Quantitative Methods for Similarity in Description Logics

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Inf. Andreas Ecke
geboren am 11. November 1987 in Ilmenau

verteidigt am 14. Juni 2016

Gutachter:
Prof. Dr.-Ing. Franz Baader
Technische Universität Dresden

Prof. Dr. Uli Sattler
University of Manchester

Dresden, im September 2016

Contents

Chapter 1

Introduction

Description Logics (DLs) [BCM+03] are a family of logic-based knowledge representation languages used to describe the knowledge of an application domain and reason about it in a formally well-defined way. DLs allow to describe the important notions and classes of the knowledge domain as concepts, by stating the conditions for which individual objects belong to that concept as a Boolean combination of atomic properties, formalized by concept names, and properties that refer to the relationship with other classes, called role restrictions. In order to encode the conceptual knowledge, the user can then state how these concepts relate to each other, for example by giving superconcept-subconcept relationships. Additionally, DLs allow to express knowledge about individual objects, to which concepts they belong and how they relate to each other.

A variety of different DLs exist, differing in the set of logical constructs one can use to express concepts, the so-called concept constructors, as well as the types of axioms available to describe the relations between concepts or individuals. However, all classical DLs have in common that they can only express exact knowledge, and correspondingly only allow exact inferences. Either we can infer that some individual belongs to a concept, or we can't, there is no in-between. In practice though, knowledge is rarely exact. Many definitions have exceptions or are vaguely formulated in the first place, and people might not only be interested in exact answers, but also in alternatives that are "close enough".

This thesis is aimed at tackling the problem how to express that something is "close enough", and how to integrate this notion into the formalism of Description Logics. To this end, we will use the notion of semantic similarity and dissimilarity measures [HRJ+13] as a way to quantify how close exactly two concepts are. We will look at how useful measures can be defined in the context of DLs, and how they can be incorporated and used in this formal framework. In particular, we will look closer at two applications of such measures to DLs: Relaxed instance queries will use a similarity measure in order to not just give the exact answer to a query concept, but all answers that are reasonably similar. Prototypical definitions on the other hand use a measure of dissimilarity or distance between a prototypical object and elements of an interpretation in order to allow the definition of and reasoning with concepts that capture not just those individuals that satisfy exactly the stated properties, but also similar ones.

1.1 Description Logics

The basic building blocks of DLs are concepts names denoting sets of objects, role names denoting relations between objects, and individuals, which point to a single object. From these building blocks complex concepts can be build by applying the concept constructors that the DL supplies. For example, if Bike, and DiscBrake are concept names that denote the set of all bicycles and the set of all disc brakes, respectively, and hasPart is a relation that connects to each object all the parts it consists of, then the concept

$$\mathsf{Bike} \sqcap \exists \mathsf{hasPart}.\mathsf{DiscBrake}$$

denotes the set of all bikes that have a disc brake. Axioms allow to formalize how different concepts relate to each other, which allows the formulation of terminological knowledge collected in a so-called TBox. For example, one can express that road racing bikes always have slim tires using

$$\mathsf{RoadRacingBike} \sqsubseteq \exists \mathsf{hasPart}.(\mathsf{Tire} \sqcap \mathsf{Slim}).$$

Assertions on the other hand can state how individuals are related to concepts and other individuals and formulate the assertional knowledge collected in an ABox. For example, if bike483 and andreas are two individual names, we can state that bike483 is a trekking bike with a hub gear, and that andreas is the owner of said bike using the following ABox:

$$\begin{aligned}\mathcal{A} = \{&(\mathsf{TrekkingBike} \sqcap \exists \mathsf{hasPart}.\mathsf{HubGear})(\mathsf{bike483}),\\ &\mathsf{ownedBy}(\mathsf{bike483}, \mathsf{andreas})\}.\end{aligned}$$

Different DLs vary in the set of concept constructors they provide, and the set of axioms one can use to formulate knowledge in terms of these concepts. The smallest propositionally closed DL is $\mathcal{ALC}$ [SS91]; however, sub-propositional DLs like $\mathcal{EL}$ [Baa03] and its extension $\mathcal{EL}^{++}$ [BBL05] are very interesting due to their tractable reasoning procedures.

The formal semantics of DLs allow for a clean definition of inferences, which can infer implicit knowledge from the given axioms and assertions. Classical inferences, like consistency and entailment, which ask whether a knowledge base consisting of a TBox and an ABox has a model and whether it entails a given axiom or assertion, are provided by almost all DL systems. Besides those, many other non-standard inferences have been investigated; these can aid in different tasks related to DLs, like the maintenance of large ontologies [LBF+06].

DLs have been successfully employed in many different areas. The most important success however is certainly their adaption by the Web Ontology language OWL [HPH03] as basis for the semantic web. As such, nowadays a huge corpus of ontologies in all different forms and sizes exists [MBP13a] and is used in many different fields. One notable example for the usage of ontologies based on tractable

DLs of the $\mathcal{EL}$ family is in the biomedical domain, with large-scale ontologies like the Gene Ontology [Gen00] and SNOMED CT [SBS+07].

1.2 Similarity and Dissimilarity

Similarity measures are thought of as one of the fundamental concepts in psychology, by which humans can form knowledge and reason. This is expressed particularly well by Tversky:

> Similarity plays a fundamental role in theories of knowledge and behavior. It serves as an organizing principle by which individuals classify objects, form concepts, and make generalizations.
>
> — Amos Tversky [Tve77]

In principle, the more similar an object is to the thing in question, the more relevant it will be. Grouping similar objects together into a category allows humans to lift properties that all these objects have in common to the whole category. Then, we can automatically make predictions about new objects, based on which category they are most similar to. As such, there has always been a huge interested in similarity in both philosophy and linguistics, and many applications of measures of similarity have evolved in many different fields. However, as popular as similarity measures seem to be, as unclear is the notion of what exactly similarity is, and how it should be measured.

Fundamentally, a similarity measure quantifies how close two things are from a conceptual point of view. Many different approaches have evolved on how to measure similarity, but most rely on the same intuitions [Lin98]:

1. The similarity between two objects increases with the commonalities that they share.
2. The similarity between two objects decreases with the differences between them.

Additionally, many similarity measures also have a notion of a maximal and minimal similarity. These should comply with the following intuitions:

3. The maximal similarity between two things is reached if they are identical, i.e., have no differences; it does not matter how much commonality they share.
4. The minimal similarity between two things is reached when they have no commonalities, no matter how many differences they have.

We are interested in *semantic similarity measures*, which compare the meanings that the objects carry instead of just their syntax [HRJ+13]. This meaning is given by some semantic proxy, which could be unstructured texts, dictionaries, or, in our case, knowledge bases. Many approaches have been proposed to measure semantic similarity. Based on the ideas and principles these measures are based on, they can be divided into different groups:

- Distance-based approaches assume that the objects can be represented as points in a high-dimensional mental space. First introduced in [She62], these approaches treat the dissimilarity between objects as a distance metric in this mental space. The distance d between objects a and b can be converted into a similarity value $a \sim b = e^{d(a,b)}$ [She87].
- Feature-based approaches assume that an object is described by a set of features. The similarity between two objects can then be understood as a function of the common and distinct features between them. This approach was made popular by the seminal work of Tversky [Tve77], but has in others forms been used long before that [Jac01].
- Structural alignment approaches introduced in [GM97] extend the feature-based approach by assuming that objects are not just represented by a set of features, but by more complex structures like labeled graphs. By trying to find the best alignment between the structures of two different objects one can estimate the similarity between them using the correspondences that the alignment reveals.
- Other approaches include similarity measures based on information-theoretic ideas [Lin98], which compute the similarity based on the information-content that both objects share; transformational approaches that estimate the similarity from the number of transformation steps that are needed to convert the first object into the second one; and hybrid approaches, that try to combine the advantages of different approaches.

In recent years, similarity measures have been getting more popular in the area of knowledge representation; some of the applications where similarity measures have been successfully employed are listed below.

1. Similarity assessment: Sometimes one is interested simply in the question whether two objects are similar, or how similar exactly they are. This can already provide some insight. One example is the Gene Ontology [Gen00], which represents genes and gene products across species. For this ontology, many different similarity measures have been proposed to measure the functional similarity between proteins [SSP+05].
2. Case-based reasoning: Case-based reasoning is the process of solving new problems based on known solutions for previously encountered problems [AP94]. In order for this to work, one needs to judge which situations are similar to the new situation, and thus which solutions are likely to be applicable in order to arrive at a solution for the new problem.
3. Ontology learning: Similarity measures can be used to cluster similar individuals in order to automatically derive new concepts.
4. Prototypical definitions: Definitions are not always exact. Oftentimes, instead of giving a set of necessary and sufficient conditions, one only knows typical

features or a typical object, and expects the defined concept to also include objects that are similar enough to this prototype.

5. Ontology alignment: In ontology alignment, on tries to integrate two different ontologies that speak about the same topic [SE13]. In order to do this, one needs to match similar concepts that are likely to be talking about the same thing.
6. Query relaxation: When one is not only interested in exact answers to a query, but also in reasonable alternatives, then similarity measures can be used to quantify how close these alternatives are to the query.

In the context of Description Logics, research into similarity measures has been started by [BWH05]. For inexpressive DLs, graph-distance approaches that count the length of the paths between two concepts and their common subsumer in the inferred concept hierarchy are popular [RMB+89], whereas for more expressive DLs generally an extensional measure [dFE05; Lin98] is used, where the overlap between the extensions of concepts (usually the number of instances from the ABox) determines the similarity value. Finally, there is a group of structural measures [Jan06; Sun13; LT12] that compute the similarity by comparing the structure of the concepts in normal form.

In the next two sections, we will look closer at two applications of similarity measures: Query relaxation and prototypical definitions.

1.3 Query Relaxation

Knowledge about individuals, the concepts they belong to, and the relations between them, is usually stored in some kind of relational database, an XML file, an RDF triple store, a Description Logic ABox, or similar storage formats. In order to access this data, one can formulate a query that describes which of the individuals one is interested in, by restricting for instance the categories or the relations to other individuals. A query answering system then selects all those individuals that satisfy the query and returns them as answers.

However, when specifying the query, one may not only be interested in the exact answers, which satisfy every single restriction that is part of the query; alternatives that do not completely satisfy the query, but most of it, may give interesting insights as well; this is in particular true if the exact answer set would be empty. For example, consider a bike shop with a DL knowledge base that contains all relevant knowledge about the bikes that are for sale. If a customer wants to buy a new bike, one can create a query that formalizes all of the requirements of the customer and query the knowledge base for all bikes that are instances of this query. In this case, alternative answers that do not completely satisfy the query could be interesting for several reasons: the costumer might not be completely sure what exactly he wants; the knowledge about the individual bikes in the knowledge base might not be complete; not everything can be formalized in a knowledge base, e.g., some bikes may simply

"feel" better than others for the customer; or the query might not have any exact answers at all because of contradictory requirements, e.g., best possible quality for the least possible price.

The process of broadening the set of answers to include similar alternatives is often called query expansion or query relaxation, and has attracted a great deal of research. Classically, query relaxation is done by rewriting the query into successively more general queries using a set of rewrite rules [CDH+06; ZGB+07; HPW08; DSW+09; HLZ08; ALP04; Lee02]. This rewriting process is repeated until enough answers were found, i.e., the k best answers for top-k query answering, or all answers above a given threshold. Furthermore, some kind of measure quantifies how much the rewritten queries deviate from the original query in order to rank the answers.

Classical query relaxation approaches usually only work in the presence of a very simple background ontology, like a concept hierarchy. Also, often the process of query relaxation can not be influenced. However, a way to specify which aspects of the query are less important and may be relaxed further can be exceedingly useful to control the query relaxation process based on user- or query-dependent preferences. For example, two customers might both want light-weight, reasonably priced road racing bicycles; however, one customer has a strict budget and places more importance one the price, while the other strictly requires a low weight and is willing to compromise on the price, if necessary. Then the queries for both customers are the same, but they should be relaxed differently.

A different approach to relax queries is by using a quantitative KR formalism that includes a way to specify the degrees inside the logic itself. This could be fuzzy DLs [Str06b; Str06a; PSS+08], rough DLs [PTT14], or DLs extended with an internal distance measure on elements [LWZ03]. However, if the quantitative measure is part of the KR formalism itself, then changing the way queries should be relaxed means that one would need to change the data as well.

In Chapter 4, we will introduce a new way to relax query concepts that is based on similarity measures on DL concepts. By choosing a suitable similarity measure this approach allows to direct the way in which the query is relaxed; by varying the similarity threshold, it allows to specify how far the query is relaxed.

1.4 Prototypical Definitions

As explained before, in practical applications one often cannot define all relevant concepts exactly by giving necessary and sufficient conditions. In fact, cognitive psychologists [RL78] argue that humans generally recognize categories by prototypes rather than concepts. As an example, taken from [Lab73], consider the notion of a cup: we can say that cups are small, cylindrical, concave containers with handles, whose top side is open; they can hold liquids and are used for drinking; and they are made of plastic or porcelain. However, these are just typical features, not strict conditions for being a cup: square metal cups are easily ima-

ginable, measuring cups are not used for drinking and may hold non-liquids such as flour, while sippy cups for toddlers are not open on the top. In order to define the concept that contains all cups, one could try to capture all exceptional cups by using a big disjunction of (exactly defined) concepts, but this would be rather clumsy and with high likelihood one would overlook some exceptions. Compared to this, prototypical definitions allow to define concepts as the set of all elements that are similar to some prototypical object.

In order to be used within a formal knowledge representation language with automated reasoning capabilities, such prototypes need to be equipped with a formal semantics. For this, the ideas underlying Gärdenfors' conceptual spaces [Gär00] are very useful, where categories are explained in terms of convex regions, which are defined using the distance from a focal point. To obtain a concrete representation language, one needs to define what focal points are and how to define the distance of an individual to such a focal point.

A different approach to formalize prototypes is by using non-monotonic logics [Bre91]. In these logics one usually tries to maximize typicality, i.e. one assumes that an individual stated to belong to a prototype concept has all the properties of the prototype, unless one is forced by other knowledge to retract this assumption. However, we think a monotonic logic where we only conclude that an individual belongs to a prototype concept if this follows from the available knowledge would be a better formalization, since non-monotonic logics with typicality only allow to guess properties of objects for which we already know that they belong to the prototypical concept, but do not allow to classify which objects might belong to the concept in the first place.

Another approach to reason with prototypes in presented in [BBF15], where concepts of the lightweight DL $\mathcal{EL}$ are used to describe prototypes. To be more precise, the paper introduces a graded membership function which, for a given $\mathcal{EL}$-concept C and an individual d of an interpretation, returns a membership degree in the interval $[0, 1]$. This is then used as "distance" to define threshold concepts and an extension of $\mathcal{EL}$ by such concepts basically in the same way as sketched above.

In Chapter 5, we will introduce a new approach to represent and reason with prototypical definitions. This approach formalizes Gärdenfors' conceptual spaces, and uses weighted tree automata in order to define focal points and their distance to an individual.

1.5 Outline and Contributions of the Thesis

In the following we will give a short outline for the remainder of this thesis.

Chapter 2 will introduce the relevant notions that are needed for this thesis. We start with the foundations of Descriptions Logics, in particular the DLs $\mathcal{ALC}$ and $\mathcal{EL}$. We next define some important $\mathcal{EL}$-specific notions like simulations and canonical models, as well as the DL $\mathcal{EL}^{++}$, that extends $\mathcal{EL}$ with concrete domains, role hierarchies, nominals and other things. Finally, we formally define what we under-

stand under similarity and dissimilarity measures, in particular concept similarity measures (CSM), and list some properties that we may want such CSMs to satisfy.

Chapter 3 is concerned with the similarity measure $\sim_c$, which is a parameterizable CSM that works w.r.t. general $\mathcal{EL}$ knowledge bases. In order to define $\sim_c$, we first introduce a similarity measure between elements of two interpretations, $\sim_i$. We show that $\sim_i$ (and thus $\sim_c$) is well-defined and computable in polynomial time. We also show the formal properties of $\sim_i$, and how they extend to the CSM $\sim_c$. Finally, we show that $\sim_c$ can be extended to a measure between $\mathcal{EL}^{++}$ concepts w.r.t. $\mathcal{EL}^{++}$ TBoxes. For this, we need to determine how the similarity measure handles values from a concrete domain.

Chapter 4 investigates the problem of instance queries relaxed by CSMs. After formally defining what relaxed instances are, we consider the case of arbitrary CSMs on an unfoldable TBox. We show that this problem is decidable as long as the CSM used for the relaxation has certain properties, we also show that the decision procedure is highly inefficient: It has non-elementary complexity. Afterwards we restrict to a single family of similarity measures, namely $\sim_c$, but instead allow for general $\mathcal{EL}$ TBoxes. In this setting we can derive an NP algorithm for both checking whether an individual is a relaxed instance of the query concept, and for finding all answers to the relaxed instance query. Again, we show that this procedure can be extended to the DL $\mathcal{EL}^{++}$. Finally, we present an implementation of the relaxed instance query answering problem; the Elastiq system. We evaluate Elastiq on different ontologies.

This and the previous chapter are based on the following publications. In [EPT13] we introduced instance queries relaxed by concept similarity measures for the first time and showed how to solve them w.r.t. unfoldable TBoxes. We introduced the CSM $\sim_c$ in [EPT14a; EPT14b; EPT15a] and showed how to extend the relaxed instance query approach to general TBoxes. The extension to $\mathcal{EL}^{++}$ is content of [Eck14]. Finally, [EPT15b] describes out implementation, Elastiq, and presents an preliminary evaluation.

[Eck14] Andreas Ecke. "Similarity-based Relaxed Instance Queries in $\mathcal{EL}^{++}$". In: *Proceedings of the First Workshop on Logics for Reasoning about Preferences, Uncertainty, and Vagueness*. Edited by Thomas Lukasiewicz, Rafael Peñaloza, and Anni-Yasmin Turhan. Volume 1205. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pages 101–113.

[EPT13] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Towards Instance Query Answering for Concepts Relaxed by Similarity Measures". In: *Workshop on Weighted Logics for AI (in conjunction with IJCAI'13)*. Beijing, China, 2013.

[EPT14a] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Answering Instance Queries Relaxed by Concept Similarity". In: *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*. Vienna, Austria: AAAI Press, 2014, pages 248–257.

[EPT14b] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Mary, What's Like All Cats?" In: *Proceedings of the 27th International Workshop on Description Logics (DL-2014)*. (Vienna, Austria). Extended Abstract. 2014.

[EPT15a] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Similarity-based Relaxed Instance Queries". In: *Journal of Applied Logic* 13.4, Part 1 (2015). Special Issue for the Workshop on Weighted Logics for AI 2013, pages 480–508. DOI: http://dx.doi.org/10.1016/j.jal.2015.01.002.

[EPT15b] Andreas Ecke, Maximilian Pensel, and Anni-Yasmin Turhan. "ELASTIQ: Answering Similarity-threshold Instance Queries in $\mathcal{EL}$". In: *Proceedings of the 28th International Workshop on Description Logics (DL-2015)*. (Athens, Greece). Edited by Diego Calvanese and Boris Konev. Volume 1350. CEUR Workshop Proceedings. CEUR-WS.org, 2015.

In Chapter 5, we study the problem of prototypical definitions in the DL $\mathcal{ALC}$. We show how prototype distance functions can be used to formalize the notion of a prototype, and that weighted alternating tree automata are a useful tool to specify such prototype distance functions. In order to show that $\mathcal{ALCP}(\text{wapta})$, the DL extended with prototypes defined via weighted alternating parity tree automata, is decidable, we first show how unweighted automata can be used to decide concept satisfiability in $\mathcal{ALC}$. Afterwards, we present a cut-point construction that computes unweighted automata which recognize exactly the cut-point language of a weighted alternating parity tree automaton, i.e., the language of all trees that have a distance of at most n. Finally, we show that one can combine the automaton used to decide concept satisfiability in $\mathcal{ALC}$ with the cut-point automata, in order to decide the concept satisfiability problem in $\mathcal{ALCP}$. This chapter is based on the publication [BE16].

[BE16] Franz Baader and Andreas Ecke. "Reasoning with Prototypes in the Description Logic $\mathcal{ALC}$ using Weighted Tree Automata". In: *Proceedings of the 10th International Conference on Language and Automata Theory and Applications (LATA 2016)*. (Prague, Czeck). Lecture Notes in Computer Science. Springer-Verlag, 2016.

In Chapter 6 we summarize our results and point out directions for future work.

Chapter 2

Preliminaries

In this chapter we will introduce the theoretical foundations that the remainder of this thesis is based on. We will start by formally introducing the basic notions of Description Logics (DLs), the logic formalism upon which everything else will be built; we will also discuss the notions of similarity and dissimilarity, both in general and in the framework of DLs.

The introduction of DLs is split in two parts: First, we will establish the basic notions of DLs that are common across all logics of the DL family. This introduction is done in terms of the DL $\mathcal{ALC}$, which forms the basis of Chapter 5. Then we will have a closer look at the $\mathcal{EL}$ family of Description Logics, which is a fragment of $\mathcal{ALC}$ with nicer computational properties, but for which some special notions arise. $\mathcal{EL}$ will serve as the base language for Chapter 3 and 4.

Finally, we will introduce the notion of similarity and dissimilarity measures, both in general and in the framework of DLs, and have a look at different methods that can be employed to measure the similarity between DL concepts.

2.1 Basic Notions of Description Logics

As described in Chapter 1, DLs are a logic-based family of languages used for knowledge representation and reasoning. This section will give a brief introduction into the syntax and semantics of languages in the DL family, using the example of the prototypical DL $\mathcal{ALC}$. We will further define some related notions that will be of help in later chapters. Note that this introduction is far from complete. A more thorough look into DLs and all its aspects can be found in [BCM+03].

2.1.1 Description Logic Concepts

DL concepts are the most important building blocks of Description Logics; they describe the categories of the application domain that one wants to model. Concepts do not have to be atomic. For example, besides the category of all bikes, we may also speak about the concept that contains all bikes with a pink frame and disc brakes. Such complex concepts are built using concept constructors on two countably infinite, disjoint sets of names: The set $\mathsf{N_C}$ of *concept names* and the set $\mathsf{N_R}$ of *role names*.

Definition 1 (Syntax of DL concepts). The set of *concepts* is the smallest set such that

- $\top$ (*top concept*), $\bot$ (*bottom concept*), and every concept name $A \in \mathsf{N_C}$ is a concept;
- if C, D are concepts then the following are also concepts: $\neg C$ (*negation*), $C \sqcap D$ (*conjunction*), and $C \sqcup D$ (*disjunction*);
- if C is a concept and $r \in \mathsf{N_R}$ is a role name, then the following are also concepts: $\exists r.C$ (*existential restriction*), and $\forall r.C$ (*universal restriction*). $\diamondsuit$

For a DL $\mathcal{L}$, we denote the set of all concepts of $\mathcal{L}$ with $\mathfrak{C}(\mathcal{L})$. In particular, the constructors above give rise to $\mathcal{ALC}$ concepts. Different DLs do offer different sets of constructors, and thus may allow to create more or less expressive concepts. Note that many more constructors exist in the literature, but are not introduced here.

When considering the complexity of certain tasks related to Description Logics, we commonly take the *size of a concept* as the input size for this task. Given a concept C, its size $|C|$ is inductively defined as

$$|C| := \begin{cases} 1 & \text{if } C \in \mathsf{N_C} \cup \{\top, \bot\}; \\ 1 + |D| & \text{if } C = \exists r.D,\ C = \forall r.D, \text{ or } C = \neg D; \\ |C_1| + |C_2| & \text{if } C = C_1 \sqcap C_2 \text{ or } C = C_1 \sqcup C_2. \end{cases}$$

The semantics of a concept is defined in a model-theoretic way by means of interpretations.

Definition 2 (Semantics of concepts). An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, the *interpretation domain*, and an *interpretation function* $\cdot^{\mathcal{I}}$, which assigns an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to each individual name $a \in \mathsf{N_I}$; a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each concept name $A \in \mathsf{N_C}$; and a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role name $r \in \mathsf{N_R}$.

The interpretation function is extended to complex concepts as follows:

- $\top^{\mathcal{I}} := \Delta^{\mathcal{I}}$;
- $\bot^{\mathcal{I}} := \emptyset$;
- $(\neg C)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- $(C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(C \sqcup D)^{\mathcal{I}} := C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- $(\exists r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}}.(d, e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$; and
- $(\forall r.C)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \forall e \in \Delta^{\mathcal{I}}.(d, e) \in r^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{I}}\}$. $\diamondsuit$

The following example illustrates the use of concepts.

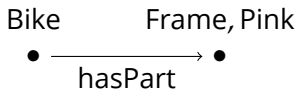

Figure 2.1: Interpretation $\mathcal{I}$ which contains an instance of the concept given in Example 3

Example 3. Consider the domain of bicycles. If we assume that Bike, DiscBrake, Frame, Pink, and Bell are concept names, and hasPart is a role name; we can express the concept of pink bikes without a bell using the following concept:

$$\mathsf{Bike} \sqcap \exists \mathsf{hasPart}.(\mathsf{Frame} \sqcap \mathsf{Pink}) \sqcap \forall \mathsf{hasPart}.\neg \mathsf{Bell}.$$

An interpretation $\mathcal{I}$ is given in Figure 2.1; the upper left element of this interpretation belongs to the given concept since it has a hasPart-successor which belongs to both Frame and Pink, and non of the hasPart-successors belong to Bell.

