesting but is outside the scope of this work. We leave this as a future task.

5.6 Implementation Methods of sim^{π}

Theorem 5.5 tells us that sim^{π} can be computed in the polynomial time. This section exhibits two algorithmic procedures of sim^{π} belonging to that class.

5.6.1 Top-Down Implementation of sim^{π}

Algorithm 1 Pseudo code for hd^{π} using top-down fashion (Part 1)

```
1: function hd^π(T_D, T_C, π)
2:     return (μ^π(T_D, π) × p-hd^π(P_D, P_C, π)) + ((1 − μ^π(T_D, π)) × e-set-hd^π(E_D, E_C, π))
3: end function
4:
5: function μ^π(T_D, π)
6:     if P_D.isEmpty() and E_D.isEmpty() then
7:         return 1
8:     end if
9:     return Σ i^c(P_D, π) / (Σ i^c(P_D, π) + Σ i^r(E_D, π))
10: end function
11:
12: function p-hd^π(P_D, P_C, π)
13:     if Σ i^c(P_D, π) = 0 then
14:         return 1
15:     else if Σ i^c(P_C, π) = 0 then
16:         return 0
17:     else
18:         w ← 0
19:         for A ∈ P_D do
20:             m ← 0
21:             for B ∈ P_C do
22:                 v ← ŝ(A, B)
23:                 if v > m then
24:                     m ← v
25:                 end if
26:             end for
27:             w ← w + (m × î(A))
28:         end for
29:         return w / Σ i^c(P_D, π)
30:     end if
31: end function
```

In Definition 5.8, hd^{π} is established by an inductive procedure. Therefore, it is a very straightforward way to implement the procedure by *recursion* (see Algorithm 1).

For the first two parts of Algorithm 1, hd^{π} is directly followed from Equation 5.10 of Definition 5.8. That is, it receives three parameters as inputs, *viz.* a description tree $\mathcal{T}_D$, a description tree $\mathcal{T}_C$, and a preference profile π. Suppose $\mathcal{T}_D$ be defined as $\mathcal{P}_D \cup \mathcal{E}_D$, $\mathcal{T}_C$ be defined as $\mathcal{P}_C \cup \mathcal{E}_C$, and $\pi = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ is given. The function $\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C, \pi)$ computes the function value for a composition in a prescribed way from the function values of the composing parts, *i.e.* $\mu^{\pi}(\mathcal{T}_D, \pi)$, $\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C, \pi)$, and $\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C, \pi)$.

Algorithm 1 Pseudo code for hd^{π} using top-down fashion (Part 2)

```
32: function e-set-hd^π(E_D, E_C, π)
33:     if Σ i^r(E_D, π) = 0 then
34:         return 1
35:     else if Σ i^r(E_C, π) = 0 then
36:         return 0
37:     else
38:         w ← 0
39:         for ∃r.X ∈ E_D do
40:             m ← 0
41:             for ∃s.Y ∈ E_C do
42:                 e ← e-hd^π(∃r.X, ∃s.Y, π)
43:                 if e > m then
44:                     m ← e
45:                 end if
46:             end for
47:             w ← w + (m × î(r))
48:         end for
49:         return w / Σ i^r(P_D, π)
50:     end if
51: end function
```

μ^{π}, $\mathsf{p\text{-}hd}^{\pi}$, $\mathsf{e\text{-}set\text{-}hd}^{\pi}$ are also followed from Equation 5.11, 5.12, and 5.13, respectively, of Definition 5.8. Each internally uses subfunctions $\sum \mathfrak{i}^{\mathsf{c}}$ and $\sum \mathfrak{i}^{\mathsf{r}}$ (see the forth part of Algorithm 1) to calculate the total number of concept importance and the total number of role importance, respectively.

For the last two parts of Algorithm 1, $\mathsf{e\text{-}hd}^{\pi}$ is directly followed from Equation 5.14 of Definition 5.8. To compute the function value $\mathsf{e\text{-}hd}^{\pi}$, we recursively compute the function value hd^{π} on the children of certain nodes (denoted by X and Y) and π. γ^{π} is directly followed from Equation 5.15 of Definition 5.8. Also, γ^{π} internally invokes subfunction $\sum \mathfrak{i}^{\mathsf{r}}$ to calculate the total number of role importance.

The reader may easily observe that the time efficiency of Algorithm 1 is quintic because the computation of $\mathsf{p\text{-}hd}^{\pi}$ is quadratic and $\mathsf{e\text{-}set\text{-}hd}^{\pi}$ contains double nested loops which indirectly make recursive calls to hd^{π}. It is also not difficult to observe that the number of recursive calls is upper bounded by the height of the description tree.

It is worth to mention that using hd^{π} requires concept descriptions to be transformed into $\mathcal{ELH}$ description trees. Taking this as an advantage, the next subsection introduces an alternative way to compute hd^{π} from bottom to up, which is approximately three times faster than the counterpart top-down approach in the worst case (*cf.* Subsection 5.7.1 for useful discussion).

Algorithm 1 Pseudo code for hd^{π} using top-down fashion (Part 3)

52: **function** $\mathsf{e\text{-}hd}^{\pi}(\exists r.X, \exists s.Y, \pi)$
53: **return** $\gamma^{\pi}(r, s, \pi) \times (\hat{\mathfrak{d}}(r) + ((1 - \hat{\mathfrak{d}}(r)) \times \mathsf{hd}^{\pi}(\mathcal{T}_X, \mathcal{T}_Y, \pi)))$
54: **end function**
55:
56: **function** $\gamma^{\pi}(r, s, \pi)$
57: **if** $\sum \mathfrak{i}^{\mathfrak{r}}(\mathcal{R}_r, \pi)$ **then**
58: **return** 1
59: **else**
60: $w \leftarrow 0$
61: **for** $\zeta \in \mathcal{R}_r$ **do**
62: $m \leftarrow 0$
63: **for** $\varrho \in \mathcal{R}_s$ **do**
64: $v \leftarrow \hat{\mathfrak{s}}(\zeta, \varrho)$
65: **if** $v > m$ **then**
66: $m \leftarrow v$
67: **end if**
68: **end for**
69: $w \leftarrow w + (m \times \hat{\mathfrak{i}}(\zeta))$
70: **end for**
71: **return** $w / \sum \mathfrak{i}^{\mathfrak{r}}(\mathcal{R}_r, \pi)$
72: **end if**
73: **end function**

5.6.2 Bottom-Up Implementation of sim^{π}

Rather than computing (possibly duplicated) value of hd^{π} again and again, Algorithm 2 employs the classical bottom-up version of dynamic programming technique to compute hd^{π} of the smaller subtrees and records the results in a table (see the variable $result[\cdot][\cdot]$ in Algorithm 2) from which a solution to the original computation of hd^{π} can be then obtained (*cf.* at line no. 20, the function returns value $result[0][0]$).

To compute hd^{π} from bottom to up, we need to know the height of the trees in advance. For Algorithm 2, we employ "breath-first search" algorithm (denoted by BFS) to determine the height of each description tree (*cf.* line no. 4 and 5 of the algorithm). Algorithm 2 reuses the methods μ^{π}, $\mathsf{p\text{-}hd}^{\pi}$, $\mathsf{e\text{-}set\text{-}hd}^{\pi}$, γ^{π}, $\sum \mathfrak{i}^{\mathfrak{c}}$, and $\sum \mathfrak{i}^{\mathfrak{r}}$ from Algorithm 1 and provides pseudo code for $\mathsf{e\text{-}hd}^{\pi}$ since it is merely overridden.

What is the time complexity of Algorithm 2? It should be quintic because the algorithm considers the similarity of all the different pairs of two concept names for h times (*cf.* line no. 6). More formally, we know $result[\mathcal{T}_{\gamma}][\mathcal{T}_{\lambda}] \in \mathcal{O}(v^2)$ where v denotes the set cardinality of $\mathcal{P}_x$ (and $\mathcal{E}_x$) for any description tree x. Let $m(i)$ and $n(i)$ be the number of nodes on level i of description trees D and C, respectively. Then, the number of times operation

Algorithm 1 Pseudo code for hd^{π} using top-down fashion (Part 4)

```
74: function ∑ i^c(P_D, π)
75:     w ← 0
76:     for A ∈ P_D do
77:         w ← w + î(A)
78:     end for
79:     return w
80: end function
81:
82: function ∑ i^r(E_D, π)
83:     w ← 0
84:     for ∃r.X ∈ E_D do
85:         w ← w + î(r)
86:     end for
87:     return w
88: end function
89:
90: function ∑ i^r(R_r, π)
91:     w ← 0
92:     for r ∈ R_r do
93:         w ← w + î(r)
94:     end for
95:     return w
96: end function
```

$result[\cdot][\cdot]$ is executed (say C) is equal to:

$$\begin{aligned}
C &= \sum_{i=0}^{h-1}\sum_{j=0}^{m(i)}\sum_{k=0}^{n(i)} v^2 \\
&= v^2 \sum_{i=0}^{h-1}\sum_{j=0}^{m(i)}\sum_{k=0}^{n(i)} 1 \\
&= v^2 \sum_{i=0}^{h-1}\sum_{j=0}^{m(i)} (n(i)+1) \\
&= v^2 \sum_{i=0}^{h-1} (n(i)+1)(m(i)+1) \\
&= v^2 \Big[[(n(0)+1)(m(0)+1)] + [(n(1)+1)(m(1)+1)] \\
&\quad + \cdots + [(n(h-1)+1)(m(h-1)+1)] \Big]
\end{aligned}$$

Thus, the algorithm makes the similar number of operations as Algorithm 1, plus an additional amount of extra space. On the positive side, the algorithm has never recursively invoked itself to determine the similarity of different pairs of nested concepts, *i.e.* it directly uses values stored in the table. The algorithm also shows that computing the

Algorithm 2 Pseudo code for hd^{π} using bottom-up fashion

Initialize a global $result[\cdot][\cdot]$ to store the degree of similarity between 2 concepts.

function $\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C, \pi)$
    Map $< \mathbb{Z}, \text{List} < \mathcal{T} >> map_D \leftarrow \mathsf{BFS}(\mathcal{T}_D)$ ▷ map_D stores nodes on each level of $\mathcal{T}_D$
    Map $< \mathbb{Z}, \text{List} < \mathcal{T} >> map_C \leftarrow \mathsf{BFS}(\mathcal{T}_C)$ ▷ map_C stores nodes on each level of $\mathcal{T}_C$
    $h \leftarrow map_D$.size()
    for $i = h - 1$ **to** 0 **do**
        List $< \mathcal{T} > list_{\mathcal{T}_\Gamma} \leftarrow map_D$.get(i)
        List $< \mathcal{T} > list_{\mathcal{T}_\Lambda} \leftarrow map_C$.get(i)
        for $\mathcal{T}_\gamma \in list_{\mathcal{T}_\Gamma}$ **do**
            for $list_{\mathcal{T}_\Lambda} \neq$ null and $\mathcal{T}_\lambda \in list_{\mathcal{T}_\Lambda}$ **do**
                if $i = h - 1$ **then**
                    $result[\mathcal{T}_\gamma][\mathcal{T}_\lambda] \leftarrow \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_\gamma, \mathcal{P}_\lambda, \pi)$
                else
                    $result[\mathcal{T}_\gamma][\mathcal{T}_\lambda] \leftarrow (\mu^{\pi}(\mathcal{T}_\gamma, \pi) \times \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_\gamma, \mathcal{P}_\lambda, \pi))$
                    $+ ((1 - \mu^{\pi}(\mathcal{T}_\gamma, \pi)) \times \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_\gamma, \mathcal{E}_\lambda, \pi))$
                end if
            end for
        end for
    end for
    return $result[0][0]$
end function

function $\mathsf{e\text{-}hd}^{\pi}(\exists r.X, \exists s.Y, \pi)$
    $hd' \leftarrow result[\mathcal{T}_X][\mathcal{T}_Y]$
    if $hd' =$ null **then**
        $hd' \leftarrow 0$
    end if
    return $\gamma^{\pi}(r, s, \pi) \times (\hat{\mathfrak{d}}(r) + ((1 - \hat{\mathfrak{d}}(r)) \times hd'))$
end function

similarity of nodes from level i, where i is greater than the minimum height of description trees (*cf.* the condition $list_{\mathcal{T}_\Lambda}! = \text{null}$ at line no. 11), is irrelevant to the computation.

Algorithm 2 does work productively in an environment where recursion is fairly expensive. For example, imperative languages, such as Java, C, and Python, are typically faster if using a loop and slower if doing a recursion. On the other hand, for some implementations of functional programming languages, iterations may be very expensive and recursion may be very cheap. In many implementations of them, recursion is transformed into a simple jump but changing the loop variables (which are mutable) requires heavy operations. Subsection 5.7.1 reports that the practical performance agrees to this theoretical analysis that the bottom-up approach is more efficient when implemented by imperative languages, such as Java.

5.7 Empirical Evaluation

This section evaluates the practical performance of both algorithms against sim[1], reassures pragmatically the backward compatibility of sim^π under π_0 (Theorem 5.2 already proves this), and discusses the applicability of sim^π in potential use cases.

5.7.1 Performance Analysis and Backward Compatibility of sim^π

Both versions of sim^π (*cf.* Subsection 5.6.1 and Subsection 5.6.2) are implemented in Java version 1.8 with the usage of Spring Boot version 1.3.3.RELEASE. All the dependencies are managed by Apache Maven version 3.2.5. We also implement unit test cases along with the development of both versions to verify the correctness of their behaviors. In the current state (when we are writing this work), there are 111 unit test cases. All of them are written to cover important parts of both implementations.

To perform benchmarking, we have selected SNOMED CT as a test ontology. As mentioned in Appendix A, it is one of the largest and the most widely used medical ontologies currently available, and also, is expressible in $\mathcal{ELH}$. In our experiments, we employ a SNOMED CT ontology version from January 2005 (hitherto referred as $\mathcal{O}_{\text{SNOMED}}$) which contains 379,691 concept names and 62 role names. Moreover, each defined concept is categorized into the 18 mutually exclusive top-level concepts. In the sense of subsumption relation, concepts belonging to the same category should be more similar than those belonging to different categories.

For our experiments, we used a 2.4 GHz Intel Core i5 with 8 GB RAM under OS X El Capitan. Unfortunately, the overall number of concept pairs in $\mathcal{O}_{\text{SNOMED}}$ is approximately 10^{11}. Suppose an execution of sim^π takes around a millisecond, we still need around 1,158 days in order to complete the entire ontology. According to this reason, we consider 2 out of 18 categories, *viz. Clinical Finding* and *Procedure*, although there are more category pairs. Then, we randomly select 0.5% of *Clinical Finding*, *i.e.* 206 concepts, denoted by $\mathbf{C}'_1$. After that, we randomly select the same number of concepts from *Procedure*, *i.e.* 206

[1]We have re-implemented sim (proposed in [151]) based on the same technologies and techniques as sim^π.

concepts, denoted by $\mathbf{C}'_2$. This sampled set is denoted by $\mathcal{O}'_{\text{SNOMED}}$, *i.e.* $\mathcal{O}'_{\text{SNOMED}} = \mathbf{C}'_1 \cup \mathbf{C}'_2$. Then, we create three test datasets from this sampled set, *viz.* $\mathbf{C}'_1 \times \mathbf{C}'_1$, $\mathbf{C}'_1 \times \mathbf{C}'_2$, and $\mathbf{C}'_2 \times \mathbf{C}'_2$.

Firstly, we estimate the practical performance of the top-down fashion. For each concept pair in each set, we **1)** employ the default preference profile π_0 on (top-down) sim^{π}; **2)** measure the similarity of concepts in $\mathcal{O}'_{\text{SNOMED}}$ by peeking on $\mathcal{O}_{\text{SNOMED}}$ to help unfolding; **3)** repeat the previous step with (top-down) sim; **4)** repeat steps **2)-3)** three times and calculate the statistical results (in milliseconds). Results are gathered on Table 5.1. We note that *avg*, *max*, and *min* represent the execution time for measuring similarity of a concept pair in the average case, in the worst case, and in the best case, respectively.

Table 5.1: Execution time of top-down sim and top-down sim^{π_0} on $\mathcal{O}'_{\text{SNOMED}}$

Pairs	Number of pairs	sim (avg/max/min)	sim^{π_0} (avg/max/min)
$\mathbf{C}'_1 \times \mathbf{C}'_1$	25	2.280/7.000/0.000	1.800/10.000/0.000
$\mathbf{C}'_1 \times \mathbf{C}'_2$	215	2.291/97.000/0.000	2.278/84.000/0.000
$\mathbf{C}'_2 \times \mathbf{C}'_2$	1,849	3.395/45.000/0.000	3.931/128.000/0.000

Secondly, we estimate the practical performance of the bottom-up fashion by following the same steps as we did previously. Indeed, we exclude the time used to determine the height of each description tree, *i.e.* our benchmark begins from line no. 7 to 21 of Algorithm 2. Table 5.2 gathers up the results.

Table 5.2: Execution time of bottom-up sim and bottom-up sim^{π_0} on $\mathcal{O}'_{\text{SNOMED}}$

Pairs	Number of pairs	sim (avg/max/min)	sim^{π_0} (avg/max/min)
$\mathbf{C}'_1 \times \mathbf{C}'_1$	25	2.200/6.000/0.000	1.693/5.000/0.000
$\mathbf{C}'_1 \times \mathbf{C}'_2$	215	2.040/32.000/0.000	1.946/10.000/0.000
$\mathbf{C}'_2 \times \mathbf{C}'_2$	1,849	3.368/55.000/0.000	3.435/45.000/0.000

The experiment shows that the practical performance of sim^{π} is likely equal to the performance obtained by sim – as ones may not expect. The results show that the bottom-up sim^{π} performs approximately three times faster that the counterpart top-down sim^{π} (in the worst case) when implemented by imperative languages (*e.g.* Java as in our case). This conforms to our analysis discussed in Subsection 5.6.2.

Lastly, we evaluate the backward compatibility of sim^{π} with sim. Our goal is to ascertain that sim^{π} can be used interchangeably as the original sim by setting preference profile to the default one (Theorem 5.2 already proves this). To this point, we have performed an experiment on concept pairs defined in $\mathcal{O}'_{\text{SNOMED}}$. The experiment evaluates results from sim and sim^{π_0} and found that both coincide, as desired. Table 5.3 gathers the results, where "td" and "bu" are abbreviation forms of top-down and bottom-up, respectively.

Table 5.3: Results of executing sim and sim^{π_0} on $\mathcal{O}'_{\text{SNOMED}}$

Pairs	Number of pairs	td sim (avg/max /min)	td sim^{π_0} (avg/max /min)	bu sim (avg/max /min)	bu sim^{π_0} (avg/max /min)
$\mathbf{C}'_1 \times \mathbf{C}'_1$	25	0.87597/ 1.00000/ 0.67953	0.87597/ 1.00000/ 0.67953	0.87597/ 1.00000/ 0.67953	0.87597/ 1.00000/ 0.67953
$\mathbf{C}'_1 \times \mathbf{C}'_2$	215	0.57801/ 0.66546/ 0.24594	0.57801/ 0.66546/ 0.24594	0.57801/ 0.66546/ 0.24594	0.57801/ 0.66546/ 0.24594
$\mathbf{C}'_2 \times \mathbf{C}'_2$	1,849	0.79690/ 1.00000/ 0.35360	0.79690/ 1.00000/ 0.35360	0.79690/ 1.00000/ 0.35360	0.79690/ 1.00000/ 0.35360

5.7.2 Applicability of sim^{π}

Tuning via $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{d}$

We show the applicability of $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{d}$ through similarity measuring on SNOMED CT. Figure 5.3 depicts an example unfoldable terminology extracted from $\mathcal{O}_{\text{SNOMED}}$.

Considering merely objective factors regardless of the agent's preferences, it yields that $\mathsf{sim}^{\pi_0}(\mathsf{NAOAF}, \mathsf{NAOM}) \approx 0.9$[1] and $\mathsf{sim}^{\pi_0}(\mathsf{NAOAF}, \mathsf{H}) = 0.2$. The results yielding to the quite similar concepts NAOAF and NAOM, which reflects the fact that both are resided in the same cluster of SNOMED CT. However, the result yielding that the concepts NAOAF and H shares a little similarity controverts the fact that both carry neither implicit nor explicit relationship. This is indeed caused by the usage of the special-purpose role called $\mathsf{roleGroup}$ – informally read as *relation group*.

In SNOMED CT, the use of relation group is widely accepted to nestedly represent a group of existential information [156]. As a consequence, it increases unintentionally the degree of similarity due to role commonality (*i.e.* γ^{π}). Since $\mathsf{roleGroup}$ precedes every existential restriction, it is useless to regard an occurrence of this as being similar. The

[1] Obvious abbreviations are used here for the sake of succinctness.

$$
\begin{aligned}
\mathsf{NeonatalAspirationOfAmnioticFluid} &\equiv \mathsf{NeonatalAspirationSyndromes} \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{causativeAgent}.\mathsf{AmnioticFluid}) \\
\mathsf{NeonatalAspirationOfMucus} &\equiv \mathsf{NeonatalAspirationSyndromes} \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{causativeAgent}.\mathsf{Mucus}) \\
\mathsf{Hypoxemia} &\equiv \mathsf{DisorderOfRespiratorySystem} \sqcap \mathsf{DisorderOfBloodGas} \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{interprets}.\mathsf{OxygenDelivery}) \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{findingSite}.\mathsf{ArterialSystemStructure}) \\
\mathsf{BodySecretion} &\sqsubseteq \mathsf{BodySubstance} \\
\mathsf{BodySubstance} &\sqsubseteq \mathsf{Substance} \\
\mathsf{BodyFluid} &\sqsubseteq \mathsf{BodySubstance} \sqcap \mathsf{LiquidSubstance} \\
\mathsf{AmnioticFluid} &\sqsubseteq \mathsf{BodyFluid} \\
\mathsf{Mucus} &\sqsubseteq \mathsf{BodySecretion} \\
\mathsf{causativeAgent} &\sqsubseteq \mathsf{associatedWith}
\end{aligned}
$$

Figure 5.3: Example of $\mathcal{ELH}$ concept definitions defined in $\mathcal{O}_{\text{SNOMED}}$

importance contribution of roleGroup in $\mathcal{O}_{\text{SNOMED}}$ should be none. Hence, the agent S who measures similarity on SNOMED CT should set $\mathfrak{d}_S(\mathsf{roleGroup}) = 0$.