Note that one could express the property of not having a bell also by $\neg\exists \mathsf{hasPart}.\mathsf{Bell}$. According to the semantics, the concepts $\neg\exists \mathsf{hasPart}.\mathsf{Bell}$ and $\forall \mathsf{hasPart}.\neg \mathsf{Bell}$ are equivalent. ♢

Sometimes, we are not just interested in the interpretation as a whole, but in specific elements of this interpretation. A *pointed interpretation* $p = (\mathcal{I}, d)$ consists of an interpretation $\mathcal{I}$ together with an element $d \in \Delta^{\mathcal{I}}$ of its domain. We use $\mathfrak{I}$ to denote the set of all pointed interpretations, and for a fixed interpretation $\mathcal{I}$, we denote the set of all pointed interpretations of the form $p = (\mathcal{I}, d)$ with $\mathfrak{I}_{\mathcal{I}}$. For a pointed interpretation $p = (\mathcal{I}, d)$, we use $\mathfrak{C}(p) = \{C \in \mathfrak{C} \mid d \in C^{\mathcal{I}}\}$ to denote the set of all concepts that d is an instance of in $\mathcal{I}$.

Note that the meaning of a concept is entirely dependent on an interpretation. Any element of an interpretation may satisfy certain concepts, and not others, but all of these are valid interpretations. However, when drawing inferences we want to restrict to certain interpretations, namely those which respect our knowledge of the domain. For example, we might only want to consider interpretations in which racing bikes have thin tires. Interpretations where this is not the case (i.e., where there exists a racing bike with thick tires) should be excluded. This is possible by capturing these restrictions in a knowledge base.

2.1.2 Knowledge Bases

Knowledge bases consist of two parts. The *TBox (terminological box)* contains terminological knowledge, i.e., knowledge about how different concepts are related; the restrictions given in the TBox hold for all elements of an interpretation. The *ABox (assertional box)* on the other hand contains knowledge about specific individuals, which are given via a third countably infinite set of names: The set $\mathsf{N_I}$ of individual names. $\mathsf{N_I}$ is disjoint to both $\mathsf{N_C}$ and $\mathsf{N_R}$.

Definition 4 (Syntax of TBoxes). A TBox is a finite set of *TBox axioms*. Axioms can be one of the following:

- A *general concept inclusion (GCI)* is of the form $C \sqsubseteq D$, where both C and D are concepts.
- A *concept definition* is of the form $A \equiv C$, where C is a concept and $A \in \mathsf{N_C}$.

A TBox $\mathcal{T}$ containing only concept definitions is called *unfoldable* if it is unambiguous and contains no cyclic definitions, i.e.:

- for every concept name $A \in \mathsf{N_C}$, there is at most one concept definition of the form $A \equiv C \in \mathcal{T}$ (unambiguity); and
- there is no $\{A_1 \equiv C_1, \ldots, A_n \equiv C_n\} \subseteq \mathcal{T}$ such that A_{i+1} occurs in C_i for $1 \leq i < n$ and A_1 occurs in C_n (acyclicity).

Concept names that occur on the left-hand side of a concept definition in an unfoldable TBox are called defined concept names. All other concept names are called primitive concept names. Unfoldable TBoxes allow the expansion of concepts by replacing defined concept names by their definition until only primitive concept names remain [BCM+03]. ♢

Note that we can replace a concept definition $A \equiv C$ by two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$; therefore, we often consider only GCIs when dealing with general TBoxes.

Conversely, for unfoldable TBoxes we can simulate a GCI of the form $A \sqsubseteq C$ by a concept definition $A \equiv A' \sqcap C$, which introduces a new concept name A' that distinguishes the elements of A from other elements of C [BCM+03]. This substitution preserves all standard inferences. Thus, TBoxes that contain GCIs of the form $A \sqsubseteq C$ may still be considered unfoldable as long as they satisfy the other constraints given in Definition 4.

The semantics of TBoxes is again defined using interpretations.

Definition 5 (Semantics of TBoxes). An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ (denoted $\mathcal{I} \models C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. It satisfies a concept definition $A \equiv C$ (written $\mathcal{I} \models A \equiv C$) iff $A^{\mathcal{I}} = C^{\mathcal{I}}$. $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ (written $\mathcal{I} \models \mathcal{T}$), if it satisfies all axioms in $\mathcal{T}$. ♢

While TBoxes define relations between the different concepts of a knowledge domain, ABoxes instead capture the knowledge about specific individuals, and are defined as follows:

Definition 6 (Syntax of ABoxes). A *concept assertion* is of the form $C(a)$, where C is a concept, and $a \in \mathsf{N_I}$ is an individual name. A *role assertion* is of the form $r(a, b)$, where $r \in \mathsf{N_R}$ is a role name and $a, b \in \mathsf{N_I}$ are individual names. Concept assertions and role assertions are called *ABox axioms*. An ABox is a finite set of ABox axioms.♢

In essence, concept assertions allow to express to which categories an individual belongs, while role assertions allow to express the relations between different individual objects. The semantics of ABoxes can be defined in a straight-forward manner.

Definition 7 (Semantics of ABoxes). An interpretation $\mathcal{I}$ satisfies a concept assertion $C(a)$ (denoted $\mathcal{I} \models C(a)$), iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$; it satisfies a role assertion $r(a,b)$ (denoted $\mathcal{I} \models r(a,b)$), iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of the ABox $\mathcal{A}$, denoted $\mathcal{I} \models \mathcal{A}$, if it satisfies all assertions in $\mathcal{A}$. ♢

Together, TBox and ABox form a knowledge base (KB).

Definition 8 (Knowledge Base). A *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We say that an interpretation $\mathcal{I}$ is a model of $\mathcal{K}$, denoted $\mathcal{I} \models \mathcal{K}$, if it is a model of $\mathcal{T}$ and $\mathcal{A}$. ♢

We sometime want to talk about the exact set of concept, role, and individual names that occur in a knowledge base or a concept, instead of the (infinite) sets $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$. We call this the *signature* of a KB $\mathcal{K}$ or a concept C, denoted $\text{sig}(\mathcal{K})$ or $\text{sig}(C)$, respectively. Similarly, we denote the set of all *sub-concepts* of a concept C with $\text{sub}(C)$, and the set of all sub-concepts of concepts occurring in $\mathcal{K}$ with $\text{sub}(\mathcal{K})$.

Formally, the signature and the set of sub-concepts of a concept C are defined inductively as follows:

$$\text{sig}(C) := \begin{cases} \{C\} & \text{if } C \in \mathsf{N_C}; \\ \emptyset & \text{if } C \in \{\top, \bot\}; \\ \{r\} \cup \text{sig}(D) & \text{if } C = \exists r.D \text{ or } C = \forall r.D; \\ \text{sig}(D) & \text{if } C = \neg D; \\ \text{sig}(C_1) \cup \text{sig}(C_2) & \text{if } C = C_1 \sqcap C_2 \text{ or } C = C_1 \sqcup C_2. \end{cases}$$

$$\text{sub}(C) := \begin{cases} \{C\} & \text{if } C \in \mathsf{N_C} \cup \{\top, \bot\}; \\ \{C\} \cup \text{sub}(D) & \text{if } C = \exists r.D,\ C = \forall r.D, \text{ or } C = \neg D; \\ \{C\} \cup \text{sub}(C_1) \cup \text{sub}(C_2) & \text{if } C = C_1 \sqcap C_2 \text{ or } C = C_1 \sqcup C_2. \end{cases}$$

and for a KB $\mathcal{K} = (\mathcal{T}, A)$ we define sig and sub as:

$$\begin{aligned} \text{sig}(\mathcal{K}) := & \bigcup_{C \sqsubseteq D \in \mathcal{T}} \big(\text{sig}(C) \cup \text{sig}(D)\big) \cup \bigcup_{A \equiv C \in \mathcal{T}} \big(\{A\} \cup \text{sig}(C)\big) \\ & \cup \bigcup_{C(a) \in \mathcal{A}} \big(\{a\} \cup \text{sig}(C)\big) \cup \bigcup_{r(a,b) \in \mathcal{A}} \{r, a, b\}; \textit{and} \\ \text{sub}(\mathcal{K}) := & \bigcup_{C \sqsubseteq D \in \mathcal{T}} \big(\text{sub}(C) \cup \text{sub}(D)\big) \cup \bigcup_{A \equiv C \in \mathcal{T}} \big(\{A\} \cup \text{sub}(C)\big) \cup \bigcup_{C(a) \in \mathcal{A}} \text{sub}(C). \end{aligned}$$

There exists a variety of inferences that can be drawn from a knowledge base. Nearly all DL systems support the following standard inferences: *consistency*, *entailment*, and *satisfiability*. Let $\mathcal{K}$ be a KB, and α an TBox axiom or ABox assertion. Then $\mathcal{K}$ is *consistent*, if there exists a model of $\mathcal{K}$. $\mathcal{K}$ *entails* α, written $\mathcal{K} \models \alpha$, if α is satisfied in every model of $\mathcal{K}$. A concept C is *satisfiable* w.r.t. a KB $\mathcal{K}$, iff there exists a model $\mathcal{I}$ of $\mathcal{K}$ with $C^{\mathcal{I}} \neq \emptyset$.

In particular, for two concepts C and D and an individual a, we say that C is *subsumed* by D w.r.t. $\mathcal{K}$ iff $\mathcal{K} \models C \sqsubseteq D$, C and D are called *equivalent* w.r.t. $\mathcal{K}$ iff $\mathcal{K} \models C \equiv D$, and a is called an *instance* of C w.r.t. $\mathcal{K}$ iff $\mathcal{K} \models C(a)$.

Note that in $\mathcal{ALC}$, if $\mathcal{K}$ is consistent then entailment of TBox axioms only depend on the TBox, i.e., one gets the same result by replacing the ABox of $\mathcal{K}$ by the empty ABox. In such cases we may write $\mathcal{T} \models \alpha$ instead of $(\mathcal{T}, \emptyset) \models \alpha$.

2.1.3 Bisimulations in $\mathcal{ALC}$

A useful tool when talking about DL interpretations and their elements is the concept of *bisimulations*. Bisimulations are equivalence relations between elements of two interpretations that partition the elements into classes that cannot be distinguished by DL concepts. For the DL $\mathcal{ALC}$ bisimulations are defined as follows [KR99].

Definition 9 ($\mathcal{ALC}$ bisimulation). Let $\mathcal{I}$ and $\mathcal{J}$ be interpretations. A relation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ is called a *bisimulation* between $\mathcal{I}$ and $\mathcal{J}$, if the following conditions hold:

1. for all $(d, e) \in S$ and $\mathcal{A} \in \mathsf{N_C}$, $d \in A^{\mathcal{I}}$ if and only if $e \in A^{\mathcal{J}}$,
2. for all $(d, e) \in S$, $r \in \mathsf{N_R}$ and $(d, d') \in r^{\mathcal{I}}$, there exists $e' \in \Delta^{\mathcal{J}}$ with $(e, e') \in r^{\mathcal{J}}$ and $(d', e') \in S$, and
3. for all $(d, e) \in S$, $r \in \mathsf{N_R}$ and $(e, e') \in r^{\mathcal{J}}$, there exists $d' \in \Delta^{\mathcal{I}}$ with $(d, d') \in r^{\mathcal{I}}$ and $(d', e') \in S$. ♢

We call two pointed interpretations *bisimilar*, denoted $(\mathcal{I}, d) \cong (\mathcal{J}, e)$, if there exists a bisimulation S between $\mathcal{I}$ and $\mathcal{J}$ with $(d, e) \in S$. Indeed, with this definition we get the following property: if $(\mathcal{I}, d) \cong (\mathcal{J}, e)$, then for every $\mathcal{ALC}$ concept C we have $d \in C^{\mathcal{I}}$ if and only if $e \in C^{\mathcal{J}}$ [KR99].

2.2 The Description Logic $\mathcal{EL}$

As said in the beginning, DLs encompass a whole family of different languages which provide different sets of concept constructors and axioms, have different expressiveness and thus also different computational properties. $\mathcal{ALC}$ is a propositionally closed DL and therefore strictly subsumes propositional logic, which implies that reasoning can never be tractable in $\mathcal{ALC}$ unless P=NP. For this reason, less expressive DLs have been investigated for which the standard inferences are tractable; some of the most popular of those are DLs from the $\mathcal{EL}$ family [BBL05].

$\mathcal{EL}$ is a fragment of $\mathcal{ALC}$ that disallows the use of negation, the bottom concept, disjunction, and universal restrictions. Thus, the set $\mathfrak{C}(\mathcal{EL})$ of $\mathcal{EL}$ concepts can be defined using the syntactic rule

$$C ::= \top \mid A \mid C \sqcap C \mid \exists r.C,$$

where $A \in \mathsf{N_C}$, and $r \in \mathsf{N_R}$.

$\mathcal{EL}$ TBoxes, ABoxes and knowledges bases are defined as before, though they may of course only contain $\mathcal{EL}$ concepts. Similarly, the semantics of $\mathcal{EL}$ concepts, TBoxes, ABoxes, and KB are also as defined previously. Note that consistency and satisfiability in $\mathcal{EL}$ are trivial to decide since $\mathcal{EL}$ lacks the means to express contradictions or falsehood. Standard entailment inferences like concept subsumption and instances can be computed in polynomial time [BBL05].

Since $\mathcal{EL}$ is less expressive than $\mathcal{ALC}$, the notion of bisimulations is too strong for $\mathcal{EL}$ concepts. There exist pointed interpretations that cannot be distinguished using only $\mathcal{EL}$ concepts, but that are not bisimilar. Instead, one can define the weaker notion of $\mathcal{EL}$ simulations:

Definition 10 ($\mathcal{EL}$ simulation). Let $\mathcal{I}$ and $\mathcal{J}$ be interpretations. A relation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ is a *simulation* between $\mathcal{I}$ and $\mathcal{J}$, if the following two conditions hold:

1. For all $(d, e) \in S$ and $A \in \mathsf{N_C}$, if $d \in A^{\mathcal{I}}$ then $e \in A^{\mathcal{J}}$.
2. For all $(d, e) \in S$, $r \in \mathsf{N_R}$ and $(d, d') \in r^{\mathcal{I}}$, there exists $e' \in \Delta^{\mathcal{J}}$ with $(e, e') \in r^{\mathcal{J}}$ and $(d', e') \in S$. ♢

Given two pointed interpretations $p = (\mathcal{I}, d)$ and $q = (\mathcal{J}, e)$, we say that

- p *simulates* q (denoted by $p \lesssim q$), if there exists a simulation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ between $\mathcal{I}$ and $\mathcal{J}$ with $(d, e) \in S$, and
- p and q are *equisimilar* (denoted by $p \simeq q$), if $p \lesssim q$ and $q \lesssim p$.

This way, the simulation relation between pointed interpretations is again connected to the question whether they can be distinguished using $\mathcal{EL}$ concepts.

Theorem 11 (Lutz and Wolter [LW10]). *Let $p = (\mathcal{I}, d), q = (\mathcal{J}, e)$ be two pointed interpretations. Then:*

1. *$p \lesssim q$ iff $d \in C^{\mathcal{I}} \Rightarrow e \in C^{\mathcal{J}}$ for all $\mathcal{EL}$ concepts C;*
2. *$p \simeq q$ iff $d \in C^{\mathcal{I}} \Leftrightarrow e \in C^{\mathcal{J}}$ for all $\mathcal{EL}$ concepts C.*

2.2.1 Canonical Models in $\mathcal{EL}$

For the DL $\mathcal{EL}$, many polynomial-time reasoning procedures rely on the fact that canonical models can be built, from which it is possible to read off entailments directly [LW10]. These canonical models represent in a sense the most general model one can construct for a concept C or the individuals in an ABox $\mathcal{A}$, w.r.t. some TBox $\mathcal{T}$.

Definition 12. (canonical models) Let C be an $\mathcal{EL}$ concept and $\mathcal{T}$ an $\mathcal{EL}$ TBox. The *canonical model* $\mathcal{I}_{C,\mathcal{T}} = (\Delta^{\mathcal{I}_{C,\mathcal{T}}}, \cdot^{\mathcal{I}_{C,\mathcal{T}}})$ of C w.r.t. $\mathcal{T}$ is defined as follows:

- $\Delta^{\mathcal{I}_{C,\mathcal{T}}} = \{d_C\} \cup \{d_D \mid \exists r.D \in \mathsf{sub}(C) \cup \mathsf{sub}(\mathcal{T})\}$
- $A^{\mathcal{I}_{C,\mathcal{T}}} = \{d_D \mid \mathcal{T} \models D \sqsubseteq A\}$, for all concept names A, and
- $r^{\mathcal{I}_{C,\mathcal{T}}} = \{(d_D, d_E) \mid \mathcal{T} \models D \sqsubseteq \exists r.E\}$, for all role names r.

The canonical model $\mathcal{I}_\mathcal{K} = (\Delta^{\mathcal{I}_\mathcal{K}}, \cdot^{\mathcal{I}_\mathcal{K}})$ of the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is defined as follows:

- $\Delta^{\mathcal{I}_\mathcal{K}} = \{d_a \mid a \in \mathsf{sig}(\mathcal{K}) \cap \mathsf{N_I}\} \cup \{d_C \mid \exists r.C \in \mathsf{sub}(\mathcal{K})\}$,
- $A^{\mathcal{I}_\mathcal{K}} = \{d_D \mid \mathcal{T} \models D \sqsubseteq A\} \cup \{d_a \mid \mathcal{K} \models A(a)\}$,
- $r^{\mathcal{I}_\mathcal{K}} = \{(d_D, d_E) \mid \mathcal{T} \models D \sqsubseteq \exists r.E\} \cup \{(d_a, d_D) \mid \mathcal{K} \models \exists r.D(a)\}$
  $\cup \{(d_a, d_b) \mid r(a,b) \in \mathcal{A}\}$. ♢

Note that canonical models for $\mathcal{EL}$ are always finite. As said before, the canonical model $\mathcal{I}_{C,\mathcal{T}}$ is the most general model for C and $\mathcal{T}$; this means that for any other model $\mathcal{J}$ of $\mathcal{T}$ with some $d \in C^\mathcal{J}$, $(\mathcal{J}, d)$ can be simulated by d_C in $\mathcal{I}_{C,\mathcal{T}}$. Similarly $\mathcal{I}_\mathcal{K}$ is the most general model of $\mathcal{K}$, i.e., for any model $\mathcal{J}$ of $\mathcal{K}$ with $d = a^\mathcal{J}$ for an individual a, $(\mathcal{J}, d)$ is simulated by d_a in $\mathcal{I}_\mathcal{K}$.

Theorem 13 (Lutz and Wolter [LW10]). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{EL}$ KB, and C, D be $\mathcal{EL}$ concepts. Then:*

1. *for all models $\mathcal{I}$ of $\mathcal{T}$ and all elements $d \in \Delta^\mathcal{I}$ it holds that $d \in C^\mathcal{I}$ iff $(\mathcal{I}_{C,\mathcal{T}}, d_C) \lesssim (\mathcal{I}, d)$;*
2. *$\mathcal{T} \models C \sqsubseteq D$ iff $d_C \in D^{\mathcal{I}_{C,\mathcal{T}}}$ iff $(\mathcal{I}_{D,\mathcal{T}}, d_D) \lesssim (\mathcal{I}_{C,\mathcal{T}}, d_C)$; and*
3. *$\mathcal{K} \models C(a)$ iff $d_a \in C^{\mathcal{I}_\mathcal{K}}$ iff $(\mathcal{I}_{C,\mathcal{T}}, d_C) \lesssim (\mathcal{I}_\mathcal{K}, d_a)$.*

Since many properties of pointed interpretations that one may want to talk about are preserved by simulations, this theorem implies that for these cases, it is enough to only check the canonical model instead of all models.

2.2.2 Non-standard Inferences

Besides typical inference services that are implemented in most reasoner systems, like subsumption, instance checking, and consistency, many other inferences have been investigated for various applications. Here, we will introduce the *most specific concept* [BK98], which, given an $\mathcal{EL}$ knowledge base $\mathcal{K}$ and an individual name a, returns the most specific concept C that has a as an instance.

Definition 14 (most specific concept). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an KB and a an individual name occurring in $\mathcal{A}$. An concept C is the *most specific concept* of a w.r.t. $\mathcal{K}$ (denoted $\mathsf{msc}_\mathcal{K}(a)$) if it satisfies:

1. $\mathcal{K} \models C(a)$, and
2. for any concept D, $\mathcal{K} \models D(a)$ implies $\mathcal{T} \models C \sqsubseteq D$. ♢

In general, the msc does not need to exist if the ABox contains any cycles [KM01]. For example, consider the $\mathcal{EL}$ KB $\mathcal{K}$ consisting of an empty TBox and the ABox $\mathcal{A} = \{r(a,a)\}$. The most specific concept of a w.r.t. $\mathcal{K}$ would be the infinite concept $\exists r.\big(\exists r.\big(\exists r.(\exists r.\dots)\big)\big)$; since concepts must be finite, $\mathsf{msc}_\mathcal{K}(a)$ cannot exist.

Instead, one can approximate the most specific concept by limiting the role-depth, i.e., the maximal nesting of existential restrictions in the resulting concept. Formally, the *role-depth* $\mathrm{rd}(C)$ of an $\mathcal{EL}$ concept C is

$$\mathrm{rd}(C) := \begin{cases} 0 & \text{if } C \in \mathsf{N_C} \cup \{\top\}, \\ 1 + \mathrm{rd}(D) & \text{if } C = \exists r.D, \\ \max(\mathrm{rd}(C_1), \mathrm{rd}(C_2)) & \text{if } C = C_1 \sqcap C_2. \end{cases}$$

Then, the role-depth bounded most specific concept can be defined as follows [KM01]:

Definition 15 (role-depth bounded most specific concept). Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a KB and a an individual occurring $\mathcal{A}$. A concept C is the *role-depth bounded most specific concept* of a w.r.t. $k \in \mathbb{N}$ and $\mathcal{K}$ (denoted k-$\mathrm{msc}_{\mathcal{K}}(a)$) if it satisfies:

1. $\mathrm{rd}(C) \leq k$,
2. $\mathcal{K} \models C(a)$, and
3. for any concept D with $\mathrm{rd}(D) \leq k$, $\mathcal{K} \models D(a)$ implies $\mathcal{T} \models C \sqsubseteq D$. ♢

The k-msc always exists since the role-depth bound will cut off any cycles at depth k. Both the msc, if it exists, and the k-msc are unique up to equivalence in $\mathcal{EL}$. Algorithms for computing the k-msc in $\mathcal{EL}$, and some of its extensions, have been studied [PT11; EPT13a], and implemented [MET11].

Instead of looking at individuals, one can also look at elements in an interpretation. This leads to the notion of model-based most specific concepts [Dis08]. We will directly introduce the role-depth bounded version.

Definition 16 (role-depth bounded model-based most specific concept). Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox, $\mathcal{I}$ be a model of $\mathcal{T}$, and $(\mathcal{I}, d) \in \mathfrak{I}_{\mathcal{I}}$ be a pointed interpretation. A concept C is the *role-depth bounded model-based most specific concept* of $(\mathcal{I}, d)$ w.r.t. $k \in \mathbb{N}$ and $\mathcal{T}$ (denoted k-$\mathrm{model\text{-}msc}_{\mathcal{T}}(p)$) if it satisfies:

1. $\mathrm{rd}(C) \leq k$,
2. $d \in C^{\mathcal{I}}$, and
3. for any concept D with $\mathrm{rd}(D) \leq k$, $d \in D^{\mathcal{I}}$ implies $\mathcal{T} \models C \sqsubseteq D$. ♢

As for the k-msc, the k-model-msc always exists in $\mathcal{EL}$, and is unique up to equivalence. The proof as direct adaptations of the corresponding proofs for the k-msc.

2.2.3 $\mathcal{EL}^{++}$: Extending $\mathcal{EL}$ with Concrete Domains, Nominals, and more

While $\mathcal{EL}$ has nice computational properties, in practice the restriction to only conjunctions and existential restrictions is often too limiting and unnecessary. Therefore, it has been investigated which constructors could be added without sacrificing

	syntax	usual semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
top concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\bot^{\mathcal{I}} = \emptyset$
nominal	$\{o\}$	$\{o\}^{\mathcal{I}} = \{o^{\mathcal{I}}\}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists e \in \Delta^{\mathcal{I}} \colon (d,e) \in r^{\mathcal{I}} \wedge e \in C^{\mathcal{I}}\}$
concrete domain	$p(f_1, \ldots, f_n)$	$p(f_1, \ldots, f_n)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (f_1^{\mathcal{I}}(d), \ldots, f_n^{\mathcal{I}}(d)) \in p^{\mathcal{D}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
role inclusion	$r_1 \circ \cdots \circ r_n \sqsubseteq s$	$r_1^{\mathcal{I}} \circ \cdots \circ r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
domain restriction	$\mathrm{dom}(r) \sqsubseteq C$	$r^{\mathcal{I}} \subseteq C^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
range restriction	$\mathrm{ran}(r) \sqsubseteq C$	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C^{\mathcal{I}}$
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 2.1: Concept constructors, TBox axioms, and ABox assertions for $\mathcal{EL}^{++}$

tractability [BBL05; BBL08]. This has resulted in the DL $\mathcal{EL}^{++}$, which adds to $\mathcal{EL}$ the bottom concept, concrete domains, nominals, domain and range restrictions, as well as role inclusions.