Furthermore, the SNOMED CT top concept SCT-TOP subsumes every defined concept of each category. This means this special concept is shared by every expanded concept description. Intuitively, this special top concept is of no importance for measuring similarity on SNOMED CT and we can treat the top-level concepts as directly subsumed by $\top$. As a result, the agent S should also set $\mathfrak{i}^{\mathfrak{c}}_S(\mathsf{SCT\text{-}TOP}) = 0$.

Tuning the measure with this expertise knowledge yields more realistic result. That is, the similarity of concepts under the same category which uses roleGroup in their definitions is slightly reduced. Also, the similarity of concepts under different categories is totally dissimilar. Continuing the case, $\mathsf{sim}^{\pi_S}(\mathsf{NAOAF}, \mathsf{NAOM}) \approx 0.84$ and $\mathsf{sim}^{\pi_S}(\mathsf{NAOAF}, \mathsf{H}) = 0.0$, as desired.

Tuning via $\mathfrak{s}^{\mathfrak{r}}$

Let us use the ontology given below to query for places similar to ActivePlace.

$$
\begin{aligned}
\mathsf{ActivePlace} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canSail}.\mathsf{Kayaking} \\
\mathsf{Mangrove} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk}.\mathsf{Trekking} \\
\mathsf{Supermarket} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canBuy}.\mathsf{FreshFood}
\end{aligned}
$$

Suppose the agent feels "walking" and "sailing" are similar and are "still satisfied much" on both actions. Taking $\mathfrak{s}^{\mathfrak{r}}(\mathsf{canWalk}, \mathsf{canSail}) = 0.6$ yields $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) > \mathsf{sim}^{\pi}(\mathsf{S}, \mathsf{AP})$, which conforms to the agent's preferences and needs.

Tuning via $\mathfrak{s}^{\mathfrak{c}}$

Let us use the ontology given below to query for a product which offers features the agent is satisfied with most.

$$
\begin{aligned}
\mathsf{WantedFeatures} &\sqsubseteq \mathsf{F}_0 \sqcap \mathsf{F}_1 \sqcap \mathsf{F}_2 \\
\mathsf{Item}_1 &\sqsubseteq \mathsf{F}_0 \sqcap \mathsf{F}_3 \\
\mathsf{Item}_2 &\sqsubseteq \mathsf{F}_0 \sqcap \mathsf{F}_4
\end{aligned}
$$

According to the ontology, $\mathsf{WantedFeatures}$ represents a collection of desired features and F_i (where $i \in \mathbb{N}$) represents a feature. A purchase decision is sometimes affected by satisfied alternations, which are varied by different people. Assume that the agent feels satisfaction to have F_3 if the agent cannot have F_1. Taking $\mathfrak{s}^{\mathfrak{c}}(\mathsf{F}_1, \mathsf{F}_3) = 0.8$ yields $\mathsf{sim}^{\pi}(\mathsf{WF}, \mathsf{I1}) > \mathsf{sim}^{\pi}(\mathsf{WF}, \mathsf{I2})$, which conforms to the agent's perceptions.

Tuning via $\mathfrak{i}^{\mathfrak{r}}$

Let us use the ontology given in Example 5.1 to query for places which are most similar to $\mathsf{ActivePlace}$. Typically, a human decision is affected by a priority of concerns, which are varied by different people. Suppose that the agent weights more on places which permit to "walk" more than other activities. Taking $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 2$ yields $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) > \mathsf{sim}^{\pi}(\mathsf{B}, \mathsf{AP})$, which conforms to the agent's preferences.

5.8 Related Work

As we develop the notion $\overset{\pi}{\sim}_{\mathcal{T}}$ as a generalization of $\sim_{\mathcal{T}}$, this section relates our development to others in two areas, *viz.* ordinary concept similarity measures (which do not take into account the agent's preferences) and preference-based concept similarity measures.

5.8.1 Ordinary Concept Similarity Measure

In the standard perception, concept similarity measure refers to the study of similar concepts inherited by nature, *i.e.* the ones similar regardless of the agent's preferences. Our concrete developments, which employ structural subsumption, can be considered as the semantic similarity approach (*cf.* Section 4.1). Hence, we merely compare our approaches to other approaches of semantic similarity as follows.

A simple method was developed in [143] for the DL $\mathcal{L}_0$ (*i.e.* no use of roles) and was known as *Jaccard Index*. Its extension to the DL $\mathcal{ELH}$ was proposed in [88]. This work also introduced important properties of concept similarity measure and suggested a general framework called *simi* which satisfied most of the properties. In *simi*, functions and operators, such as t-conorm and the fuzzy connector, were to be parameterized and thus left to be specified. The framework also did not contain implementation details. This may cause implementation difficulties since merely promising properties were given and no guideline of how concrete operators are chosen is provided.

In [146], the measure $simdl_A$ was proposed for the DL $\mathcal{SHI}$. The measure was not completely defined in mathematical terms and some text descriptions were not precise. Roughly, the measure had three stages. First, two concept descriptions in question were converted into the negation normal form (NNF). A modified version of the tableau was used to generate a completion tree for each NNF concept. In this modified version, the $\sqcup$-rule was modified and another $\forall$-rule was added. Second, a set of *proxy models* was generated from the completion tree. A proxy model was a tree where each was labeled by a role name and each node was labeled by a concept name. Third, both sets of

proxy models were used to compute the degree of similarity. This was done by measuring similarity among all pairs of proxy models (using tree similarity). The final result was evaluated from either the maximum, the minimum, or the average. However, the paper did not explain the selection rule when more than one tableau rules could be applied.

The notion of skeptical subsumption degree and credulous subsumption degree are originally introduced in [89]. Also, the skeptical subsumption degree is generalized and is extended toward the development of $\overset{\pi}{\sim}s$ for the DL $\mathcal{FL}_0$ in this chapter. Theorem 5.1 suggests that $\overset{\pi}{\sim}s$ can be used to measure similarity of concepts inherently by nature through the setting π_0, *i.e.* $\overset{\pi_0}{\sim}s$.

The notion of homomorphism degree is originally introduced in [145] and is thereof extended toward the development of sim^π for the DL $\mathcal{ELH}$ in this chapter. Theorem 5.2 suggests that sim^π can be used to measure similarity of concepts inherently by nature through the setting π_0, *i.e.* sim^{π_0}.

The measures $\overset{\pi}{\sim}s$ and sim^π per se are categorized as a structure-based measure; however, reconciling each one with the strategies (*cf.* Section 5.5) to help tuning $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$ may be considered as the hybrid approach as it uses the canonical interpretation $\mathcal{I}_\mathcal{A}$ for measuring similarity of primitive concept names and similarity of primitive role names.

As inspired by the tree homomorphism, sim^π differs [88] from the use of μ^π to determine how important the primitive concepts are to be considered and the use of γ^π to determine a degree of role commonality between matching edges of the description trees.

5.8.2 Preference-based Concept Similarity Measure

Most concept similarity measures are objective-based. However, there exists work [88,151] which provides methodologies for tuning. We discuss their differences to our approaches in the following.

In an extended work of sim [151], a range of number for discount factor (ν) and the neglect of special concept names were used in the similarity application of SNOMED CT. For instance, when $\mathsf{roleGroup}$ was found, the value of ν was set to 0. These ad hoc approaches can be viewed as specific applications of $\mathfrak{d}$ and $\mathfrak{i}^{\mathfrak{c}}$, respectively, of preference profile. Unfortunately, no other aspects of π appear in its use.

In *simi* [88], the function *pm* was used to define the similarity degree of primitive concept pairs and role pairs. Using *pm* with primitive concept pairs invokes the equivalent intuition as $\mathfrak{s}^{\mathfrak{c}}$; however, this does not mean so in the aspect $\mathfrak{s}^{\mathfrak{r}}$. Allowing to define the similarity of defined role names, as in [88], may be not appropriate since defined role names are contributed by primitive role names. For example, let $r_1 \sqsubseteq s_1$ and $r_2 \sqsubseteq s_2$ are defined in $\mathcal{T}$. It is clear that $r_1, r_2 \in \mathsf{RN}^{\mathsf{def}}$. By defining $pm(r_1, r_2)$, the defined similarity should be also propagated to the similarity of s_1 and s_2. However, this point was not discussed in [88]. In respect of this, $\mathsf{RN}^{\mathsf{pri}}$ is merely used in $\mathfrak{s}^{\mathfrak{r}}$ and γ^π is defined for the similarity of defined role names. The authors of [88] also defined the function $g : N_A \to \mathbb{R}_{>0}$ representing the weight for concept names and existential restriction atoms (based on their definition). Ones may feel the resemblance of g and $\mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}$; however, they are also different in three perspectives. Firstly, the mapping of g is reached to the infinity whereas $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{i}^{\mathfrak{r}}$ are bounded. This characteristic of g is impractical to use as it may

Table 5.4: Concept similarity measures which embed preference elements

Similarity Measure	DL	$\mathfrak{i}^{\mathfrak{c}}$	$\mathfrak{i}^{\mathfrak{r}}$	$\mathfrak{s}^{\mathfrak{c}}$	$\mathfrak{s}^{\mathfrak{r}}$	$\mathfrak{d}$
sim^{π}	$\mathcal{ELH}$	✔	✔	✔	✔	✔
$\overset{\pi}{\sim}_s$	$\mathcal{FL}_0$	✔		✔		
the extended work of sim [151]	$\mathcal{ELH}$	✔				✔
simi [88]	$\mathcal{EL}$	✓		✔	✓	

lead to the unbalance of weight assignments. For instance, one may define $g(A_1) = 1$ but $g(A_2) = 10^{12}$ where $A_1, A_2 \in \mathsf{CN}^{\mathsf{pri}}$. To avoid this situation, the authors should provide a guideline for weight assignments. Secondly, the mapping of g is lower bounded by one. This clearly makes an impossibility to define the intuition of having no importance. Thus, the situation given in Subsubsection 5.7.2 is not expressible. Lastly, the domain of g is the set of atoms whereas $\mathfrak{i}^{\mathfrak{c}}$ (and $\mathfrak{i}^{\mathfrak{r}}$) is the set of primitive concept names (and the set of role names, respectively). Using the set of atoms as the domain is also impractical since there can be infinitely many existential restriction atoms and the interpretation of functions is slightly dubious. For instance, given $g(\exists r.C) = 2$ and $g(\exists r.D) = 3$, do both r intentionally contribute the equal importance? Thus, this definition is inappropriate to represent the agent's perception. Moreover, the aspect $\mathfrak{d}$ disappeared from [88]. Lacking of fully $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{d}$ makes the framework inappropriate to use for SNOMED CT applications. These distinctions of *simi* and ours are radically caused by their different motivations. Table 5.4 summarizes this discussion, where ✔ denotes totally identical to the specified function whereas ✓ denotes partially identical to the specified function.

Not only distinct on the mathematical representation of *simi* and our measures, the desired properties presented in each work are also different. While the properties introduced in [88] were motivated for (ordinary) concept similarity measure, our properties are developed under the consideration of the agent's preferences ($\overset{\pi}{\sim}_{\mathcal{T}}$). Hence, some properties introduced for concept similarity measure are revised in subjective manners and the new property is introduced.

5.9 Potential Applications

Our proposed approaches have great potential use in knowledge engineering, such as the development of recommendation systems based on the agent's preferences, the development of domain-specific knowledge bases, and the ontology engineering. We exemplify in Subsection 5.7.2 a development of recommendation systems based on the agent's preferences via the sections about tuning $\mathfrak{s}^{\mathfrak{r}}$, $\mathfrak{s}^{\mathfrak{c}}$, and $\mathfrak{s}^{\mathfrak{r}}$, and a development of the domain-specific knowledge base in case of SNOMED CT (*cf.* page 109).

Furthermore, our proposed approaches may be also used with heterogeneous ontologies by identifying duplicated primitive concepts and primitive roles among ontologies via $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$, respectively. It has been revealed in [157] that concepts used by different

terminologies may unintentionally mean the same. In [157], 30% of Human Phenotype Ontology (HPO) concepts are semantically duplicated with SNOMED CT concepts. For example, the HPO concept "Multicystic Dysplastic Kidney" (HP:0000003) is identical to the SNOMED CT concept "Multicystic Renal Dysplasia" (SCTID:204962002). In such a case, a mapping between these two ontologies should be formed. After the mapping, similar concepts from multiple ontologies can be found out.

Last but not least, the approaches can contribute to the development of several kinds of similarity-based reasoning such as approximate reasoning (*cf.* [158, 159]) and analogical reasoning (*cf.* [95, 96]). We also discuss several approaches to extend the proposed measures for analogical reasoning in the next chapter.

5.10 Summary

- As aforementioned, when two concepts are not equivalent, subjective factors may play an important role in identifying the degree of concept similarity. To achieve this, preferential elements relevant to concept similarity in DLs are defined and are used in defining concept similarity measure under preference profile. Basically, a concept similarity measure under preference profile is defined as a function which maps a concept pair under preference profile π to a unit interval;

- Concrete concept similarity measures under preference profile were developed by generalizing the approaches for concrete concept similarity measures w.r.t. each preferential element of preference profile. Then, they were shown to be procedures which ensure termination and can be used as indicators for similarity under the agent's preferences between concepts in an ontology. Their underlying properties were also clearly investigated;

- As realistic ontologies are generally complex – consisting in plenty of concept names and role names, having some strategies to tune a measure helps ontology engineers, researchers, and application users to use a measure for similarity-based under the agent's profile applications. That is, instead of specifying each aspect of preference profile individually and manually, an agent may automatically assign an importance of each primitive concept name through a defined concept name with $\mathfrak{I}^{\mathfrak{c}}$. Similarly, an agent may automatically assign an importance of each role name through a more general role name with $\mathfrak{I}^{\mathfrak{r}}$. However, these notions have some restrictions and we discuss each of them in Subsection 5.5.1 and Subsection 5.5.2, respectively. If an ABox is presented, the canonical interpretation can be induced and be used to compute $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$ for each primitive concept pair and primitive role pair (*cf.* Subsection 5.5.3 and Subsection 5.5.4). These strategies are recommended to use for the initial preference tuning and may be refined wherever the agent wishes; and

- The empirical evaluation was carried out w.r.t. realistic ontologies and their potential use cases and guidance were suggested and discussed.

Chapter 6

Application in Analogical Reasoning

6.1 Introduction

Analogical reasoning is a complex process based on a comparison between two pairs of concepts or states of affairs (*aka.* the *source* and the *target*) sharing some common features [160]. This comparison is the ground of a specific type of inference called *argument from analogy*, in which the conclusion of an argument attributes to a specific feature characterized from the source to the target (*cf.* the proposed models in [161–165]). Despite the diversity, those models can be represented by the generic structure called an *argumentation scheme for argument from analogy* introduced by Walton [164] (also see Section C.1) as follows:

Similarity Premise	Generally, case C_1 is similar to case C_2
Base Premise	A is true (false) in case C_1
Conclusion	A is true (false) in case C_2

This generic structure can be explained as follows. The similarity is regarded to hold between two cases. These cases could be two different concepts or states of affairs. Consequently, a property (*e.g.* a feature A) attributes to both cases. Intuitively, this kind of structure can be represented as a logic program where A and C_i are appeared at the head and the body of an inference rule, respectively. Several attempts similar to this approach were developed in [95, 96, 158, 166]

A fundamental problem for this kind of reasoning is how to evaluate an analogical argument, *i.e.* its acceptibility. Basically, this problem amounts to investigations of the structure of analogical arguments and its defeasibility characteristics. At the abstract level, critical questions (CQ) [164] associated to the argument scheme outlines several conditions of defeasibility:

CQ1	Is A true (false) in C_1?
CQ2	Are C_1 and C_2 similar in the respects cited?
CQ3	Are there important differences (dissimilarities) between C_1 and C_2
CQ4	Is there some other case C_3 that is also similar to C_1 except that A is false (true) in C_3?

These critical questions can be used to understand which analogical arguments should not be accepted. However, they do not address the following three basic problems: (1) how similarity/dissimilarity should be determined (which amounts to understand the notion of similarity); (2) how an analogical arguments is constructed (which amounts to understand the structure of an analogical argument); and (3) how a conclusion drawn from the similarity premise and the base premise is warranted (which amounts to understand the evaluation of analogical arguments). The argumentation scheme and its critical questions do not involve these aspects concretely.

It is obvious that the approaches we have developed so far for concept similarity measure under the agent's preferences can be used to address the first problem if cases (in the argumentation scheme) are expressible in the form of DL concepts (*cf.* Chapter 3). Therefore, we have remained to investigate the second and the third problems, which are indeed our main focus of this chapter.

It is worth mentioning that analogical reasoning is used quite often by human beings in real-life situations, especially when humans have to encounter an unseen situation. Its applications may be also seen in some practical domains such as law and medical diagnosis. The following example illustrates an application in legal reasoning where attorney Gerry Spence has involved in the case of Silkwood v. Kerr-McGee Corporation (1984) [167].

Example 6.1. (The Silkwood case) Karen Silkwood was a technician who had the job of grinding and polishing plutonium pins used to make fuel rods for nuclear reactors. Tests in 1974 showed that she had been exposed to dangerously high levels of plutonium radiation. After she died in an automobile accident, her father brought an action against Kerr-McGee in which the corporation was held to be at fault for her death on the basis of strict liability. In strict liability, a person can be held accountable for the harmful consequences of some dangerous activity he was engaged in, without having to prove that he intended the outcome. □

Spence's closing argument uses the analogy of the escaping lion, which had great rhetorical effect on the jury. According to his speech ([167, p.129]), he emphasized the statement "If the lion got away, Kerr-McGee has to pay".

> Some guy brought an old lion on his ground, and he put it in a cage – and lions are dangerous – and through no negligence of his own through no fault of his own, the lion got away. Nobody knew how – like in this case, "nobody knew how". And, the lion went out and he ate up some people – and they sued the man. And they said, you know: "Pay. It was your lion, and he got away". And the man says: "But I did everything in my power – I had a good cage – had a lock on the door – I did everything that I could – I had security – I had trained people watching the lion – and it is not my fault that he got away". Why should we punish him? They said: "We have to punish him – we have to punish you – you have to pay". You have to pay because it was your lion – unless the person who was hurt let the lion out himself.

Spence was using an analogy to "compare" what happened in the Silkwood case and what happened in the lion example. In fact, the lion example is the source case in

argument from analogy to illustrate the jury how cases of this kind had originated in English common law. The plutonium case is the target case, which is shown to be "similar to" the source case. Therefore, all Spence had to prove is that Karen Silkwood was harmed by the plutonium. In his summary argument, Spence reminded the jury that this claim has been proved during the trial by the evidence brought forward. This argument was a rhetorically powerful one and it could persuade the jury to find Kerr-McGee liable for over ten million dollars [168].

This chapter explains two different approaches (*cf.* Section 6.2 and Section 6.3) which can be used to reconstruct arguments from analogy. Both are similar to each other in a sense that models the Walton's scheme and use the notion $\overset{\pi}{\sim}$ for identifying the similarity between the source case and the target case.[1] Their usages of the scheme are based on the assumption that the proponent and the opponent have the same ground-truth preferences. However, they are different to each other in a way the arguments are represented.

6.2 First Approach: Integrating DLs with Rules

Analogical arguments can be reconstructed and evaluated by a semantics of logic programming. Answer set semantics for logic programs [169] is one of the most widely adopted semantics for logic programs. It provides the theoretical foundation for answer set programming (ASP) (*cf.* Subsection 6.2.1 for the fundamentals) [170, 171] and was proven in [172] to coincide with the stable semantics of argumentation framework (*cf.* Section C.2).

An approach developed in this section exploits benefits of two different formalism *i.e.* DLs and rules. In particular, DLs are used for extracting information about conceptual schemata whereas rules are applied to data-centric problems. Both formalisms exhibit certain shortcomings that can be compensated for by advantages of the other. Using DLs, situations are defined in form of concept definitions, and similarity premises can be identified by the use of a concept similarity measure under preference profile *e.g.* sim^{π} and $\overset{\pi}{\sim} s$ for $\mathcal{ELH}$ and $\mathcal{FL}_0$ concepts, respectively.

Concretely, we introduce a knowledge base[2] $\mathcal{K}$ (*cf.* Subsection 6.2.2) which makes it possible to find analogical consequences. We note that, whenever we refer to a knowledge base $\mathcal{K}$, we mean our setting defined in this section. Informally, $\mathcal{K}$ has three components, *viz.* a logic program $\mathcal{LP}$, a DL ontology $\mathcal{O}$, an instance of concept similarity measure under preference profile $\overset{\pi}{\sim}_{\mathcal{T}}$. Subsection 6.2.2 introduces a "declarative language" for the specification of $\mathcal{LP}$ and gives a formal definition of $\mathcal{K}$. Subsection 6.2.3 addresses the problem of computing conclusions from analogy.

6.2.1 Background: Answer Set Programming

Answer set semantics for logic programs [169] is one of the most widely adopted semantics for logic programs, *i.e.* logic programs that allow negation-as-failure in the body of the

[1] Consequently, their applications are limited to analogical conclusions inferred by the similarity between two DL concepts.

[2] This extended definition of knowledge base is mainly used by this section for analogical reasoning.

rules. This semantics provides the theoretical foundation for ASP which has been proven to be useful in several applications such as diagnosis, bioinformatics, planning, *etc.* (see [173–178] for some applications' detail)

Answer set semantics is inherently nonmonotonic, *i.e.* the set of logical consequences does not necessarily grow monotonically if the size of knowledge base increases. This happens due to the allowed negation-as-failure. Answer set semantics is defined for *extended logic programs*, in which not only negation-as-failure can be used in program rules, but also strong negation (or "classical negation") and disjunction are also allowed.

In ASP, the computation of a logic program involves two phases. First, a *grounder* constructs a ground instances of program clauses, *i.e.* replacing variables by grounded (variable-free) terms (possibly with some further transformation and simplification). Second, the resulting grounded program is passed to a solver for computing answer sets. One of the main reasons for its gaining in popularity is the availability of sophisticated grounders and solvers, *e.g.* SMODELS[1], DLV[2], and CLASP[3]. Figure 6.1 illustrates the design of an ASP system in general.

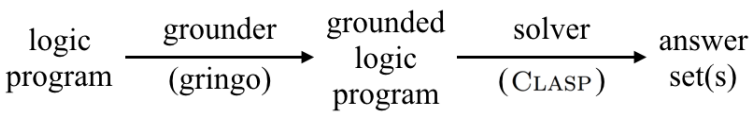


Figure 6.1: Architecture of an ASP system

According to the figure, *gringo* is a grounder and CLASP is a solver. There is also a utility software *e.g. clingo* which automatically pipes the output from gringo to CLASP. Another widely used grounder is *lparse* which comes with both SMODELS and DLV systems. Let us note that CLASP also supports the result of lparse.