The full syntax and semantics of all $\mathcal{EL}^{++}$ concept constructors as well as the TBox and ABox axioms is given in Table 2.1. In the following, we will briefly explain the new constructors and axioms.

- Nominals allow to incorporate individuals directly into concepts. For example, if japan is an individual that represents the respective country, then one can define a japanese bike by

$$\mathsf{JapaneseBike} \equiv \mathsf{Bike} \sqcap \exists \mathsf{madeIn}.(\mathsf{Country} \sqcap \{\mathsf{japan}\}).$$

- Concrete domains allow to attach concrete values like strings or numbers to elements of an interpretation via feature names, and to reason over those values via a set of predicates. For example, using the concrete domain $\boldsymbol{Q}$ of the rational numbers with predicates $\{=, \geq_p, =_p\}$ with the obvious meanings, one can express that a heavy bike is a bike whose frame weights at least 8 kilogram:

$$\mathsf{HeavyBike} \sqsubseteq \mathsf{Bike} \sqcap \exists \mathsf{hasPart}.(\mathsf{Frame} \sqcap \geq_8(\mathsf{weight})),$$

  or that the front and rear wheel of all bikes have the same size:

$$\mathsf{Bike} \sqsubseteq =(\mathsf{frontWheelSize}, \mathsf{rearWheelSize}).$$

- Domain and range restriction can restrict the context in which roles are used,

by saying that elements connected via such a role must belong to certain concepts. For example, we might say that any element that has an incoming hasPart-edge must be a BikePart:

$$\text{ran}(\mathsf{hasPart}) \sqsubseteq \mathsf{BikePart}.$$

- Role-inclusion axioms allow to express, among other things, role hierarchies and transitive roles. For example, one can say that the hasPart-relation is transitive using

$$\mathsf{hasPart} \circ \mathsf{hasPart} \sqsubseteq \mathsf{hasPart}.$$

In order to avoid intricate interactions between role-inclusions and range restrictions that may lead to intractability or even undecidability, $\mathcal{EL}^{++}$ TBoxes need to satisfy an additional syntactic restriction [BBL05]. For a TBox $\mathcal{T}$ and role name r, we write $\mathcal{T} \models \text{ran}(r) \sqsubseteq C$, iff there are $r_1 \sqsubseteq r_2, \ldots, r_{n-1} \sqsubseteq r_n \in \mathcal{T}$ with $r_1 = r$ and $\text{ran}(r_n) \sqsubseteq C \in \mathcal{T}$, i.e., the range of r is implicitly restricted via a superrole. Then, the syntactic restriction is as follows: If $r_1 \circ r_2 \circ \cdots \circ r_n \sqsubseteq s \in \mathcal{T}$ for $n > 1$ and $\mathcal{T} \models \text{ran}(s) \sqsubseteq C$, then also $\mathcal{T} \models \text{ran}(r_n) \sqsubseteq C$.

We will give a formal definition of concrete domains [BH91; BBL05].

Definition 17 (concrete domain). A concrete domain $\mathcal{D} = (\Delta^{\mathcal{D}}, P^{\mathcal{D}})$ consists of a set of concrete values $\Delta^{\mathcal{D}}$ and a set of predicates $p \in P$, each associated with an arity $n > 0$ and an extension $p^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^n$.

A countable infinite set of feature names $\mathsf{N_F}$ is used to connect elements of an interpretation to concrete values. Description Logics extended with concrete domains allow for the constructor $p(f_1, \ldots, f_n)$, where $p \in P$ is an n-ary predicate, and $f_1, \ldots, f_n \in \mathsf{N_F}$ are feature names.

An interpretation assigns to each feature name $f \in \mathsf{N_F}$ a partial function $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightharpoonup \Delta^{\mathcal{D}}$ from the interpretation domain to the concrete domain. The semantics of the predicate constructor are as follows:

$$(p(f_1, \ldots, f_n))^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid (f_1^{\mathcal{I}}(d), \ldots, f_n^{\mathcal{I}}(d)) \in p^{\mathcal{D}}\} \qquad \diamondsuit$$

In order for the complexity of standard reasoning tasks to stay tractable, the concrete domain needs to satisfy certain properties. Such concrete domains are called p-admissible.

Definition 18 (p-admissible). A concrete domain $\mathcal{D}$ is *p-admissible*, if satisfiability and entailment in $\mathcal{D}$ are decidable in PTime, and the concrete domain is convex, i.e., if a set of predicated implies a disjunction of predicates, then it needs to imply already one of its disjuncts. $\diamondsuit$

P-admissibility is quite a strong restriction to a concrete domains. In fact, most non-trivial concrete domains are not p-admissible, as are all those with a finite domain. However, there are a few examples of non-trivial p-admissible concrete domains given in [BBL05] which allow the use of rational numbers and strings.

Furthermore, we will later argue that even trivial concrete domains can be very useful for use in similarity measures and for relaxing instance queries. In fact, for any infinite set V of values, the concrete domain $\mathcal{V} = (V, =_v)$ which consists only of unary predicates $=_v$ for $v \in V$ with $(=_v)^{\mathcal{V}} = \{v\}$, i.e., only allows for value assignments, is p-admissible and can be very useful.

2.3 Similarity and Dissimilarity

As we have see in Chapter 1, a large variety of approaches to similarity exist, most of which treat the notion of similarity slightly different. However, we have established four principle that are commonly used to guide the construction of similarity measures.

1. The similarity between two objects increases with the commonalities that they share.
2. The similarity between two objects decreases with the differences between them.
3. The maximal similarity between two things is reached if they are identical, i.e., have no differences; it does not matter how much commonality they share.
4. The minimal similarity between two things is reached when they have no commonalities, no matter how many differences they have.

As we require a similarity measure to have a minimal and a maximal value, we can actually fix the range of similarity measures. Most commonly, the value 0 is used to denote complete dissimilarity, and the value 1 is used to denote complete similarity. With this, we can define similarity measures formally:

Definition 19. Given a (possible infinite) set Δ of objects. A similarity measure $\sim$ on Δ is a function $\sim\colon \Delta \times \Delta \to [0, 1]$ with $d \sim d = 1$ for all $d \in \Delta$. ♢

Dissimilarity on the other hand is much harder to define. It can either mean the inverse of a similarity measure, which also has a minimal and maximal value, or it can be unbounded. In the latter case, the more differences there are between two objects, the higher the dissimilarity can grow, often even disregarding commonalities between the two objects completely. Then, the dissimilarity measure is simply a measure of the differences between the objects and thus behaves like a distance metric. We will use this view as well.

2.3.1 Concept Similarity in DLs

In the contexts of DLs, we are usually interested in concept similarity measures (CSMs), i.e, measures where the objects to be compared are DL concepts. Additionally, when working with DL knowledge bases we always have some background knowledge in the form of a TBox. We require CSMs to respect this background knowledge.

The introduction already gave an overview over general approaches to similarity measures. For DLs, some specific CSMs have been proposed. Early measures for DLs did usually use one of three approaches:

- for very inexpressive terminologies, graph-distance approaches that count the length of the paths between the concepts and their common subsumer [RMB+89] have been used. These approaches often only respect direct subsumption relations between concept names, and are hard to generalize for more expressive DLs. However, they have been used to great effect for the Gene Ontology, where they are used to measure the functional similarity between different genes and gene products. Graph-distances measures are often combined with information-content approaches, as both are quite similar.
- For more expressive Description Logics, generally an extensional measure [dFE05; Lin98] is used, which uses the overlap between the extensions of concepts (usually from the ABox) to compute the similarity value. These measures require a large and well-balanced ABox in order to achieve good similarity estimations. If the ABox is small or unbalanced, then the results can be very counter-intuitive. One example is that extensional measures assign similarity 1 to completely different concept names if both have an empty extension in the ABox.
- Finally, there is a group of structural measures [Jan06; Sun13; LT12] that compute the similarity by comparing the structure of the concepts in some kind of normal form. However, these usually only work for unfoldable TBoxes and for rather inexpressive DLs [Sun13; LT12]. If the presence of more more expressive DLs, these measures can become very complex, and the normal form might not be unique any longer, which quickly leads to problems.

Recently, a number of other measures have been defined. A transformational dissimilarity measure for $\mathcal{EL}$ was introduced in [DAB14], which works by counting the number of tree operations it takes to transform the first concept into the second. This measure has many nice properties; for example it fulfills the triangle-property known from distance metrics.

A family of different CSMs has been introduced in [APS14]. These measures are conceptually very simple, they work by extracting two sets S_1, S_2 of concepts from the concepts that are compared and using the Jaccard index

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

to measure the similarity between these sets. By varying the set of concepts one is allowed to extract, i.e., just concept names, subconcepts, or concepts generated by some kind of grammar, one can create CSMs of different power and complexity. They work on a large set of DLs, however they can not be parameterized and do not allow for discounting.

2.3.2 Formal Properties of Concept Similarity Measures

A set of formal properties for CSMs was presented in [LT12]. The framework devised in [LT12] allows to construct CSMs for $\mathcal{EL}$ concepts (possibly defined w.r.t. unfoldable TBoxes) that satisfy most of these properties.

Here, we want to investigate CSMs for $\mathcal{EL}$ concepts defined w.r.t. general TBoxes. Thus, we extend the definition of the properties of CSMs from [LT12] to the case where general TBoxes are used.

Definition 20. Let $\mathcal{L}$ be a DL and $\mathcal{T}$ be an $\mathcal{L}$ TBox. Then a concept similarity measure $\sim : \mathfrak{C}(\mathcal{L}) \times \mathfrak{C}(\mathcal{L}) \to [0, 1]$ w.r.t. $\mathcal{T}$ is called:

- *symmetric*, if $C \sim D = D \sim C$ for all $C, D \in \mathfrak{C}(\mathcal{L})$;
- *equivalence-invariant*, if for all $C, C', D, D' \in \mathfrak{C}(\mathcal{L})$ with $\mathcal{T} \models C \equiv C'$ and $\mathcal{T} \models D \equiv D'$ it holds that $C \sim D = C' \sim D'$;
- *equivalence-closed*, if $\mathcal{T} \models C \equiv D \iff C \sim D = 1$ for all $C, D \in \mathfrak{C}(\mathcal{L})$;
- *bounded*, if the existence of an $\mathcal{L}$-concept $E \neq \top$ with $\mathcal{T} \models C \sqsubseteq E$ and $\mathcal{T} \models D \sqsubseteq E$ implies $C \sim D > 0$ for all $C, D \in \mathfrak{C}(\mathcal{L})$; and
- *dissimilar-closed*, if for all $C, D \in \mathfrak{C}(\mathcal{L})$ with $C, D \neq \top$, the non-existence of $E \neq \top$ with $\mathcal{T} \models C \sqsubseteq E$ and $\mathcal{T} \models D \sqsubseteq E$ implies that $C \sim D = 0$. ♢

Many of these properties can be motivated by the intuition about similarity measures given above: Equivalence-closed means that the maximal similarity 1 should only occur for concepts that are equivalent, i.e., have no differences (w.r.t. the given TBox). Equivalence invariance further requires that equivalent concepts always behave exactly the same w.r.t. $\sim$. Boundedness requires that as soon as two concepts share some commonality (in form of a common subsumer E), the similarity must be different from the minimum similarity 0, while dissimilar-closed implies that similarity 0 occurs exactly when two concepts have no commonalities. Symmetry is a common property that most measures satisfy, though some authors do reject this property [Tve77].

More properties have been defined in the literature, like the triangle property or the weakened forms of this property called subsumption preserving and reverse subsumption preserving [LT12]. However, while these properties are useful and easy to explain for distance or dissimilarity measures, they may not always be intuitive for similarity measures. For example, while the triangle property has an intuitive geometric meaning on distance metrics, this meaning is lost when adapting this property to similarity measures, and the need to introduce a transformation function to translate between distances and similarities means that different formulations of the triangle property for similarity measures are possible, none of which are natural.

We will define an additional property, which is important for the definition of relaxed instance queries later.

Definition 21. A CSM $\sim$ is called *role-depth bounded*, if $C \sim D = C_k \sim D_k$ for any $k > \min(\text{rd}(C), \text{rd}(D))$, where C_k and D_k are the restrictions of C and D to role-depth bound k, i.e., the concepts one gets by replacing all existential restrictions at a role-depth of k with $\top$. $\diamondsuit$

Basically, this property means one does not need to take into account the exact shape of features that occur in one concept but not the other. For example, when comparing two bikes, one with battery lights and one without any light, the similarity does not depend on the type, color, or power of the lights the first bike has – the other bike does not have any lights either way.

These defined properties make the outcome of a CSM with these properties more predictable for ontology users. The measures described in [Sun13; LT12] fulfill most of these properties, as will the measure that we will introduce in the next chapter.

Chapter 3

A Concept Similarity Measure for $\mathcal{EL}$

In this chapter we present $\sim_c$, a structural similarity measure for $\mathcal{EL}$ concepts w.r.t. a general $\mathcal{EL}$ TBox. This measure is based on the measure *simi* introduced in [LT12], however, there are some differences. The biggest difference is certainly that *simi* only works w.r.t. unfoldable TBoxes: in the first step, it expands all concepts with the definitions given in the TBox and then discards it; later it uses only the expanded concepts to compute the similarity. This approach has two big problems: First, it cannot be easily generalized to general TBoxes; and second, the expansion may lead to an unnecessary exponential blowup of the concept descriptions.

The similarity measure $\sim_c$ introduced in this chapter instead relies on a so-called interpretation similarity measure $\sim_i$, that assigns a similarity value to a pair of pointed interpretations. It is then expanded to a concept similarity measure by comparing the canonical models of the concepts w.r.t. the general TBox. This approach avoids the exponential blowup introduced by the expansion step, works w.r.t. general TBoxes, and retains most of the formal properties of the measure *simi*.

3.1 Interpretation Similarity

An *interpretation similarity measure* (ISM) is defined as a similarity measure on finite pointed interpretations, i.e., a function of the type $\mathfrak{I} \times \mathfrak{I} \to [0, 1]$. It maps any pair of pointed interpretations to a similarity value between 0 and 1.

It is possible to transfer the formal properties of CSMs to ISMs.

Definition 22. Given a DL $\mathcal{L}$ and suitable simulation relations $\lesssim$ and $\simeq$ (as given in Definition 10 for $\mathcal{EL}$), we call an interpretation similarity measure $\sim_i$:

- *symmetric*, iff $p \sim_i q = q \sim_i p$ for all $p, q \in \mathfrak{I}$;
- *bounded*, iff $\mathfrak{C}(p) \cap \mathfrak{C}(q) \supsetneq \{\top\}$ implies $p \sim_i q > 0$ for all $p, q \in \mathfrak{I}$;
- *dissimilar closed*, iff $\mathfrak{C}(p) \cap \mathfrak{C}(q) = \{\top\}$ implies $p \sim_i q = 0$ for all $p, q \in \mathfrak{I}$ with $\mathfrak{C}(p) \supsetneq \{\top\}$ and $\mathfrak{C}(q) \supsetneq \{\top\}$;
- *equisimulation invariant*, iff $p \simeq q$ implies $p \sim_i u = q \sim_i u$ for all $p, q, u \in \mathfrak{I}$;
- *equisimulation closed*, iff $p \simeq q \iff p \sim_i q = 1$ for all $p, q \in \mathfrak{I}$;
- *simulation preserving*, iff $r \lesssim q \lesssim p$ implies $p \sim_i q \geq p \sim_i r$ for all $p, q, r \in \mathfrak{I}$;

- *reverse simulation preserving*, iff $r \lesssim q \lesssim p$ implies $q \sim_i r \geq p \sim_i r$ for all $p, q, r \in \mathfrak{I}$. ♢

We will now introduce the interpretation similarity measure $\sim_i$.

3.1.1 The Interpretation Similarity Measure $\sim_i$

For two pointed interpretations to be perfectly similar, they need to have the same set of concept names and have edges labeled with the same roles going to perfectly similar successor elements. Otherwise, the most similar concept names and the most similar direct successors are compared and a similarity value is computed from these pairs. In essence, $\sim_i$ is a feature-based similarity measure where the concept names and successors of an interpretation element are its features.

Before defining the interpretation similarity measure $\sim_i$, we need to define two auxiliary functions:

- $\text{CN} : \mathfrak{I} \to \mathcal{P}(\mathsf{N_C})$ with $\text{CN}((\mathcal{I}, d)) = \{A \in \mathsf{N_C} \mid d \in A^{\mathcal{I}}\}$ returns the set of concept names that d is an instance of in $\mathcal{I}$.
- $\text{SC} : \mathfrak{I} \to \mathcal{P}(\mathsf{N_R} \times \mathfrak{I})$ with $\text{SC}((\mathcal{I}, d)) = \{(r, (\mathcal{I}, e)) \mid (d, e) \in r^{\mathcal{I}}\}$ returns the set of successors, i.e., elements e that are connected to d via role r in $\mathcal{I}$.

$\sim_i$ is defined as a recursive function, where the similarity of two elements in their respective interpretations depends on the concept names that they have in common and in which they differ, as well as the similarity of all the elements that are connected to them via roles. In order to control how these features influence the similarity, $\sim_i$ depends on a number of parameters:

Primitive Measure The primitive measure $\sim_p$: $\mathsf{N_C} \times \mathsf{N_C} \cup \mathsf{N_R} \times \mathsf{N_R} \to [0, 1]$ assigns a similarity value to pairs of concept names or pairs of role names, such that $x \sim_p x = 1$ for every concept names $x \in \mathsf{N_C}$ or role name $x \in \mathsf{N_R}$. We give a default primitive measure as follows:

$$x \sim_{\text{default}} y = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases}$$

Weighting Function The weighting function $g : \mathsf{N_C} \cup \mathsf{N_R} \to \mathbb{R}_{>0}$ assigns a positive weight to each concept and role name. These weights allow one to control the influence that different concept names and roles have on the overall similarity. A high weight means that the concept or role name is important and has a great impact of the overall similarity. Again, we will give a default weighting function g_{default} that assigns the weight 1 to all concept and role names.

Discounting Factor The discounting factor $0 < w < 1$ controls the discounting of existential restrictions, i.e., how big of an impact the similarity of the successors of an element have on the overall similarity.

The primitive measure can be used to compute the similarity between single concept names. Since elements are usually instance of more than a single concept name, we need to match the concept names of both elements in order to find the maximal correspondence w.r.t. the primitive measure. This gives rise to the following function $\mathrm{cm} : \mathfrak{I} \times \mathfrak{I} \to \mathbb{R}$:

$$\mathrm{cm}(p, q) = \sum_{A \in \mathrm{CN}(p)} \left(g(A) \max_{B \in \mathrm{CN}(q)} A \sim_p B \right).$$

Essentially, this directed measure from p to q simply adds for each concept name that p belongs to the maximal primitive similarity to the concept names of q, weighted according to the weighting function g.

For the successors of the pointed interpretations p and q, we can define a similar directed measure $\mathrm{sm} : \mathfrak{I} \times \mathfrak{I} \to \mathbb{R}$. For this, assume that we already know the interpretation similarity $p' \sim_i q'$ between the successor elements.

$$\mathrm{sm}(p, q) = \sum_{(r,p') \in \mathrm{SC}(p)} \left(g(r) \max_{(s,q') \in \mathrm{SC}(q)} \left((r \sim_p s)\big((1 - w) + w(p' \sim_i q')\big) \right) \right)$$

This function again finds for each successor of p the best successor of q, i.e., the one where the primitive similarity between the role names multiplied with the discounted interpretations similarity between the elements itself is maximal.

With this in place, we can define the ISM $\sim_i$. In order to compute the similarity between to pointed interpretations p and q, $\sim_i$ simply add the values of the measures cm and sm. Since these measures are directed, this is done in both directions; from p to q and from q to p, to arrive at a symmetric measure. Finally, the sum is divided by the weights of all concept names that p and q are instance of and the weights of the role names of all successors of p and q. This normalizes the similarity value to the interval $[0, 1]$.

Definition 23. Given a primitive measure $\sim_p$, a weighting function g, a discounting factor w, as well as two interpretations $\mathcal{I}$ and $\mathcal{J}$, the interpretation similarity measure $\sim_i \ (\sim_p, g, w) : \mathfrak{I} \times \mathfrak{I} \to [0, 1]$ is defined as follows, for all $p, q \in \mathfrak{I}$: If $\mathrm{CN}(p) = \mathrm{CN}(q) = \mathrm{SC}(p) = \mathrm{SC}(q) = \emptyset$, then $p \sim_i q = 1$, otherwise

$$p \sim_i q = \frac{\mathrm{cm}(p, q) + \mathrm{cm}(q, p) + \mathrm{sm}(p, q) + \mathrm{sm}(q, p)}{\sum_{A \in \mathrm{CN}(p)} g(A) + \sum_{A \in \mathrm{CN}(q)} g(A) + \sum_{(r,p') \in \mathrm{SC}(p)} g(r) + \sum_{(r,q') \in \mathrm{SC}(q)} g(r)} \tag{3.1}$$

♢

We often speak simply of $\sim_i$ instead of $\sim_i \ (\sim_p, g, w)$. In this case, we assume that $\sim_p$, g and w are either clear from the context, or arbitrary.

Note that, since sm depends again on $\sim_i$, this definition is mutually recursive. Simply put, the similarity of to pointed interpretations p and q will depend on the similarity between all its successor elements, which depend on the similarities of their successors and so on. Since interpretations can contain cycles, in order for $\sim_i$ to be well-defined we need to show that a unique solution to equation (3.1) exists.

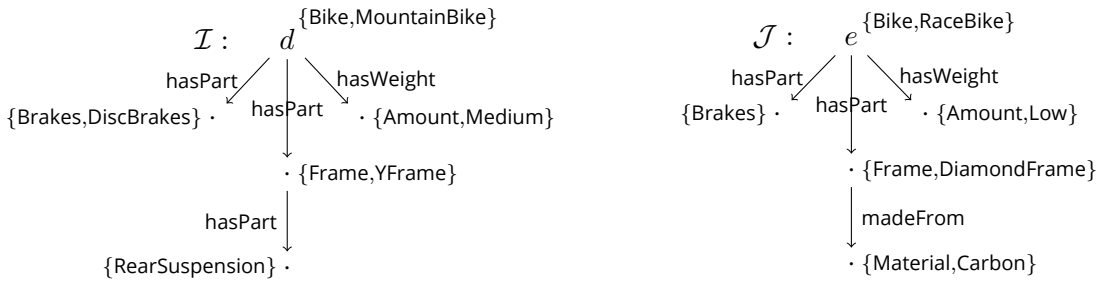

Figure 3.1: The pointed interpretations $(\mathcal{I}, d)$ and $(\mathcal{J}, e)$ from Example 24

Before proving that $\sim_i$ is well-defined, we will illustrate its definition using a brief example.

Example 24. Consider the interpretations $\mathcal{I}$ and $\mathcal{J}$ given in Figure 3.1. The pointed interpretation $p = (\mathcal{I}, d)$ describes a medium weight mountain bike with disc brakes and a Y-frame with rear suspension. The pointed interpretation $q = (\mathcal{J}, e)$ describes a road racing bike with unspecified brakes, a carbon diamond frame and low weight. In the following, we will use the default weighting function g_{default} that assigns weight 1 to all concept and role names, and a discounting factor of $w = 0.8$. We also use a primitive measure $\sim_p$ that agrees with the default primitive measure on all pairs of concept and role names with the exception of the following: The primitive similarity between Low and Medium as well as between Medium and High is 0.5 instead of 0.

To compute the similarity between the two pointed interpretations p and q, we need to find, for each concept name and each successor of any of the two elements, the best matching concept name or successor of the other element. For this we need the similarities of all successors of the elements d and e. Since both d and e have three successors, we would need to compute the similarity between all nine pairs. However, since hasPart and hasWeight have a primitive similarity of 0, and the brakes and frame have no concept names in common, it is easy to see that the best matching successors will be the ones that both speak about the weight, the brakes, or the frame, respectively.

- The hasPart-successors speaking about the brakes have a similarity 0.667, as both elements are instance of the concept name Brakes, but the successor of e is missing the concept name DiscBrake, resulting in a similarity value of $\frac{(1+0)+1}{3} = 0.667$.
- The most similar concept names for the two hasWeight-successor of d and e are $(\mathsf{Amount}, \mathsf{Amount})$ and $(\mathsf{Medium}, \mathsf{Low})$, which results in a similarity value of $\frac{(1+0.5)+(1+0.5)}{4} = 0.75$.
- For the two hasPart-successors of d and e speaking about the frame, both are instance of Frame, while the concept names YFrame and DiamondFrame have no correspondence in the other element. Similarly, the hasPart-successor in $\mathcal{I}$ and the madeFrom-successor in $\mathcal{J}$ have no corresponding successors the

other interpretation. Overall, this yields a similarity of $\frac{(1+0)+(1+0)+0+0}{2+2+1+1} = 0.333$ for the two services.

Using this, we can finally compute the similarity between d and e by computing cm and sm for both directions (p, q) and (q, p) and dividing by the sum of all weights:

$$\begin{aligned}
\text{cm}(p,q) &= \text{cm}(q,p) = 1+1,\\
\text{sm}(p,q) &= \text{sm}(q,p) = (0.2 + 0.8 \cdot 0.667) + (0.2 + 0.8 \cdot 0.75) + (0.2 + 0.8 \cdot 0.333)\\
&= 2,\\
(\mathcal{I}, d) \sim_i (\mathcal{I}, e) &= \frac{1+1+2+2}{2+2+3+3} = 0.6.
\end{aligned}$$

Despite the fact that these bikes are fairly different, the similarity between them is rather large. The main reason for this is that the general concept names like Bike, Amount, Brakes, and Frame have the same influence on the similarity as the more interesting subconcepts like MountainBike and RaceBike or Low and Medium. By modifying the weighting function g such that it decreases the weight of the concepts Bike, Amount, Brakes, and Frame to 0.1, we arrive at the following similarity values:

- The similarity between the brakes is now $\frac{(0.1+0)+0.1}{1.2} = 0.167$.
- The similarity between the weights is now $\frac{0.1+0.5+0.1+0.5}{2.2} = 0.545$.
- The similarity between the frames is now $\frac{0.2}{4.2} = 0.048$.
- The total similarity between the bikes is $p \sim_i q = 0.21$.

With this new weighting function, the similarity comes mainly from the fact that both bikes have a weight that is not completely dissimilar, and that they have the same roles (albeit with dissimilar successors), which the discounting formula honors with a factor of 0.2. ♢

3.1.2 Existence and Uniqueness of the Solution for $\sim_i$

In order to show that a unique solution to Equation (3.1) exists even if the interpretations contain cycles, we will use the Banach fixed-point theorem [Ban22]. For this, we need the notion of a contraction mapping.

Definition 25 (contraction mapping). Given a metric space (X, d), a function $f : X \to X$ is called a contraction mapping on X, if there exists a $\lambda \in [0, 1)$ such that for all $x, y \in X$ we have

$$d(f(x), f(y)) \leq \lambda d(x, y).$$ ♢

The Banach fixed-point theorem basically says that any contraction mapping on a complete metric space has a unique fixed point.

Theorem 26 (Banach fixed-point theorem [Ban22]). *Let (X, d) be a complete metric space, and $f : X \to X$ be a contraction mapping on X. Then f admits a unique fixed-point x^* in X, i.e., there exists a unique $x^* \in X$ with $f(x^*) = x^*$. Furthermore, for any $x_0 \in X$, the sequence x_n with $x_n = f(x_{n-1})$ converges to x^*, i.e., $x^* = \lim_{n\to\infty} x_n$.* ♢

In order to be able to use the Banach fixed-point theorem, we need to transform the equation system (3.1) into a (contraction) mapping. In particular, given two pointed interpretations $p = (\mathcal{I}, d)$ and $q = (\mathcal{J}, e)$ with $\mathcal{I}$ and $\mathcal{J}$ being finite, we need to consider the similarity values between all elements of $\mathcal{I}$ and $\mathcal{J}$, and thus we transform equation system (3.1) into a mapping $f_{\text{sim}} : \mathbb{R}^{\mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}} \to \mathbb{R}^{\mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}}$ (where $x_{p,q}$ denote the value at index p, q of vector $\boldsymbol{x}$):

$$f_{\text{sim}}(\boldsymbol{x}) = \boldsymbol{x'}, \tag{3.2}$$

where

$$x'_{p,q} = \frac{c(p,q) + c(q,p) + s_{\boldsymbol{x}}(p,q) + s_{\boldsymbol{x}}(q,p)}{\sum\limits_{A \in \text{CN}(p)} g(A) + \sum\limits_{B \in \text{CN}(q)} g(B) + \sum\limits_{(r,p') \in \text{SC}(p)} g(r) + \sum\limits_{(s,q') \in \text{SC}(q)} g(s)},$$

$$c(p,q) = \sum_{A \in \text{CN}(p)} \left(g(A) \max_{B \in \text{CN}(q)} (A \sim_p B) \right), \text{ and}$$

$$s_{\boldsymbol{x}}(p,q) = \sum_{(r,p') \in \text{SC}(p)} \left(g(r) \max_{(s,q') \in \text{SC}(q)} \left((r \sim_p s)((1-w) + w(x_{p',q'})) \right) \right).$$

Note that the fixed-points of f_{sim} and the solutions of Equation system (3.1) are equivalent, since for a fixed point $\boldsymbol{x}^*$, the equation $f_{\text{sim}}(\boldsymbol{x}^*) = \boldsymbol{x}^*$ yields exactly the equation system (3.1).

Given the Chebyshev distance metric $d_{\max}$ on $\mathbb{R}^{\mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}}$ with

$$d_{\max}(\boldsymbol{x}, \boldsymbol{y}) = \max_{(p,q) \in \mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}} |x_{p,q} - y_{p,q}|,$$

we can now show that f_{sim} is indeed a contraction mapping on the complete metric space $(\mathbb{R}^{\mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}}, d_{\max})$.

Lemma 27. *f_{sim} is a contraction mapping on $(\mathbb{R}^{\mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}}, d_{\max})$.*

Proof. We need to show that for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{\mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}}$, we have

$$d_{\max}(f_{\text{sim}}(\boldsymbol{x}), f_{\text{sim}}(\boldsymbol{y})) \leq \lambda d_{\max}(\boldsymbol{x}, \boldsymbol{y})$$

for some $\lambda \in [0, 1)$. In order to show this, we need the following claim:

Claim 28. *For some index set I and two vectors $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^I$, we have*

$$|\max_{i \in I} x_i - \max_{i \in I} y_i| \leq \max_{i \in I} |x_i - y_i|.$$

Proof. Let $\max_{i \in I} x_i = x_{i_k}$ and $\max_{i \in I} y_i = y_{i_l}$. Assume that $x_{i_k} \geq y_{i_l}$; then $|\max_{i \in I} x_i - \max_{i \in I} y_i| = |x_{i_k} - y_{i_l}| = x_{i_k} - y_{i_l}$. As y_{i_l} is the maximum over all y_i, we further get $x_{i_k} - y_{i_l} \leq x_{i_k} - y_{i_k} \leq \max_{i \in I} |x_i - y_i|$. The case $x_{i_k} < y_{i_l}$ is analogous. □

Let now $\boldsymbol{x'} = f_{\text{sim}}(\boldsymbol{x})$ and $\boldsymbol{y'} = f_{\text{sim}}(\boldsymbol{y})$. Then we have $d_{\max}(f_{\text{sim}}(\boldsymbol{x}), f_{\text{sim}}(\boldsymbol{y})) = \max_{(p,q) \in \mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}} |x'_{p,q} - y'_{p,q}|$. In particular, using claim 28, we have for any $p, q \in \mathfrak{I}$:

$$
\begin{aligned}
s_{\boldsymbol{x}}(p,q) - s_{\boldsymbol{y}}(p,q) = \Bigg| & \sum_{(r,p')\in \mathsf{SC}(p)} g(r) \left(\max_{(s,q')\in \mathsf{SC}(q)} (r \sim_p s)((1-w) + w(x_{p',q'})) \right) \\
& \qquad - \sum_{(r,p')\in \mathsf{SC}(p)} g(r) \left(\max_{(s,q')\in \mathsf{SC}(q)} (r \sim_p s)((1-w) + w(y_{p',q'})) \right) \Bigg| \\
\leq & \sum_{(r,p')\in \mathsf{SC}(p)} g(r) \max_{(s,q')\in \mathsf{SC}(q)} \big| (r \sim_p s)((1-w) + w(x_{p',q'})) \\
& \qquad - (r \sim_p s)((1-w) + w(y_{p',q'})) \big| \\
= & \sum_{(r,p')\in \mathsf{SC}(p)} g(r) \left(w \max_{(s,q')\in \mathsf{SC}(q)} (r \sim_p s) \left| x_{p',q'} - y_{p',q'} \right| \right) \\
\leq & \sum_{(r,p')\in \mathsf{SC}(p)} g(r) \left(w \max_{(s,q')\in \mathsf{SC}(q)} \left| x_{p',q'} - y_{p',q'} \right| \right) \\
\leq & \sum_{(r,p')\in \mathsf{SC}(p)} g(r) \left(w \, d_{\mathsf{max}}(\boldsymbol{x}, \boldsymbol{y}) \right)
\end{aligned}
$$

This implies the following:

$$
\begin{aligned}
|x'_{p,q} - y'_{p,q}| = \Bigg| & \frac{(c(p,q) + c(q,p) + s_{\boldsymbol{x}}(p,q) + s_{\boldsymbol{x}}(q,p))}{\sum_{A\in \mathsf{CN}(p)} g(A) + \sum_{B\in \mathsf{CN}(q)} g(B) + \sum_{(r,p')\in \mathsf{SC}(p)} g(r) + \sum_{(s,q')\in \mathsf{SC}(q)} g(s)} \\
& \qquad - \frac{(c(p,q) + c(q,p) + s_{\boldsymbol{y}}(p,q) + s_{\boldsymbol{y}}(q,p))}{\sum_{A\in \mathsf{CN}(p)} g(A) + \sum_{B\in \mathsf{CN}(q)} g(B) + \sum_{(r,p')\in \mathsf{SC}(p)} g(r) + \sum_{(s,q')\in \mathsf{SC}(q)} g(s)} \Bigg| \\
\leq & \frac{|(s_{\boldsymbol{x}}(p,q) - s_{\boldsymbol{y}}(p,q)) + (s_{\boldsymbol{x}}(q,p) - s_{\boldsymbol{y}}(q,p))|}{\sum_{A\in \mathsf{CN}(p)} g(A) + \sum_{B\in \mathsf{CN}(q)} g(B) + \sum_{(r,p')\in \mathsf{SC}(p)} g(r) + \sum_{(s,q')\in \mathsf{SC}(q)} g(s)} \\
\leq & \frac{\sum_{(r,p')\in \mathsf{SC}(p)} g(r) \, (w \, d_{\mathsf{max}}(\boldsymbol{x}, \boldsymbol{y})) + \sum_{(s,q')\in \mathsf{SC}(q)} g(s) \, (w \, d_{\mathsf{max}}(\boldsymbol{x}, \boldsymbol{y}))}{\sum_{A\in \mathsf{CN}(p)} g(A) + \sum_{B\in \mathsf{CN}(q)} g(B) + \sum_{(r,p')\in \mathsf{SC}(p)} g(r) + \sum_{(s,q')\in \mathsf{SC}(q)} g(s)} \\
\leq & \, w \, d_{\mathsf{max}}(\boldsymbol{x}, \boldsymbol{y}) \frac{\sum_{(r,p')\in \mathsf{SC}(p)} g(r) + \sum_{(s,q')\in \mathsf{SC}(q)} g(s)}{\sum_{A\in \mathsf{CN}(p)} g(A) + \sum_{B\in \mathsf{CN}(q)} g(B) + \sum_{(r,p')\in \mathsf{SC}(p)} g(r) + \sum_{(s,q')\in \mathsf{SC}(q)} g(s)} \\
\leq & \, w \, d_{\mathsf{max}}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

Since $|x'_{p,q} - y'_{p,q}| \leq w \, d_{\mathsf{max}}(\boldsymbol{x}, \boldsymbol{y})$ for any $p, q \in \mathfrak{I}$, this implies that also

$$
d_{\mathsf{max}}(f_{\mathsf{sim}}(\boldsymbol{x}), f_{\mathsf{sim}}(\boldsymbol{y})) = \max_{(p,q)\in \mathfrak{I}_{\mathcal{I}} \times \mathfrak{I}_{\mathcal{J}}} |x'_{p,q} - y'_{p,q}| \leq w \, d_{\mathsf{max}}(\boldsymbol{x}, \boldsymbol{y}),
$$

and thus f_{sim} is a contraction mapping with Lipschitz constant w. □

This allows us to apply the Banach fixed-point theorem to get the following result.

Theorem 29. *The similarity measure $\sim_i$ is well-defined, i.e., $p \sim_i q$ defined in Equation* (3.1) *has a unique solution for all finite pointed interpretations p, q.*

Proof. f_{sim} is a contraction mapping on $(\mathbb{R}^{\mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}}, d_{\max})$ due to Lemma 27. Thus, the Banach fixed-point Theorem 26 implies that f_{sim} always has a unique fixed-point $\boldsymbol{x}^*$. Since the fixed-point equation $f_{\text{sim}}(\boldsymbol{x}^*) = \boldsymbol{x}^*$ yields exactly the equation system (3.1), this also implies that equation system (3.1) always has a unique solution, and thus the similarity measure $\sim_i$ is well-defined. □

3.1.3 Polynomial Time Complexity

In this section, we want to show that $\sim_i$ can be computed in polynomial time in the size of the interpretations $\mathcal{I}$ and $\mathcal{J}$. For this, we transform the equation system (3.1) into a linear optimization problem P_{sim}:

Definition 30. Let $\mathcal{I}$ and $\mathcal{J}$ be finite two interpretations. The linear optimization problem $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$ is defined by the following linear constraints with variables $x_{p,q}$, $x_{(r,p'),q}$, and $x_{p,(s,q')}$ for all $p \in \mathfrak{I}_\mathcal{I}$, $q \in \mathfrak{I}_\mathcal{J}$, $(r, p') \in \text{SC}(p)$ and $(s, q') \in \text{SC}(q)$:

$$x_{p,q} = \frac{\text{cm}(p,q) + \text{cm}(q,p) + \sum\limits_{(r,p') \in \text{SC}(p)} x_{(r,p'),q} + \sum\limits_{(s,q') \in \text{SC}(q)} x_{p,(s,q')}}{\sum\limits_{A \in \text{CN}(p)} g(A) + \sum\limits_{B \in \text{CN}(q)} g(B) + \sum\limits_{(r,p') \in \text{SC}(p)} g(r) + \sum\limits_{(s,q') \in \text{SC}(q)} g(s)}$$

$$\text{cm}(p,q) = \sum_{A \in \text{CN}(p)} \left(g(A) \max_{B \in \text{CN}(q)} (A \sim_p B) \right)$$

$$x_{(r,p'),q} \geq g(r)(r \sim_p s)((1-w) + w \cdot x_{p',q'}) \qquad \text{for all } (s,q') \in \text{SC}(q)$$

$$x_{q,(s,q')} \geq g(s)(r \sim_p s)((1-w) + w \cdot x_{p',q'}) \qquad \text{for all } (r,p') \in \text{SC}(p)$$

and the objective function

$$\sum_{(p,q) \in \mathfrak{I}_\mathcal{I} \times \mathfrak{I}_\mathcal{J}} x_{p,q},$$

which should be minimized. ♢

In $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$, the variables $x_{p,q}$ correspond to $p \sim_i q$, while variables $x_{(r,p'),q}$ (and $x_{p,(s,q')}$, respectively) correspond to $\text{sm}(p,q)$ without the outer sum. By stating the constraints that $x_{(r,p'),q}$ must be greater or equal to the given expression for all successors $(s, q') \in \text{SC}(q)$, minimizing the objective function will result in the variables $x_{(r,p'),q}$ being assigned exactly the maximum of all those expressions.

In fact, the values for variables $x_{p,q}$ in the minimal solution to $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$ are exactly the similarity values $p \sim_i q$ for all $p \in \mathfrak{I}_\mathcal{I}$ and $q \in \mathfrak{I}_\mathcal{J}$.

Lemma 31. *Let $\mathcal{I}$ and $\mathcal{J}$ be two finite interpretations, and $p \in \mathfrak{I}_\mathcal{I}$, $q \in \mathfrak{I}_\mathcal{J}$ be pointed interpretations. Then any minimal solution of P_{sim} corresponds to a solution of equation system* (3.1) *and vice versa.*

Proof. We have to show that every solution to equation system (3.1) satisfies all constraints of the linear optimization problem $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$, and that this solution is indeed minimal. Since equation system (3.1) and the feasible region of $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$ is convex, this implies that the solution of equation system (3.1) and the minimal solution of $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$ coincide.

1. By setting

$$\begin{aligned} x_{p,q} &= p \sim_i q, \\ x_{(r,p'),q} &= g(r) \max_{(s,q') \in \text{SC}(q)} \Big((r \sim_p s)\big((1-w) + w(p' \sim_i q')\big) \Big), \text{ and} \\ x_{p,(s,q')} &= g(s) \max_{(r,p') \in \text{SC}(p)} \Big((r \sim_p s)\big((1-w) + w(p' \sim_i q')\big) \Big) \end{aligned}$$

   for all $p \in \mathfrak{I}_\mathcal{I}$, $q \in \mathfrak{I}_\mathcal{J}$, $(r, p') \in \text{SC}(p)$ and $(s, q') \in \text{SC}(q)$ it is easy to see that all constraints of $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$ are satisfied.

2. Let $\boldsymbol{v}$ be a minimal solution to $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$, that assigns to each variable $x_{p,q}$, $x_{(r,p'),q}$, and $x_{p,(s,q')}$ the value $v_{p,q}$, $v_{(r,p'),q}$, and $v_{p,(s,q')}$, respectively.

   Claim 32. *For each of the variables $x_{(r,p'),q}$ ($x_{p,(s,q')}$ analogously), one of the inequalities*

$$v_{(r,p'),q} \geq g(r)(r \sim_p s)((1-w) + w \cdot v_{p',q'})$$

   holds exactly, i.e.,

$$v_{(r,p'),q} = g(r)(r \sim_p s)((1-w) + w \cdot v_{p',q'})$$

   for some $(s, q') \in SC(q)$.

   Proof. Assume to the contrary that there is a variable $x_{(r,p'),q}$ ($x_{p,(s,q')}$ analogously) with $v_{(r,p'),q} > g(r)(r \sim_p s)((1-w) + w \cdot v_{p',q'})$. But then setting

$$v_{(r,p'),q} := \max_{(s,q') \in \text{SC}(q)} \big(g(r)(r \sim_p s)((1-w) + w \cdot v_{p',q'}) \big)$$

   and updating those values $v_{p,q}$ with $(r, p') \in \text{SC}(p)$ with

$$v_{p,q} := \frac{\text{cm}(p,q) + \text{cm}(q,p) + \sum\limits_{(r,p') \in \text{SC}(p)} v_{(r,p'),q} + \sum\limits_{(s,q') \in \text{SC}(q)} v_{p,(s,q')}}{\sum\limits_{A \in \text{CN}(p)} g(A) + \sum\limits_{B \in \text{CN}(q)} g(B) + \sum\limits_{(r,p') \in \text{SC}(p)} g(r) + \sum\limits_{(s,q') \in \text{SC}(q)} g(s)},$$

   so that the equations for $x_{p,q}$ are satisfied again yields a smaller solution that still satisfies all equalities and inequalities in $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$. This is a contradiction to the fact that v is a minimal solution. □

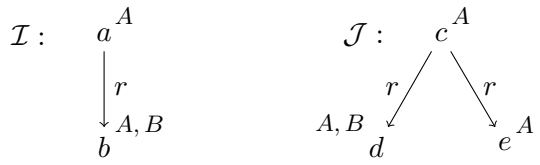

Figure 3.2: Two equisimilar interpretations $\mathcal{I}$ and $\mathcal{J}$ for which $(\mathcal{I}, a) \sim_i (\mathcal{J}, c) \neq 1$.

Using this claim, we can show that each minimal solution to $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$ also satisfies equation system (3.1). This is easy to see, as Claim 32 implies that for any variable $x_{(r,p'),q}$, we indeed have

$$v_{(r,p'),q} = \max_{(s,q') \in \text{SC}(q)} \big(g(r)(r \sim_p s)((1-w) + w \cdot v_{p',q'})\big)$$

(analogous for $x_{p,(s,q')}$). Since equation system (3.1) has a unique solution as shown in Theorem 29, this solution must be exactly the same as the minimal solution for $P_{\text{sim}}(\mathcal{I}, \mathcal{J})$. □

This finally gives us a polynomial time computation procedure for the interpretation similarity measure $\sim_i$.

Corollary 33. *Given two finite interpretations $\mathcal{I}$ and $\mathcal{J}$, all interpretation similarities $(\mathcal{I}, d) \sim_i (\mathcal{J}, e)$ for $e \in \Delta^{\mathcal{I}}$ and $d \in \Delta^{\mathcal{J}}$ can be computed simultaneously in time polynomial in the size of $\mathcal{I}$ and $\mathcal{J}$.*

Proof. This follows directly from Lemma 31, the fact that the linear optimization problem in Definition 30 is of polynomial size, and that linear optimization problems can be solved in polynomial time [Kar84]. □

3.1.4 Properties of $\sim_i$

The measure $\sim_i$ is not equisimulation-closed or equisimulation-invariant. This is easy to see for the interpretations $\mathcal{I}$ and $\mathcal{J}$ given in Figure 3.2. In this example, $(\mathcal{I}, a) \simeq (\mathcal{J}, c)$, but $(\mathcal{I}, a) \sim_i (\mathcal{J}, c) \neq 1$, as it depends on $(\mathcal{I}, b) \sim_i (\mathcal{J}, e)$, which is less than 1 since b is an instance of B, but e is not. In order to regain equisimulation-closure and equisimulation-invariance, we will introduce a normalization procedure for interpretations, which in the example above would remove the edge (c, e) from $\mathcal{J}$ after which $(\mathcal{I}, a) \sim_i (\mathcal{J}, c) = 1$ does hold.

Definition 34 (normal form for interpretations). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is in *normal form* if for all elements $a, b, c \in \Delta^{\mathcal{I}}$, $\{(a, b), (a, c)\} \subseteq r^{\mathcal{I}}$ and $(\mathcal{I}, b) \lesssim (\mathcal{I}, c)$ implies $b = c$, i.e., no node has two successor nodes for the same role name that are in a simulation relation. ♢

Any interpretation $\mathcal{I}$ can be transformed into normal form as follows:

1. For all edges $(a, b_0) \in r^{\mathcal{I}}$, check if there are other edges $(a, b_i) \in r^{\mathcal{I}}$, $i > 0$, with $(\mathcal{I}, b_0) \simeq (\mathcal{I}, b_i)$ and choose one representative b_j; then remove all other edges (a, b_i), $i \neq j$, from $r^{\mathcal{I}}$.
2. Remove all edges $(a, b) \in r^{\mathcal{I}}$ in the interpretation graph, for which there exists an edge $(a, c) \in r^{\mathcal{I}}$ with $(\mathcal{I}, b) \lesssim (\mathcal{I}, c)$.

Equisimilar pointed interpretations will always be normalized into a unique structural normal form, i.e., both pointed interpretations will have the same number of pairwise equisimilar successors. This is true even though the normalization steps given above are nondeterministic.

We say that two pointed interpretations p and q are *structurally equivalent*) if for any successor $(r, p') \in \mathrm{SC}(p)$ there exists a unique successor $(r, q') \in \mathrm{SC}(q)$ with $p' \simeq q'$ and vice versa.

Lemma 35. *Let $(\mathcal{I}, a)$ and $(\mathcal{J}, b)$ be two pointed interpretations and let $\mathcal{I}'$ and $\mathcal{J}'$ be the results of normalizing $\mathcal{I}$ and $\mathcal{J}$, respectively. Then the following holds:*

1. *Normalization preserves simulations, i.e., if $(\mathcal{I}, a) \lesssim (\mathcal{J}, b)$ then also $(\mathcal{I}', a) \lesssim (\mathcal{J}', b)$.*
2. *If $(\mathcal{I}, a) \simeq (\mathcal{J}, b)$, then $(\mathcal{I}, a) \simeq (\mathcal{J}, b)$ are structurally equivalent.*

Proof.

1. Let $(\mathcal{I}, a)$ and $(\mathcal{J}, b)$ be two pointed interpretations with $(\mathcal{I}, a) \lesssim (\mathcal{J}, b)$. Then for each concept name A, we have $a \in A^{\mathcal{I}'} \Leftrightarrow a \in A^{\mathcal{I}} \Rightarrow b \in A^{\mathcal{J}} \Leftrightarrow b \in A^{\mathcal{J}'}$. Additionally, for each role name r, we have $(a, a') \in r^{\mathcal{I}'} \Rightarrow (a, a') \in r^{\mathcal{I}} \Rightarrow \exists b' : (b, b') \in r^{\mathcal{J}} \wedge (\mathcal{I}, a') \lesssim (\mathcal{J}, b')$. If $(b, b') \in r^{\mathcal{J}'}$, we are done: $(\mathcal{I}', a) \lesssim (\mathcal{J}', b)$ follows directly.