6.2.2 The Knowledge Base Setting

The object language of $\mathcal{LP}$ conforms to the familiar logic-programming-like style. That is, let $\Sigma = \langle \mathcal{C}, \mathcal{V}, \mathcal{P} \rangle$ be a signature with a finite set of constants $\mathcal{C}$, an infinite set of variables $\mathcal{V}$, and a finite set of predicate symbols $\mathcal{P}$. Let L_Σ be the first-order language constructed over Σ. There are two types of literals. A *strong* literal is an atomic first-order formula A (of L_Σ) or such a formula preceded by classical negation, *i.e.* $\neg A$. A *weak* literal is a literal of the form *not* A, where A is a strong literal and *not* denotes *negation-as-failure* (or default negation). Informally, *not* A reads as "there is no evidence that A is the case" whereas $\neg A$ reads as "A is definitely not the case". In what follows, we use the standard typographic conventions of logic programming.

Definition 6.1 (Program Clause)**.** A *definite program clause* is a clause of the form $A_0 \leftarrow L_1, \ldots, L_n$ where A_0 is a strong literal and L_i $(1 \leq i \leq n)$ is a literal. If $n = 0$, it is referred to as a *fact*. Otherwise, it is referred to as a *rule*.

[1] http://www.tcs.hut.fi/Software/smodels/
[2] http://www.dlvsystem.com/dlv/
[3] https://potassco.org/clasp/

Definition 6.2 (Logic Program)**.** A *definite logic program* $\mathcal{LP}$ is a finite set of definite program clauses.

Example 6.2. (Continuation of Example 6.1) We translate the Silkwood case into our logic program $\mathcal{LP}$. For the sake of clarity, we distinguish in $\mathcal{LP}$ the legal rules $\mathcal{LP}_L$, the hypothetical case $\mathcal{LP}_H$, and the current case $\mathcal{LP}_C$, *i.e.* $\mathcal{LP} = \mathcal{LP}_L \cup \mathcal{LP}_H \cup \mathcal{LP}_C$. The literal $exception(X)$ means "X is an exception to inactivate the goal". To avoid confusion, we separate each program clause by a semicolon.
$\mathcal{LP}_L = \{\ defendant(X) \leftarrow owner(X, Y), danger(Y), killer(Y, Z);$
$liable(X) \leftarrow defendant(X), not\ exception(X);\ \}$
$\mathcal{LP}_H = \{\ danger(X) \leftarrow lion(X);\ lion(l_1);\ owner(guy, l_1);\ person(man);$
$killer(l_1, man);\ \}$
$\mathcal{LP}_C = \{\ plutonium\_plant(p_1);\ owner(kerr\_mcgee, p_1);\ person(silkwood);$
$killer(p_1, silkwood).\ \}$ □

We note that there could be many ways to transform a problem domain into $\mathcal{LP}$. Addressing this issue is also important but it is outside the scope of this work. Our intention is to determine the similarity of two predicate symbols from a DL ontology by using the notion $\overset{\pi}{\sim}_{\mathcal{T}}$. The following gives a formal definition of our knowledge base setting.

Definition 6.3 (Knowledge Base)**.** A *knowledge base* $\mathcal{K}$ is a triple $\langle \mathcal{LP}, \mathcal{O}, \overset{\pi}{\sim}_{\mathcal{T}} \rangle$, where $\mathcal{LP}$ is a logic program, $\mathcal{O}$ is a DL ontology, $\overset{\pi}{\sim}_{\mathcal{T}}$ represents an instance of concept similarity measure under preference profile in DLs.

One may observe that not every knowledge base is meaningful to give conclusions from analogy, *e.g.* when the set $\mathsf{Pred}(\mathcal{LP})$ of predicate symbols appearing in $\mathcal{LP}$ and the set $\mathsf{CN}(\mathcal{O})$ of concept names appearing in $\mathcal{O}$ do not intersect.

Example 6.3. (Continuation of Example 6.2) We assume that our working ontology $\mathcal{O}$ has been modeled as follows:

$$\begin{aligned} \mathsf{lion} &\sqsubseteq \mathsf{carnivore} \sqcap \mathsf{wild} \\ \mathsf{plutonium\_plant} &\sqsubseteq \mathsf{power\_plant} \sqcap \mathsf{radiation} \\ \mathsf{carnivore} &\sqsubseteq \mathsf{harm} \\ \mathsf{radiation} &\sqsubseteq \mathsf{harm} \end{aligned}$$

A knowledge base $\mathcal{K}$ may be represented as a triple $\langle \mathcal{LP}, \mathcal{O}, \mathsf{sim}^{\pi} \rangle$. We note that sim^{π} is shown to be an instance of $\overset{\pi}{\sim}_{\mathcal{T}}$ (*cf.* Subsection 5.4.1). □

6.2.3 Computing Analogical Conclusions

One may observe from Example 6.3 that $\mathcal{K}$ successfully models the decision of the hypothetical case by having logical conclusions "$defendant(guy)$" and "$liable(guy)$". However, $\mathcal{K}$ does not model the decision of the current case stated in Spence's closing argument. $\mathcal{K}$ can be twisted a bit with additional rules (*e.g.* rules representing extra evidences) so that

"*liable*(*kerr_mcgee*)" is logically concluded. Nevertheless, this approach does not correctly reconstruct Spence's analogical argument, which is not based on purely logical models.

In this subsection, we assume that there is no extra evidence about the case. To reconstruct Spence's argument, we extend $\mathcal{K}$ with analogical knowledge extracted from the ontological component. This extension is technically defined as the operator $\cdot^+$. Intuitively, $\cdot^+$ provides transforming steps to extend $\mathcal{LP}$ with $\mathcal{O}$ via $\overset{\pi}{\sim}_{\mathcal{T}}$. The result of executing $\cdot^+$ on $\mathcal{K}$, *i.e.* $\mathcal{K}^+$, conforms to the input language of grounder **gringo** [179] and can be used with an answer set engine.

Transforming Logic Program $\mathcal{LP}$

We achieve this by transforming each clause of $\mathcal{LP}$ as a set of answer set program clauses in $\mathcal{K}^+$. Our transformation uses the predicate symbol *atom* as a basic predicate symbol. For each $\varphi_0 \leftarrow \varphi_1, \ldots, \varphi_n$ $(0 \leq i \leq n) \in \mathcal{LP}$ and let φ'_0 *:-* $\varphi'_1, \ldots, \varphi'_n$ $(0 \leq i \leq n)$ be a corresponding clause in $\mathcal{K}^+$, then the transformation is performed as follows:

- If $\varphi_i = A_i(X_0, \ldots, X_m)$, then $\varphi'_i = atom(X_0, \ldots, X_m, A_i)$;
- If $\varphi_i = \neg A_i(X_0, \ldots, X_m)$, then $\varphi'_i =$ *-atom*$(X_0, \ldots, X_m, A_i)$;
- If $\varphi_i = not\ A_i(X_0, \ldots, X_m)$, then $\varphi'_i = not\ atom(X_0, \ldots, X_m, A_i)$.

Extending with Similarity from Ontology $\mathcal{O}$

Let Sim be a set of pairs of predicates whose similarity is maximal among each matching predicate, *i.e.* $\mathsf{Sim} = \{(\varphi, \psi) \mid \forall \varphi \in \Lambda : (\varphi \overset{\pi}{\sim}_{\mathcal{T}} \psi = \max_{\psi \in \Lambda}\{\varphi \overset{\pi}{\sim}_{\mathcal{T}} \psi\})\}$ where $\Lambda = \mathsf{Pred}(\mathcal{LP}) \cap \mathsf{CN}(\mathcal{O})$ and $\varphi \not\equiv \psi$. Let $arity(\varphi, \psi)$ gives the number of arguments that both φ and ψ take. We note that *dsim* is additionally reserved. For each $(\varphi, \psi) \in \mathsf{Sim}$:

- $\mathcal{K}^+ := \mathcal{K}^+ \cup \{dsim(\varphi, \psi); atom(A_1, \ldots, A_m, \varphi)$ *:-* $atom(A_1, \ldots, A_m, \psi)$, $dsim(\varphi, \psi).\}$ where $m = arity(\varphi, \psi)$.

It is worth to observe that φ is fixed for each $\varphi \in \Lambda$ to determine the maximal pair. Thus, either a symmetric measure or an asymmetric measure can be employed by the same rule of extending with similarity. We refer the readers to Subsection 5.4.1 for useful discussion about inherited properties of concept similarity measures in DLs, *e.g.* sim^π and $\overset{\pi}{\sim}s$ are symmetric.

Example 6.4. (Continuation of Example 6.3) Let $\mathcal{K}^+$ be the result of transforming the logic program (*i.e.* the first step of $\cdot^+$). Then, we enrich $\mathcal{K}^+$ by the following additional set of clauses. That is, $\mathcal{K}^+ = \mathcal{K}^+ \cup \{$
dsim(*lion*, *plutonium_plant*);
atom$(X,$ *lion*$)$ *:-* *atom*$(X,$ *plutonium_plant*$)$, *dsim*(*lion*, *plutonium_plant*);
dsim(*plutonium_plant*, *lion*);
atom$(X,$ *plutonium_plant*$)$ *:-* *atom*$(X,$ *lion*$)$, *dsim*(*plutonium_plant*, *lion*). $\}$ □

Using Critical Questions as Constraints

It is not difficult to see that the first critical question and the second one are automatically configured by $\overset{\pi}{\sim}_{\mathcal{T}}$ and the logic program, respectively. Also, counter-analogies can be discovered through answer set semantics. These make $\mathcal{K}^+$ to incorporate critical questions.

Finding Logical Entailment

We successfully explain each execution step of $\cdot^+$. As we can use an answer set engine (*e.g.* **clasp** [179]) to determine answer sets as analogical conclusions from $\mathcal{K}^+$, we include the original definition of entailment w.r.t answer set semantics here for self-containment. For a logic program Π and a ground atom a, Π entails a w.r.t. answer set semantics (in symbols, $\Pi \models a$) if $a \in S$ for every answer set S of Π. Similarly, for a logic program Π and a ground atom a, Π entails $\neg a$ w.r.t. answer set semantics (in symbols, $\Pi \models \neg a$) if $\neg a \in S$ for every answer set S of Π. If neither $\Pi \models a$ nor $\Pi \models \neg a$, then we say that a is *unknown* w.r.t. Π. Hence, we say an atom $A(X_0, \ldots, X_m)$ is an *analogical conclusion* from $\mathcal{K}$ if $\mathcal{K}^+ \models atom(X_0, \ldots, X_m, A)$.

6.2.4 Relationship to Argumentation Framework

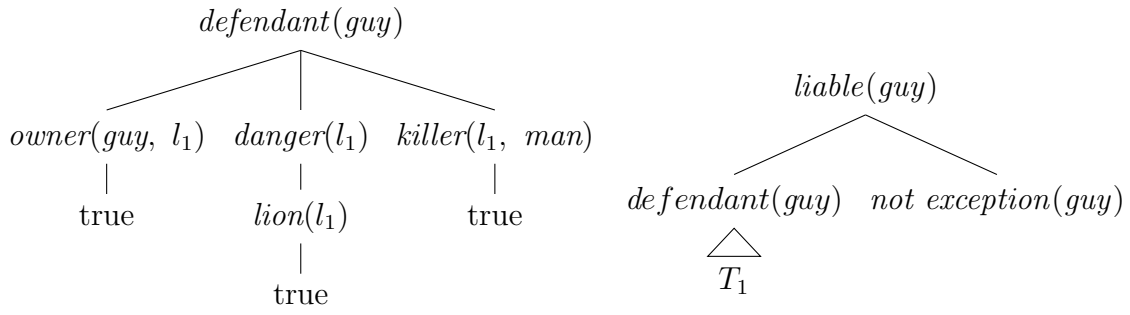


Figure 6.2: Possible proof trees for *defendant*(*guy*) and *liable*(*guy*)

Now, our intention is to analyze the underlying mechanisms of $\mathcal{K}$ and $\mathcal{K}^+$ under the lens of argumentation framework. It is not difficult to see that logical conclusions obtained from $\mathcal{K}$ and $\mathcal{K}^+$ can be seen as proof trees constructed from $\mathcal{K}$ and $\mathcal{K}^+$, respectively. Each proof tree represents an argument supporting the conclusion at its root. For instance, possible proof trees for "*defendant*(*guy*)" and "*liable*(*guy*)" are depicted on the left-hand side and the right-hand side of Figure 6.2, where T_1 denotes a proof tree for "*defendant*(*guy*)". This explains that the guy is a defendant for the lion case and the guy is liable for the case.

As discussed in Subsection 6.2.3, the knowledge base $\mathcal{K}$ cannot successfully model the legal rules with the current case. To achieve this, $\mathcal{K}$ must be extended to $\mathcal{K}^+$ so that $\mathcal{K}^+ \models atom(ker\_mcgee, defendant)$. Figure 6.3 depicts a proof tree for "$atom(ker\_mcgee,$ *defendant*)" from $\mathcal{K}^+$, where T_1' denotes a proof tree for "$atom(p_1,$ *danger*)". Such trees represent *arguments from analogy* supporting the conclusion at their roots.

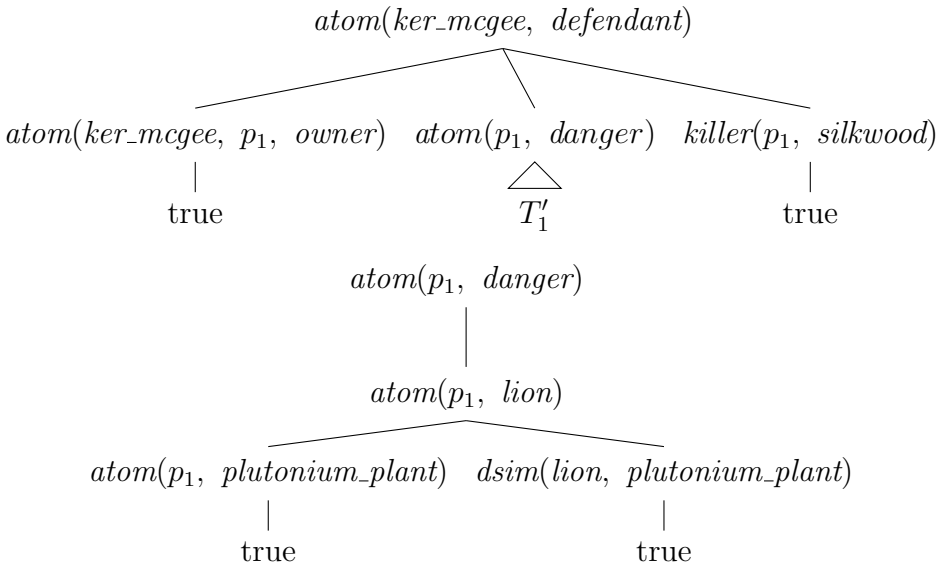


Figure 6.3: Proof trees for $atom(ker\_mcgee,\ defendant)$ and $atom(p_1,\ danger)$

It is not difficult to see that our proposed operator formalizes the form of Walton's scheme. That is, when desired conclusions cannot be logically inferred from a knowledge base, that knowledge base can be extended with similarity information. Conclusions obtained from the extended one are called *analogical conclusions*.

6.3 Second Approach: Argument-based Logic Programming

The first approach may seem rather easy to be implemented due to the availability of answer set engines in the current state-of-the-art technologies and our developed application programming interfaces (APIs) for concept similarity measures under preference profile (*cf.* Appendix B). However, it does not preserve similarity degrees on the computation of analogical conclusions. This ignorance may cause the system to behave w.r.t. the different degrees of similarity premises equally. Regarding this, ones may consider the first approach as a rough-and-ready implementation method.

On the other hand, the second approach considers to model the notion of an argument's structure and proposes a computational method to decide an "acceptable" analogical argument. Basically, a formalism proposed in the section has adopted from an approach of combining logic programming with argumentation called *Defeasible Logic Programming* (DeLP) in [180] (see also Section C.2). We note that the representational language of DeLP is defined as an extension of a logic programming language that considers two types of rules *viz.* strict and defeasible and allows for both strong and default negation. Providing a capability of constructing analogical arguments, this section extends DeLP with similarity rules and a new dialectical analysis is developed to evaluate the constructed

arguments.

6.3.1 The Language

Our object language conforms to the familiar logic-programming-like style. That is, let $\Sigma = \langle \mathcal{C}, \mathcal{V}, \mathcal{P} \rangle$ be a signature with a finite set of constants $\mathcal{C}$, an infinite set of variables $\mathcal{V}$, and a finite set of predicate symbols $\mathcal{P}$. Let L_Σ be the first-order language constructed over Σ. There are two types of literals. A *strong* literal is an atomic first-order formula A (of L_Σ) or such a formula preceded by the classical negation, *i.e.* $\neg A$. A *weak* literal is a literal of the form *not* A, where A is a strong literal and *not* denotes *negation-as-failure* (or default negation). Informally, *not* A reads as "there is no evidence that A is the case" whereas $\neg A$ reads as "A is definitely not the case". In what follows, we use the standard typographic conventions of logic programming.

Definition 6.4 (Strict Rule)**.** A strict rule is an expression of the form $L_0 \leftarrow L_1, \ldots, L_n$ where $n \geq 0$ and L_i $(0 \leq i \leq n)$ is a *strong* literal. If $n = 0$, it is referred to as a *fact*.

Definition 6.5 (Defeasible Rule)**.** A defeasible rule is an expression of the form $L_0 \Leftarrow L_1, \ldots, L_n$ where $n \geq 0$, L_0 is a *strong* literal, and L_i $(1 \leq i \leq n)$ is a literal. If $n = 0$, it is referred to as a *presumption*.

Defeasible rules are used to represent tentative information which will be used if no one could disprove it whereas strict rules are used to represent strict information. For example, $\mathit{fly}(X) \Leftarrow \mathit{bird}(X)$ expresses "usually, a bird can fly" whereas $\mathit{bird}(X) \leftarrow \mathit{penguin}(X)$ expresses "all penguins are birds".

It is worth noting that in general account of defeasible logic, particularly Nute's d-Prolog [181], it contains a facility to define *defeater rules*, *e.g.* "sick birds do not fly". The purpose of defeater rules is to express exceptions to defeasible rules. However, [182] shows that defeater rules can be simulated by means of strict and defeasible rules (in Nute's sense). As we will also show soon, our system does not need to supply with defeater rules. The system will find counter-arguments, including counter-analogies, among arguments it is able to build.

Definition 6.6 (Similarity Rule)**.** A similarity rule is an expression of the form $L_0 \overset{x}{\leftleftarrows} L_1$[1] where L_0, L_1 are first-order predicates and $0 < x \leq 1$ for any real number x, such that $L_0 \overset{x}{\leftleftarrows} L_1$ means L_1 is similar to L_0 at x degree (but, not vice versa) and $L_0 \overset{1}{\leftleftarrows} L_1$ indicates L_1 is *totally similar* to L_0, *i.e.* $L_1 \equiv L_0$.

Similarity rules are used to form similarity premises, *e.g.* $\mathit{plutonium\_plant} \overset{1}{\leftleftarrows} \mathit{lion}$ expresses "lions are totally similar to plutonium plants". One methodology for building similarity rules is to query similarity information of corresponding concepts defined in DL ontologies (Definition 5.9). To the best of our knowledge, $\overset{x}{\leftleftarrows}$ is first introduced for analogical reasoning in this work.

[1] When $x = 1$, we may remove it.

Definition 6.7 (Logic Program)**.** A logic program $\mathcal{P}$ is a triple $\langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ where SR is a finite set of of strict rules, DR is a finite set of defeasible rules, and SIM is a finite set of similarity rules.

We assume that every rule of $\mathcal{P}$ is grounded. Nevertheless, our examples may use the usual convention, *i.e. schematic rules with variables*, in logic programs.

Definition 6.8 (Derivation)**.** Let $\mathcal{P}$ be a logic program and L be a ground literal. A derivation for L from $\mathcal{P}$ with an *analogy degree* w, in symbols $\mathcal{P} \overset{w}{\vdash} L$[1], is a finite sequence $L_1, \ldots, L_n$ of ground literals such that $L_n = L$ and each literal L_i $(1 \leq i \leq n)$ satisfies the following conditions:

1. L_i is a fact or a presumption;

2. There exists a rule R_i of $\mathcal{P}$ (strict or defeasible) with head L_i and body $L_1, \ldots, L_j$ such that every literal of the body, except ones preceded by negation-as-failure, is an element of the sequence appearing before L_i $(j < i)$;

3. There exist similarity rules $A'_1 \overset{x_1}{\leftleftarrows} L'_1, \ldots, A'_k \overset{x_k}{\leftleftarrows} L'_k$ of $\mathcal{P}$ and another rule R_i of $\mathcal{P}$ (strict or defeasible) with head L_i and body $L_1, \ldots, L_k, \ldots, L_j$ such that the predicate of $L_1, \ldots, L_k$ are $L'_1, \ldots, L'_k$, respectively. The substitution on each predicate of $L_1, \ldots, L_k$ with $A'_1, \ldots, A'_k$, respectively, and other non-substituted literals, except ones preceded by negation-as-failure, is an element of the sequence appearing before L_i $(k \leq j < i)$.

4. $w = \bigotimes_{A'_i \overset{x_i}{\leftleftarrows} L'_i} \{x_i\}$, where $\bigotimes$ is a triangular norm (t-norm)[2] and $A'_i \overset{x_i}{\leftleftarrows} L'_i$ is used to derive L. Otherwise, we set $w = 1$ as the default value.

Basically, $\mathcal{P} \overset{1}{\vdash} L$ means that L may be derived without any use of analogies. Condition 2 to 3 assume two implicit inference rules, *viz.* a rule of defeasible modus ponens (MP) and a rule of defeasible analogical rule (AR) as follows:

$$\frac{L_1, \ldots, L_j \qquad L_i \Leftarrow L_1, \ldots, L_j, \mathit{not}\ L_m, \ldots, \mathit{not}\ L_n}{L_i}\ \text{MP}$$

$$\frac{A'_1 \overset{x_1}{\leftleftarrows} L'_1\ \ldots\ A'_k \overset{x_k}{\leftleftarrows} L'_k \qquad L_i \Leftarrow L_1, \ldots, L_k, \ldots, L_j, \mathit{not}\ L_m, \ldots, \mathit{not}\ L_n}{L_i \Leftarrow A_1, \ldots, A_k, \ldots, L_j, \mathit{not}\ L_m, \ldots, \mathit{not}\ L_n}\ \text{AR}$$

Since this work employs the notion of t-norm, we include its definition here for self-containment. A function $\otimes : [0,1]^2 \to [0,1]$ is called a t-norm iff it fulfills the following

[1] When $w = 1$, we may remove it.

[2] The precise definition of t-norm is given later.

properties for all $x, y, z, w \in [0, 1]$: **(1)** $x \otimes y = y \otimes x$ (commutativity); **(2)** $x \leq z$ and $y \leq w \Longrightarrow x \otimes y \leq z \otimes w$ (monotonicity); **(3)** $(x \otimes y) \otimes z = x \otimes (y \otimes z)$ (associativity); **(4)** $x \otimes 1 = x$ (identity). A t-norm is called *bounded* iff $x \otimes y = 0 \Longrightarrow x = 0$ or $y = 0$. There are several reasons for the use of a t-norm. Firstly, it is the generalization of the conjunction in propositional logic. Secondly, the operator *min* (*i.e.* $x \otimes y = min\{x, y\}$) is an instance of a bounded t-norm. This reflects an intuition on the use of analogical reasoning that the strength of a consequence depends upon the use of similarities. Lastly, 1 acts as the neutral element for t-norms. Table 6.1 shows other instances of $\otimes$.

Table 6.1: Some instances of the operator $\otimes$

Name	Notation	$x_1 \otimes x_2 =$
Minimum	$\otimes_{\min}$	$\min\{x_1, x_2\}$
Product	$\otimes_{\text{prod}}$	$x_1 \cdot x_2$
Hamacher product	$\otimes_{H_0}$	0 if $x_1 = x_2 = 0$; otherwise $\frac{x_1 \cdot x_2}{x_1 + x_2 - x_1 \cdot x_2}$

Example 6.5. (Continuation of Example 6.1) We translate the Silkwood case into our logic program $\mathcal{P}$ as follows. The literal *exception(X)* means "X is an exception to inactivate the goal". To avoid confusion, we separate each rule by a semicolon.

$\mathsf{SR} = \{defendant(X) \leftarrow owner(X, Y), danger(Y), killer(Y, Z);\ lion(l_1);\ owner\ (guy, l_1);$ $plutonium\_plant(p_1);\ owner(kerr\_mcgee, p_1);\ person(man);\ killer(l_1, man);\ person(silkwood);$ $killer(p_1, silkwood)\}$;

$\mathsf{DR} = \{liable(X) \Leftarrow defendant(X), not\ exception(X);\ danger(X) \Leftarrow lion(X)\}$;

$\mathsf{SIM} = \{plutonium\_plant \overset{0.8}{\leftleftarrows} lion\}$[1].

Let $\bigotimes$ be the *min* operator, we have $\mathcal{P} \overset{0.8}{\vdash} liable(kerr\_mcgee)$. □

6.3.2 Structured Argument

We are ready to extend the notion of derivation (Definition 6.8) for constructing arguments. As aforementioned, our notion of building arguments is adapted from DeLP (*cf.* [180,183,184] for the similar notion of arguments' structure) for discovering inconsistency in the strict knowledge base.

Definition 6.9 (Argument)**.** Let $\mathcal{P} = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ be a logic program and Q be a ground literal. A structure of an argument $\mathcal{A}$ for Q is a quadruple $\langle \mathsf{D}, \mathsf{S}, Q, w \rangle$, where $\mathsf{D} \subseteq \mathsf{DR}$ and $\mathsf{S} \subseteq \mathsf{SIM}$ such that:

1. There exists a derivation for Q from $\langle \mathsf{SR}, \mathsf{D}, \mathsf{S} \rangle$, *i.e.* $\langle \mathsf{SR}, \mathsf{D}, \mathsf{S} \rangle \overset{w}{\vdash} Q$;
2. If L is a literal in the derivation for Q, then there is no defeasible rule in D containing not L in the body;

[1]We may employ the notion of $\overset{\pi}{\sim}_{\mathcal{T}}$ to obtain 0.8 from realistic ontologies.

3. $\langle \mathsf{SR}, \mathsf{D}, \mathsf{S} \rangle$ is consistent, *i.e.* $\langle \mathsf{SR}, \mathsf{D}, \mathsf{S} \rangle \overset{w}{\nvdash} A, \overline{A}$ for some ground literal A;

4. $\mathcal{A}$ is minimal, *i.e.* there is no $\mathsf{D}' \cup \mathsf{S}' \subseteq \mathsf{D} \cup \mathsf{S}$ such that $\langle \mathsf{D}', \mathsf{S}', Q, w \rangle$.

It is worth noting that Condition 2 helps avoiding an introduction of a self-conflicting argument. For example, let $\mathcal{P} = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ where $\mathsf{SR} = \emptyset$, $\mathsf{DR} = \{a \Leftarrow b; b \Leftarrow \text{not } a\}$, and $\mathsf{SIM} = \emptyset$. Without Condition 2, it is possible to derive a from $\mathcal{P}$, *i.e.* assuming *not* a obtains a.

By distinguishing a set of defeasible rules of DR and a set of similarity rules of SIM in an argument's structure, we can clearly identify analogical arguments (or arguments from analogy) apart from standard arguments.

Definition 6.10. An argument $\mathcal{A} = \langle \mathsf{D}, \mathsf{S}, Q, w \rangle$ is called an *analogical argument* if $\mathsf{S} \neq \emptyset$ and is called a *standard argument* if otherwise. Alternatively, $\mathcal{A}$ is called a *strict argument* if $\mathsf{D} = \mathsf{S} = \emptyset$ and is called a *defeasible argument* if otherwise.

From the definition, $\mathcal{A}_1 = \langle \emptyset, \mathsf{S}_1, Q_1, 0.8 \rangle$, where $\mathsf{S}_1 = \{\mathit{plutonium\_plant} \overset{0.8}{\leftleftarrows} \mathit{lion}\}$ and $Q_1 = \mathit{danger(kerr\_mcgee)}$, is an analogical argument. Observe that an analogical argument is also a defeasible argument (such as $\mathcal{A}_1$).

Definition 6.11. Let $\mathcal{P} = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ be a logic program and $\mathcal{A}_1 = \langle \mathsf{D}_1, \mathsf{S}_1, Q_1, w_1 \rangle$, $\mathcal{A}_2 = \langle \mathsf{D}_2, \mathsf{S}_2, Q_2, w_2 \rangle$ be arguments. Then, we say that $\mathcal{A}_1$ *attacks* $\mathcal{A}_2$ iff one of the following conditions hold:

1. $\langle \mathsf{SR}, \mathsf{D}_1, \mathsf{S}_1 \rangle \overset{w_1'}{\vdash} A$ and $\langle \mathsf{SR}, \mathsf{D}_2, \mathsf{S}_2 \rangle \overset{w_2'}{\vdash} \overline{A}$ for some ground literal A;

2. $\langle \mathsf{SR}, \mathsf{D}_1, \mathsf{S}_1 \rangle \overset{w_1'}{\vdash} A$ and there exists $r \in \mathsf{D}_2$ which contains *not* A in the body;

Definition 6.11 does not include ways of comparing which arguments are better. It only says which arguments are in conflict. We illustrate this in Example 6.6.

Example 6.6. (Continuation of Example 6.5) Let us enrich the example that
$\mathsf{SR} = \{\mathit{defendant}(X) \leftarrow \mathit{owner}(X, Y), \mathit{danger}(Y), \mathit{killer}(Y, Z);\ \mathit{lion}(l_1);\ \mathit{owner}\ (\mathit{guy}, l_1);$
$\mathit{plutonium\_plant}(p_1);\ \mathit{owner}(\mathit{kerr\_mcgee}, p_1);\ \mathit{person}(\mathit{man});\ \mathit{killer}(l_1, \mathit{man});\ \mathit{person}(\mathit{silkwood});$
$\mathit{killer}(p_1, \mathit{silkwood});\ \neg \mathit{danger}(X) \leftarrow \mathit{green\_environment}(X)\}$;
$\mathsf{DR} = \{\mathit{liable}(X) \Leftarrow \mathit{defendant}(X), \mathit{not\ exception}(X);\ \mathit{danger}(X) \Leftarrow \mathit{lion}(X);$
$\mathit{green\_environment}(X) \Leftarrow \mathit{wind\_turbine}(X);$
$\mathit{federal\_law}(X) \Leftarrow \mathit{owner}(X, Y), \mathit{plutonium\_plant}(Y);\ \neg \mathit{liable}(X) \Leftarrow \mathit{federal\_law}(X)\}$;
$\mathsf{SIM} = \{\mathit{plutonium\_plant} \overset{0.8}{\leftleftarrows} \mathit{lion};\ \mathit{plutonium\_plant} \overset{0.4}{\leftleftarrows} \mathit{wind\_turbine}\}$.

We note that $\mathit{federal\_law}(X)$ refers to "the federal preemption of state regulation of the safety aspects of nuclear energy". Hence, we can find that:
$\mathcal{A}_1 = \langle \{\mathit{danger}(X) \Leftarrow \mathit{lion}(X); \mathit{liable}(X) \Leftarrow \mathit{defendant}(X), \mathit{not\ exception}(X)\},$
$\{\mathit{plutonium\_plant} \overset{0.8}{\leftleftarrows} \mathit{lion}\}, \mathit{liable}(\mathit{kerr\_mcgee}), 0.8 \rangle$ as an analogical argument,

$\mathcal{A}_2 = \langle \{federal\_law(X) \Leftarrow owner(X,Y), plutonium\_plant(Y);$
$\neg liable(X) \Leftarrow federal\_law(X)\}, \emptyset, \neg liable(kerr\_mcgee), 1\rangle$ as a defeasible argument, and $\mathcal{A}_3 = \langle \{green\_environment(X) \Leftarrow wind\_turbine(X)\}, \{plutonium\_plant \overset{0.4}{\Leftarrow} wind\_turbine\}, \neg danger(p_1), 0.4\rangle$ as an analogical argument. Thus, we have $\mathcal{A}_1$ attacks $\mathcal{A}_2$ and $\mathcal{A}_2$ attacks $\mathcal{A}_1$ by Condition (1). In addition, $\mathcal{A}_1$ attacks $\mathcal{A}_3$ and $\mathcal{A}_3$ also attacks $\mathcal{A}_1$ by Condition (1). □

The argument $\mathcal{A}_1$ attacks $\mathcal{A}_2$ since it satisfies the first condition, *i.e.* $\langle \mathsf{SR}, \mathsf{D}_1, \mathsf{S}_1 \rangle \overset{w_1'}{\vdash}$ *liable(kerr_mcgee)* and $\langle \mathsf{SR}, \mathsf{D}_2, \mathsf{S}_2 \rangle \overset{w_2'}{\vdash} \neg$*liable(kerr_mcgee)*. This way of attack is called *rebuttal*. Basically, a rebuttal attacks an argument by drawing the complement of a derived literal. On the other hand, the second condition is called *undercut*. An undercut attacks by showing an exceptional situation without drawing the complement of a literal.

Formalizing the scheme makes a special way of comparing arguments. This is because use of the scheme imposes some specialties (from the critical questions) for adjudicating conflicting arguments. There are three kinds of the comparison as discussed following.

Firstly, we consider the case of a standard argument $\mathcal{A}_1$ attacking another standard argument $\mathcal{A}_2$, *i.e.* $\mathcal{A}_1 = \langle \mathsf{D}_1, \mathsf{S}_1, Q_1, w_1\rangle$, $\mathcal{A}_2 = \langle \mathsf{D}_2, \mathsf{S}_2, Q_2, w_2\rangle$, and $\mathsf{S}_1, \mathsf{S}_2 = \emptyset$. Comparing arguments is treated in usual ways, *e.g.* some preference criteria are required to compare which argument is better for the rebuttal case. Such criteria can be defined as a relation $> \subseteq \mathsf{DR} \times \mathsf{DR}$ in a standard way, *i.e.* $r_1 > r_2$ means r_1 is preferred over r_2 for any $r_1, r_2 \in \mathsf{DR}$. Then, we say an argument $\mathcal{A}_1 = \langle \mathsf{S}_1, \mathsf{D}_1, Q_1, w_1\rangle$ is better than $\mathcal{A}_2 = \langle \mathsf{S}_2, \mathsf{D}_2, Q_2, w_2\rangle$ (denoted by $\mathcal{A}_1 \succ \mathcal{A}_2$[1]) if **(1)** $\exists r_1 \in \mathsf{D}_1 . \exists r_2 \in \mathsf{D}_2 : r_1 > r_2$; and **(2)** $\forall r_1 \in \mathsf{D}_1 . \forall r_2 \in \mathsf{D}_2 : r_2 \not> r_1$. We also establish that an argument structure based on facts is preferable to an argument structure based on presumptions.

Secondly, we consider the case of an analogical argument $\mathcal{A}_1$ attacking another analogical argument $\mathcal{A}_2$, *i.e.* $\mathcal{A}_1 = \langle \mathsf{D}_1, \mathsf{S}_1, Q_1, w_1\rangle$, $\mathcal{A}_2 = \langle \mathsf{D}_2, \mathsf{S}_2, Q_2, w_2\rangle$, and $\mathsf{S}_1, \mathsf{S}_2 \neq \emptyset$. In these cases, we compare the analogy degrees of both arguments. To defeat another, the degree must be at least equal to one's another.

Lastly, we consider the case of an analogical argument $\mathcal{A}_1$ attacking another standard argument $\mathcal{A}_2$, *i.e.* $\mathcal{A}_1 = \langle \mathsf{D}_1, \mathsf{S}_1, Q_1, w_1\rangle$, $\mathcal{A}_2 = \langle \mathsf{D}_2, \mathsf{S}_2, Q_2, w_2\rangle$, $\mathsf{S}_1 \neq \emptyset$, and $\mathsf{S}_2 = \emptyset$. In these cases, we base our reasoning on the use of argumentation schemes. Uttering an instance of Argument from Analogy like this in a dialogue creates a presumption in favor of the conclusion from analogy and a corresponding proof for the other side in the dialogue to defeat the conclusion by asking critical questions. This also conforms to Waller's use of analogical arguments [185] in the way of persuasion. Thus, an analogical argument is preferable.

In the following, conditions 1 to 3 capture these three kinds of comparing for rebuttal. Conditions 4 to 6 capture these three kinds of comparing for undercut.

Definition 6.12. Let $\mathcal{A}_1 = \langle \mathsf{D}_1, \mathsf{S}_1, Q_1, w_1\rangle$ and $\mathcal{A}_2 = \langle \mathsf{D}_2, \mathsf{S}_2, Q_2, w_2\rangle$ be two arguments. Then, $\mathcal{A}_1$ *defeats* $\mathcal{A}_2$ iff one of the following holds:

[1]Later, this definition is used by Definition 6.12 for comparing between rebuttal attacks.

1. $\mathcal{A}_1$ attacks $\mathcal{A}_2$ under Condition (1) of Definition 6.11, $(\mathsf{S}_1, \mathsf{S}_2 = \emptyset)$, $\mathcal{A}_2 \not\succ \mathcal{A}_1$, and $\mathcal{A}_2$ does not attack $\mathcal{A}_1$ under Condition (2) of Definition 6.11;

2. $\mathcal{A}_1$ attacks $\mathcal{A}_2$ under Condition (1) of Definition 6.11, $(\mathsf{S}_1, \mathsf{S}_2 \neq \emptyset)$, $w_1 \geq w_2$, and $\mathcal{A}_2$ does not attack $\mathcal{A}_1$ under Condition (2) of Definition 6.11;

3. $\mathcal{A}_1$ attacks $\mathcal{A}_2$ under Condition (1) of Definition 6.11, $\mathsf{S}_1 \neq \emptyset$, $\mathsf{S}_2 = \emptyset$, and $\mathcal{A}_2$ does not attack $\mathcal{A}_1$ under Condition (2) of Definition 6.11;

4. $\mathcal{A}_1$ attacks $\mathcal{A}_2$ under Condition (2) of Definition 6.11 and $(\mathsf{S}_1, \mathsf{S}_2 = \emptyset)$;

5. $\mathcal{A}_1$ attacks $\mathcal{A}_2$ under Condition (2) of Definition 6.11, $(\mathsf{S}_1, \mathsf{S}_2 \neq \emptyset)$, and $w_1 \geq w_2$; and

6. $\mathcal{A}_1$ attacks $\mathcal{A}_2$ under Condition (2) of Definition 6.11, $\mathsf{S}_1 \neq \emptyset$, and $\mathsf{S}_2 = \emptyset$.

We say that $\mathcal{A}_1$ *strictly defeats* $\mathcal{A}_2$ iff $\mathcal{A}_1$ defeats $\mathcal{A}_2$ and $\mathcal{A}_2$ does not defeat $\mathcal{A}_1$.

We note that many researchers in the area of argumentation with priority have defined many ways to compare arguments. Addressing this issue is outside the scope and is irrelevant to the Condition 1. We leave this as a future task.

Example 6.7. (Continuation of Example 6.6) We have that $\mathcal{A}_1$ strictly defeats $\mathcal{A}_2$ and also strictly defeats $\mathcal{A}_3$. □

Theorem 6.1. There does not exist an argument which attacks an argument $\mathcal{A} = \langle \emptyset, \emptyset, Q, w \rangle$, where Q is a ground literal and $w \in (0, 1]$.

Proof. Let $\mathcal{P} = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ be a logic program and suppose that there exists an argument $\mathcal{B} = \langle \mathsf{D}_b, \mathsf{S}_b, Q_b, w_b \rangle$ which attacks $\mathcal{A}$. By Definition 6.11, we show contradiction by cases:

- (Rebuttal) We have $\langle \mathsf{SR}, \mathsf{D}_b, \mathsf{S}_b \rangle \overset{w_b}{\vdash} A$ and $\langle \mathsf{SR}, \emptyset, \emptyset \rangle \overset{w_a}{\vdash} \overline{A}$ for some ground literal A. This means $\langle \mathsf{SR}, \mathsf{D}_b, \mathsf{S}_b \rangle$ is inconsistent and $\mathcal{B}$ is not an argument.

- (Undercut) We have $\langle \mathsf{SR}, \mathsf{D}_b, \mathsf{S}_b \rangle \overset{w_b}{\vdash} A$ and there exists $r \in \emptyset$ which contains *not* A in the body. This case is trivial.

❏

Corollary 6.1. Strict arguments always strictly defeat defeasible arguments, including analogical arguments.

Proof. This immediately follows from Theorem 6.1 and Definition 6.12. ❏

Corollary 6.1 exhibits that any conclusions drawn from analogy cannot strictly defeat conclusions drawn from the use of only strict rules. This shows that our system conforms to how the legal (and similar) uses analogy in reasoning.

6.3.3 Justification through Dialectical Analysis

In this work, we base our semantics on the grounded semantics of Dung's theory (*cf.* Section C.2 for the formal definition). A sound and complete calculus under the grounded semantics has a form of the dialectical style between a proponent (P) and an opponent (O) of an argument. A proponent starts with an argument to be justified and then the players take turn (*cf.* the first condition of Definition 6.13). An opponent must defeat or strictly defeat a proponent's last argument (*cf.* the forth condition of Definition 6.13) while a proponent must strictly defeat opponent's last argument (*cf.* the third condition of Definition 6.13). Moreover, a proponent is not allowed to repeat his arguments (*cf.* the second of Definition 6.13).

Definition 6.13 ([186])**.** A dialogue is a finite nonempty sequence of moves $move_i = \langle Player_i, \mathcal{A}_i \rangle$ where $i > 0$ such that:

1. $(Player_i = P \Longleftrightarrow i \text{ is odd})$ and $(Player_i = O \Longleftrightarrow i \text{ is even})$;
2. $(Player_i = P \text{ and } Player_j = P \text{ and } i \neq j) \Longrightarrow \mathcal{A}_i \neq \mathcal{A}_j$;
3. $Player_i = P$ $(i > 1) \Longrightarrow \mathcal{A}_i$ strictly defeats $\mathcal{A}_{i-1}$;
4. $Player_i = O \Longrightarrow \mathcal{A}_i$ defeats $\mathcal{A}_{i-1}$.

Definition 6.14 ([186])**.** A dialogue tree is a finite tree of moves such that:

1. Each path of the tree is a dialogue;
2. If $Player_i = P$, then children of $move_i$ are all defeaters of $\mathcal{A}_i$.

A player wins a dialogue if there are no moves for another player. Furthermore, a player wins a dialogue tree iff that player wins all paths of the tree.

Definition 6.15. An argument $\mathcal{A} = \langle \mathsf{D}, \mathsf{S}, Q, w \rangle$ is a *justified argument* from a logic program $\mathcal{P}$ (in symbols, $\mathcal{P} \vdash \mathcal{A}$) iff there exists a dialogue tree which $\mathcal{A}$ appears at the root and is won by the proponent. If $\mathcal{A}$ is justified, then Q is called a *justified conclusion* of $\mathcal{A}$. The justification degree, denoted by $|\mathcal{A}|$, is defined as follows:

$$|\mathcal{A}| = \bigoplus_{\mathcal{A}_i = \langle \mathsf{D}_i, \mathsf{S}_i, Q_i, w_i \rangle} \{w_i\}, \text{ for all } \mathcal{A}_i \text{ of each P along the tree and}$$

an accumulator $\bigoplus^1 : [0,1]^n \to [0,1]$ holds the following properties where n is the cardinality of the set $\{w_i\}$:

[1]This definition is used by this chapter only.

- Identity closed: $\forall a, \ldots, z \in \{w_i\} : \bigoplus\{a, \ldots, z\} = 1 \iff a = \cdots = z = 1$;
- Monotonicity: $\forall a, \ldots, z, a', \ldots, z' \in \{w_i\} : a \leq a' \wedge \cdots \wedge z \leq z' \Longrightarrow \bigoplus\{a, \ldots, z\} \leq \bigoplus\{a', \ldots, z'\}$.

Theorem 6.2. If an argument $\mathcal{A} = \langle \mathsf{D}, \mathsf{S}, Q, w \rangle$ is justified and its dialogue tree does not use analogical arguments, then $|\mathcal{A}| = 1$.

Let us illustrate this based on our motivative example. In the following, we define $\bigotimes$ as the *min* operator and $\bigoplus$ as the *arithmetic mean*.