   Otherwise, we know by the construction of $\mathcal{J}'$, that there exists an element $c \in \Delta^{\mathcal{J}'}$ with $(b, c) \in r^{\mathcal{J}'}$ and $(\mathcal{J}', b') \lesssim (\mathcal{J}', c)$ or $(\mathcal{J}', b') \simeq (\mathcal{J}', c)$. Since $\lesssim$ is transitive and $(\mathcal{I}', a') \lesssim (\mathcal{I}, a) \lesssim (\mathcal{J}, c)$, this means that $(\mathcal{I}', a') \lesssim (\mathcal{J}', c)$ and the claim that $(\mathcal{I}', a) \lesssim (\mathcal{J}', b)$ again follows.
2. Let $(\mathcal{I}, a)$ and $(\mathcal{J}, b)$ be two pointed interpretations with $(\mathcal{I}, a) \simeq (\mathcal{J}, b)$. Let further $(a, c) \in r^{\mathcal{I}'}$, which implies that also $(a, c) \in r^{\mathcal{I}}$. Since $\mathcal{I}'$ is in normal form, this means that there is no $c' \in \Delta^{\mathcal{I}}$ with $(a, c') \in r^{\mathcal{I}}$ and $(\mathcal{I}, c) \lesssim (\mathcal{I}, c')$, and $(\mathcal{I}, c') \not\lesssim (\mathcal{I}, c)$. Since $(\mathcal{I}, a) \simeq (\mathcal{J}, b)$, there exists an element $d \in \Delta^{\mathcal{J}}$ with $(b, d) \in r^{\mathcal{J}}$ and $(\mathcal{I}, c) \lesssim (\mathcal{J}, d)$, but not necessarily $(b, d) \in r^{\mathcal{J}'}$. By the construction of $\mathcal{J}'$, we know that there is an element $e \in \Delta^{\mathcal{J}'}$ with $(b, e) \in r^{\mathcal{J}'}$ and $(\mathcal{J}, d) \lesssim (\mathcal{J}, e)$. Again, $(\mathcal{I}, a) \simeq (\mathcal{J}, b)$ implies that a must have a successor $(a, f) \in r^{\mathcal{I}}$ with $(\mathcal{J}, e) \lesssim (\mathcal{I}, f)$; however, since with $(\mathcal{I}, c) \lesssim (\mathcal{J}, d)$ and $(\mathcal{J}, d) \lesssim (\mathcal{J}, e)$, this also means $(\mathcal{I}, c) \lesssim (\mathcal{I}, f)$. Since we know that there is no $c' \in \Delta^{\mathcal{I}}$ with $(a, c') \in r^{\mathcal{I}}$ and $(\mathcal{I}, c) \lesssim (\mathcal{I}, c')$, this means that $f = c$ and thus $(\mathcal{I}, c) \simeq (\mathcal{J}, e)$ and by point 1. also $(\mathcal{I}', c) \simeq (\mathcal{J}', e)$. e must be the unique successor of this kind since $\mathcal{J}'$ is in normal form and thus there cannot be an $f' \in \Delta^{\mathcal{J}'}$ with $(b, f') \in r^{\mathcal{J}'}$ and $(\mathcal{J}', e) \simeq (\mathcal{J}', f')$. The other direction is analogous. □

Since we only consider finite pointed interpretations, the normalization procedure is well-defined and can be computed in polynomial time in the size of the pointed interpretation, as simulations can also be computed in P-time.

Now, we can finally show the properties of the ISM $\sim_i$. For the default primitive measure, $\sim_i$ is symmetric, bounded, dissimilar-closed, equisimulation-invariant, and equisimulation-closed for all normalized pointed interpretations. For other primitive measures $\sim_p$, $\sim_i$ will always be bounded and equisimulation-invariant, but the other properties depend on the properties of $\sim_p$, as given by the following theorem.

Theorem 36. *Let $\sim_i(\sim_p, g, w)$ be instantiated with a primitive measure $\sim_p$, a weighting function g, and discounting factor $w \in (0, 1)$. Then $\sim_i$ has the following properties:*

1. *$\sim_i$ is symmetric, if the primitive measure $\sim_p$ is symmetric;*
2. *$\sim_i$ is bounded;*
3. *$\sim_i$ is dissimilar-closed, if the primitive measure $\sim_p$ does not assign a similarity value greater than 0 to different concept or role names.*
4. *$\sim_i$ is equisimulation-invariant for normalized interpretations; and*
5. *$\sim_i$ is equisimulation-closed for normalized interpretations, if the primitive measure $\sim_p$ does not assign the similarity value 1 to different concept or role names.*

Proof.

1. *symmetric:* $\sim_i$ is symmetric, if the primitive measure $\sim_p$ is symmetric, as the definition of $\sim_i$ only uses commutative operators.
2. *bounded:* $\sim_i$ is bounded, if $\mathfrak{C}(p) \cap \mathfrak{C}(q) \supsetneq \{\top\}$ implies $p \sim_i q > 0$ for all $p, q \in \mathfrak{I}$. Assume that there exists a concept $C \neq \top$ in $\mathfrak{C}(p) \cap \mathfrak{C}(q)$. Then, there also exists either a concept name A or an existential restriction of the form $\exists r.\top$ in $\mathfrak{C}(p) \cap \mathfrak{C}(q)$ since, for all conjunctions $C_1 \sqcap C_2 \in \mathfrak{C}(p) \cap \mathfrak{C}(q)$ we also have $C_1, C_2 \in \mathfrak{C}(p) \cap \mathfrak{C}(q)$ and for all $\exists r.C \in \mathfrak{C}(p) \cap \mathfrak{C}(q)$ we also have $\exists r.\top \in \mathfrak{C}(p) \cap \mathfrak{C}(q)$.

   However, for a concept name $A \in \mathfrak{C}(p) \cap \mathfrak{C}(q)$, we have that $A \sim_p A = 1$ and thus
   $$\sum_{A \in \mathsf{CN}(p)} \left(g(A) \max_{B \in \mathsf{CN}(q)} (A \sim_p B) \right) > 0.$$
   This yields $p \sim_i q > 0$. Correspondingly, for $\exists r.\top \in \mathfrak{C}(p) \cap \mathfrak{C}(q)$, we have $r \sim_p r = 1$ and thus $(r \sim_p r)((1-w) + w(p' \sim_i q')) > 1 - w > 0$ and
   $$\sum_{(s,q') \in \mathsf{SC}(p)} \left(g(r) \max_{(p,q) \in \mathsf{SC}(q)} (r \sim_p s)((1-w) + w(p' \sim_i q')) \right) > 0.$$
   Again, this yields $p \sim_i q > 0$.

3. *dissimilar-closed:* $\sim_i$ is dissimilar-closed, if $\mathfrak{C}(p) \cap \mathfrak{C}(q) = \{\top\}$ implies $p \sim_i q = 0$ for all $p, q \in \mathfrak{I}$ with $\mathfrak{C}(p) \supsetneq \{\top\}$ and $\mathfrak{C}(q) \supsetneq \{\top\}$; of course, $\sim_i$ can only be dissimilarity-closed if the primitive measure does not assign a similarity value greater than 0 to different concept or role names. Hence we only show this property for the default primitive measure $\sim_{\text{default}}$.

   Let $p, q \in \mathfrak{I}$ with $\mathfrak{C}(p) \supsetneq \{\top\}$ and $\mathfrak{C}(q) \supsetneq \{\top\}$, i.e., both p and q are instance of some concept name or have a successor. If $\mathfrak{C}(p) \cap \mathfrak{C}(q) = \{\top\}$, then $A \sim_{\text{default}} B = 0$ for all $A \in \text{CN}(p)$ and $B \in \text{CN}(q)$. Similarly, as there is no role name r with $(r, p') \in \text{SC}(p)$ and $(r, q') \in \text{SC}(q)$, we have $r \sim_{\text{default}} s = 0$ for all $(r, p') \in S(p)$ and $(s, q') \in S(q)$. This then yields $p \sim_i q = 0$.

4. *equisimulation-invariant:* $\sim_i$ is equisimulation-invariant for normalized interpretations, if $p \simeq q$ implies $p \sim_i u = q \sim_i u$ for all normalized pointed interpretations $p, q, u \in \mathfrak{I}$; it is a direct consequence of the fact that if $p \simeq q$, then the normalized pointed interpretations do not just simulate each other, but are structurally equivalent, as stated in Point 2 in Lemma 35. Thus the computation of $p \sim_i u$ can be modified to compute $q \sim_i u$ by simply replacing the successors of p by the unique equisimilar successors of q and vice versa; this will always yield the same similarity value.

5. *equisimulation-closed:* The direction from left to right, i.e., $p \simeq q$ implies $p \sim_i q = 1$, follows again by Point 2 in Lemma 35. For the other direction, that $p \sim_i q = 1$ also implies $p \simeq q$, we need the property that the primitive measure does not assign a similarity value of 1 to different concept or role names. In this case, assume that $p \not\simeq q$ for $p = (\mathcal{I}, a)$ and $q = (\mathcal{J}, b)$. Then, w.l.o.g., we have one of the following conditions:

   a) there exists a concept name A with $a \in A^{\mathcal{I}}$ and $b \notin A^{\mathcal{J}}$, or

   b) a has a successor $(a, c) \in r^{\mathcal{I}}$ and there is no d with $(b, d) \in r^{\mathcal{J}}$, or

   c) a has a successor $(a, c) \in r^{\mathcal{I}}$ and for all successors $t = (\mathcal{J}, d)$ of b with $(b, d) \in r^{\mathcal{J}}$ we have that $s = (\mathcal{I}, c) \not\simeq t$. In this case, there must be a finite chain of such successors s_i, t_i starting from a, b such that condition 1 or 2 holds for s_n, t_n.

   Now, we can prove inductively that $p \sim_i q < 1$. In the first two cases a) and b), Equation 3.1 directly gives a similarity value < 1, since the concept name A in case a) or the role name r in case b) will always be matched with a different concept or role name and $\sim_p$ never assigns similarity 1 to different concept or role names. In the third case, we assume that $c \sim_i d < 1$ by induction for all successors d of b. Then Equation 3.1 again yields a similarity value $p \sim_i q < 1$. Thus $\sim_i$ must equisimulation-closed. □

Note that $\sim_i$ is dissimilar-closed only if the primitive measure always assigns value 0 to different names, which is a huge restriction. Indeed, dissimilar-closure is not always a desirable property, since it implies that primitive concept names

should always be completely dissimilar (i.e., have similarity 0). In practice however, every ontology needs a certain amount of abstraction, so concepts may appear primitive in the ontology even though they could have been defined using even further and thus may have hidden similarities. For example, one may not want to define the exact chemical structure of ethanol and mineral oil because this is unnecessary for the given application, but still consider both substances to be slightly similar as they both are organic compounds. In this case dissimilar-closure is an unwanted property that is lost by giving ethanol and mineral oil a primitive similarity larger than 0.

With the interpretation similarity $\sim_i$ in place, we can finally define the CSM $\sim_c$.

3.2 The Concept Similarity Measure $\sim_c$

Given a TBox $\mathcal{T}$, we can measure the similarity between concepts C and D by applying $\sim_i$ to the normalized canonical models for the concepts w.r.t. $\mathcal{T}$. We define the concept similarity measure $\sim_c$ as follows:

Definition 37. Given a TBox $\mathcal{T}$ and an interpretation similarity measure $\sim_i$ $(\sim_p, g, w)$, the concept similarity measure $\sim_c$ is defined as follows:

$$C \sim_c D = \mathcal{I}'_{C,\mathcal{T}} \sim_i \mathcal{I}'_{D,\mathcal{T}},$$

where $\mathcal{I}'_{C,\mathcal{T}}$ and $\mathcal{I}'_{D,\mathcal{T}}$ are the normalized canonical models of C and D w.r.t. $\mathcal{T}$, respectively. ♢

Since the canonical models can be constructed and normalized in polynomial time, and the interpretations similarity $\sim_i$ can be computed in P-time as well, we get the following result:

Corollary 38. *The similarity $C \sim_c D$ w.r.t. $\mathcal{T}$ can be computed in polynomial time in the size of C, D, and $\mathcal{T}$.*

Proof. Follows from the fact that canonical models can be constructed and normalized in polynomial time and Corollary 33. □

We will show that the concept similarity measure $\sim_c$ indeed satisfies all the properties given in Section 2.3.2.

The concept similarity measure $\sim_c$ inherits the formal properties of the ISM $\sim_i$, since the properties for interpretation similarity measures were defined to correspond exactly to the properties for concept similarity measures given in the preliminaries.

Theorem 39 (Properties of $\sim_c$). *For a primitive measure $\sim_p$, a weighting function g, and a discounting factor w, the concept similarity measure $\sim_c(\sim_p, g, w)$ is symmetric, bounded, dissimilar-closed, equivalence-invariant, and equivalence-closed, if $\sim_i(\sim_p, g, w)$ is symmetric, bounded dissimilar-closed, equisimulation-invariant and equisimulation-closed, respectively.*

Proof. We show that the properties of $\sim_i$ transfer to $\sim_c$:

1. symmetry: $C \sim_c D = (\mathcal{I}'_{C,\mathcal{T}}, d_C) \sim_i (\mathcal{I}'_{D,\mathcal{T}}, d_D) = (\mathcal{I}'_{D,\mathcal{T}}, d_D) \sim_i (\mathcal{I}'_{C,\mathcal{T}}, d_C) = D \sim_c C$ follows from the symmetry of $\sim_i$.
2. bounded: Assume that for two $\mathcal{EL}$-concept C and D, there exists a concept $E \neq \top$ with $\mathcal{T} \models C \sqsubseteq E$ and $\mathcal{T} \models D \sqsubseteq E$. Then Theorem 13 and Lemma 35 yield $E \in \mathfrak{C}(p) \cap \mathfrak{C}(q)$ for $p = (\mathcal{I}'_{C,\mathcal{T}}, d_C)$ and $q = (\mathcal{I}'_{D,\mathcal{T}}, d_D)$. Therefore boundedness of $\sim_i$ implies $C \sim_c D = p \sim_i q > 0$.
3. dissimilar-closed: Assume that for two $\mathcal{EL}$-concept $C, D \neq \top$, there is no concept $E \neq \top$ with $\mathcal{T} \models C \sqsubseteq E$ and $\mathcal{T} \models D \sqsubseteq E$. Then Theorem 13 and Lemma 35 imply that $\mathfrak{C}(p) \cap \mathfrak{C}(q) = \{\top\}$ for $p = (\mathcal{I}'_{C,\mathcal{T}}, d_C)$ and $q = (\mathcal{I}'_{D,\mathcal{T}}, d_D)$, and thus, if we assume that $\sim_i$ is dissimilar-closed, $C \sim_c D = p \sim_i q = 0$.
4. equivalence-invariant: Assume that $C \equiv_{\mathcal{T}} D$. Then by Theorem 13 and Lemma 35 we have $(\mathcal{I}'_{C,\mathcal{T}}, d_C) \simeq (\mathcal{I}'_{D,\mathcal{T}}, d_D)$ and thus equisimulation-invariance of $\sim_i$ implies $(\mathcal{I}'_{C,\mathcal{T}}, d_C) \sim_i (\mathcal{J}, e) = (\mathcal{I}'_{D,\mathcal{T}}, d_D) \sim_i (\mathcal{J}, e)$ for any pointed interpretation $(\mathcal{J}, e)$, in particular pointed interpretations of the form $(\mathcal{I}'_{E,\mathcal{T}}, d_E)$. This then yields $C \sim_c E = D \sim_c E$ for any $\mathcal{EL}$-concept E.
5. equivalence-closed: Assume that $C \equiv_{\mathcal{T}} D$. Then by Theorem 13 and Lemma 35 we have $(\mathcal{I}'_{C,\mathcal{T}}, d_C) \simeq (\mathcal{I}'_{D,\mathcal{T}}, d_D)$ and thus $(\mathcal{I}'_{C,\mathcal{T}}, d_C) \sim_i (\mathcal{I}'_{D,\mathcal{T}}, d_D) = 1$ if $\sim_i$ is equisimulation-closed. But then we also have $C \sim_c D = 1$.

   Similarly, assume that $C \sim_c D = (\mathcal{I}'_{C,\mathcal{T}}, d_C) \sim_i (\mathcal{I}'_{D,\mathcal{T}}, d_D) = 1$. Then $(\mathcal{I}'_{C,\mathcal{T}}, d_C) \simeq (\mathcal{I}'_{D,\mathcal{T}}, d_D)$ since $\sim_i$ is equisimulation-closed, and thus Theorem 13 yields $C \equiv_{\mathcal{T}} D$. □

The following example illustrates the CSM $\sim_c$.

Example 40. Consider the TBox $\mathcal{T}$ describing knowledge about bikes defined as follows:

$$\begin{aligned}\mathcal{T} = \{&C \sqsubseteq \mathsf{Bike} \text{ for } C \in \{\mathsf{MountainBike}, \mathsf{RaceBike}\},\\ &C \sqsubseteq \mathsf{Frame} \text{ for } C \in \{\mathsf{YFrame}, \mathsf{DiamondFrame}\},\\ &C \sqsubseteq \mathsf{Material} \text{ for } C \in \{\mathsf{Low}, \mathsf{Medium}, \mathsf{High}\}\\ &\mathsf{Carbon} \sqsubseteq \mathsf{Material},\\ &\mathsf{Bike} \sqsubseteq \exists \mathsf{hasPart}.\mathsf{Brakes}\}\end{aligned}$$

and let

$$\begin{aligned}C = \mathsf{MountainBike} &\sqcap \exists \mathsf{hasPart}.(\mathsf{YFrame} \sqcap \exists \mathsf{hasPart}.\mathsf{RearSuspension})\\ &\sqcap \exists \mathsf{hasPart}.\mathsf{DiscBrakes} \sqcap \exists \mathsf{hasWeight}.\mathsf{Medium},\\ D = \mathsf{RaceBike} &\sqcap \exists \mathsf{hasPart}.(\mathsf{DiamondFrame} \sqcap \exists \mathsf{madeFrom}.\mathsf{Carbon})\\ &\sqcap \exists \mathsf{hasWeight}.\mathsf{Low}.\end{aligned}$$

We assume the same primitive measure, weighting function, and discounting factor as in Example 24. In order to compute the similarity between C and D w.r.t. $\mathcal{T}$, we

have

$$C \sim_c D = (\mathcal{I}'_{C,\mathcal{T}}, d_C) \sim_i (\mathcal{I}'_{D,\mathcal{T}}, d_D).$$

However, the normalized canonical models are exactly those given in Example 24. Thus, $C \sim_c D = 0.6$ or $C \sim_c D = 0.21$, depending on which weighting function is used. ♢

3.2.1 Comparison to Other Approaches

So far, we have only defined the CSM $\sim_c$ and showed that it is well-defined and has nice properties. However, showing that the measure is actually useful in practice is much harder, as there does not exist a gold-standard, to which similarity measures for DL concepts can be compared. In fact, even such a gold standard would have limited significance, since the exact meaning of similarity, and thus the requirements to the CSM, can vary wildly for different applications.

In this section we will compare the CSM $\sim_c$ to some measures that have been defined in the literature. In Chapter 4, we will give a practical evaluation of $\sim_c$ in one specific application: Answering relaxed instance queries. Between this comparison, the practical evaluation and the examples given in this chapter, this should give some insights into the question when $\sim_c$ might be a useful measure.

First, we will discuss the measure *simi* [LT12], on which $\sim_c$ is based. While both measures have many similarities, there are a number of differences as well:

- $\sim_c$ is defined to work w.r.t. general TBoxes, while *simi* only works for unfoldable TBoxes. Additionally, [LT12] does not give any complexity bounds for *simi*: A naive implementation would have an ExpTime worst case complexity due to the unfolding, while $\sim_c$ works in polynomial time.
- The unfolding done in *simi* replaces all defined concept names until only atomic concept names remain. If an unfoldable TBox is used for $\sim_c$, then the canonical models will still contain all the defined concept names in addition to the atomic names. However, if the TBox contains definitions of the form $A \sqsubseteq C$, the unfolding done in *simi* will introduce temporary concept names A' (see Section 2.1.2) that act very similar to keeping the defined name A in the canonical model when computing the similarity. For definitions of the form $A \equiv C$, the difference due to keeping defined concept names in $\sim_c$ can be minimized by simply assigning them low weights.
- *simi* is defined as the average of the directed similarity $simi_d$ in both directions, while $\sim_c$ computes the average between both directions already inside the recursive formula. Basically, this makes sure that both directions are weighted by the number of features, i.e., concept names and successor, that the corresponding elements in the canonical models contain. In our opinion this behavior is more intuitive. For example, consider the concepts $C = A$ and $D_i = A \sqcap \prod_{1 \leq j \leq i} A_j$ w.r.t. the empty TBox. *simi* would give those concepts the similarity $\lim_{i \to \infty} simi(C, D_i) = 1 \otimes 0 = 0.5$, while for $\sim_c$ we get

$\lim_{i\to\infty} C \sim_c D_i = 0$. Since for $i \to \infty$ the number common features between C and D stays one while the number of different features increases, we feel that the similarity should tend towards 0 as well.

- *simi* and $\sim_c$ are both parameterizable using a primitive measure, a weighting function, and a discounting factor. These parameters allow to adapt the CSM to different use cases and preferences. However, *simi* has two additional parameters that $\sim_c$ is missing: Instead of the average to combine the directed similarities it can use arbitrary fuzzy connectors, and instead of the maximum to find the best matching concept name or successor it can use a t-conorm. The t-conorm can be incorporated into $\sim_c$ as well, but might increase the complexity. The fuzzy connector cannot be simply added to $\sim_c$. However, we feel the default values average and maximum are the most intuitive choices.

The measure defined in [Sun13] is very similar to [LT12], with all its problems. A different measure to compute the dissimilarity between $\mathcal{EL}$ concepts is given in [DAB14]; this measure does fulfill triangle inequality of metrics, but does not consider TBoxes.

Other CSMs have been defined for more expressive DLs, e.g. [dFE05; Jan06; APS14]. However, these usually do not satisfy most of the formal properties given in Definition 20. In fact, these measures are always suffering from one three problems: Either the normal form they use is not unique, which means that equivalent concepts may be treated differently, violating equivalence invariance [Jan06; JW09]; they use the ABox to measure the similarity between complex or at least atomic concepts, which usually violates the properties bounded and dissimilar-closed and means the measure is only useful when applied to a KB with a rich ABox [dFE05; dSF08; BWH05]; or the simply do not use all the information that is available in the TBox and thus violate equivalence-closure and boundedness [APS14]. Additionally, the structural measures that do not rely on the canonical model of the ABox usually do not consider a TBox, and thus are only applicable w.r.t. unfoldable definitions. A more thorough discussion of the different measures can be found in [LT12].

3.3 Extension to the Description Logic $\mathcal{EL}^{++}$

In practice, $\mathcal{EL}$ has very limited expressiveness. For instance, in example 24, we could distinguish between the weights of the two bikes only using the concepts Low, Medium, and High. By using concrete domains, one could attach the exact weights to the bikes as rational numbers. Similarly, the other features of $\mathcal{EL}^{++}$ are highly useful in practice: Using the bottom concept one can express the disjointness of concepts, and many classical examples of large-scale $\mathcal{EL}$ ontologies, like SNOMED CT or Gene Ontology, actually use role hierarchies and transitivity constraints as well as domain and range restrictions.

In order to extend the similarity measure $\sim_c$ to work for $\mathcal{EL}^{++}$ concepts w.r.t. an $\mathcal{EL}^{++}$ KB, we need also extend the basic notions like simulations and canonical models. However, unlike in $\mathcal{EL}$ without concrete domains, the definition of

interpretations for $\mathcal{EL}^{++}$ given in the preliminaries does not admit canonical models. For example, in the concrete domain of the rational numbers $Q = (\mathbb{Q}, P^Q)$ introduced before, a concept like $>_0(f)$ will have infinitely many models (one for each positive rational number) without any of them being preferable and therefore canonical. One way to avoid this problem and ensure the existence of canonical models is to consider *pseudo-interpretations*.

Normally, the interpretation function $\cdot^{\mathcal{I}}$ maps each individual, concept, role, and feature name to an element of, a subset of, or a binary relation on the interpretation domain, or to a partial function from domain elements to concrete elements, respectively. However, one could easily define interpretations the other way round; then, the interpretation function would map each element of the domain to the set of concept names the element belongs to, the set of individual names that are mapped to this element, a set of successors consisting of a role name and a successor element from the domain, and a partial function from the feature names to concrete elements. If we require that all individual names occur in the "reverse interpretation" of exactly one element, then these two views are equivalent.

Pseudo-interpretations adapt this new view, but they differ from the usual interpretations in the fourth component: Instead of assigning each element a partial function from feature names to the concrete domain, they assign to each element directly a subset of the set of all predicates of $\mathcal{D}$ over the feature names, denoted with $\text{Pred}^{\mathcal{D}}(\mathsf{N_F})$. In that way, each pseudo-interpretation corresponds to a set of usual interpretations, namely all those whose concrete elements assigned to the feature names of an element of the interpretation domain satisfy all the predicates mapped to the domain element by the pseudo-interpretation.

Definition 41. A *pseudo-interpretation* $\mathcal{J} = (\Delta^{\mathcal{J}}, f_C^{\mathcal{J}}, f_R^{\mathcal{J}}, f_I^{\mathcal{J}}, f_F^{\mathcal{J}})$ consists of an interpretation domain $\Delta^{\mathcal{J}}$ and the interpretation functions $f_C^{\mathcal{J}} : \Delta^{\mathcal{J}} \to \mathcal{P}(\mathsf{N_C})$, $f_R^{\mathcal{J}} : \Delta^{\mathcal{J}} \to \mathcal{P}(\mathsf{N_R} \times \Delta^{\mathcal{J}})$, $f_I^{\mathcal{J}} : \Delta^{\mathcal{J}} \to \mathcal{P}(\mathsf{N_I})$, and $f_F^{\mathcal{J}} : \Delta^{\mathcal{J}} \to \mathcal{P}(\text{Pred}^{\mathcal{D}}(\mathsf{N_F}))$, such that for each $a \in \mathsf{N_I}$ there exists exactly one $d \in \Delta^{\mathcal{J}}$ with $a \in f_I^{\mathcal{J}}(d)$, and the conjunction

$$\text{conj}((\mathcal{J}, d)) = \bigwedge_{p(f_1,\ldots,f_n) \in f_F^{\mathcal{J}}(d)} p(f_1, \ldots, f_n)$$

is satisfiable in $\mathcal{D}$ for any $d \in \Delta^{\mathcal{J}}$. ♢

Pseudo-interpretations can be used exactly as usual interpretations, with the exception that they do not interpret feature names itself; however, it does interpret predicates of the concrete domain, in such a way that an element of a pseudo-interpretation satisfies such a predicate iff the element satisfies it in all the inter-

pretations that are contained in this pseudo-interpretation:

$$\begin{aligned} A^{\mathcal{J}} &= \{d \in \Delta^{\mathcal{J}} \mid A \in f_C^{\mathcal{J}}(d)\} \\ r^{\mathcal{J}} &= \{(d,e) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid (r,e) \in f_R^{\mathcal{J}}(d)\} \\ a^{\mathcal{J}} &= d \iff a \in f_I^{\mathcal{J}}(d) \\ p(f_1,\ldots,f_n)^{\mathcal{J}} &= \{d \in \Delta^{\mathcal{J}} \mid \mathcal{D} \models \text{conj}((\mathcal{J},d)) \Rightarrow p(f_1,\ldots,f_n)\} \end{aligned}$$

All other concept constructors, axioms, and assertions can then be interpreted as given in Definition 2. We say that a pseudo-interpretation $\mathcal{J}$ is a model a KB $\mathcal{K}$, if it satisfies all axioms and assertions in $\mathcal{K}$. This is the case if and only if all corresponding usual interpretations are models of $\mathcal{K}$.

We call a pair $(\mathcal{J}, d)$ consisting of a pseudo-interpretation $\mathcal{J}$ and an element $d \in \Delta^{\mathcal{J}}$ a *pointed pseudo-interpretation* and denote the set of all pointed pseudo-interpretations as $\mathfrak{P}$. We sometimes write $f_C(p)$ (and similarly for f_R, f_I and f_F) instead of $f_C^{\mathcal{J}}(d)$ for $p = (\mathcal{J}, d)$.

3.3.1 Pseudo-simulations and canonical models for $\mathcal{EL}^{++}$

Simulations allow the characterization of elements of interpretations w.r.t. the concepts they are instance of. To extend the simulation relation between interpretations w.r.t. $\mathcal{EL}$ given in [LW10] to pseudo-interpretations w.r.t. $\mathcal{EL}^{++}$, we observe the following:

- role inclusions, range and domain restrictions are not concept constructors, and thus do not matter for the set of concepts that an element of a pseudo-interpretation is an instance of;
- the bottom concept $\bot$ cannot occur in pseudo-interpretations;
- nominals allow to use individual names in concepts, and thus pseudo-simulations need to preserve individuals; and
- for concrete domains, simulations need to preserve the valuations that satisfy the elements, which can be formalized using implications between the predicate sets of pointed pseudo-interpretations.

Thus, we can define a pseudo-simulation relation for $\mathcal{EL}^{++}$ as follows:

Definition 42. Let $\mathcal{I}$ and $\mathcal{J}$ be pseudo-interpretations. A relation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ is a *pseudo-simulation* between $\mathcal{I}$ and $\mathcal{J}$, if the following conditions hold:

1. For all $(d,e) \in S$ and $A \in \mathsf{N_C}$, if $d \in A^{\mathcal{I}}$ then $e \in A^{\mathcal{J}}$.
2. For all $(d,e) \in S$, $r \in \mathsf{N_R}$ and $(d,d') \in r^{\mathcal{I}}$, there is an $(e,e') \in r^{\mathcal{J}}$ with $(d',e') \in S$.
3. For all $(d,e) \in S$ and $a \in \mathsf{N_I}$, if $d = a^{\mathcal{I}}$ then $e = a^{\mathcal{J}}$.
4. For all $(d,e) \in S$, we have that $\mathcal{D} \models \text{conj}((\mathcal{J},e)) \Rightarrow \text{conj}((\mathcal{I},d))$. $\diamondsuit$

Given two pointed pseudo-interpretations $p = (\mathcal{I}, d)$ and $q = (\mathcal{J}, e)$, we say that p simulates q (denoted $p \lesssim q$), if there exists a pseudo-simulation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ between $\mathcal{I}$ and $\mathcal{J}$ with $(d, e) \in S$. p and q are equisimilar (denoted $p \simeq q$), if $p \lesssim q$ and $q \lesssim p$.

This definition of pseudo-simulation behaves as expected since there is again the correspondence between similar elements and the set of concepts that the elements in the simulation are instances of. Indeed, we can extent the result from Theorem 11 to pseudo-simulations in $\mathcal{EL}^{++}$:

Theorem 43. *Let p and q be pointed pseudo-interpretations, then:*

1. *$p \lesssim q$ iff $\mathfrak{C}(p) \subseteq \mathfrak{C}(q)$, and*
2. *$p \simeq q$ iff $\mathfrak{C}(p) = \mathfrak{C}(q)$.*

Proof. Let $p = (\mathcal{I}, d)$ and $q = (\mathcal{J}, e)$.

1. Let $p \lesssim q$, and $C \in \mathfrak{C}(p)$. We show $C \in \mathfrak{C}(q)$ by induction on the structure of C.

   - If $C = A \in \mathsf{N_C}$, then we have $d \in A^{\mathcal{I}}$ and by Definition 42 also $e \in A^{\mathcal{J}}$. Thus $A \in \mathfrak{C}(q)$.
   - If $C = \{o\}$ for $o \in \mathsf{N_I}$, then we have $d = o^{\mathcal{I}}$ and by Definition 42 also $e = o^{\mathcal{J}}$, thus $\{o\} \in \mathfrak{C}(q)$.
   - If $C = \top$, then trivially $\top \in \mathfrak{C}(q)$
   - If $C = C_1 \sqcap C_2$, then $d \in (C_1 \sqcap C_2)^{\mathcal{I}}$, which implies $d \in C_1^{\mathcal{I}}$ and $d \in C_2^{\mathcal{I}}$. By induction this means that $e \in C_1^{\mathcal{J}}$ and $e \in C_2^{\mathcal{J}}$, and thus $e \in (C_1 \sqcap C_2)^{\mathcal{J}}$, i.e., $(C_1 \sqcap C_2) \in \mathfrak{C}(q)$.
   - If $C = \exists r.D$, then we know that $d \in (\exists r.D)^{\mathcal{I}}$, i.e., there exists $d' \in \Delta^{\mathcal{I}}$ with $(d, d') \in r^{\mathcal{I}}$ and $d' \in D^{\mathcal{I}}$. By Definition 42, this implies that there is also an $e' \in \Delta^{\mathcal{J}}$ with $(e, e') \in r^{\mathcal{J}}$ and $(\mathcal{I}, e) \lesssim (\mathcal{J}, e')$, thus the induction hypothesis yields $e' \in D^{\mathcal{J}}$. But this means that $e \in (\exists r.D)^{\mathcal{J}}$ and thus $\exists r.D \in \mathfrak{C}(q)$.
   - Finally, if $C = p(f_1, \ldots, f_n)$, then $d \in p(f_1, \ldots, f_n)^{\mathcal{I}}$ and thus

     $$\mathcal{D} \models \text{conj}((\mathcal{I}, d)) \Rightarrow p(f_1, \ldots, f_n).$$

     By Definition 42, we also have $\mathcal{D} \models \text{conj}((\mathcal{J}, e)) \Rightarrow \text{conj}((\mathcal{I}, d))$, which yields $\mathcal{D} \models \text{conj}((\mathcal{J}, e)) \Rightarrow p(f_1, \ldots, f_n)$, and thus $e \in p(f_1, \ldots, f_n)^{\mathcal{J}}$, i.e., $p(f_1, \ldots, f_n) \in \mathfrak{C}(q)$.

   In the other direction, let $\mathfrak{C}(p) \subseteq \mathfrak{C}(q)$. Then it is easy to show that the relation $S \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ with $(d', e') \in S$ iff $\mathfrak{C}((\mathcal{I}, d')) \subseteq \mathfrak{C}((\mathcal{J}, e'))$ satisfies all conditions from Definition 42 and is thus a pseudo-simulation. This implies that $p \lesssim q$.
2. Follows directly from 1. and the fact that $p \simeq q$ iff $p \lesssim q$ and $q \lesssim p$. □

Next, we need to define canonical pseudo-models for $\mathcal{EL}^{++}$. For these, the additional axioms like role inclusions *are* important. However, if the concept C contains the bottom concept $\bot$, it must be equivalent to $\bot$, and thus cannot have any element in an interpretation as instance – in particular, $\bot$ does not have a canonical model. Thus, by requiring that C is satisfiable w.r.t. $\mathcal{K}$, we do not have to worry about $\bot$ at all.

Since individuals can be part of concepts via nominals, we need to take care of the case that two individuals are equivalent, e.g. by the GCI $\{a\} \sqsubseteq \{b\}$. In this case, we cannot create two elements in the canonical interpretation for the two concepts $\{a\}$ and $\{b\}$, since this would not yield a model of the TBox anymore. Instead, we need to take one representative for all equivalence classes of concepts that are subsumed by the same individual:

$$[C] = \{D \in \mathfrak{C}(\mathcal{EL}^{++}) \mid \exists a \in \mathsf{N_I} \colon \mathcal{K} \models C \sqsubseteq \{a\} \wedge \mathcal{K} \models D \sqsubseteq \{a\}\}$$

Using this equivalence relation, we can define canonical pseudo-models as follows:

Definition 44. Let $\mathcal{K}$ be a satisfiable $\mathcal{EL}^{++}$-KB and $C \in \mathfrak{C}(\mathcal{EL}^{++})$ be an $\mathcal{EL}^{++}$-concept with $C \not\equiv_{\mathcal{K}} \bot$. The *canonical pseudo-model* $\mathcal{J}_{C,\mathcal{K}} = (\Delta^{\mathcal{J}_{C,\mathcal{K}}}, f_C, f_R, f_I, f_F)$ *of* C w.r.t. $\mathcal{K}$ is a pseudo-interpretations defined as follows:

- $\Delta^{\mathcal{J}_{C,\mathcal{K}}} = \{d_{[C]}\} \cup \{d_{[\{a\}]} \mid a \in (\mathsf{sig}(\mathcal{K}) \cup \mathsf{sig}(C)) \cap \mathsf{N_I}\} \cup \{d_{[D]} \mid \exists r.D \in \mathsf{sub}(C) \cup \mathsf{sub}(\mathcal{K})\}$,

and for all $d_{[D]}$ in $\Delta^{\mathcal{J}_{C,\mathcal{K}}}$:

- $f_C(d_{[D]}) = \{A \in \mathsf{N_C} \mid \mathcal{K} \models D \sqsubseteq A\}$,
- $f_R(d_{[D]}) = \{(r, d_{[E]}) \in \mathsf{N_R} \times \Delta^{\mathcal{J}_{C,\mathcal{K}}} \mid \mathcal{K} \models D \sqsubseteq \exists r.E\}$,
- $f_I(d_{[D]}) = \{a \in \mathsf{N_I} \mid \mathcal{K} \models D \sqsubseteq \{a\}\}$, and
- $f_F(d_{[D]}) = \{p(f_1, \ldots, f_n) \in \mathrm{Pred}^{\mathcal{D}}(\mathsf{N_F}) \mid \mathcal{K} \models D \sqsubseteq p(f_1, \ldots, f_n)\}$. ♢

It can be shown that the canonical pseudo-model $\mathcal{J}_{C,\mathcal{K}}$ is indeed a model of the KB $\mathcal{K}$, and its elements $d_{[D]}$ are instances of the corresponding concept D, for all $d_{[D]} \in \Delta^{\mathcal{J}_{C,\mathcal{K}}}$.

Lemma 45. *Let $\mathcal{K}$ be a satisfiable $\mathcal{EL}^{++}$-KB and let C, D be $\mathcal{EL}^{++}$-concepts with $C \not\equiv_{\mathcal{K}} \bot$. Then:*

1. *if $d_{[D]} \in \Delta^{\mathcal{J}_{C,\mathcal{K}}}$, then $d_{[D]} \in D^{\mathcal{J}_{C,\mathcal{K}}}$, and*
2. *$\mathcal{J}_{C,\mathcal{K}} \models \mathcal{K}$.*

We will skip the proof, since it is an easy adaption of the same results for $\mathcal{EL}$.

Finally, it can be shown that the canonical model is indeed 'canonical', i.e., it can simulate all other models (and is thus least w.r.t. $\lesssim$):

Theorem 46. *Let $\mathcal{K}$ be a satisfiable $\mathcal{EL}^{++}$-KB and C, D be $\mathcal{EL}^{++}$-concepts with $C \not\equiv_{\mathcal{K}} \bot$. Then:*

1. *for all pseudo-models $\mathcal{J}$ of $\mathcal{K}$ and all elements $d \in \Delta^{\mathcal{J}}$ it holds $d \in C^{\mathcal{J}}$ iff $(\mathcal{J}_{C,\mathcal{K}}, d_{[C]}) \lesssim (\mathcal{J}, d)$,*
2. *for all pseudo-models $\mathcal{J}$ of $\mathcal{K}$, all individuals a occurring in $\mathcal{K}$, and all elements $d \in \Delta^{\mathcal{J}}$ it holds $d = a^{\mathcal{J}}$ iff $(\mathcal{J}_K, d_{[\{a\}]}) \lesssim (\mathcal{J}, d)$, and*
3. *$\mathcal{K} \models C \sqsubseteq D$ iff $d_{[C]} \in D^{\mathcal{J}_{C,\mathcal{K}}}$ iff $(\mathcal{J}_{D,\mathcal{K}}, d_{[D]}) \lesssim (\mathcal{J}_{C,\mathcal{K}}, d_{[C]})$.*

Again, this is shown very similar to the corresponding result for $\mathcal{EL}$ [LW10].

3.3.2 Extending $\sim_c$ to $\mathcal{EL}^{++}$

Analogous to the $\mathcal{EL}$ case, we will define the CSM via a similarity measure on pointed pseudo-interpretations. Compared to $\mathcal{EL}$, the pseudo-interpretation similarity for $\mathcal{EL}^{++}$ will also have as parameters a primitive measure, a weighting function, and a discounting factor, but those need to be extended to handle the new individual and feature names. Additionally, it will require a similarity measure on elements of the concrete domain and a factor c to specify the influence of the concrete domain on the similarity value. Thus, for $\mathcal{EL}^{++}$ we have the following parameters:

- a primitive measure $\sim_{prim}$: $\mathsf{N_C} \times \mathsf{N_C} \cup \mathsf{N_R} \times \mathsf{N_R} \cup \mathsf{N_I} \times \mathsf{N_I} \to [0,1]$ that assigns a similarity value to each pair of concept names, role names, and individual names;
- a weighting function $g : \mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I} \cup \mathsf{N_F} \to \mathbb{R}_{>0}$, which assigns a weight to each concept, role, individual an feature name;
- a similarity measure $\sim_{\mathcal{D}}$: $\Delta^{\mathcal{D}} \times \Delta^{\mathcal{D}} \to [0,1]$ on the concrete domain;
- a discounting factor $w \in (0,1)$; and
- a concrete domain factor $c > 0$.

We will extend the concrete similarity measure $\sim_{\mathcal{D}}$ to handle undefined values, i.e., $\sim_{\mathcal{D}}$: $(\Delta^{\mathcal{D}} \cup \{\bot\}) \times (\Delta^{\mathcal{D}} \cup \{\bot\}) \to [0,1]$ by setting $\bot \sim_{\mathcal{D}} d = d \sim_{\mathcal{D}} \bot = 0$ for $d \in \Delta^{\mathcal{D}}$ and $\bot \sim_{\mathcal{D}} \bot = 1$. This can be further extended to a similarity measure on valuations, i.e., partial functions $u, v : \mathsf{N_F} \nrightarrow \Delta^{\mathcal{D}}$, by computing the weighted average of the similarity values for all features:

$$u \sim_{\mathcal{D}} v = \frac{\sum_{f \in \mathrm{dom}(u) \cup \mathrm{dom}(v)} g(f)(u(f) \sim_{\mathcal{D}} v(f))}{\sum_{f \in \mathrm{dom}(u) \cup \mathrm{dom}(v)} g(f)}.$$

Finally, we can define the similarity of conjunctions of predicates on the concrete domain using a similar construction to the Hausdorff metric, where the valuations

u, v are restricted to those feature names occurring in $f_F(p)$ or $f_F(q)$:

$$\mathrm{sim}_{\mathcal{D}}(p,q) = \min\left\{ \inf_{u \models \mathrm{conj}(p)} \sup_{v \models \mathrm{conj}(q)} u \sim_{\mathcal{D}} v, \inf_{u \models \mathrm{conj}(q)} \sup_{v \models \mathrm{conj}(p)} u \sim_{\mathcal{D}} v \right\}$$

where $u \models p$ means that the valuation u satisfies the formula p. This Hausdorff measure captures the following intuition. Basically, it finds the satisfying valuation u for p that has the least similarity to any of the satisfying valuations v for q, as the other way round. If $\sim_{\mathcal{D}}$ has the property $d \sim_{\mathcal{D}} d' = 1$ iff $d = d'$ for all $d, d' \in \Delta^{\mathcal{D}}$, then $\sim_{\mathcal{D}}$ is equivalence-closed as well, i.e., $\mathrm{sim}_{\mathcal{D}}(p,q) = 1$ iff $\mathcal{D} \models \mathrm{conj}((\mathcal{J}, e)) \Leftrightarrow \mathrm{conj}((\mathcal{I}, d))$. Otherwise, there always exists a valuation that satisfies all predicates on one side but not on the other side, which will lead to a similarity value less than 1. Similarly, if there is a valuation satisfying the predicates of one pointed interpretation that is completely dissimilar to all valuations satisfying the other predicates of the other pointed interpretation, then the similarity value will be 0.

To compare how similar two pointed pseudo-interpretations are for these aspects, we again introduce measures for the concept names, successors, and individual names of pointed interpretations. The functions $\mathrm{cm} : \mathfrak{P} \times \mathfrak{P} \to \mathbb{R}$ and $\mathrm{sm} : \mathfrak{P} \times \mathfrak{P} \to \mathbb{R}$ are defined essentially as in the $\mathcal{EL}$ case, the only difference being that p and q are now pointed pseudo-interpretations:

$$\mathrm{cm}(p,q) = \sum_{A \in f_C(p)} \left(g(A) \max_{B \in f_C(q)} A \sim_p B \right)$$

$$\mathrm{sm}(p,q) = \sum_{(r,p') \in f_R(p)} \left(g(r) \max_{(s,q') \in f_R(q)} \left((r \sim_p s)\big((1-w) + w(p' \sim_i q')\big) \right) \right)$$

Since $\mathcal{EL}^{++}$ concepts may also contain nominals, we need an additional measure to compare the individual names of two pointed pseudo-interpretations. This function, $\mathrm{im} : \mathfrak{P} \times \mathfrak{P} \to \mathbb{R}$, works exactly like cm, only comparing individual names instead of concept names:

$$\mathrm{im}(p,q) = \sum_{a \in f_I(p)} \left(g(a) \max_{b \in f_I(q)} a \sim_p b \right)$$

Finally, we define a function $\mathrm{fm} : \mathfrak{P} \times \mathfrak{P} \to \mathbb{R}$, that measures the similarity between the concrete domain parts of two pointed interpretations p and q. Essentially, this is a product of $\mathrm{sim}_{\mathcal{D}}(p,q)$ and the concrete domain factor c; however, when neither of the pointed interpretations p or q has any concrete feature, we set

the weight to 0:

$$g_F(p,q) = \begin{cases} c & \text{if } f_F(p) \neq \emptyset \vee f_F(q) \neq \emptyset \\ 0 & \text{otherwise} \end{cases},$$
$$\text{fm}(p,q) = g_F(p,q) \cdot \text{sim}_{\mathcal{D}}(p,q).$$

With this in place, we can now extend $\sim_i$ to $\mathcal{EL}^{++}$. This extension mainly adds two terms to the equation from Definition 23: The measure that compares the individual names (in both directions), and the term that estimates the similarity between the concrete domain parts, $\text{sim}_{\mathcal{D}}(p,q)$:

Definition 47. Given a primitive measure $\sim_p$, a weighting function g, a similarity measure $\sim_{\mathcal{D}}$ on the concrete domain, a discounting factor w, a concrete domain factor c, as well as two interpretations $\mathcal{I}$ and $\mathcal{J}$, the interpretation similarity measure $\sim_i$ $(\sim_p, \sim_{\mathcal{D}}, g, w, c) : \mathfrak{P} \times \mathfrak{P} \to [0,1]$ is defined as follows, for all $p, q \in \mathfrak{P}$: If $f_C(p) = f_C(q) = f_R(p) = f_R(q) = f_I(p) = f_I(q) = f_F(p) = f_F(q) = \emptyset$, then $p \sim_i q = 1$, otherwise

$$p \sim_i q = \frac{\text{cm}(p,q) + \text{cm}(q,p) + \text{sm}(p,q) + \text{sm}(q,p) + \text{im}(p,q) + \text{im}(q,p) + \text{fm}(p,q)}{\text{weight}}$$

where

$$\text{weight} = \sum_{A \in f_C(p)} g(A) + \sum_{A \in f_C(q)} g(A) + \sum_{(r,p') \in f_R(p)} g(r) + \sum_{(r,q') \in f_R(q)} g(r) + \sum_{a \in f_I(q)} g(a) + \sum_{b \in f_I(q)} g(b) + g_F(p,q) \qquad \diamondsuit$$

The proof that $\sim_i$ is well-defined can be transferred from Theorem 29 in a straightforward way, as $\text{sim}_{\mathcal{D}}$ is well-defined as well. We will give this result without repeating the proof.

Theorem 48. *$\sim_i$ is well-defined, i.e., $p \sim_i q$ has a unique solution.*

In order to lift $\sim_i$ to a CSM $\sim_c$ for $\mathcal{EL}^{++}$ concepts, we again need to normalize the canonical models. The definition and computation of normal forms is exactly the same as in the $\mathcal{EL}$ case given in Definition 34, except for using pseudo-simulations instead of simulations.

Thus, we can define the CSM $\sim_c$ for $\mathcal{EL}^{++}$ concepts w.r.t. an $\mathcal{EL}^{++}$ KB $\mathcal{K}$ as follows:

$$C \sim_c D = (\mathcal{J}'_{C,\mathcal{K}}, d_{[C]}) \sim_i (\mathcal{J}'_{D,\mathcal{K}}, d_{[D]}),$$

where $\mathcal{J}'_{C,\mathcal{K}}$ and $\mathcal{J}'_{D,\mathcal{K}}$ are the normalized canonical pseudo-models of C and D w.r.t. $\mathcal{K}$. If C or D are equivalent to $\bot$, they do not have a canonical model. In this case, we set $C \sim_c \bot = \bot \sim_c D = 0$ and $\bot \sim_c \bot = 1$. We can show that $\sim_c$ has all of the properties given in Definition 20:

Theorem 49. *Let $\sim_c(\sim_p, g, \sim_\mathcal{D}, w, c)$ be instantiated with a primitive measure $\sim_p$, a weighting function g, a concrete similarity measure $\sim_\mathcal{D}$, a discounting factor $w \in (0, 1)$, and a concrete domain factor c. Then $\sim_c$ has the following properties:*

1. *$\sim_c$ is symmetric, if the primitive measures $\sim_p$ and $\sim_\mathcal{D}$ are symmetric;*
2. *$\sim_c$ is bounded;*
3. *$\sim_c$ is dissimilar-closed, if the primitive measure $\sim_p$ does not assign a similarity value greater than 0 to different concept, role, or individual names, and the concrete similarity measure $\sim_\mathcal{D}$ does not assign a similarity value greater than 0 to different elements of the concrete domain.*
4. *$\sim_i$ is equivalence-invariant for normalized interpretations; and*
5. *$\sim_i$ is equivalence-closed for normalized interpretations, if the primitive measure $\sim_p$ does not assign the similarity value 1 to different concept, role, or individual names, and the concrete similarity measure $\sim_\mathcal{D}$ does not assign value 1 to different elements of the concrete domain.*

Again, the proof is a straight-forward adaption of the proofs of Theorem 36 and Theorem 39, together with the properties of $\sim_\mathcal{D}$ mentioned before.