Example 6.8. (Continuation of Example 6.7) There are 10 arguments from the logic program, *i.e.* $\mathcal{A}_1 = \langle \{danger(X) \Leftarrow lion(X)\}, \{plutonium\_plant \overset{0.8}{\leftleftarrows} lion\}\ , danger(p_1), 0.8 \rangle$,
$\mathcal{A}_2 = \langle \{green\_environment(X) \Leftarrow wind\_turbine(X)\},$
$\{plutonium\_plant \overset{0.4}{\leftleftarrows} wind\_turbine\}, green\_environment(p_1), 0{,}4 \rangle$,
$\mathcal{A}_3 = \langle \{danger\ (X) \Leftarrow lion(X)\}, \emptyset, danger(l_1), 1 \rangle$,
$\mathcal{A}_4 = \langle \mathcal{A}_3^{\mathsf{D}}, \emptyset, defendant\ (guy), 1 \rangle$[1],
$\mathcal{A}_5 = \langle \mathcal{A}_1^{\mathsf{D}}, \mathcal{A}_1^{\mathsf{S}}, defendant\ (kerr\_mcgee), 0{,}8 \rangle$,
$\mathcal{A}_6 = \langle \mathcal{A}_3^{\mathsf{D}} \cup \{liable(X) \Leftarrow defendant(X), not\ exception(X)\}, \emptyset, liable(guy), 1 \rangle$,
$\mathcal{A}_7 = \langle \mathcal{A}_1^{\mathsf{D}} \cup \{liable(X) \Leftarrow defendant(X), not\ exception(X)\}, \mathcal{A}_1^{\mathsf{S}}, liable(kerr\_mcgee), 0{,}8 \rangle$,
$\mathcal{A}_8 = \langle \{federal\_law(X) \Leftarrow owner(X\ , Y), plutonium\_plant(Y)\}, \emptyset, federa\_law(kerr\_mcgee),$
$1 \rangle$, $\mathcal{A}_9 = \langle \mathcal{A}_8^{\mathsf{D}} \cup \{\neg liable(X) \Leftarrow federal\_law(X)\}, \emptyset, \neg liable(kerr\_mcgee), 1 \rangle$,
$\mathcal{A}_{10} = \langle \mathcal{A}_2^{\mathsf{D}}, \mathcal{A}_2^{\mathsf{S}}, \neg danger(p_1), 0{,}4 \rangle$.

We can show that $\mathcal{A}_7$ is a justified argument through the dialectical analysis by starting a dispute with $\mathcal{A}_7$. Now, the opponent has to defeat this argument. However, there are no arguments defeating $\mathcal{A}_7$. Since the opponent runs out of moves, $\mathcal{A}_7$ is a justified argument with the degree $|\mathcal{A}_7| = 0.8/1 = 0.8$. □

The following definition identifies components for the implementation of an analogical reasoner. Intuitively, we would like to make the reasoner possible to be also fine tuned with a user-defined similarity threshold. Hence, we include a *relevant degree* s where $s \in (0, 1]$ apart from the logic program.

Definition 6.16 (Analogical Reasoner)**.** An analogical reasoner $\mathcal{R}$ is a pair $\langle \mathcal{P}, s \rangle$ where $\mathcal{P} = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ is a logic program and $s \in (0, 1]$ is a relevant degree. An argument $\mathcal{A}$ is *justified by* $\mathcal{P}$ *under* s iff $\mathcal{A}$ is a justified argument of $\mathcal{P}'$ where $\mathcal{P}' = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM}' \rangle$ and $\mathsf{SIM}' = \{L_0 \overset{x}{\leftleftarrows} L_1 \subseteq \mathsf{SIM} : x \geq s\}$.

6.3.4 Guideline of Choosing Operator $\otimes$ and $\oplus$

The following theorems are aids to help deciding which operator $\otimes$ and $\oplus$ should be chosen. We remind that $\otimes$ determines the analogy degree of an argument and $\oplus$ determines the overall degree of a justified conclusion.

[1]For the sake of succinctness, $\cdot^{\mathsf{D}}$ (and $\cdot^{\mathsf{S}}$) indicates a duplicated set of defeasible rules (and similarity rules, respectively) from a specified argument structure.

Table 6.2: Some instances of the operator $\oplus$

Name	Notation	$x_1 \oplus \cdots \oplus x_n =$
Minimum	$\oplus_{\min}$	$\min\{x_1, \ldots, x_n\}$
Product	$\oplus_{\text{prod}}$	$x_1 \cdot x_2 \cdot \cdots \cdot x_n$
Average	$\oplus_{avg}$	$\frac{x_1+\cdots+x_n}{n}$

Theorem 6.3. From Table 6.1 and let $x_1, x_2 \in (0, 1]$. Then, $\otimes_{\text{prod}} \leq \otimes_{H_0} \leq \otimes_{\min}$.

Proof. We show the following inequality:

$$x_1 \cdot x_2 \leq \frac{x_1 \cdot x_2}{x_1 + x_2 - x_1 \cdot x_2} \leq \min\{x_1, x_2\}$$

That is, we show $x_1 \cdot x_2 \leq \frac{x_1 \cdot x_2}{x_1+x_2-x_1 \cdot x_2}$ as follows:

$$x_1 \cdot x_2 \leq \frac{x_1 \cdot x_2}{x_1 + x_2 - x_1 \cdot x_2} \iff 1 \leq \frac{1}{x_1 + x_2 - x_1 \cdot x_2} \iff x_1 + x_2 - x_1 \cdot x_2 \leq 1$$
$$\iff x_2 - x_1 \cdot x_2 \leq 1 - x_1 \iff (1 - x_1) \cdot x_2 \leq 1 - x_1 \iff x_2 \leq 1 \text{ (by assumption)}$$

Lastly, we show $\frac{x_1 \cdot x_2}{x_1+x_2-x_1 \cdot x_2} \leq \min\{x_1, x_2\}$ in the similar fashion. ❑

Theorem 6.4. From Table 6.2 and let $x_1, \ldots, x_n \in (0, 1]$. Then, $\oplus_{\text{prod}} \leq \oplus_{min} \leq \oplus_{\text{avg}}$.

Proof. (Sketch) We show $x_1 \oplus_{\text{prod}} \cdots \oplus_{\text{prod}} x_n \leq x_1 \oplus_{min} \cdots \oplus_{min} x_n \leq x_1 \oplus_{\text{avg}} \cdots \oplus_{\text{avg}} x_n$ in the similar fashion of Theorem 6.3 by induction on n. ❑

Theorem 6.3 and Theorem 6.4 shows ordering of those instances of $\otimes$ and $\oplus$. This suggests that we can choose specific operators based on an application domain. For an analogical reasoner which strongly recognizes analogical principles, we may choose the weakest operators for both (*i.e.* $\otimes_{\min}$ and $\oplus_{\text{avg}}$). On the other hand, we may choose the strongest ones (*i.e.* $\otimes_{\text{prod}}$ and $\oplus_{\text{prod}}$) for an analogical reasoner which weakly recognizes analogical principles. We generalize this observation toward the nature of pessimistic in analogical reasoning. For the sake of succinctness, we may simply denote the chosen operators with superscripts, *e.g.* $\mathcal{P}^{\otimes_{\min}, \oplus_{\text{avg}}}$ and $|\cdot|^{\otimes_{\min}, \oplus_{\text{avg}}}$.

Definition 6.17. Let $\mathcal{P}$ be a logic program and $\otimes_1, \otimes_2, \oplus_1, \oplus_2$ be concrete operators. Let $\mathcal{A}^*$ be the set of all arguments from $\mathcal{P}$. Then, $\mathcal{P}^{\otimes_1, \oplus_1}$ is more *pessimistic* than $\mathcal{P}^{\otimes_2, \oplus_2}$ if $\forall a \in \mathcal{A}^* : (\mathcal{P}^{\otimes_1, \oplus_1} \vdash a \text{ and } \mathcal{P}^{\otimes_2, \oplus_2} \vdash a \Longrightarrow |a|^{\otimes_1, \oplus_1} < |a|^{\otimes_2, \oplus_2})$.

Dually, the nature of optimistic analogical reasoning is defined in the opposite direction, *i.e.* $\mathcal{P}^{\otimes_1, \oplus_1}$ is more *optimistic* than $\mathcal{P}^{\otimes_2, \oplus_2}$ if $\forall a \in \mathcal{A}^* : (\mathcal{P}^{\otimes_1, \oplus_1} \vdash a \text{ and } \mathcal{P}^{\otimes_2, \oplus_2} \vdash a \Longrightarrow |a|^{\otimes_1, \oplus_1} > |a|^{\otimes_2, \oplus_2})$.

6.3.5 Implementation Design: Analogist

Analogist is an inference system which reasons from analogy. Basically, it is a hybrid reasoner of concept similarity measure under preference profile in DLs and the argument-based logic program. We discusses a design sketch of the system in this section. It is also worth to mention that the argument from analogy has a great importance in legal practice, both in common law (because of the *stare decisis* principle, which implies that a court should use precedents to guide new decisions) and in civil law [187]. Unfortunately, this canon is prohibited in criminal law as it is the shared idea that judges shall not create new law in criminal matters.

The framework presented in Section 6.3 supplies the mechanism to define knowledge base in declarative ways. Taking this as advantage, we employ concept similarity measure techniques in DLs (*cf.* Chapter 4 - 5) to induce similarity rules used in Analogist. Figure 6.4 depicts this conceptual idea.

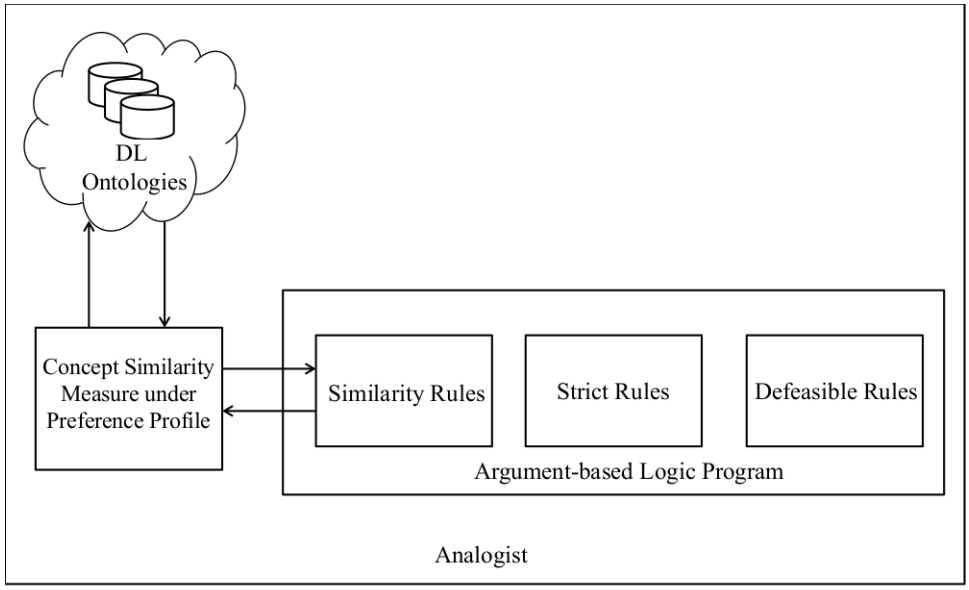


Figure 6.4: The design of Analogist

With the use of DLs, the semantics of each axiomatic information is formally defined. Traditional work on analogical reasoning indicates that, to measure similarity between two situations, both mapping of relation among objects and mapping of individual objects should take into account. Choosing appropriate measures can remedy this difficulty. For example, the measure sim^{π} [92] determines the similarity of $\mathcal{ELH}$ concepts based on axiomatic information in TBox. Preference profile (Definition 5.6) may be also used to define the preferred aspects for similarity identification at stake. For instance, Spence may define a highly importance on Danger and omit to consider other aspects, *i.e.* $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Danger}) = 2$ and $\mathfrak{i}^{\mathfrak{c}}(A) = 0$ for $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) \setminus \{\mathsf{Danger}\}$, and thereby Lion is totally similar to $\mathsf{PowerPlant}$, *i.e.* $\mathsf{Lion} \overset{\pi}{\sim}_{\mathcal{T}} \mathsf{PowerPlant} = 1$. This adds a similarity rule $\mathit{plutonium\_plant} \overset{1}{\leftleftarrows} \mathit{lion}$ (and vice versa) into the program.

6.4 Related Work

After surveying the literature on Argument from Analogy in many fields, such as logic, law, philosophy of science, and computer science, it appears to us that there are two different forms of Argument from Analogy. The first form (*cf.* Section 6.1), on which our two approaches are based, is the most widely accepted version whereas the second one compares factors of two cases (*e.g.* [188, 189]), which may be regarded as an instance of the first form. As the second one makes no reference to the notion of similarity, it becomes simpler to use, such as in standard case-based reasoning. The method of evaluating an argument from analogy in case-based reasoning (CBR) uses respects (*i.e.* dimensions and factors) in which two cases are similar or different. In CBR, the decision in the best precedent case is then taken as the decision into the current case. A dimension is a relevant aspect of the case whereas a factor is an argument favoring one side or the other in relation to the issue being disputed. The HYPO system [188] used dimensions. CATO [189] was a simpler CBR system that used factors. Systems which employ factors use pro factors to represent similarities for supporting an argument whereas con factors representing dissimilarity to undermine the argument. Factors may be weighted. In contrast, our approaches formalize the first form of Argument from Analogy in which $\overset{\pi}{\sim}_{\mathcal{T}}$ is used to identify similarity premises. This creates an advantage, *i.e.* many aspects of preference profile can be exploited.

ASPIC+ [190] was also a framework which constructed an argumentation framework based on the notion of proof tree with strict rules and defeasible rules. According to our literature, there are two common methodologies for building arguments, *i.e.* proof tree and the similar notion to our approach (*e.g.* [180, 183, 184]). The motivation of the notion used in [180, 183, 184] was to discover inconsistency in the strict knowledge base. For instance, let $\mathcal{P} = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ be a program where $\mathsf{SR} = \{a \leftarrow b; \neg a \leftarrow b\}$, $\mathsf{DR} = \{b \Leftarrow\}$, and $\mathsf{SIM} = \emptyset$. Those approaches which build arguments based on proof tree will accept an argument $\{b\}$; however, there will be no arguments for those systems similar to us since accepting b will derive inconsistency in the strict knowledge base via $a \leftarrow b$ and $\neg a \leftarrow b$.

There were also substantial efforts of linking analogical reasoning to existing logical models of non-monotonic reasoning. For example, [191] proposed a form of analogical reasoning based on hypothetical reasoning. In that approach, similarity was expressed as an equality hypothesis and a goal-directed theorem prover was used to search relevant hypotheses. In [192,193], an analogical reasoning was considered as deductive reasoning by inserting the rule (in our language): *has_property(t, p)* $\leftarrow$ *has_property(s, p)*, *similar(s, t)* as a strict rule to be used in deriving analogical conclusions.

Regarding the first approach, our operator $\cdot^{+}$ also has a rule similar to the above. However, its approach is different to others on the constraint and similarity identification. Existing logical approaches require consistency on logic programs and employs a preference, *i.e.* maximizing the number of common properties. In contrast, the approach relies on the notion of counter-analogy and exploits the notion of preference profile for specifying preferences over similarity. Using consistency as the only constraint is not sufficient. For instance, let us exemplify a counter-example (in its language):

$\mathcal{LP} = \{danger(X) \leftarrow lion(X); \neg danger(X) \leftarrow solar\_plant(X); plutonium\_plant(p_1).\}$ and $\mathcal{O} = \{\mathsf{lion} \sqsubseteq \mathsf{harm} \sqcap \mathsf{wild}; \mathsf{plutonium\_plant} \sqsubseteq \mathsf{power\_plant} \sqcap \mathsf{harm}; \mathsf{solar\_plant} \sqsubseteq \mathsf{power\_plant} \sqcap \mathsf{green\_environment}\}$. Most of existing approaches would conclude either "$danger(p_1)$" or "$\neg danger(p_1)$" from the logic program. However, this approach concludes nothing as a counter-analogy is discovered. Using the number of common properties to identify preferred similarity also has less flexibility than using preference profile, *e.g.* it cannot express importance over different names.

Definition 6.8 of the second approach also uses a rule which functions similar to the above. Like the first approach, it also differs from the others on constraint checking and similarity identification. As aforementioned, existing logical approaches require consistency on logic programs and uses the number of common properties (in the model theory) for similarity identification. This approach also relies on the notion of counter-analogy and uses similarity rules to define similarity of two predicates. Using only consistency for constraint checking is not enough. We exemplify this case in the language of our argument-based logic programming. For instance, let $\mathcal{P} = \langle \mathsf{SR}, \mathsf{DR}, \mathsf{SIM} \rangle$ be a program where $\mathsf{SR} = \{plutonium\_plant(p)\}$, $\mathsf{DR} = \{danger(X) \Leftarrow lion(X); \neg danger(X) \Leftarrow solar\_plant(X)\}$, and $\mathsf{SIM} = \{plutonium\_plant \leftrightharpoons lion; plutonium\_plant \leftrightharpoons solar\_plant\}$. Most existing approaches would conclude either "$danger(p)$" or "$\neg danger(p)$" from logic programs. However, our approach concludes nothing as a counter-analogy is discovered. Using similarity rules is also a pro since it enables integration with external systems for building similarity rules, as designed in Analogist. This creates more dimensions of configurable aspects for evaluating the similarity of concepts.

The main differences between the proposed two approaches are the notion of their argument's structure and the used semantics for their argument's evaluation. While the second approach considers on the notion of arguments' structure, the first approach relies on proof trees (*cf.* Subsection 6.2.4). Furthermore, the second approach employs the grounded semantics whereas the first approach employs the stable semantics. Due to these differences, their practical applications can be different.

Finally, we remark that there are extremely interesting relations between analogical reasoning and defeasible argumentation. We have covered some of them in this chapter.

6.5 Summary

- Analogical reasoning is a form of non-deductive reasoning in which a conclusion is inferred base on the similarity of concepts or states of affairs; and
- Two formalisms were proposed based on concept similarity measure under preference profile introduced in Chapter 5. Their relationship to defeasible argumentation were also discussed. Finally, we showed that analogical arguments driven by the similarity of concepts can be reconstructed from the proposed formalisms.

Chapter 7

Concluding Remarks

This thesis investigated and formally defined the notion of concept similarity measure (in Description Logics), which can also be used when the agent's preferences are provided. The main objectives of this work were to investigate the computational approaches for identifying the degree of similarity between two concept descriptions and to demonstrate their applicability on several areas such as recommendation systems based on the agent's preferences, domain-specific knowledge base, and analogical reasoning.

To achieve these goals, we tool a look into the classical notion of concept equivalence in Description Logics because concept equivalence can be seen as the basis operation for comparing concepts. Since concept equivalence of concepts can be determined by evaluating concept subsumption of concepts w.r.t. the two corresponding directions, existing and efficient subsumption checking algorithms have been investigated and generalized for our proposed notion of concept similarity measure under the agent's preferences. In the following sections, major technical and empirical results of this thesis are discussed.

7.1 Discussion of Achieved Results

The major results achieved in this thesis can be classified as follows:

- The development of concept similarity measure under preference profile in Description Logics (with its emphasis on sub-Boolean logics);
- The design of algorithmic procedures for our proposed measure sim^{π} and their empirical evaluation w.r.t. realistic ontologies; and
- The extended development for computational approaches of analogical reasoning based on our proposed concept similarity measure under preference profile.

7.1.1 The Development of Concept Similarity Measure under Preference Profile in Description Logics

Concept similarity measure can be regarded as a generalization of the classical reasoning problem of equivalence. That is, any two concepts are equivalent if and only if their

similarity degree is one (*cf.* Equation 4.22). Regarding this observation, we have investigated several approaches to compute the degree of subsumption between concepts in sub-Boolean DLs since their subsumption reasoning problems are tractable and they are expressive enough to formulate realistic ontologies. Several results according to this step included two subsumption degree algorithms for the DL $\mathcal{FL}_0$ (*cf.* Chapter 4). Using these results, we can compute the degree of concept similarity based on the degree of subsumption degrees w.r.t. two corresponding directions. For instance, the measures $\sim^s$, $\sim^c$, and sim are defined as the average of the two corresponding subsumption degrees of $\mathcal{FL}_0$ concepts, $\mathcal{FL}_0$ concepts, and $\mathcal{ELH}$ concepts, respectively. These results are capable of providing the degree of relation w.r.t. their common and different features even though the concepts are not in equivalence relationship. Accordingly, they play a major role in the discovery of similar concepts in an ontology and are often employed by ontology alignment algorithms.

The existence of having two proposed measures for the same DL $\mathcal{FL}_0$ corresponds to an experiment in [154] that similarity measure might depend on target applications and should be personalized to the agent's similarity judgment style. Our investigation on the relationship between these two measures (*cf.* Proposition 4.2) guides an approach to apply a suitable measure for an applying agent. For example, when applying a similarity measure for building a query answering system and an agent would like to see many possible matching results, the measure $\sim^c$ could be applied.

It is obvious that choosing the right measure is one way of representing the agent's preferences. Fortunately, the definition of concept similarity measure is extended and generalized in Chapter 5 in such a way that the degree of concept similarity is also identified w.r.t. subjective factors (*e.g.* the agent's preferences). This generalized notion is called *concept similarity measure under preference profile*. In Section 5.1, a formalism for expressing the agent's preferences in concept similarity called *preference profile* was developed. Two existing measures *viz.* $\sim^s$ and sim are refined according to each aspect of preference profile and result in $\overset{\pi}{\sim} s$ for the DL $\mathcal{FL}_0$ and sim^π for the DL $\mathcal{ELH}$, respectively.

Apart from the definition of concept similarity measure under preference profile, Chapter 5 also identified a set of desirable properties that any concrete measures of this notion should satisfy. We have provided proofs of satisfied properties for the developed concrete measures. Understanding their satisfied properties is important for employing the measures in any applicable areas since their users can predict the expected behaviors. The measures can also be used regardless of the agent's preferences. In such cases, both $\overset{\pi}{\sim} s$ and sim^π can be tuned with the special preference profile called the default preference profile π_0. We have also provided several guidance to identify suitable values of preference profile (*cf.* Section 5.5). Finally, we have provided proofs that when the TBox is unfoldable, both $\overset{\pi}{\sim} s$ and sim^π can be computed in polynomial time.

7.1.2 The Design of Algorithmic Procedures for sim^{π} and Their Empirical Evaluation w.r.t. Realistic Ontologies

In Section 5.6, two concrete algorithms *viz.* the top-down approach and the bottom-up approach for implementations of sim^{π} were developed. The computational complexity of each algorithm was clearly analyzed. Concretely, both algorithms make the similar number of executions; however, the bottom-up additionally requires an amount of extra space due to the employed dynamic programming technique. Unlike the top-down approach, the bottom-up approach has never recursively invoked itself to determine the similarity of different pair of nested concepts. The algorithm directly uses values stored in the table. Both approaches have different benefits and drawbacks. On the one hand, the bottom-up requires an additional extra space. On the other hand, it does work productively an environment where recursion is fairly expensive.

In Section 5.7, our defined notion $\overset{\pi}{\sim}$ has been evaluated with realistic ontologies w.r.t. several use cases. In this thesis, we used sim^{π} to show the practical performance of both developed algorithms and usefulness of tuning the measure via preference profile. Both algorithms of sim^{π} were implemented using Java version 1.8 with the usage of Spring Boot version 1.3.3.RELEASE as application programming interfaces (APIs). These APIs can also be used by application developers to use sim^{π} with their working ontologies. Results of the empirical evaluation are summarized as follows:

1. We compared the practical performance of the top-down sim^{π} and the bottom-up sim^{π} w.r.t. the medical ontology SNOMED CT. The experiment showed that the bottom-up sim^{π} performs approximately three times faster than the top-down sim^{π}. This result conforms to our theoretical analysis as discussed earlier;

2. We re-implemented the existing measure sim based on the same technologies and techniques as sim^{π}. Then, we compared the practical performance of sim^{π} and sim w.r.t. SNOMED CT and found that they perform equally.;

3. We evaluated the backward compatibility of sim^{π} with sim. This experiment would like to ensure that the default preference profile can be used when preferences are not given by the agent. Our experiment has guaranteed this;

4. We showed the usefulness of our defined notion through measuring the similarity of SNOMED CT concepts. Due to its special characteristics, measuring similarity of SNOMED CT concepts requires special ways of tuning the measure. We showed that tuning sim^{π} under the special setting yields the more intuitive results. We also compared the use of *simi*, which is another measure for the same $\mathcal{ELH}$, and found that lacking (even some) aspects of preference profile may not be suitable to use with an ontology where some special cases of tuning are required; and

5. We also showed the usefulness of our defined notion in several use cases of query answering systems with realistic ontologies. The discussion showed that $\overset{\pi}{\sim}$ is appropriate to identify the degree of similarity w.r.t. the agent's preferences.

Our defined notion has great potential use in knowledge engineering, such as the development of recommendation systems based on the agent's preferences, the development of domain-specific knowledge bases, and the ontology engineering. Moreover, it may be used with heterogeneous ontologies by identifying duplicated primitive concepts and primitive roles among ontologies via $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$ (*cf.* Section 5.9), respectively, and used in the implementation of an analogical inference engine. We summarize our study on the computational approaches for analogical reasoning in the next subsection.

7.1.3 Extending Concept Similarity Measure under Preference Profile for Analogical Reasoning

Reasoning by analogy is a form of non-deductive reasoning in which we infer a conclusion based on similarity of two concepts or states of affairs. Two formalisms were proposed in Chapter 6 based on the argumentation scheme for argument from analogy. However, their approaches of development were different. The first approach has exploited benefits of extensive tools from answer set programming together with our developed notions (and our developed APIs). This provides a rough-and-ready method for building an analogical inference engine. On the other hand, the second approach developed an argument-based logic-programming-like language which provides the possibility of representing information in terms of strict, defeasible, and similarity rules in a declarative manner. In this formalism, critical questions can be reconfigured as the defeating relation and the acceptability of constructed arguments is evaluated w.r.t. the grounded semantics.

A realistic argument in a legal case, which was made from analogies between concepts, was shown to be "reconstructible" by the proposed formalisms to show their applicability. For other viewpoints, ones may concern only the precedent is applicable to the current case or not – but do not concern if the precedent is similar to the current case or not. This in fact reflects to the two different phases of an analogical reasoning system, *viz.* the similarity phase and the applicability phase. Based on this viewpoint, our proposed system is positioned in the similarity phase in which the system will suggest the reasoning by analogy to a target situation (*e.g.* the legal field) and the expert (*e.g.* the law people) may consider the applicability after our suggestion; this is at their disposal.

7.2 Directions of Future Research

Several directions for further research on computational approaches of concept similarity measure in DLs and analogical reasoning are in order:

- Our proposed measures in this thesis are not meant to be the universal measures. Indeed, they are restricted to the DLs $\mathcal{FL}_0$ and $\mathcal{ELH}$ with an unfoldable TBox. While they came with the limitation in terms of expressivity, their computations were proven to be tractable; thereby, provided practically acceptable response time which is a key requirement in the design and the development of large-scale ontologies. As for future work, we are interested in exploring other techniques of concept

similarity measure under preference profile for more expressive DLs;

- The current structure of preference profile also restricts its expressivity on sub-Boolean logics, particularly $\mathcal{ELH}$. Hence, it appears to be a natural step to extend preference profile to support more expressive DLs *e.g.* concept negation, and also, to support capabilities to express preferences on an ABox;

- As reported in [154] about the need of having multiple measures, we are interested to investigate the possible classes of similarity measures w.r.t. their potential use cases and applications. Understanding this would help the agent to select the right measure for a dealing situation;

- The proposed approaches of concept similarity measure under preference profile has an advantage of computing the degree of commonalities under the agent's preferences. On the other hand, they do not provide a good reason why two concepts are considered as "being similar". As for future work, we are interested in extracting the computational content which makes two concepts considered as being similar. To do this, we may investigate the deduction systems *e.g.* a sequent calculus and a natural deduction system as developed in [69].

- Apart from our developed Java APIs, we intend to extend our development as a plug-in of ontology editors such as Protégé. Doing this would undoubtedly spread out their usability to a wider group of users;

- The current usage of preference profile appears only in the task of concept similarity measure, which is a TBox-related problem. Now, we are interested in exploring ways to adopt preference profile on ABox-related problem *e.g.* non-standard instance checking under preference profile. The idea in the nutshell is to use concept similarity measure under preference profile for ABox instance checking rather than using the standard instance checking techniques. This may also involve extending the structure of preference profile with some capabilities of defining preferences over each instance in the ABox;

- Concept similarity measure is the core functionality of various problems. Analogical reasoning is one of such problems and we demonstrated in Chapter 6 how ones could build an inference engine based on our proposed measures. However, there were still problems and open questions remained for further investigation. This might involve continuously study in the area of argumentation and structured argumentation. According to this point, we are interested to investigate how ones can construct (and reconstruct) analogical arguments based on other formalisms in structured argumentation such as ASPIC$^+$ [190] and ABA [194].

Appendix A

The Systematized Nomenclature of Medicine: SNOMED CT

The Systematized Nomenclature of Medicine, Clinical Terms (*aka.* SNOMED CT)[2] is one of the largest and the most widely used medical ontologies currently available. Figure A.1 depicts its web interface which can be accessed via the link given at the footnote. It was produced by merging SNOMED Reference Terminology (RT) [195, 196] with Clinical Terms version 3 (CTV3) [197].

Historically, SNOMED RT was developed by the College of American Pathologists (CAP) with the aim to be a comprehensive clinical reference terminology *e.g.* the retrieval and analysis of data relating the causes of diseases, the treatment of patients, and retrieval of health care information [195]. The RT version was the first generation of the SNOMED terminology to use the formal semantics through the KRSS syntax [198].

In 1993, the UK National Health Service (NHS) has adopted the Read codes, which had been developed by a medical practitioner Read, for health electronic records. Later on, the terminology has been expanded and enhanced to become Clinical Terms version 3 (CTV3).

Between 1999 and 2002, CAP and the UK NHS together with Keiser Permanente have jointly worked to merge SNOMED RT and CTV3. Its resulting SNOMED CT contained 55% of the source concepts from CTV3 and 31% from RT. Moreover, the ontology become freely available in both the US and UK.

Nowadays, SNOMED CT is already used by more than 50 countries. Furthermore, it is the most comprehensive, multilingual clinical health-care terminology in the world and is mapped to other international standards. As reported in [105, 106], SNOMED CT can be seen as the DL $\mathcal{ELH}$ with an unfoldable TBox. SNOMED CT has several inherent characteristics. We discuss several of them in the following.

Firstly, SNOMED CT purposefully uses the special role roleGroup to group two or more existential quantifications in a definition. Spackman *et al.* has illustrated the use of roleGroup for the concept "Tetralogy of Fallot" in [199] as follows.

[2] http://bioportal.bioontology.org/ontologies/SNOMEDCT

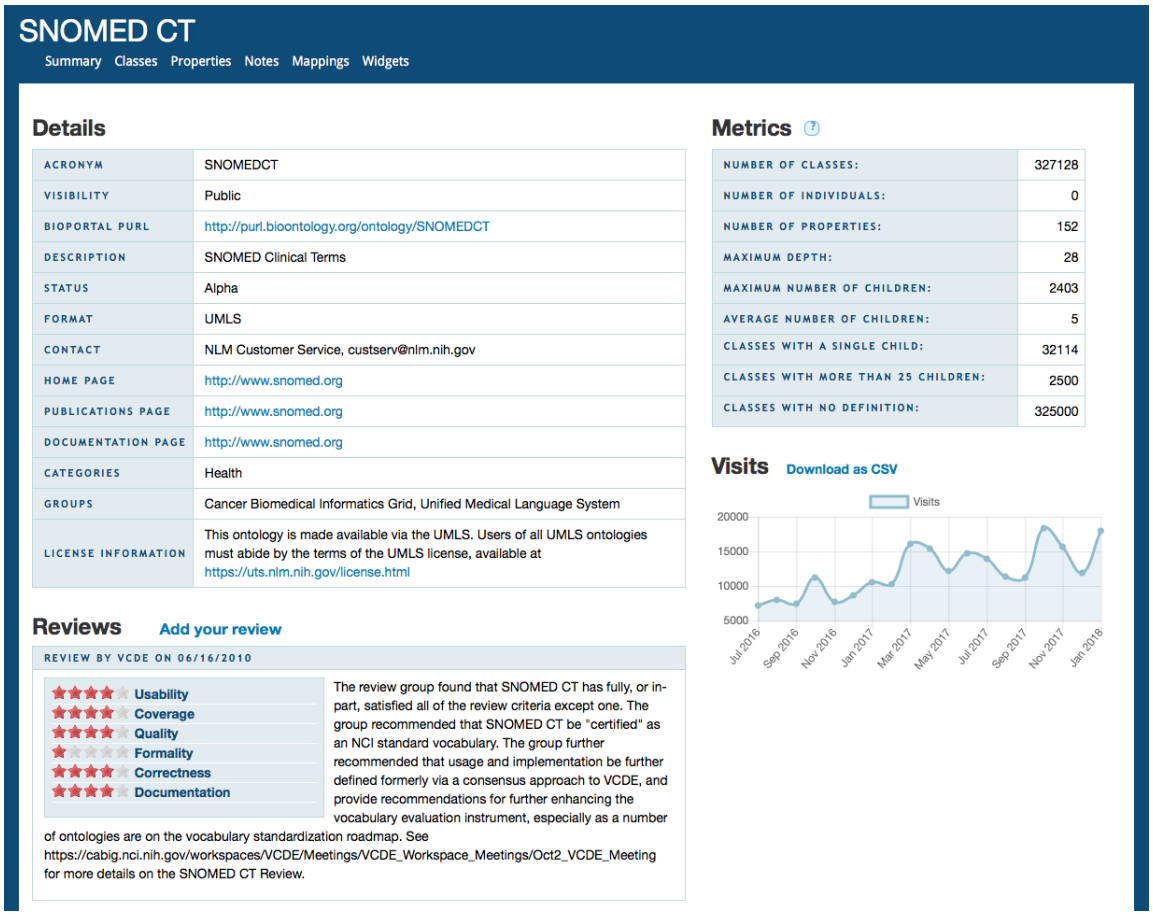


Figure A.1: SNOMED CT BioPortal (accessed on February 21, 2018)

$$\begin{aligned}\mathsf{TetralogyOfFallot} \equiv\ & \exists \mathsf{rG}.(\exists \mathsf{s}.\mathsf{RightVentricle} \sqcap \exists \mathsf{m}.\mathsf{Hypertrophy}) \sqcap \\ & \exists \mathsf{rG}.(\exists \mathsf{s}.\mathsf{Aorta} \sqcap \exists \mathsf{m}.\mathsf{Overriding}) \sqcap \\ & \exists \mathsf{rG}.(\exists \mathsf{s}.\mathsf{Pulmonary} \sqcap \exists \mathsf{m}.\mathsf{Stenosis}) \sqcap \\ & \exists \mathsf{rG}.(\exists \mathsf{s}.\mathsf{InterventricularSeptum} \sqcap \exists \mathsf{m}.\mathsf{IncompleteClosure})\end{aligned}$$

where rG, s, and m are abbreviations for roles roleGroup, site, and morphology, respectively.

Secondly, individuals (*i.e.* the ABox) are omitted. On the other hand, SNOMED concepts such as Germany, Japan, and Thailand are used to represent unique individuals. Indeed, they are seen as "instances" of the concept GeographicLocation.

Thirdly, SNOMED CT has 18 mutually exclusive top-level concepts for dividing the entire ontology into disjoint categories. However, the disjointness is not logically specified as axioms; hence, some concept names may happen to belong to more than one category.

Lastly, the SNOMED CT top concept SCT-Top subsumes every defined concept of each category. This means this special concept is shared by every expanded concept.

In this thesis, we use SNOMED CT ct ontology version from January 2005 which contains 13 role inclusions, 38,719 concept definitions, 340,972 primitive concept definitions, 379,691 concept names and 62 role names.

Appendix B

Implementation of The Measure sim^{π}

We have implemented sim^{π} as a collection of application programming interfaces (APIs) as well as command-line interfaces (CLIs) using Java version 1.8 with the usage of Spring Boot version 1.3.3.RELEASE. The ultimate goal of these APIs is to provide a tool for identifying the degree of concept similarity under preference profile for the DL $\mathcal{ELH}$. As shown in Theorem 5.5, the computation of sim^{π} can be performed in polynomial time.

Since sim^{π} is targeted on $\mathcal{ELH}$, we summarize the provided constructors as follows:

- top concept "$\top$",
- conjunction "$C \sqcap D$", and
- full existential quantification "$\exists r.C$"; and

the following means of expressivity to construct an ontology as follows:

- primitive concept definition "$A \sqsubseteq D$",
- concept definition "$A \equiv D$", and
- role hierarchy axiom "$r \sqsubseteq s$".

Currently, sim^{π} accepts two formats of inputs *viz.* in KRSS[2] (Knowledge Representation System Specification) [198], OWL (Web Ontology Language), and OWL 2[3]. Our APIs wrap OWL API[4] version 3.4.4. In the following, we have summarized shortly both KRSS and OWL syntaxes only the parts relevant to our APIs for self-containment of the thesis.

In KRSS, an ontology contains the following sorts of statements:

- primitive concept definition "(define-primitive-concept CN C)",
- concept definition "(define-concept CN C)", and
- role hierarchy axiom "(define-primitive-role RN_1 RN_2)",

[2] http://dl.kr.org/krss-spec.ps

[3] `https://www.w3.org/TR/owl2-overview/`

[4] `http://semanticweb.org/wiki/OWL_API.html`

where CN be a concept name, RN_1 and RN_2 be two different role names, and concept C can be either CN, TOP, or formed as follows:

- conjunction "(and C_1 ... C_n)",
- full existential quantification "(some RN C)",

where concept $\mathsf{C}, \mathsf{C}_1, \ldots, \mathsf{C}_n$ are recursively defined as above.

Figure B.1 depicts an overview of OWL 2. In the center, the ellipse represents the abstract notion of an ontology, which can be thought of as an abstract ontology structure or an RDF graph. The top of the figure shows each concrete syntax based on the abstract notion which can be serialized and exchanged. The bottom shows the two specification of semantics defining the meaning of an ontology. As aforementioned, our APIs wrap the OWL API, which can handle these various syntaxes and semantics. Thus, this capability automatically transfer to our APIs for free. We refer the readers to check the official documentation for the full descriptions of each syntax and semantics.

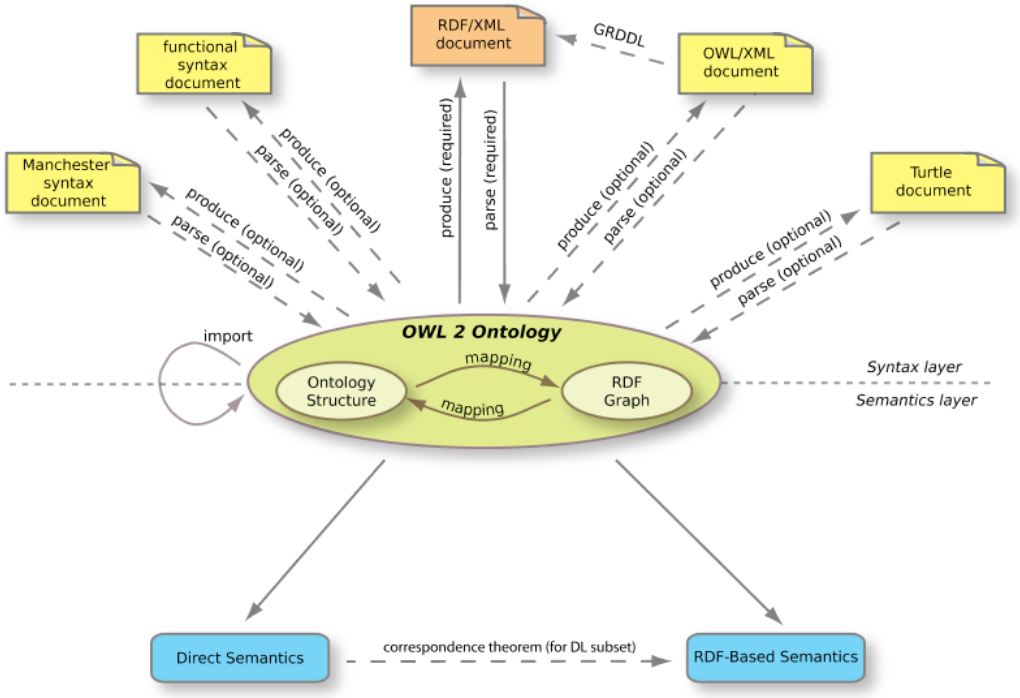


Figure B.1: The structure of OWL 2
(source: `https://www.w3.org/TR/owl2-overview/OWL2-structure2-800.png`)

To use our APIs in Java, four classes may be involved *viz.* "KRSSServiceContext", "OWLServiceContext", "PreferenceProfile", and "SimilarityService". First, KRSSServiceContext and OWLServiceContext are used to initialize the ontology from a given file path. Initializing the ontology is mandatory and is required to do once prior to the query of concept similarity. Second, PreferenceProfile is used to configure each aspect of preference profile (*cf.* Section 5.1). If this class is not explicitly used, it will automatically

use the default value (*cf.* the default preference profile). Third, SimilarityService encapsulates functionalities to compute the degree of concept similarity based on a syntax and a computational approach (*cf.* Section 5.6) as follows:

- measureOWLConceptsWithTopDownSimPi(conceptName1 : String, conceptName2 : String) : BigDecimal;
- measureOWLConceptsWithDynamicProgrammingSimPi(conceptName1 : String, conceptName2 : String) : BigDecimal;
- measureKRSSConceptsWithTopDownSimPi(conceptName1 : String, conceptName2 : String) : BigDecimal; and
- measureKRSSConceptsWithDynamicProgrammingSimPi(conceptName1 : String, conceptName2 : String) : BigDecimal.

The following demonstrates how ones can use our APIs in Java.

```
public class Example {

   ... // Initialize logger, owlServiceContext and similarityService

   private void run(String... args) {
      String owlFilepath = StringUtils.trimWhitespace(args[0]);
      String conceptName1 = StringUtils.trimWhitespace(args[1]);
      String conceptName2 = StringUtils.trimWhitespace(args[2]);

      owlServiceContext.init(owlFilepath);
      BigDecimal value =
         similarityService.measureOWLConceptsWithTopDownSimPi(conceptName1,
         conceptName2);

      logger.info("Done! The similarity between " + conceptName1 + " and " +
         conceptName2 + " is " + value.toPlainString() + " %.");
    }
}
```

Listing B.1: Example of using sim^{π} APIs in Java

We have also implemented several batch programs based on the APIs and used them on the part of our empirical evaluation of the thesis (*cf.* Section 5.7). For the current implementation, each program stores each concept pair in question as a text file separated by a space. Each aspect of preference profile is stored on its own file but is collectively kept together in the same folder. Their outputs after the execution is stored in another text file. Figure B.2 depicts the idea as described above. We also implemented other batch programs based on the same techniques for the measure sim (as discussed in Section 5.7) with the purpose of benchmarking. In total, we have implemented 8 programs. Each uses the same structure as shown in the figure.

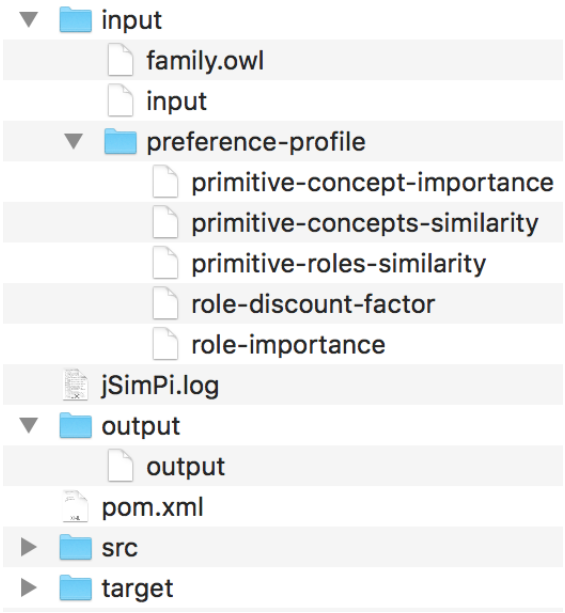


Figure B.2: Our batch program's structure

To run each batch program, we have to execute the command "mvn spring-boot:run". When the program is run, it will take each concept pair defined in a given ontology (such as "family.owl" in this case), compute the degree of similarity under a defined preference profile, and pipe the results to output file. Figure B.3 illustrates an example after the execution. The figure shows that the degree of similarity between both concepts is 0.96.

```
Son     SonInLaw        0.96000
```

Figure B.3: The degree of similarity between Son and SonInLaw

Finally, we have written 111 unit test cases to ensure that all batch programs and the core APIs function correctly. These test cases were written to cover important parts of the implementation. Concepts in both the family ontology (family.owl) and SNOMED CT were used by the test cases. To execute the test, we use the command "mvn test". Figure B.4 depicts the results.

```
Results :

Tests run: 110, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------
[INFO] Total time: 49.587 s
[INFO] Finished at: 2018-02-14T22:55:32+09:00
[INFO] Final Memory: 16M/226M
[INFO] ------------------------------------------
```

Figure B.4: Results of unit tests

Appendix C

Defeasible Argumentation

C.1 Argumentation Schemes

Argumentation schemes [164] are stereotypical non-deductive patterns of reasoning, consisting of a set of premises and a conclusion that is "presumed" to follow from the premises. Many of important schemes have been identified and analyzed by Hastings in 1963 [200], Perelman and Olbrechts-Tyteca in 1969 [201], Kienpointner in 1992 [202], Walton in 1996 [203], and Grennan in 1997 [204].