While the addition of nominals and other features of $\mathcal{EL}^{++}$ was relatively straightforward, we haven't found any CSM in the literature that can handle concrete domains. We believe that the measure introduced here is a good starting point for applications that handle concrete data inside the knowledge base and need a notion of similarity. In fact, often the full power of concrete domains is not necessary: For similarity measures, even simple value assignments (i.e., predicates of the for $=_d$ for $d \in \Delta^\mathcal{D}$) can be highly useful, as the primitive measure $\sim_\mathcal{D}$ can compare those values very flexibly. The next chapter will introduce one application of similarity measures where even such simple concrete domains can be exploited to great effect.

Chapter 4

Relaxed Instance Queries

In this chapter, we want to introduce instance queries relaxed by concept similarity measures. Recall that, given a KB $\mathcal{K}$ together with a concept C and an individual name a, we call a an instance of C w.r.t. $\mathcal{K}$, if $a^{\mathcal{I}}$ is in the interpretation of C in all models $\mathcal{I}$ of $\mathcal{K}$. Often, we are not interested only in checking whether an individual is an instance of a concept C, but instead we want to find all such instances. This is called the instance query problem.

Definition 50. Given a KB $\mathcal{K}$ and a query concept Q, we call an individual $a \in \mathsf{N_I}$ an instance of Q w.r.t. $\mathcal{K}$, if $\mathcal{K} \models Q(a)$. The instance query problem is to compute all instances of Q w.r.t. $\mathcal{K}$. ♢

However, as said in the introduction, it is often a good idea to not just consider the perfect matches, but also alternatives that are similar to the query. The following motivating example shows a situation where query relaxation could be preferable to exact query answering.

Example 51. Consider a bike shop that has created a TBox $\mathcal{T}$ with relevant knowledge about the bicycles, and an ABox $\mathcal{A}$ that describes all bikes the store has in stock as individuals. If a customer wants to buy a new bike, the store can create a query concept Q that formalizes all of the requirements of the customer, and query the knowledge base for all bikes that are instances of Q w.r.t. the KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. The answers are all those bikes that satisfy exactly the requirements of the customer.

However, usually it makes sense for the shop and the customer to not just consider the exact answer, but also alternatives that are similar enough to the query. Reasons to consider reasonable alternatives can be manifold: the costumer might not be completely sure what exactly he wants; the KB might not be complete because of laziness or mistakes and therefore skip relevant instances; or it might be simply because of reasons that cannot be formalized in a knowledge base, e.g., a bike may simply "feel" better to the customer, even though it does not perfectly satisfy her requirements.

Another reason where alternatives are useful is if no bikes satisfies all the requirements perfectly. This is easily imaginable, since often desired properties are contradictory: Customers often want the best possible quality for the least possible price. In this case, returning alternatives to the query that can be checked out would be much more preferable for the bike shop than telling the customer that they have no such bike in stock. ♢

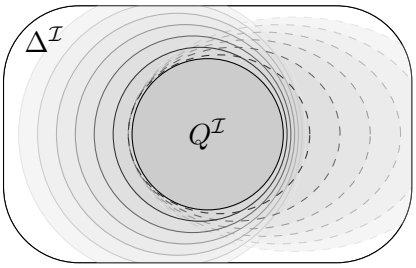

Figure 4.1: Relaxed instances w.r.t. two different CSMs, represented by continuous and dashed lines, respectively. Darker colors represent the relaxed instances of Q w.r.t. higher degrees t.

There are of course many different approaches how one can relax a query. We will discuss some of them later. The approach we take in this paper is to use a CSM $\sim$ to find concepts that are similar to the query concept Q, and then return all instances of these similar concepts as answers as well. A parameter $t \in [0, 1]$, the threshold, allows one to set a lower bound for how similar concepts need to be to the query concept in order to be considered.

Definition 52 (relaxed instance). Let $\mathcal{L}$ be a DL, $\mathcal{K}$ be an $\mathcal{L}$ knowledge base, Q be an $\mathcal{L}$ concept, $\sim$ a concept similarity measure over $\mathcal{L}$ concepts, and $t \in [0, 1]$. An individual $a \in \mathsf{N_I}$ is a *relaxed instance* of Q w.r.t. $\mathcal{K}$, $\sim$ and the threshold t, denoted $a \in_t^{\sim} Q$, iff

$$\sup_{X \in \mathfrak{C}(\mathcal{L}) \text{ with } \mathcal{K} \models X(a)} Q \sim X \geq t.$$

$\diamondsuit$

For brevity, we will denote as $\text{Relax}_t^{\sim}(Q)$ the set of all relaxed instances of the concept Q w.r.t. $\mathcal{K}$, $\sim$ and t. Clearly, the elements of $\text{Relax}_t^{\sim}(Q)$ depend strongly on the value of t, but also on the similarity measure $\sim$ chosen; this dependency is depicted in Figure 4.1. For a fixed concept similarity measure $\sim$ and thresholds $t \leq t'$, we have $\text{Relax}_{t'}^{\sim}(Q) \subseteq \text{Relax}_t^{\sim}(Q)$. In the figure, the central circle represents the interpretation of the concept Q. The other lines show the interpretation of $\text{Relax}_t^{\sim}(Q)$ with darker lines gradually representing larger thresholds t. We use two different kinds of lines (continuous and dashed, respectively) to represent two different similarity measures that relax the concepts based on different features. This allows to relax the query concept into different directions, and can be used for example to state which features are more important and should not be relaxed as far as other features.

In the following, we will restrict our attention to a finite signature Σ which contains all concept and role names occurring in the query concept Q and the KB $\mathcal{K}$. Considering other names in the concepts X is not necessary for finding the supremum of the similarities towards the query concept Q.

Comparison to Other Approaches

Instead of relying on similarity measures to relax queries, there exist many other approaches one could choose from. In fact, in the context of databases, the query re-

laxation problem has been investigated by many other authors, for example for relational databases [CDH+06; ZGB+07], RDF triple stores [HPW08; DSW+09; HLZ08], and XML data [ALP04; Lee02]. These approaches usually work by rewriting the query into a set of more general queries, e.g., by deleting some parts of the query or replacing relations with more general ones. This process is repeated, and the answers to the rewritten queries are collected until a threshold is reached or a set number of answers has been found (this case is also called top-k query answering). Some kind of measure quantifies how much the rewritten queries deviate from the original query in order to rank the answers.

This approach has several disadvantages. The background knowledge used to rewrite is often rather inexpressive, usually not more then a simple concept hierarchy. The measure used for ranking the answers is usually fixed, and can not be adjusted in order to choose which parts of the query are more or less important given the current context. Finally, the whole process of query relaxation depends heavily on the set of rewrite rules that are available, which makes it hard to define easy to understand semantics for the relaxed answers.

Instead of relying on a classical knowledge base and relax the query, one could also use approaches where the knowledge base is already equipped with quantitative information, like fuzzy or rough logics. In these logics one does not reason about subsumption and membership, but instead about the degrees with which those relations hold or an indiscernibility relation on the individuals. Relaxed instances of the concept C would then be those with a large membership degree to C in fuzzy, and those that are indiscernible from any certain instance of C in rough logics. Query answering over fuzzy DLs has been studied, see for instance [Str06b; Str06a; PSS+08], whereas for rough DLs the investigation of query answering techniques is only at its beginning [PTT14]. However, in order to use fuzzy logics for relaxed query answering, the whole KB needs to be converted into a fuzzy KB, which may not be appropriate in many cases. Also, since the quantitative information is now part of the KB, it is again quite hard to adapt the direction of the query relaxation process without changing the KB itself.

One could also try to decide which individuals are *similar* to any of the certain instances of C. Such a method requires a similarity measure defined over the *elements* of the domain, rather than on the concepts. A DL with a similarity measure over the domain elements was introduced in [LWZ03]. However, for this DL the similarity measure (or more precisely, a distance metric) is part of the interpretation and cannot be adjusted to different user needs. Also, this approach would not work when the query has no certain instances to begin with.

In comparison to the previous alternatives, our approach using a CSM to relax queries allows to vary the degree of relaxation, and the way how the query is relaxed by choosing a suitable similarity measure. If the measure is parameterizable like the one introduced in Chapter 3, then one can vary the parameters to choose which parts of the query are more important and should be relaxed very little, and which parts of the query are less important.

Finally, an idea that can be used to answer relaxed queries has been introduced in [BBF15]. It defined the Logic $\tau\mathcal{EL}$, which extends the DL $\mathcal{EL}$ with graded membership functions m that specify the degree to which elements of an interpretation belong to a concept, and are used specify threshold concepts of the form $C_{\sim t}$ for $\sim \in \{<, >, \leq, \geq\}$ and $t \in [0, 1]$. These threshold concepts are interpreted as the set of all elements that have a matching degree to the concept C according to m. By asking for instances to a threshold concept $C_{\geq t}$ w.r.t. a classical TBox $\mathcal{T}$, one could achieve very similar effects as the relaxed instance queries defined here. In fact, the graded membership function deg is defined similarly to a directed version of $\sim_c$, i.e., where only one direction of cm and sm is evaluated instead of both. Indeed, [BBF15] shows that when converting an instance of the similarity measure *simi* introduced in [LT12] into a graded membership function in a straight-forward way, then the resulting graded membership function m is equivalent to deg. This implies that relaxed instances of Q w.r.t. t, an unfoldable KB $\mathcal{K}$, and the CSM $\sim_i$ can be computed in polynomial time by computing all instances of the threshold concept $Q_{\geq t}$ w.r.t. the membership degree function deg. We will discuss this approach in more detail in Section 4.4.

In the following, we will show how to answer relaxed instance queries in the presence of both unfoldable and general $\mathcal{EL}$ TBoxes.

4.1 Relaxed Instance Queries for Unfoldable $\mathcal{EL}$ TBoxes

The main difficulty we need to deal with when computing relaxed instances is that for an individual a, there might be infinitely many concepts C which have a as instance. In order to compute the degree to which a belongs to C w.r.t. the query concept Q, we may need to compute the supremum of $Q \sim C$ for all these infinitely many concepts C.

However, for unfoldable TBoxes $\mathcal{T}$, we know that the query concept Q is of finite depth, even after expanding it according to the definitions in $\mathcal{T}$. Most similarity measures will only compare concepts up to the lesser role-depth of the both concepts; if one concept has an existential restriction at some point, but the other does not, the similarity between those parts is 0 anyway and does not need to be compared any further. More formally, the *restriction of an $\mathcal{EL}$ concept C to role-depth k*, denoted C_k, is defined as

$$C_k := \begin{cases} C & \text{if } C = A \in \mathsf{N_C} \text{ or } C = \top, \\ D_k \sqcap E_k & \text{if } C = D \sqcap E, \\ \exists r.D_{k-1} & \text{if } C = \exists r.D \wedge k > 0, \\ \top & \text{if } C = \exists r.D \wedge k = 0. \end{cases}$$

We call a CSM $\sim$ *role-depth bounded* iff $C \sim D = C_k \sim D_k$ for all $C, D \in \mathfrak{C}(\mathcal{EL})$ and $k > \min(\mathrm{rd}(C), \mathrm{rd}(D))$. Note that this property only makes sense for an empty TBox. For unfoldable TBoxes, we assume that C, D and all concept assertions in

the ABox have been expanded and then can remove the TBox as well.

If $\sim$ is role-depth bounded then we only need to check concepts up to the role-depth $k = \text{rd}(Q) + 1$ of the expanded query concept Q to see if an individual a is a relaxed instance of Q w.r.t. an unfoldable TBox. If $\sim$ is also equivalence invariant, then equivalent concepts will always have the same similarity to Q, and thus we only need to check different concepts. However, it is known that in $\mathcal{EL}$ for a finite signature Σ, there only exists a finite number of different concepts with a bounded role-depth. Since we can restrict the signature to those names actually occurring in $\mathcal{K}$ and Q, this immediately gives us decidability.

Theorem 53. *Given an unfoldable $\mathcal{EL}$ KB $\mathcal{K}$, an $\mathcal{EL}$ concept Q, an individual $a \in \mathsf{N_I}$, and an equivalence-invariant, role-depth bounded CSM $\sim$, it is decidable whether $a \in_t^{\sim} Q$ w.r.t. $\mathcal{K}$.*

Proof. By definition of $\in_t^{\sim}$ we have

$$a \in_t^{\sim} Q \text{ iff } \sup_{X \in \mathfrak{C}(\mathcal{L}) \text{ with } \mathcal{K} \models X(a)} Q \sim X \geq t.$$

Let Q' be the concept Q expanded w.r.t. $\mathcal{K}$, and $k = \text{rd}(Q)$. Since $\sim$ is equivalence invariant and role-depth bounded, this implies that we can restrict our search to concepts X that have a role-depth bound of $k + 1$ and which are not equivalent. However, since the number of such concepts X is finite, simply checking each of them gives a valid decision procedure. □

One can compute all answers to a relaxed instance query by simply checking all individuals occurring in $\mathcal{K}$ if they are relaxed instances or not, and returning all those that are. Such an algorithm is given in Figure 4.2.

We previously [EPT13b; EPT14; EPT15] claimed that on can guess a generalized concepts of the k-msc(a) (see Definition 15) in order to compute the relaxed instances. Such a generalized concept arises from k-msc(a) by deleting certain subconcepts. This algorithm would then decide whether an individual a is a relaxed instance in NExpTime. However, we recently found out that this algorithm and the underlying idea were indeed wrong. In fact, the size of k-msc(a) is at most exponential in k, and thus there can only be 2Exp many different generalized concepts of k-msc(a). However, the total number of different concepts with bounded role-depth is at least non-elementary. In fact, for a given signature Σ and a role-depth bound k one can easily construct an ABox $\mathcal{A}$ with an individual a that is an instance of all concepts C with $\text{rd}(C) \leq k$:

$$\mathcal{A} = \{A(a) \mid A \in \Sigma \cap \mathsf{N_C}\} \cup \{r(a, a) \mid r \in \Sigma \cap \mathsf{N_R}\}.$$

Due to the non-elementary lower bound of the number of such concepts C and the 2Epx upper bound of the number of generalized concepts of k-msc(a), this implies that there are concepts with role-depth of k or less that have a as an instance, but are not generalized concepts of k-msc(a). We will show the non-elementary lower

Procedure: relaxed-instance?$(a, Q, \mathcal{K}, \sim, t)$
Input: a: individual in $\mathcal{K}$; Q: $\mathcal{EL}$-concept; $\mathcal{K}$: $\mathcal{EL}$-KB with unfoldable TBox; $\sim$: equivalence-invariant and role-depth bounded CSM; t: threshold;
Output: true if $a \in_t^{\sim} Q$ w.r.t. $\mathcal{K}$; otherwise false

1: expand Q w.r.t. $\mathcal{K}$ into Q'
2: $k \leftarrow \mathrm{rd}(Q')$
3: Ans $\leftarrow \emptyset$
4: **for all** $a \in \mathrm{sig}(\mathcal{K}) \cap \mathsf{N_I}$ **do**
5: **for all** non-equivalent $X \in \mathfrak{C}(\mathcal{EL})$ with $\mathrm{rd}(X) \leq k+1$ and $\mathcal{K} \models X(a)$ **do**
6: **if** $Q \sim X \geq t$ **then**
7: Ans $\leftarrow$ Ans $\cup \{a\}$
8: **end if**
9: **end for**
10: **end for**
11: **return** Ans

Figure 4.2: Algorithm to compute all relaxed instances w.r.t. unfoldable $\mathcal{EL}$ TBoxes.

bound to the number of different role-depth bounded $\mathcal{EL}$ concepts in the following lemma.

Lemma 54. *There is a non-elementary number of non-equivalent $\mathcal{EL}$ concepts with bounded role-depth, even when restricting to only 2 concept names and 1 role name.*

Proof. First we show that $\binom{2n}{n} \geq 2^n$, which can be done by induction on n: $\binom{2}{1} = 2 \geq 2^1$, and if $\binom{2n}{n} \geq 2^n$, then $\binom{2n+2}{n+1} = \binom{2n}{n-1} + 2\binom{2n}{n} + \binom{2n}{n+1} \geq 2 \cdot 2^n = 2^{n+1}$.

We will now show that there exists a non-elementary number of $\mathcal{EL}$ concepts of bounded role-depth by induction on the role-depth k, assuming that we have at least two concept names A and B and one role name r. In particular, we will show by induction on k that there are

$$2n = 2 \cdot 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\} k$$

concepts of role-depth k, such that non of these concepts subsume each other.

- For $k = 0$, there are 2 such concepts: A and B.
- Assume that there exist $2n$ concepts of role-depth k, such that none of these concepts is subsumed by another concept. We need to show that we can construct $2 \cdot 2^n$ pairwise non-subsuming $\mathcal{EL}$ concepts of role-depth $k+1$. By choosing exactly n of those $2n$ concepts $C_1, C_2, \ldots, C_n$ we can create a concept $A \sqcap \exists r.C_1 \sqcap \exists r.C_2 \sqcap \cdots \sqcap \exists r.C_n$ of role-depth $k+1$. In fact, we can create exactly $\binom{2n}{n} \geq 2^n$ such concepts, for which again none subsume each other. Similarly, by replacing concept name A with B, we can create another 2^n concepts of role-depth $k+1$, for a total of $2 \cdot 2^n$. □

One example where the approach based on generalized concepts fails is the following. Given the KB $\mathcal{K} = (\emptyset, \mathcal{A})$ consisting of the empty TBox and the ABox $\mathcal{A} = \{r(a,b), A \sqcap B(b)\}$, as well as the query concept $Q = \exists r.A \sqcap \exists r.B$, we have that $\text{msc}(a) = \exists r.(A \sqcap B)$. It is easy to see that $\mathcal{K} \models Q(a)$ and thus for any equivalence invariant CSM $\sim$ we have $a \in_1^\sim Q$. However, $\exists r.A \sqcap \exists r.B$ is not a generalized concept of $\exists r.(A \sqcap B)$, since it can not be derived from the latter by deleting subconcepts. Thus, if $\sim$ is also equivalence closed, then no generalized concept of $\text{msc}(a)$ has similarity 1 to Q and the old algorithm would falsely decide that $a \notin_t^\sim Q$.

Without further information about the similarity measure (besides being equivalence invariant and role-depth bounded), we were not able to create better algorithms than naively checking all the different role-depth bounded concepts X that have the individual as instance. Since in general there may be a non-elementary number of concepts X that need to be compared to the query concept Q, this is hardly possible in practice. Therefore, in the next section, we will restrict to a single family of concept similarity measures in order to exploit their structure and arrive at a more efficient way to compute relaxed instances. In fact, we will use the similarity measure $\sim_c$ introduced in Chapter 3.

4.2 Relaxed Instance Queries for General $\mathcal{EL}$ TBoxes using $\sim_c$

For general TBoxes, the approach presented in Algorithm 4.2 does not work anymore, since the expansion of the query concept Q w.r.t. a general TBox may not terminate. However, we can exploit the fact that $\sim_c$ is defined via the interpretation similarity measure $\sim_i$:

$$a \in_t^\sim Q \quad \text{iff} \quad \left(\sup_{X \in \mathfrak{C}(\mathcal{EL}) \text{ with } \mathcal{K} \models X(a)} Q \sim_c X \right) \leq t$$

$$\text{iff} \quad \left(\sup_{X \in \mathfrak{C}(\mathcal{EL}) \text{ with } \mathcal{K} \models X(a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i (\mathcal{I}'_{X,\mathcal{T}}, d_X) \right) \leq t.$$

Since we compute the supremum over all such concepts X, and those concepts can grow arbitrarily large, instead of iterating over concepts and then using their canonical models, we can also directly iterate over (normalized) pointed interpretations since, for each pointed interpretation, the model-based msc has a canonical model which coincides with the original pointed interpretation up to an arbitrarily large role-depth. However, we only consider concepts X for which $\mathcal{K} \models X(a)$; according to Theorem 13, we must then also restrict to pointed interpretations p with $p \lesssim (\mathcal{I}_\mathcal{K}, d_a)$. Thus we get:

$$a \in_t^\sim Q \quad \text{iff} \quad \left(\sup_{p \in \mathfrak{I} \text{ with } p \lesssim (\mathcal{I}_\mathcal{K}, d_a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p \right) \leq t.$$

Pointed interpretations p that are simulated by $(\mathcal{I}_\mathcal{K}, d_a)$ must have a certain form: all concept names $A \in \mathsf{CN}(p)$ must also have d_a as an instance in $\mathcal{I}_\mathcal{K}$, and for all successors of p there must be a successor of d_a labeled with the same role such that these are again in a simulation relation. Thus, instead of checking all pointed interpretations p, we instead modify $\sim_i$ so that it implicitly constructs the best such pointed interpretation from the canonical model $(\mathcal{I}_\mathcal{K}, d_a)$. We call this the maximal interpretation similarity $\sim_i^{\max}$.

Definition 55. Given a primitive measure $\sim_p$, a weighting function g, a discounting factor w, as well as two interpretations $\mathcal{I}$ and $\mathcal{J}$, the maximal interpretation similarity measure $\sim_i^{\max}\ (\sim_p, g, w) : \mathfrak{I} \times \mathfrak{I} \to [0,1]$ is defined as follows, for all $p, q \in \mathfrak{I}$: If $\mathsf{CN}(p) = \mathsf{CN}(q) = \mathsf{SC}(p) = \mathsf{SC}(q) = \emptyset$, then $p \sim_i^{\max} q = 1$, otherwise

$$p \sim_i^{\max} q = \max_{\substack{C \subseteq \mathsf{CN}(q) \\ S \subseteq \mathsf{SC}(q)}} \frac{\mathrm{cm}(\mathsf{CN}(p), C) + \mathrm{cm}(C, \mathsf{CN}(p)) + \mathrm{sm}(\mathsf{SC}(p), S) + \mathrm{sm}(S, \mathsf{SC}(p))}{\sum\limits_{A \in \mathsf{CN}(p)} g(A) + \sum\limits_{A \in C} g(A) + \sum\limits_{(r,p') \in \mathsf{SC}(p)} g(r) + \sum\limits_{(r,q') \in S} g(r)}$$

with

$$\mathrm{cm}(S_1, S_2) = \sum_{A \in S_1} \left(g(A) \max_{B \in S_2} A \sim_p B \right), \text{ and}$$

$$\mathrm{sm}(S_1, S_2) = \sum_{(r,p') \in S_1} \left(g(r) \max_{(s,q') \in S_2} \left((r \sim_p s)\big((1-w) + w(p' \sim_i^{\max} q')\big) \right) \right). \qquad \diamondsuit$$

In order to show that $\sim_i^{\max}$ indeed allows us to answer relaxed instance queries w.r.t. general $\mathcal{EL}$ TBoxes, we need to show that the maximal interpretation similarity $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_\mathcal{K}, d_a)$ is indeed always greater or equal to $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p$ for any pointed interpretation p with $p \lesssim (\mathcal{I}_K, d_a)$.

Lemma 56. *Given the CSM $\sim_i$ $(\sim_p, g, w)$, an $\mathcal{EL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, an $\mathcal{EL}$ concept Q, an individual $a \in \mathsf{N_I}$ and a pointed interpretation $p = (\mathcal{I}, d)$, we have*

$$(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p \leq (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{max} (\mathcal{I}'_\mathcal{K}, d_a).$$

Proof. Let $p = (\mathcal{I}, d)$. We assume that for all elements $e \in \Delta^\mathcal{I}$, there exists an element $e' \in \Delta^{\mathcal{I}'_\mathcal{K}}$ with $e \lesssim e'$. Otherwise, e is not reachable from d in $\mathcal{I}$ and can be removed without changing the similarity $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p$.

We will show that for each $(p_1, p_2, p_3) \in \mathfrak{I}_{\mathcal{I}'_{Q,\mathcal{T}}} \times \mathfrak{I}_{\mathcal{I}'_\mathcal{K}} \times \mathfrak{I}_\mathcal{I}$ with $p_3 \lesssim p_2$ we have $p_1 \sim_i^{\max} p_2 \geq p_1 \sim_i p_3$:

- for any concept name $A \in \mathsf{CN}(p_3)$, $p_3 \lesssim p_2$ implies that $A \in \mathsf{CN}(p_2)$. Thus, when computing $p_1 \sim_i^{\max} p_2$ we can chose the subset $C = \mathsf{CN}(p_3)$, which means that $\mathrm{cm}(\mathsf{CN}(p_1), C)$ and $\mathrm{cm}(C, \mathsf{CN}(p_1))$ for $(\mathcal{I}'_{Q,\mathcal{T}}, e) \sim_i^{\max} (\mathcal{I}'_\mathcal{K}, f)$ will be at least as large as $\mathrm{cm}(p_1, p_3)$ and $\mathrm{cm}(p_3, p_1)$, respectively.
- for any successor $(s, p'_3) \in \mathsf{SC}(p_3)$, $p_3 \lesssim p_2$ implies that we have $(s, p'_2) \in \mathsf{SC}(p_2)$ with $p'_3 \lesssim p'_2$. But then for any $(r, p'_1) \in \mathsf{SC}(p_1)$ for any choice of $(s, p'_3) \in \mathsf{SC}(p_3)$ that maximizes $(r \sim_p s)\big((1-w) + w(p'_1 \sim_i^{\max} p'_3)\big)$, we can

similarly add $(r, p_2') \in \mathsf{SC}(p_2)$ with $p_3' \lesssim p_2'$ to the chosen set $S \subseteq \mathsf{SC}(p_2)$ of successors. This is done analogously for the other direction. But then $\mathrm{sm}(\mathsf{SC}(p_1), S)$ and $\mathrm{sm}(S, \mathsf{SC}(p_1))$ for $(\mathcal{I}'_{Q,\mathcal{T}}, e) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, f)$ will again be at least as large as $\mathrm{sm}(p_1, p_3)$ and $\mathrm{sm}(p_3, p_1)$, respectively.