Nowadays, there have been considerable interest on argumentation schemes in the filed of artificial intelligence, particularly in agent reasoning (*cf.* [205, 206]). In the area of argumentation study, schemes can be used to identify arguments, find missing premises, analyzing arguments, and finally evaluate them. Use of argumentation schemes is evaluated by a specific set of critical questions corresponding to each scheme. If such questions have not been answered adequately, conclusions drawn by the schemes will fail to hold. Since Chapter 6 uses a scheme called *argument from analogy*, we illustrate how this argumentation scheme works as follows:

Similarity Premise	Generally, case C_1 is similar to case C_2
Base Premise	A is true (false) in case C_1
Conclusion	A is true (false) in case C_2

This generic structure can be explained as follows. The similarity is regarded to hold between two cases. These cases could be two different concepts or states of affairs. Consequently, a property (e.g. a feature A) attributes to both cases.

As aforementioned, any arguments fitting the scheme for argument from analogy are evaluated in a dialogue framework in which another party can ask critical questions. Basically, this amounts to investigate its acceptability. Following outlines critical questions (CQ) associated to the scheme.

CQ1	Is A true (false) in C_1?
CQ2	Are C_1 and C_2 similar in the respects cited?
CQ3	Are there important differences (dissimilarities) between C_1 and C_2
CQ4	Is there some other case C_3 that is also similar to C_1 except that A is false (true) in C_3?

The first critical question nicely ensures the right conclusion. The second and the third critical questions relate to differences between the two cases that could detract from the strength of the argument from analogy. Lastly, the forth critical question is associated with a familiar type of counter-analogy. The function of this critical question is to suggest doubt that could possibly lead to a plausible counter-argument that could be used to attack the original conclusion.

Argument from analogy is immensely employed in the procedure of analogical reasoning and case-based reasoning [188, 189]. Several concrete representations of the above structure are studied in [168] and are demonstrated in practical areas *e.g.* laws and clinical practices in [168, 207]. However, due to its generality, the argumentation scheme and its critical questions do not deal with the three basic questions (as mentioned in Chapter 6), we have tackled those in the thesis.

C.2 Argumentation Framework

An abstract argumentation framework (AF) [172] is a pair $(\mathcal{A}, \mathcal{R})$ where $\mathcal{A}$ is a set of *arguments* and $\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$ is called an *attack relation*. Arguments may attack each other; hence, it is clear that they may not stand together and their statuses are subject to an evaluation. In this sense, semantics for AF returns sets of arguments called *extensions*, which are *conflict-free* and *defend* themselves against attacks [172]. Their formal definitions are given as follows:

Definition C.1. Let $(\mathcal{A}, \mathcal{R})$ be an AF. An argument $a \in \mathcal{A}$ is *acceptable* w.r.t. a set $S \subseteq \mathcal{A}$ iff $\forall b \in \mathcal{A} : (b, a) \in \mathcal{R} \implies (S, b) \in \mathcal{R}$. Also, let S be *conflict-free*, i.e. $\nexists a, b \in S : (a, b) \in \mathcal{R}$[1]. Then, S is called:

- an *admissible* extension iff $x \in S \implies x$ is acceptable w.r.t. S;
- a *complete* extension iff $x \in S \iff x$ is acceptable w.r.t. S;
- a *preferred* extension iff S is a set inclusion maximal complete extension;
- a *grounded* extension iff S is a set inclusion minimal complete extension;
- a *stable* extension iff $\forall x \in \mathcal{A} : x \notin S \implies (S, x) \in \mathcal{R}$.

It its worth noting that argumentation framework is another line of research which has its root in the study of logic programming and non-monotonic reasoning. Concerning logic programming, it has been also shown in [172] that different semantics of the abstract framework *e.g.* grounded extensions and stable extensions correspond to the well-founded and answer set semantics of normal logic programs.

Unfortunately, the structure and meaning of arguments and attacks are abstract in AF. On the one hand, these characteristics enable the study of properties which are

[1]With a little abuse of notation, we define $(S, b) \in \mathcal{R}$ as $\exists a \in S : (a, b) \in \mathcal{R}$. Similarly, we define $(b, S) \in \mathcal{R}$ as $\exists a \in S : (b, a) \in \mathcal{R}$.

independent of any specific aspects [208]. On the other hand, this generality features a limited expressivity and can be hardly adopted to model practical target situations. To fill out this gap, less abstract formalisms were considered, dealing in particular with the construction of arguments and the conditions for an argument to attack another, such as ASPIC$^+$ [209], DeLP [180], and assumption-based argumentation (ABA) [194].

Research on the relation between logic programming and argumentation has been fruitful and compensated for each other, *i.e.* some argumentation formalisms were used to define semantics for logic programming, and also, logic programming was used to provide an underlying representational language for non-abstract argumentation formalisms (similar to the mentioned above). In the following, we give a brief description of DeLP.

DeLP (Defeasible Logic Programming) is a formalism that combines techniques of both logic programming and defeasible argumentation. The representational language of DeLP is defined as an extension of a logic programming language that considers two types of rules *viz.* strict and defeasible rules. A DeLP program is denoted by a pair (Π, Δ) where Π is a set of strict rules (representing facts and non-defeasible knowledge) and Δ is a set of defeasible rules. Rules in a DeLP program must be ground.

Defeasible rules allow to infer tentative conclusions. A *defeasible derivation* of a literal Q from a DeLP program (Π, Δ) (denoted by $(\Pi, \Delta) \vdash Q$) is a finite sequence of ground literals $L_1, L_2, \ldots, L_n = Q$ where either:

1. L_i is a fact in Π; or
2. There exists a rule R_i in (Π, Δ) with head L_i and body $B_1, B_2, \ldots, B_k$ and every literal of the body is an element L_j of the sequence appearing before $L_i (j < i)$.

Unlike ASPIC$^+$ and ABA, DeLP considers the notion of arguments' structure other than proof trees as follows.

Definition C.2. Let H be a ground literal, (Π, Δ) a DeLP program, and $\mathcal{A} \subseteq \Delta$. The pair $\langle \mathcal{A}, H \rangle$ is an argument structure if:

1. There exists a defeasible derivation for H from $(\Pi, \mathcal{A})$;
2. There is no defeasible derivation from $(\Pi, \mathcal{A})$ of contradictory literals; and
3. There is no proper subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}'$ satisfies (1) and (2).

A DeLP query is a ground literal that DeLP will try to warrant, *i.e.* evaluate its acceptability. Basically, a query will succeed if it is possible to build an argument that supports the query and this argument is also found to be undefeated by a warrant procedure. This process implements an exhaustive dialectical analysis that involves the construction and evaluation of arguments. The DeLP dialectical analysis is formally described in [180].

Bibliography

[1] N. Guarino and P. Giaretta, "Ontologies and knowledge bases: towards a terminological clarification," in *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, pp. 25–32, IOS Press, 1995.

[2] N. Guarino, D. Oberle, and S. Staab, *What Is an ontology?*, pp. 1–17. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[3] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, pp. 199–220, June 1993.

[4] W. N. Borst, *Construction of engineering ontologies for knowledge sharing and reuse*. Universiteit Twente, 1997.

[5] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: principles and methods," *Data & Knowledge Engineering*, vol. 25, pp. 161–197, Mar. 1998.

[6] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds., *The description logic handbook: theory, implementation, and applications*. New York, NY, USA: Cambridge University Press, 2007.

[7] F. Baader and U. Sattler, "An overview of tableau algorithms for description logics," *Studia Logica*, vol. 69, no. 1, pp. 5–40, 2001.

[8] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi, "Reasoning in expressive description logics.," *Handbook of Automated Reasoning*, vol. 2, pp. 1581–1634, 2001.

[9] F. Baader, I. Horrocks, and U. Sattler, *Description logics*, pp. 3–28. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[10] D. Fensel, F. Van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider, "OIL: an ontology infrastructure for the semantic web," *IEEE intelligent systems*, vol. 16, no. 2, pp. 38–45, 2001.

[11] I. Horrocks and P. F. Patel-Schneider, "The generation of DAML+OIL.," in *Description logics*, 2001.

[12] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen, "Reviewing the design of DAML+OIL: an ontology language for the semantic web," *AAAI/IAAI*, vol. 2002, pp. 792–797, 2002.

[13] W3C OWL Working Group, "OWL Web ontology language semantics and abstract syntax," 2004.

[14] W3C OWL Working Group, "OWL 2 Web ontology language document overview (second edition)," 2015.

[15] F. Van Harmelen, V. Lifschitz, and B. Porter, *Handbook of knowledge representation*, vol. 1. Elsevier, 2008.

[16] M. R. Quillian, "Word concepts: a theory and simulation of some basic semantic capabilities," *Systems Research and Behavioral Science*, vol. 12, no. 5, pp. 410–430, 1967.

[17] M. Minsky, "A framework for representing knowledge," 1974.

[18] W. Woods, "What's in a link: foundations for semantic networks, representation and understanding: studies in cognitive science, bob row, d. g., and collins, a," 1975.

[19] R. J. Brachman, "What's in a concept: structural foundations for semantic networks," *International Journal of Man-machine Studies*, vol. 9, no. 2, pp. 127–152, 1977.

[20] P. J. Hayes, "In defence of logic," in *Proc. IJCAI-77*, pp. 559–565, 1977.

[21] P. J. Hayes, "The logic of frames," *Frame Conceptions and Text Understanding*, vol. 46, p. 61, 1979.

[22] R. J. Brachman, "Structured inheritance networks," *Research in Natural Language Understanding, Quarterly Progress Report*, vol. 1, pp. 36–78, 1978.

[23] R. J. Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system," *Cognitive science*, vol. 9, no. 2, pp. 171–216, 1985.

[24] E. Mays, R. Dionne, and R. Weida, "K-Rep system overview," *ACM SIGART Bulletin*, vol. 2, no. 3, pp. 93–97, 1991.

[25] R. J. Brachman, R. E. Fikes, and H. J. Levesque, "Krypton: a functional approach to knowledge representation," *Computer;(United States)*, vol. 10, 1983.

[26] R. J. Brachman and H. J. Levesque, *Readings in knowledge representation*. Morgan Kaufmann Publishers Inc., 1985.

[27] B. Nebel, "Terminological reasoning is inherently intractable," *Artificial Intelligence*, vol. 43, no. 2, pp. 235–249, 1990.

[28] B. Parsia, E. Sivrin, M. Grove, and R. Alford, "Pellet owl reasoner. Maryland Information and Networks Dynamics Lab," 2003.

[29] R. J. Brachman, ""Reducing" CLASSIC to practice: knowledge representation theory meets reality.," in *KR*, pp. 247–258, 1992.

[30] A. S. Sidhu, T. S. Dillon, E. Chang, and B. S. Sidhu, "Protein ontology development using OWL.," in *OWLED*, vol. 188, 2005.

[31] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt, "The complexity of concept languages," *Information and Computation*, vol. 134, no. 1, pp. 1–58, 1997.

[32] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt, "Tractable concept languages.," in *IJCAI*, vol. 91, pp. 458–463, 1991.

[33] F. M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. M. Spaccamela, "The complexity of existential quantification in concept languages," *Artificial Intelligence*, vol. 53, no. 2-3, pp. 309–327, 1992.

[34] K. Schild, "Terminological cycles and the propositional μ-calculus," in *KR*, 1994.

[35] I. Horrocks and U. Sattler, "A description logic with transitive and inverse roles and role hierarchies.," *Journal of Logic and Computation*, vol. 9, no. 3, pp. 385–410, 1999.

[36] I. Horrocks, U. Sattler, and S. Tobies, "Practical reasoning for expressive description logics," in *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pp. 161–180, Springer, 1999.

[37] G. De Giacomo and M. Lenzerini, "Boosting the correspondence between description logics and propositional dynamic logics," in *AAAI*, vol. 94, pp. 205–212, 1994.

[38] G. De Giacomo and M. Lenzerini, "Concept language with number restrictions and fixpoints, and its relationship with mu-calculus," in *ECAI*, vol. 94, pp. 411–415, PITMAN, 1994.

[39] G. De Giacomo, "Decidability of class-based knowledge representation formalisms," 1995.

[40] G. De Giacomo and M. Lenzerini, "TBox and ABox reasoning in expressive description logics.," *KR*, vol. 96, no. 316-327, p. 10, 1996.

[41] I. Horrocks, "Using an expressive description logic: FaCT or fiction?," *KR*, vol. 98, pp. 636–645, 1998.

[42] V. Haarslev and R. Müller, "RACER system description," *Automated Reasoning*, pp. 701–705, 2001.

[43] S. Schlobach, R. Cornet, *et al.*, "Non-standard reasoning services for the debugging of description logic terminologies," in *IJCAI*, vol. 3, pp. 355–362, 2003.

[44] A. Borgida, "On the relative expressiveness of description logics and predicate logics," *Artificial intelligence*, vol. 82, no. 1-2, pp. 353–367, 1996.

[45] L. Pacholski, W. Szwast, and L. Tendera, "Complexity of two-variable logic with counting," in *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, pp. 318–327, IEEE, 1997.

[46] E. Grädel, P. G. Kolaitis, and M. Y. Vardi, "On the decision problem for two-variable first-order logic," *Bulletin of symbolic logic*, vol. 3, no. 01, pp. 53–69, 1997.

[47] E. Grädel, "Guarded fragments of first-order logic: a perspective for new description logics?," in *In Proc. of 1998 Int. Workshop on Description Logics DL98, Trento, CEUR Electronic Workshop Proceedings*, 1998.

[48] E. Grädel, "On the restraining power of guards," *The Journal of Symbolic Logic*, vol. 64, no. 04, pp. 1719–1742, 1999.

[49] M. Buchheit, F. M. Donini, W. Nutt, and A. Schaerf, "A refined architecture for terminological systems: terminology = schema + views," *Artificial Intelligence*, vol. 99, no. 2, pp. 209–260, 1998.

[50] D. Calvanese, G. De Giacomo, and M. Lenzerini, "On the decidability of query containment under constraints," in *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 149–158, ACM, 1998.

[51] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, "Description logic framework for information integration," in *KR*, pp. 2–13, 1998.

[52] D. Tsarkov and I. Horrocks, FaCT^{++} *Description logic reasoner: system description*, pp. 292–297. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[53] U. Hustadt and R. A. Schmidt, "On the relation of resolution and tableaux proof systems for description logics," in *Automated Reasoning Workshop: Bridging the Gap between Theory and Practice*, 1999.

[54] U. Hustadt and R. A. Schmidt, "Issues of decidability for description logics in the framework of resolution," in *Automated Deduction in Classical and Non-Classical Logics*, pp. 191–205, Springer, 2000.

[55] C. Areces, M. de Rijke, and H. de Nivelle, "Resolution in modal, description and hybrid logic," *Journal of Logic and Computation*, vol. 11, no. 5, pp. 717–736, 2001.

[56] U. Hustadt, B. Motik, and U. Sattler, "Reducing $\mathcal{SHOIQ}$-description logic to disjunctive datalog programs.," *KR*, vol. 2004, pp. 152–162, 2004.

[57] N. Kurtonina and M. De Rijke, "Expressiveness of concept expressions in first-order description logics," *Artificial Intelligence*, vol. 107, no. 2, pp. 303–333, 1999.

[58] C. Lutz and U. Sattler, "Mary likes all cats," in *Proceedings of the 2000 International Workshop in Description Logics (DL2000), number 33 in CEUR-WS*, Citeseer, 2000.

[59] S. Tobies, "Complexity results and practical algorithms for logics in knowledge representation," *arXiv preprint cs/0106031*, 2001.

[60] F. Baader and S. Tobies, "The inverse method implements the automata approach for modal satisfiability," in *International Joint Conference on Automated Reasoning*, pp. 92–106, Springer, 2001.

[61] F. Baader, J. Hladik, C. Lutz, and F. Wolter, "From tableaux to automata for description logics," *Fundamenta Informaticae*, vol. 57, no. 2-4, pp. 247–279, 2003.

[62] I. Horrocks, O. Kutz, and U. Sattler, "The even more irresistible sroiq.," *Kr*, vol. 6, pp. 57–67, 2006.

[63] F. Baader, S. Brandt, and C. Lutz, "Pushing the $\mathcal{EL}$ envelope," in *IJCAI*, vol. 5, pp. 364–369, 2005.

[64] F. Baader, S. Brandt, and C. Lutz, "Pushing the el envelope further," 2008.

[65] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "DL-Lite: Tractable description logics for ontologies," in *AAAI*, vol. 5, pp. 602–607, 2005.

[66] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "Tractable reasoning and efficient query answering in description logics: The dl-lite family," *Journal of Automated reasoning*, vol. 39, no. 3, pp. 385–429, 2007.

[67] D. L. McGuinness and A. Borgida, "Explaining subsumption in description logics," in *IJCAI (1)*, pp. 816–821, 1995.

[68] T. Liebig and M. Halfmann, "Explaining subsumption in $\mathcal{ALEH}_{\mathcal{R}^+}$ tboxes," *Proc. of DL05*, pp. 144–151, 2005.

[69] A. Rademaker, *A Proof Theory for Description Logics*. Springer Science & Business Media, 2012.

[70] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati, "QUONTO: querying ontologies," in *AAAI*, vol. 5, pp. 1670–1671, 2005.

[71] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler, "Conjunctive query answering for the description logic," *Journal of Artificial Intelligence Research*, vol. 31, pp. 157–204, 2008.

[72] C. Lutz, "The complexity of conjunctive query answering in expressive description logics," *Automated Reasoning*, pp. 179–193, 2008.

[73] M. Ortiz, D. Calvanese, and T. Eiter, "Data complexity of query answering in expressive description logics via tableaux," *Journal of Automated Reasoning*, vol. 41, no. 1, pp. 61–98, 2008.

[74] B. Parsia, E. Sirin, and A. Kalyanpur, "Debugging OWL ontologies," in *Proceedings of the 14th international conference on World Wide Web*, pp. 633–640, ACM, 2005.

[75] F. Baader, R. Penaloza, and B. Suntisrivaraporn, "Pinpointing in the description logic $\mathcal{EL}^+$," *Advances in Artificial Intelligence*, p. 52, 2007.

[76] F. Baader and B. Suntisrivaraporn, "Debugging SNOMED CT using axiom pinpointing in the description logic el," *KR-MED 2008*, p. 1, 2008.

[77] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler, "A logical framework for modularity of ontologies.," in *IJCAI*, vol. 2007, pp. 298–303, 2007.

[78] B. Konev, C. Lutz, D. Walther, and F. Wolter, "Semantic modularity and module extraction in description logics.," in *ECAI*, pp. 55–59, 2008.

[79] B. Suntisrivaraporn, "Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies," *The Semantic Web: Research and Applications*, pp. 230–244, 2008.

[80] M. Sheremet, D. Tishkovsky, F. Wolter, and M. Zakharyaschev, "A logic for concepts and similarity," *Journal of Logic and Computation*, vol. 17, no. 3, pp. 415–452, 2007.

[81] M. Sheremet, D. Tishkovsky, F. Wolter, and M. Zakharyaschev, "Comparative similarity, tree automata, and diophantine equations," in *Logic for Programming, Artificial Intelligence, and Reasoning* (G. Sutcliffe and A. Voronkov, eds.), (Berlin, Heidelberg), pp. 651–665, Springer Berlin Heidelberg, 2005.

[82] A. Tversky, "Features of similarity," *Psychological Review*, vol. 84, no. 4, pp. 327–352, 1977.

[83] J. H. LEE, M. H. KIM, and Y. J. LEE, "Information retrieval based on conceptual distance in IS-A hierarchies," *Journal of Documentation*, vol. 49, no. 2, pp. 188–207, 1993.

[84] R. Rada, H. Mili, E. Bicknell, and M. Blettner, "Development and application of a metric on semantic nets," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, pp. 17–30, Jan 1989.

[85] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[86] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[87] C. D'Amato, S. Staab, and N. Fanizzi, "On the influence of description logics ontologies on conceptual similarity," *In Proceedings of Knowledge Engineering: Practice and Patterns*, pp. 48–63, 2008.

[88] K. Lehmann and A.-Y. Turhan, "A framework for semantic-based similarity measures for $\mathcal{ELH}$-concepts.," in *JELIA* (L. F. del Cerro, A. Herzig, and J. Mengin, eds.), vol. 7519 of *Lecture Notes in Computer Science*, pp. 307–319, Springer, 2012.

[89] T. Racharak and B. Suntisrivaraporn, "Similarity measures for $\mathcal{FL}_0$ concept descriptions from an automata-theoretic point of view," *In Proceedings of the 6th Annual International Conference on Information and Communication Technology for Embedded Systems (ICICTES 2015)*, 2015.

[90] T. Racharak and S. Tojo, "Concept similarity under the agent?s preferences for the description logic $\mathcal{FL}_0$ with unfoldable tbox," in *Proceedings of the 10th International Conference on Agents and Artificial Intelligence*, pp. 201–210, INSTICC, SciTePress, 2018.

[91] T. Racharak, B. Suntisrivaraporn, and S. Tojo, "Identifying an agent's preferences toward similarity measures in description logics," in *Semantic Technology: the 5th Joint International Conference, JIST 2015, Yichang, China, November 11-13, 2015, Revised Selected Papers*, pp. 201–208, Springer International Publishing, 2016.

[92] T. Racharak, B. Suntisrivaraporn, and S. Tojo, "sim^{π}: a concept similarity measure under an agent's preferences in description logic $\mathcal{ELH}$," in *Proceedings of the 8th International Conference on Agents and Artificial Intelligence*, pp. 480–487, 2016.

[93] T. Racharak, B. Suntisrivaraporn, and S. Tojo, "Personalizing a concept similarity measure in the description logic $\mathcal{ELH}$ with preference profile," *Journal of Computing and Informatics (accepted on July 11; to appear)*, 2017.

[94] T. Racharak and S. Tojo, "Tuning agent's profile for similarity measure in description logic $\mathcal{ELH}$," in *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017.*, pp. 287–298, 2017.

[95] T. Racharak, S. Tojo, N. D. Hung, and P. Boonkwan, *Combining answer set programming with description logics for analogical reasoning under an agent's preferences*, pp. 306–316. Cham: Springer International Publishing, 2017.

[96] T. Racharak, S. Tojo, N. D. Hung, and P. Boonkwan, *Argument-based logic programming for analogical reasoning*, pp. 253–269. Cham: Springer International Publishing, 2017.

[97] D. Van Dalen, *Logic and structure*. Springer, 1997.

[98] K. E. Sabri, "Semantic tableau proof system for first-order logic," 2009.

[99] M. Schmidt-Schauß and G. Smolka, "Attributive concept descriptions with complements," *Artificial Intelligence*, vol. 48, no. 1, pp. 1–26, 1991.

[100] R. J. Brachman and H. J. Levesque, "The tractability of subsumption in frame-based description languages," in *AAAI*, vol. 84, pp. 34–37, 1984.

[101] F. Baader, R. Küsters, and R. Molitor, "Computing least common subsumers in description logics with existential restrictions," in *IJCAI*, vol. 99, pp. 96–101, 1999.

[102] F. Baader, "Terminological cycles in a description logic with existential restrictions," in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pp. 325–330, Morgan Kaufmann Publishers Inc., 2003.

[103] K. Schild, "A correspondence theory for terminological logics: preliminary report," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, (San Francisco, CA, USA), pp. 466–471, Morgan Kaufmann Publishers Inc., 1991.

[104] S. Brandt, "Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and – what else?," in *Proceedings of the 16th European Conference on Artificial Intelligence*, pp. 298–302, IOS Press, 2004.

[105] M. Q. Stearns, C. Price, K. A. Spackman, and A. Y. Wang, "SNOMED clinical terms: overview of the development process and project status.," in *Proceedings of the AMIA Symposium*, (College of American Pathologists, Northfield, IL, USA.), pp. 662–666, 2001.

[106] K. A. Spackman, "Rates of change in a large clinical terminology: three years experience with SNOMED clinical terms.," in *AMIA*, 2005.

[107] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene Ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, pp. 25–29, May 2000.

[108] B. Nebel, "Terminological cycles: semantics and computational properties," 1991.

[109] M. Buchheit, F. M. Donini, and A. Schaerf, "Decidable reasoning in terminological knowledge representation systems," *Journal of Artificial Intelligence Research*, vol. 1, pp. 109–138, Dec. 1993.

[110] F. M. Donini, M. lenzerini, D. Nardi, and A. Schaerf, "Principles of knowledge representation," ch. Reasoning in description logics, pp. 191–236, Stanford, CA, USA: Center for the Study of Language and Information, 1996.

[111] M. Mortimer, "On languages with two variables," *Mathematical Logic Quarterly*, vol. 21, no. 1, pp. 135–140, 1975.

[112] H. Andréka, I. Németi, and J. van Benthem, "Modal languages and bounded fragments of predicate logic," *Journal of Philosophical Logic*, vol. 27, no. 3, pp. 217–274, 1998.

[113] W. V. Quine, "Variables explained away," *Proceedings of the american philosophical society*, vol. 104, no. 3, pp. 343–347, 1960.

[114] W. Quine, "Algebraic logic and predicate functors," 1971.

[115] S. Y. Maslov, "The inverse method for establishing deducibility for logical calculi," *Trudy Matematicheskogo Instituta imeni VA Steklova*, vol. 98, pp. 26–87, 1968.

[116] U. Hustadt, R. A. Schmidt, and L. Georgieva, "A survey of decidable first-order fragments and description logics," *Journal of Relational Methods in Computer Science*, vol. 1, no. 251-276, p. 3, 2004.

[117] A. Church, "A note on the entscheidungsproblem," *The journal of symbolic logic*, vol. 1, no. 1, pp. 40–41, 1936.

[118] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937.

[119] L. Henkin, "Logical systems containing only a finite number of symbols," 1967.

[120] D. Scott, "A decision method for validity of sentences in two variables," *Journal of Symbolic Logic*, vol. 27, no. 377, p. 74, 1962.

[121] E. Graedel, M. Otto, and E. Rosen, "Two-variable logic with counting is decidable," in *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, LICS '97, (Washington, DC, USA), pp. 306–, IEEE Computer Society, 1997.

[122] A. Schaerf, "Reasoning with individuals in concept languages," *Data & Knowledge Engineering*, vol. 13, no. 2, pp. 141 – 176, 1994.

[123] B. Nebel, *Reasoning and revision in hybrid representation systems*. New York, NY, USA: Springer-Verlag New York, Inc., 1990.

[124] F. Baader, "Using automata theory for characterizing the semantics of terminological cycles," *Annals of Mathematics and Artificial Intelligence*, vol. 18, no. 2, pp. 175–219, 1996.

[125] F. Baader, S. Brandt, and R. Küsters, "Matching under side conditions in description logics," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'01, pp. 213–218, Morgan Kaufmann Publishers Inc., 2001.

[126] J. Euzenat and P. Valtchev, "Similarity-based ontology alignment in OWL-Lite," in *Proceedings of the 16th European Conference on Artificial Intelligence*, ECAI'04, (Amsterdam, The Netherlands, The Netherlands), pp. 323–327, IOS Press, 2004.

[127] J. E. Caviedes and J. J. Cimino, "Towards the development of a conceptual distance metric for the UMLS," *Journal of Biomedical Informatics*, vol. 37, pp. 77–85, Apr. 2004.

[128] J. Ge and Y. Qiu, "Concept Similarity Matching Based on Semantic Distance," *2008 Fourth International Conference on Semantics, Knowledge and Grid*, pp. 380–383, Dec. 2008.

[129] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, (Stroudsburg, PA, USA), pp. 133–138, Association for Computational Linguistics, 1994.

[130] C. Leacock and M. Chodorow, "Combining local context and wordnet similarity for word sense identification," in *MIT Press* (C. Fellfaum, ed.), (Cambridge, Massachusetts), pp. 265–283, 1998.

[131] F. Giunchiglia, M. Yatskevich, and P. Shvaiko, "Journal on data semantics ix," ch. Semantic Matching: Algorithms and Implementation, pp. 1–38, Berlin, Heidelberg: Springer-Verlag, 2007.

[132] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, (San Francisco, CA, USA), pp. 448–453, Morgan Kaufmann Publishers Inc., 1995.

[133] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *CoRR*, 1997.

[134] D. Lin, "An information-theoretic definition of similarity," in *Proceedings of the 15th International Conference on Machine Learning*, ICML'98, (San Francisco, CA, USA), pp. 296–304, Morgan Kaufmann Publishers Inc., 1998.

[135] T. Pedersen, S. V. Pakhomov, S. Patwardhan, and C. G. Chute, "Measures of semantic similarity and relatedness in the biomedical domain," *Journal of Biomedical Informatics*, vol. 40, no. 3, pp. 288 – 299, 2007.

[136] S. Patwardhan, "Using wordnet-based context vectors to estimate the semantic relatedness of concepts," in *Proceedings of the EACL 2006 Workshop Making Sense of Sense-bringing Computational Linguistics and Psycholinguistics Together*, vol. 1501, pp. 1–8, 2006.

[137] H. Schütze, "Automatic word sense discrimination," *Computational Linguistics*, vol. 24, no. 1, pp. 97–123, 1998.

[138] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of The American Society For Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[139] K. H. Lai, M. Topaz, F. R. Goss, and L. Zhou, "Automated misspelling detection and correction in clinical free-text records," *Journal of biomedical informatics*, vol. 55, pp. 188–195, 2015.

[140] R. Gabrys, E. Yaakobi, and O. Milenkovic, "Codes in the damerau distance for dna storage," in *Information Theory (ISIT), 2016 IEEE International Symposium on*, pp. 2644–2648, IEEE, 2016.

[141] P. Bille, "A survey on tree edit distance and related problems," *Theoretical computer science*, vol. 337, no. 1-3, pp. 217–239, 2005.

[142] M. Collins and N. Duffy, "Convolution kernels for natural language," in *Advances in neural information processing systems*, pp. 625–632, 2002.

[143] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpeset des jura," *Bulletin de la Societe Vaudoise des Sciences Naturellese*, vol. 37, pp. 547–579, 1901.

[144] T. Alsubait, B. Parsia, and U. Sattler, "Measuring conceptual similarity in ontologies: how bad is a cheap measure?," in *Informal Proceedings of the 27th International Workshop on Description Logics, Vienna, Austria, July 17-20, 2014.*, pp. 365–377, 2014.

[145] B. Suntisrivaraporn, "A similarity measure for the description logic $\mathcal{EL}$ with unfoldable terminologies," *In INCoS*, pp. 408–413, 2013.

[146] K. Janowicz and M. Wilkes, "SIM-$\mathrm{DL_A}$: A novel semantic similarity measure for description logics reducing inter-concept to inter-instance similarity," *In Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, pp. 353–367, 2009.

[147] C. D'Amato, N. Fanizzi, and F. Esposito, "A dissimilarity measure for $\mathcal{ALC}$ concept descriptions," *In Proceedings of the 2006 ACM Symposium on Applied Computing*, pp. 1695–1699, 2006.

[148] N. Fanizzi and C. D'Amato, "A similarity measure for the $\mathcal{ALN}$ description logic," *In Proceedings of CILC 2006 - Italian Conference on Computational Logic*, pp. 26–27.

[149] S. Tongphu and B. Suntisrivaraporn, "On desirable properties of the structural subsumption-based similarity measure," *In Lecture Notes in Computer Science (LNCS)*, vol. 8943, pp. 19–32, 2015.

[150] C. D'Amato, N. Fanizzi, and F. Esposito, "A semantic similarity measure for expressive description logics," *In CoRR*, vol. abs/0911.5043, 2009.

[151] S. Tongphu and B. Suntisrivaraporn, "Algorithms for measuring similarity between $\mathcal{ELH}$ concept descriptions: a case study on SNOMED CT," *Journal of Computing and Informatics (accepted on May 7; to appear)*, 2015.

[152] A. Borgida, T. J. Walsh, and H. Hirsh, "Towards measuring similarity in description logics," in *Working Notes of the International Description Logics Workshop, volume 147 of CEUR Workshop Proceedings*, 2005.

[153] K. Janowicz, "Sim-DL: towards a semantic similarity measurement theory for the description logic $\mathcal{ALCNR}$ in geographic information retrieval," in *SeBGIS 2006, OTM Workshops 2006. Volume 4278 of Lecture Notes in Computer Science*, Springer, 2006.

[154] A. Bernstein, E. Kaufmann, C. Bürki, and M. Klein, *How Similar Is It? Towards Personalized Similarity Measures in Ontologies*, pp. 1347–1366. Heidelberg: Physica-Verlag HD, 2005.

[155] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[156] S. Schulz, B. Suntisrivaraporn, and F. Baader, "Snomed ct's problem list: ontologists' and logicians' therapy suggestions," *Studies in health technology and informatics*, vol. 129, no. 1, p. 802, 2007.

[157] F. Dhombres and O. Bodenreider, "Interoperability between phenotypes in research and healthcare terminologies?investigating partial mappings between hpo and SNOMED CT," *Journal of biomedical semantics*, vol. 7, no. 1, p. 3, 2016.

[158] S. Raha, A. Hossain, and S. Ghosh, "Similarity based approximate reasoning: fuzzy control," *Journal of Applied Logic*, vol. 6, no. 1, pp. 47 – 71, 2008.

[159] M. I. Sessa, "Approximate reasoning by similarity-based SLD resolution," *Theoretical Computer Science*, vol. 275, no. 1?2, pp. 389 – 426, 2002.

[160] P. Bartha, "By parallel reasoning: The construction and evaluation of analogical arguments," pp. 1–384, 01 2010.

[161] I. M. Copi, C. Cohen, and K. McMahon, *Introduction to logic*. Routledge, 2016.

[162] T. R. Davies, "Determination, uniformity, and relevance: Normative criteria for generalization and reasoning by analogy," in *Analogical reasoning*, pp. 227–250, Springer, 1988.

[163] M. Guarini, A. Butchart, P. S. Smith, and A. Moldovan, "Resources for research on analogy: A multi-disciplinary guide," *Informal Logic*, vol. 29.

[164] D. Walton, C. Reed, and F. Macagno, *Argumentation Schemes*. Cambridge University Press, 2008.

[165] D. Walton, "Similarity, precedent and argument from analogy," *Artificial Intelligence and Law*, vol. 18, no. 3, pp. 217–246, 2010.

[166] R. Sun, "Robust reasoning: Integrating rule-based and similarity-based reasoning," *Artificial Intelligence*, vol. 75, no. 2, pp. 241–295, 1995.

[167] M. S. Lief, H. M. Caldwell, and B. Bycel, *Ladies And Gentlemen Of The Jury: Greatest Closing Arguments In Modern Law*. Scribner, 2000.

[168] D. Walton, "Argumentation schemes for argument from analogy," *Systematic Approaches to Argument by Analogy*, pp. 23–40, 2014.

[169] M. Gelfond and V. Lifschitz, "The stable model semantics for logic programming.," in *ICLP/SLP*, vol. 88, pp. 1070–1080, 1988.

[170] V. W. Marek and M. Truszczyński, "Stable models and an alternative logic programming paradigm," in *The Logic Programming Paradigm: A 25-Year Perspective*, pp. 375–398, Springer Berlin Heidelberg, 1999.

[171] I. Niemelä, "Logic programs with stable model semantics as a constraint programming paradigm," *Annals of Mathematics and Artificial Intelligence*, vol. 25, pp. 241–273, Feb. 1999.

[172] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial intelligence*, vol. 77, no. 2, pp. 321–357, 1995.

[173] M. Balduccini, M. Gelfond, R. Watson, and M. Nogueira, "The usa-advisor: A case study in answer set planning," in *International Conference on Logic Programming and Nonmonotonic Reasoning*, pp. 439–442, Springer, 2001.

[174] C. Baral, "Knowledge representation, reasoning and declarative problem solving with answer sets," 2006.

[175] C. Baral, A. Provetti, and T. Son, "Theory and practice of logic programming special issue on answer set programming, vol. 3," 2003.

[176] B. Chisham, E. Pontelli, T. C. Son, and B. Wright, "Cdaostore: A phylogenetic repository using logic programming and web services," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 11, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011.

[177] M. Gebser, C. Guziolowski, M. Ivanchev, T. Schaub, A. Siegel, S. Thiele, and P. Veber, "Repair and prediction (under inconsistency) in large biological networks with answer set programming.," in *KR*, 2010.

[178] V. Lifschitz, "Answer set programming and plan generation," *Artificial Intelligence*, vol. 138, no. 1-2, pp. 39–54, 2002.

[179] Potsdam answer set solving collection. `http://potassco.sourceforge.net/`. [Online; accessed 13-December-2016].

[180] A. J. García and G. R. Simari, "Defeasible logic programming: An argumentative approach," *Journal of Theory and Practice of Logic Programming*, vol. 4, pp. 95–138, Jan. 2004.

[181] D. Nute, "Handbook of logic in artificial intelligence and logic programming (vol. 3)," ch. Defeasible Logic, pp. 353–395, New York, NY, USA: Oxford University Press, Inc., 1994.

[182] G. Antoniou, D. Billington, G. Governatori, and M. J. Maher, "Representation results for defeasible logic," *ACM Trans. Comput. Logic*, vol. 2, pp. 255–287, Apr. 2001.

[183] L. Amgoud and C. Cayrol, "On the acceptability of arguments in preference-based argumentation," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, UAI'98, (San Francisco, CA, USA), pp. 1–7, Morgan Kaufmann Publishers Inc., 1998.

[184] M. Elvang-Gøransson, P. J. Krause, and J. Fox, *Acceptability of arguments as 'logical uncertainty'*, pp. 85–90. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993.

[185] B. N. Waller, "Classifying and analyzing analogies," *Informal Logic*, vol. 21, no. 3, 2001.

[186] H. Prakken and G. Sartor, "Argument-based extended logic programming with defeasible priorities," *Journal of Applied Non-classical Logics*, vol. 7, pp. 25–75, 1997.

[187] G. Damele, *Analogia Legis and Analogia Iuris: An Overview from a Rhetorical Perspective*, pp. 243–256. Cham: Springer International Publishing, 2014.

[188] K. Ashley, *Case-based reasoning*, pp. 23–60. Dordrecht: Springer Netherlands, 2006.

[189] V. Aleven, "Teaching case-based argumentation through a model and examples," 1997.

[190] H. Prakken, A. Z. Wyner, T. J. M. Bench-Capon, and K. Atkinson, "A formalization of argumentation schemes for legal case-based reasoning in ASPIC+," *J. Log. Comput.*, vol. 25, no. 5, pp. 1141–1166, 2015.

[191] R. Goebel, *A sketch of analogy as reasoning with equality hypotheses*, pp. 243–253. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989.

[192] R. Greiner, *Learning by Understanding Analogies*, pp. 81–84. Boston, MA: Springer US, 1986.

[193] P. H. Winston, "Learning and reasoning by analogy," *Commun. ACM*, vol. 23, pp. 689–703, Dec. 1980.

[194] P. M. Dung, R. A. Kowalski, and F. Toni, *Assumption-based argumentation*, pp. 199–218. Boston, MA: Springer US, 2009.

[195] K. A. Spackman, K. E. Campbell, and R. A. Côté, "Snomed rt: a reference terminology for health care.," in *Proceedings of the AMIA annual fall symposium*, p. 640, American Medical Informatics Association, 1997.

[196] A. Rector, "Medical informatics," in *The description logic handbook*, pp. 406–426, Cambridge University Press, 2003.

[197] M. O'Neil, C. Payne, and J. Read, "Read codes version 3: a user led terminology.," *Methods of information in medicine*, vol. 34, no. 1-2, pp. 187–192, 1995.

[198] P. F. Patel-Schneider and B. Swartout, "Description-logic knowledge representation system specification from the krss group of the arpa knowledge sharing effort," *KRSS group of the ARPA*, 1993.

[199] K. A. Spackman, R. Dionne, E. Mays, and J. Weis, "Role grouping as an extension to the description logic of ontylog, motivated by concept modeling in SNOMED.," in *Proceedings of the AMIA Symposium*, p. 712, American Medical Informatics Association, 2002.

[200] A. C. Hastings, "A reformulation of the modes of reasoning in argumentation," 1963.

[201] C. Perelman and L. Olbrechts-Tyteca, "The new rhetoric, notre dame," *IN: University of Notre Dame*, 1969.

[202] M. Kienpointner, "Alltagslogik struktur und funktion von argumentationsmustern," 1992.

[203] D. Walton, "Argumentation schemes for presumptive reasoning lawrence erlbaum associates mahwah," 1996.

[204] W. Grennan, *Informal logic*. Kingston: McGill-Queen?s University Press, 1997.

[205] C. Reed and T. Norman, *Argumentation machines: New frontiers in argument and computation*, vol. 9. Springer Science & Business Media, 2003.

[206] B. Verheij, "Dialectical argumentation with argumentation schemes: An approach to legal logic," *Artificial intelligence and Law*, vol. 11, no. 2, pp. 167–195, 2003.

[207] N. Guallart, "Analogical reasoning in clinical practice," *Systematic Approaches to Argument by Analogy*, pp. 257–273, 2014.

[208] P. Baroni and M. Giacomin, *Semantics of abstract argument systems*, pp. 25–44. Boston, MA: Springer US, 2009.

[209] S. Modgil and H. Prakken, "The ASPIC$^+$ framework for structured argumentation: A tutorial," *Argument and Computation*, vol. 5, no. 1, pp. 31–62, 2014.

Publications

International Journal

[1] Teeradaj Racharak, Boontawee Suntisrivaraporn, and Satoshi Tojo, **Personalizing a concept similarity measure in the description logic $\mathcal{ELH}$ with preference profile**, Computing and Informatics (accepted).

Proceeding International Conferences

[2] Teeradaj Racharak and Satoshi Tojo, **Concept similarity under the agent's preference for the description logic $\mathcal{FL}_0$ with unfoldable TBox**, in Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART), Madeira, Portugal, pp. 201-210, 2018.

[3] Teeradaj Racharak and Satoshi Tojo, **Tuning agent's profile for similarity measure in description logic $\mathcal{ELH}$**, in Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART), Porto, Portugal, pp. 287-298, 2017.

[4] Teeradaj Racharak, Satoshi Tojo, Nguyen Duy Hung, Prachya Boonkwan, **Combining answer set programming with description logics for analogical reasoning under an agent's preferences**, in Proceedings of International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE), Arras, France, pp. 306-316, 2017.

[5] Teeradaj Racharak, Boontawee Suntisrivaraporn, and Satoshi Tojo, **sim^{π}: a concept similarity measure under an agent's preferences in description logic $\mathcal{ELH}$**, in Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART), Rome, Italy, pp. 480-487, 2016.

[6] Teeradaj Racharak, Boontawee Suntisrivaraporn, and Satoshi Tojo, **Identifying an agent's preferences toward similarity measures in description logics**, in Proceedings of the 5th Joint International Semantic Technology Conference (JIST), Yichang, China, pp. 201-208, 2015.

Peer-reviewed International Workshop (Post-proceeding in Parenthesis)

[7] Teeradaj Racharak, Satoshi Tojo, Nguyen Duy Hung, Prachya Boonkwan, **Argument-based logic programming for analogical reasoning**, In Proceedings of the 10th International Workshop on Juris-Informatics (JURISIN), Kanagawa, Japan, 2016 (Kurahashi S., Ohta Y., Arai S., Satoh K., Bekki D. (eds) New Frontiers in Artificial Intelligence, JSAI-isAI 2016, Lecture Notes in Computer Science, vol 10247, Springer, Cham).

Post-proceeding International Conference

[8] Teeradaj Racharak and Satoshi Tojo, **Inherited Properties of $\mathcal{FL}_0$ Concept Similarity Measure under Preference Profile**, In Agents and Artificial Intelligence (submitted), Lecture Notes in Computer Science.

Related Post-proceeding International Conference which is not in the thesis

[9] Teeradaj Racharak and Satoshi Tojo, **Analogical reasoning in clinical practice with description logic $\mathcal{ELH}$**, In Jaap van den Herik, Ana Paula Rocha and Joaquim Filipe (eds) Agents and Artificial Intelligence, Lecture Notes in Computer Science.