Putting everything together, we get that indeed $p_1 \sim_i^{\max} p_2 \geq p_1 \sim_i p_3$, in particular $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a) \geq (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p$. □

This means that $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a)$ must also be greater or equal to the supremum $\sup_{p \in \mathfrak{I} \text{ with } p \lesssim (\mathcal{I}_{\mathcal{K}}, d_a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p$. In fact, one can show that both values are equal. But this implies that $\sim_i^{\max}$ indeed decides the relaxed instance answering problem:

Theorem 57. *Given the CSM $\sim_i$ $(\sim_p, g, w)$, an $\mathcal{EL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, an $\mathcal{EL}$ concept Q, and an individual $a \in \mathsf{N_I}$. Then $a \in_t^\sim Q$ iff $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{max} (\mathcal{I}'_{\mathcal{K}}, d_a) \geq t$.*

Proof. For any arbitrary pointed interpretation p, we can construct a sequence of concepts $(X_k)_k$by simply traversing up to role-depth k, such that

$$\lim_{k \to \infty} ((\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i (\mathcal{I}'_{X_k,\mathcal{T}}, d_{X_k})) = (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p,$$

as $(\mathcal{I}'_{X_k,\mathcal{T}}, d_{X_k})$ coincides with p up to role-depth k and $\sim_i$ is discounting. But then

$$\sup_{X \in \mathfrak{C}(\mathcal{EL}) \text{ with } \mathcal{K} \models X(a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i (\mathcal{I}'_{X,\mathcal{T}}, d_X) = \sup_{p \in \mathfrak{I} \text{ with } p \lesssim (\mathcal{I}_{\mathcal{K}}, d_a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p,$$

and thus

$$a \in_t^\sim Q \quad \text{iff} \quad \sup_{p \in \mathfrak{I} \text{ with } p \lesssim (\mathcal{I}_{\mathcal{K}}, d_a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p \geq t.$$

From Lemma 56 it follows directly that

$$(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a) \geq \sup_{p \in \mathfrak{I} \text{ with } p \lesssim (\mathcal{I}_{\mathcal{K}}, d_a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p.$$

All that is left to show is the existence of a pointed interpretation $p_{\max}$ with $p_{\max} \lesssim (\mathcal{I}_{\mathcal{K}}, d_a)$ and

$$(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a) = (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p_{\max}.$$

We can construct such a pointed interpretation $p_{\max}$ by following along the guesses of the subsets $C \subseteq \mathsf{CN}(q)$ and $S \subseteq \mathsf{SC}(q)$ that $\sim_i^{\max}$ had to make for each pair p, q when computing $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a)$. We denote the set $C \subseteq \mathsf{CN}(q)$ that was chosen during the computation of $p \sim_i^{\max} q$ with $C_{p,q}$; similarly, we denote the chosen set $S \subseteq \mathsf{SC}(q)$ with $S_{p,q}$. Then, we can define $p_{\max} = (\mathcal{I}_{\max}, (d_Q, d_a))$ as

follows:

$$\begin{aligned}
\Delta^{\mathcal{I}_{\max}} &= \Delta^{\mathcal{I}'_{Q,\mathcal{T}}} \times \Delta^{\mathcal{I}'_{\mathcal{K}}},\\
A^{\mathcal{I}_{\max}} &= \{(p,q) \mid A \in C_{p,q}\},\\
r^{\mathcal{I}_{\max}} &= \{((p,q),(p',q')) \mid (r,q') \in S_{p,q} \text{ and } (p',q') \text{ was}\\
&\qquad\qquad \text{a best successor-pair in } \mathrm{sm}(\mathrm{SC}(p), S_{p,q})\}
\end{aligned}$$

It is easy to see that $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a) = (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p_{\max}$. But then we have

$$(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a) = \sup_{p \in \mathfrak{I} \text{ with } p \lesssim (\mathcal{I}_{\mathcal{K}}, d_a)} (\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i p$$

and thus $a \in_t^{\sim} Q$ iff $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, d_a) \geq t$. □

Similarly to $\sim_i$, computing $(\mathcal{I}, a) \sim_i^{\max} (\mathcal{J}, b)$ involves collecting all the equations for $p \sim_i q$ with $p \in \mathfrak{I}_{\mathcal{I}}$ and $q \in \mathfrak{I}_{\mathcal{J}}$ into an equation system and solving it. Also similarly to $\sim_i$, one can show that $\sim_i^{\max}$ is well-defined, but this also follows from the correspondence to $\sim_i$ established in the proof of Theorem 57.

However, the PTime complexity of $\sim_i$ does not carry over to $\sim_i^{\max}$.

4.2.1 Complexity

Recall that we could show that $\sim_i$ can be computed in PTime by transforming it into an equivalent linear optimization problem of polynomial size. The maxima in $\sim_i$ could be eliminated by introducing new variables $x_{(r,p'),q}$ and $x_{p,(s,q')}$ (compare Definition 30). However, such an elimination is not possible for $\sim_i^{\max}$: since we choose subsets $C \subseteq \mathrm{CN}(q)$ and $S \subseteq \mathrm{SC}(q)$, and there are exponentially many such subsets, eliminating the maximum $\max_{C \subseteq \mathrm{CN}(q), S \subseteq \mathrm{SC}(q)}$ would require the introduction of exponentially many new variables.

However, one can simply guess all the subsets non-deterministically before creating the linear optimization problem. This yields an NP-algorithm for relaxed instance checking.

Theorem 58. *Given the CSM $\sim_i$ $(\sim_p, g, w)$, an $\mathcal{EL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, an $\mathcal{EL}$ concept Q, and an individual $a \in \mathsf{N_I}$. Checking whether $a \in_t^{\sim} Q$ can be done in non-deterministic polynomial time.*

Proof. For two interpretations $\mathcal{I}$ and $\mathcal{J}$ and each pair $(p,q) \in \mathfrak{I}_{\mathcal{I}} \times \mathfrak{I}_{\mathcal{J}}$, we guess the subsets $C_{p,q} \subseteq \mathrm{CN}(q)$ and $S_{p,q} \subseteq \mathrm{SC}(q)$. The linear optimization problem $P_{\mathrm{sim}}^{\max}(\mathcal{I}, \mathcal{J}, \boldsymbol{C}, \boldsymbol{S})$ is defined by the following linear constraints with variables $x_{p,q}$,

$x_{(r,p'),q}$, $x_{p,(s,q')}$ for all $p \in \mathfrak{I}_\mathcal{I}$, $q \in \mathfrak{I}_\mathcal{J}$, $(r,p') \in \mathsf{SC}(p)$ and $(s,q') \in \mathsf{SC}(q)$:

$$x_{p,q} = \frac{\mathrm{cm}(\mathsf{CN}(p), C_{p,q}) + \mathrm{cm}(C_{p,q}, \mathsf{CN}(p)) + \sum_{(r,p')\in\mathsf{SC}(p)} x_{(r,p'),S_{p,q}} + \sum_{(s,q')\in S_{p,q}} x_{\mathsf{SC}(p),(s,q')}}{\sum_{A\in\mathsf{CN}(p)} g(A) + \sum_{B\in C_{p,q}} g(B) + \sum_{(r,p')\in\mathsf{SC}(p)} g(r) + \sum_{(s,q')\in S_{p,q}} g(s)} \tag{4.1}$$

$$\mathrm{cm}(C_1, C_2) = \sum_{A\in C_1} \left(g(A) \max_{B\in C_2} (A \sim_p B) \right)$$

$$x_{(r,p'),S} \geq g(r)(r \sim_p s)((1-w) + w \cdot x_{p',q'}) \qquad \text{for all } (s,q') \in S$$

$$x_{S,(s,q')} \geq g(s)(r \sim_p s)((1-w) + w \cdot x_{p',q'}) \qquad \text{for all } (r,p') \in S$$

and the objective function

$$\sum_{(p,q)\in\mathfrak{I}_\mathcal{I}\times\mathfrak{I}_\mathcal{J}} x_{p,q},$$

which should be minimized.

Analogously to Lemma 31, the value of $x_{(\mathcal{I},d),(\mathcal{J},e)}$ in the minimal solution to $P^{\max}_{\mathsf{sim}}(\mathcal{I}, \mathcal{J}, \boldsymbol{C}, \boldsymbol{S})$ corresponds exactly to the value $(\mathcal{I}, d) \sim_i^{\max} (\mathcal{J}, e)$, provided that $\sim_i^{\max}$ would always choose exactly the sets $C_{p,q}$ and $S_{p,q}$.

Let now $\mathcal{I}'_{Q,\mathcal{T}}$ and $\mathcal{I}'_\mathcal{K}$ be the normalized canonical models of Q and $\mathcal{A}$. If $a \in_t^\sim (Q)$, then $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_\mathcal{K}, d_a) \geq t$ by Theorem 57, and thus for every $(p,q) \in \mathfrak{I}_{\mathcal{I}'_{Q,\mathcal{T}}} \times \mathfrak{I}_{\mathcal{I}'_\mathcal{K}}$ there exist subsets $C_{p,q} \subseteq \mathsf{CN}(q)$ and $S_{p,q} \subseteq \mathsf{SC}(q)$ such that the value of $x_{(\mathcal{I}'_{Q,\mathcal{T}},d_Q),(\mathcal{I}'_\mathcal{K},d_a)}$ in the minimal solution of $P^{\max}_{\mathsf{sim}}(\mathcal{I}'_{Q,\mathcal{T}}, \mathcal{I}'_\mathcal{K}, \boldsymbol{C}, \boldsymbol{S})$ for those choices is greater or equal to t. Similarly, if $a \notin_t^\sim (Q)$, then for any choice of $\boldsymbol{C}, \boldsymbol{S}$ the value of $x_{(\mathcal{I}'_{Q,\mathcal{T}},d_Q),(\mathcal{I}'_\mathcal{K},d_a)}$ in the minimal solution of $P^{\max}_{\mathsf{sim}}(\mathcal{I}'_{Q,\mathcal{T}}, \mathcal{I}'_\mathcal{K}, \boldsymbol{C}, \boldsymbol{S})$ must be smaller than t, since otherwise $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{\max} (\mathcal{I}'_\mathcal{K}, d_a)$ could not be smaller than t either.

Thus, we can decide whether $a \in_t^\sim (Q)$ in NP time. □

Besides deciding whether an element is a relaxed instance of Q, we can also give a non-deterministic polynomial time algorithm to compute all relaxed instances of a query concept Q. Instead of checking each individual by itself, the linear optimization problem in (4.1) already contains all the elements d_a for each individual a occurring in the knowledge base $\mathcal{K}$. As such, it is possible to solve the linear optimization problem only once and read of the degrees $x_{(\mathcal{I}'_{Q,\mathcal{T}},d_Q),(\mathcal{I}'_\mathcal{K},d_a)}$ for each such individual.

Corollary 59. *Given the CSM $\sim_i$ $(\sim_p, g, w)$, an $\mathcal{EL}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, and an $\mathcal{EL}$ concept Q. Computing all relaxed instances of Q w.r.t. $\mathcal{K}$ and $\sim_i$ can be done in non-deterministic polynomial time.*

Compared the case of unfoldable TBoxes, the intimate knowledge about the used CSM allows us to reduce the complexity from non-elementary to NP, and even handle general $\mathcal{EL}$ TBoxes. However, we no longer allow arbitrary similarity measures, but require a CSM from the $\sim_c$ family.

4.2.2 Extension to $\mathcal{EL}^{++}$

Similarly to the CSM $\sim_c$, one can also extend relaxed instance query answering to work w.r.t. $\mathcal{EL}^{++}$ KBs. However, for this we restrict the p-admissible concrete domains even further; we only allow infinite data-types [OWL09], i.e., the only predicate is an assignment. Concrete domains may therefore only have the form $\mathcal{V} = (V, =_v)$ consisting of an infinite set V of values and unary predicates $=_v$ for $v \in V$ with $(=_v)^{\mathcal{V}} = \{v\}$.

This has a couple of reasons. First of all, data-types guarantee that if the similarity measure on the concrete domain, $\sim_{\mathcal{D}}$, can be computed in polynomial time, then this is also true for the Hausdorff measure $\text{sim}_{\mathcal{D}}$. Second, in many ontologies used in practice data-types are only attached to individuals in the ABox as meta data. Third, in the context of similarity measures, even simple data-type assignments can be very useful. For this, consider the following example:

Example 60. Consider the concrete domain G to represent geographic coordinates as a pair of latitude and longitude, with $\Delta^G = [-90, 90] \times [-180, 180] \subseteq \mathbb{R} \times \mathbb{R}$ and the unary predicates $=_p$ for $p \in \Delta^G$. This allows to attach a location to any individual in the ABox using assertions like $(=_{(51.026,13.723)}(\mathsf{location}))(\mathsf{a})$. If we construct the concrete similarity measure $\sim_{\mathcal{D}}$ used for relaxing the queries in such a way that it assigns larger similarities to locations closer together, an instance query which includes the predicate $=_l(location)$ for the location of the user will find the closest individuals that also match the rest of the query. Indeed, one could also construct a similarity measure that returns similarity 0 for locations more than a set distance away, allowing the user to specify the maximum distance. Thus, while the concrete domain itself is extremely inexpressive, it allows the relaxed instance queries to include the distance between locations in its similarity evaluation. ♢

The maximal interpretation similarity measure $\sim_i^{\max}$ can again be straight-forwardly extended to $\mathcal{EL}^{++}$. The only thing we need to be careful about for this extension are concrete domains. As data-types have assignments as the only predicate, any conjunction of predicates will have a simple form: It is essentially a valuation that maps each feature name to its assigned value (or $\bot$ if there is no predicate for this feature). But then, we do not need to guess a subset of all predicates of q, but we simply take all predicates q, for which there also exists a predicate with the same feature name in p. It is easy to see that this will maximize the term fm of the similarity measure. Thus, we can define the maximal interpretation similarity measure for $\mathcal{EL}^{++}$ as follows:

Definition 61. Given a primitive measure $\sim_p$, a weighting function g, a similarity measure $\sim_{\mathcal{D}}$ on the concrete domain, a discounting factor w, a concrete domain factor c, as well as two interpretations $\mathcal{I}$ and $\mathcal{J}$, the maximal interpretation similarity measure $\sim_i^{\max}(\sim_p, \sim_{\mathcal{D}}, g, w, c) : \mathfrak{P} \times \mathfrak{P} \to [0, 1]$ is defined as follows, for all $p, q \in \mathfrak{P}$: If $f_C(p) = f_C(q) = f_R(p) = f_R(q) = f_I(p) = f_I(q) = f_F(p) = f_F(q) = \emptyset$,

then $p \sim_i q = 1$, otherwise

$$p \sim_i q = \max_{\substack{C \subseteq f_C(q) \\ S \subseteq f_R(q) \\ I \subseteq f_I(q)}} \left(\frac{\mathrm{cm}(f_C(p), C) + \mathrm{cm}(C, f_C(p)) + \mathrm{sm}(f_R(p), S) + \mathrm{sm}(S, f_R(p))}{\text{weight}} + \frac{\mathrm{im}(f_I(p), I) + \mathrm{im}(I, f_I(p)) + \mathrm{fm}(p, q)}{\text{weight}} \right)$$

with

$$\begin{aligned} \text{weight} = &\sum_{A \in f_C(p)} g(A) + \sum_{A \in f_C(q)} g(A) + \sum_{(r,p') \in f_R(p)} g(r) + \sum_{(r,q') \in f_R(q)} g(r) \\ &+ \sum_{a \in f_I(q)} g(a) + \sum_{b \in f_I(q)} g(b) + g_F(p, q), \end{aligned}$$

$$\mathrm{cm}(S_1, S_2) = \sum_{A \in S_1} \left(g(A) \max_{B \in S_2} A \sim_p B \right),$$

$$\mathrm{sm}(S_1, S_2) = \sum_{(r,p') \in S_1} \left(g(r) \max_{(s,q') \in S_2} \left((r \sim_p s) \big((1 - w) + w(p' \sim_i^{\max} q') \big) \right) \right),$$

$$\mathrm{im}(S_1, S_2) = \sum_{a \in S_1} \left(g(a) \max_{b \in S_2} a \sim_p b \right),$$

$$g_F(p, q) = \begin{cases} c & \text{if } f_F(p) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}, \text{ and}$$

$$\mathrm{fm}(p, q) = g_F(p, q) \cdot \frac{\sum_{f \in \mathrm{dom}(u)} g(f) \cdot \mathrm{sim}_{\mathcal{D}}(u(f), v(f))}{\sum_{f \in \mathrm{dom}(u)} g(f)},$$

where u and v are the unique minimal valuations satisfying $f_F(p)$ and $f_F(q)$. ♢

It is straight-forward to show that $\sim_i^{\max}$ indeed decides the relaxed instance answering problem for $\mathcal{EL}^{++}$:

Theorem 62. *Given the CSM $\sim_i$ $(\sim_p, \sim_{\mathcal{D}}, g, w, c)$, an $\mathcal{EL}^{++}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, an $\mathcal{EL}^{++}$ concept Q, and an individual $a \in \mathsf{N_I}$. Then $a \in_t^{\sim} Q$ iff $(\mathcal{I}'_{Q,\mathcal{T}}, d_Q) \sim_i^{max} (\mathcal{I}'_{\mathcal{K}}, d_a) \geq t$.*

Again, we will skip the proof, since it is mostly the same as the proofs to Lemma 56 and Theorem 57. Indeed, if $\sim_{\mathcal{D}}$ can be computed in polynomial time, then it also also easy to show that relaxed instances can be decided in NP for $\mathcal{EL}^{++}$ as well.

4.3 Implementation and Evaluation

In this section, we want to present a first implementation of the relaxed instance answering problem, the ELASTIQ system[1], and perform an evaluation of both the usefulness of relaxed instance queries in practice, and the performance of ELASTIQ.

[1] ELASTIQ is open source and can be found at `https://github.com/MaxPensel/ELastiq`

4.3.1 The ELASTIQ System

Recall that the upper complexity bound of NP for answering relaxed instance queries w.r.t. the CSM $\sim_c$ was shown by non-deterministically guessing subsets of concept names and successors for the canonical model of the knowledge base, and then translating $\sim_i^{\max}$ into a linear optimization problem of polynomial size and solving it. However, this approach is not practical. Instead, ELASTIQ implements an iterative approach that refines the similarity values in each iteration and converges to $\sim_i^{\max}$ in the limit. In essence, this is very similar to the contraction mapping defined in Eq. (3.2) for $\sim_i$. This iterative approach is sound, as it converges to the maximal similarity values from below, but it is not necessarily complete. However, it allows to approximate the maximal similarity values arbitrarily close and converges quite fast (application of the Banach fixed point-theorem indeed implies linear convergence speed [Ban22]), so the incompleteness is not a huge problem in practice.

In order to compute the answers to a relaxed instance query, ELASTIQ proceeds in four main steps.

Step 1: Global preprocessing.
The canonical model $\mathcal{I}_\mathcal{K}$ of the ABox and the TBox is generated by the use of a standard DL reasoner; currently ELASTIQ uses the ELK system [KKS14].

Step 2: Local preprocessing.
The canonical model of the query concept w.r.t. the TBox $\mathcal{I}_{Q,\mathcal{T}}$ is generated—as in Step 1 by the use of ELK.

ELASTIQ distinguishes the two preprocessing steps for the sake of computing several relaxed instance queries against the same KB faster. In such a case, $\mathcal{I}_\mathcal{K}$ does not depend on the query concept and can therefore be reused for every subsequent queries. The canonical model $\mathcal{I}_{Q,\mathcal{T}}$ however needs to be recreated for every query concept Q. In both steps we use the ELK reasoner to compute classification and realization of the ontology, and then retrieve subsumption and instance relationships from the results. In the canonical models that are built, each domain element corresponds to an individual or subconcept occurring in Q or $\mathcal{K}$. ELASTIQ only needs to consider those domain elements that are reachable from elements representing ABox individuals or Q and thus can be used by the main algorithm.

Similar to database theory, we assume that in practice the model $\mathcal{I}_{Q,\mathcal{T}}$ stays fairly small (depending on the structure of the query concept), whereas the size of $\mathcal{I}_\mathcal{K}$ depends on the number of individuals in the ABox, which can grow quite large. The normal forms of the canonical models are computed on-the-fly during the creation of the canonical models, by reusing the classification results.

Step 3: Computing the maximal interpretation similarity $\sim_i^{\max}$.
Recall that ELASTIQ implements an iterative approach, that refines the similarity values and converges to $\sim_i^{\max}$ in the limit. Thus the main computation yields a sequence of matrices $M_0, M_1, M_2, \ldots$, each representing an iteration of the compu-

tation. The rows of such a matrix M_j represent domain elements from $\mathcal{I}'_{Q,\mathcal{T}}$ and the columns domain elements from $\mathcal{I}'_{\mathcal{K}}$. The values inside each cell of M_j are then identified by two domain elements $d \in \Delta^{\mathcal{I}_{Q,\mathcal{T}}}$ and $e \in \Delta^{\mathcal{I}_{\mathcal{K}}}$, and converge towards $(\mathcal{I}'_{Q,\mathcal{T}}, d) \sim_i^{\max} (\mathcal{I}'_{\mathcal{K}}, e)$ for $j \to \infty$.

Instead of computing the similarity values for all pairs of elements from the canonical models in each iteration, ELASTIQ restricts the entries in M_j to those elements that are reachable from pairs (d_Q, d_{a_i}) of the query concept and an ABox individual a_i by mutual paths in $\mathcal{I}'_{Q,\mathcal{T}}$ in $\mathcal{I}'_{\mathcal{K}}$. To this end M_0 is initialized with one row (for d_Q) and as many columns as there are individuals in the ABox. The set of columns is extended with new elements (d', e') if there exists an element (d, e) in M_0 such that d and d' are connected in $\mathcal{I}'_{Q,\mathcal{T}}$ via some role r, and e and e' are connected in $\mathcal{I}'_{\mathcal{K}}$ via some role s. Since the canonical models $\mathcal{I}'_{Q\mathcal{T}}$ and $\mathcal{I}'_{\mathcal{K}}$ are finite, the size of M_0 is bounded by $|\Delta^{\mathcal{I}'_{Q,\mathcal{T}}}| \cdot |\Delta^{\mathcal{I}'_{\mathcal{K}}}|$. Once all reachable pairs have been added to M_0, it contains values exactly for those pairs that are necessary for computing similarities between the domain elements that we are interested in—namely similarities between the query concept Q and each ABox individual: (d_Q, d_{a_i}).

In the beginning, each element of M_0 is initialized with the value 0. Each iteration $j+1$ creates a new matrix M_{j+1}, and computes the values by applying Equation (55) to the values in M_j. At each point, ELASTIQ needs only to keep the current matrix M_{j+1} and the last matrix M_j ($j \geq 0$) in memory. The iterations for the refinement of similarity values proceeds until one of the following termination criteria is met:

- the maximal amount of iterations i_{max} specified by the user is reached; or
- no values have changed during the last iteration by more than a relative factor specified by the user.

Step 4: Comparison with t.
After the iteration stopped, the similarity values $M_j(d_Q, d_a)$ are compared to the input threshold t and the answer set of individuals is compiled. This set is then listed in descending order of similarity.

4.3.2 Optimizations for Computing Relaxed Instances

A naive implementation of the algorithm cannot reasonably compute relaxed instances for large ontologies in acceptable time. As mentioned before, a highly effective optimization is the reuse of $\mathcal{I}_{\mathcal{K}}$ for multiple queries. Since ABoxes are usually much larger than query concepts, the model $\mathcal{I}_{\mathcal{K}}$ is also much more costly to create than the models $\mathcal{I}_{Q,\mathcal{T}}$.

Additionally, the normalization of canonical models can be done more efficiently than by computing simulations to determine unnecessary role-successors. Before adding a domain element d_C as an r-successor to some element d_D, ELASTIQ checks whether there already exists an r-successor d_E for d_D such that $E \sqsubseteq C$. In this case normalization would eliminate d_C, thus avoiding the introduction of d_C (and its role successors) improves the runtime of the canonical model generation fur-

ther. Similarly, when adding d_C as an r-successor to d_D, ELASTIQ eliminates all r-successors d_E of d_D for which $C \sqsubseteq E$.

During the generation of the canonical models, ELASTIQ performs many subsumption checks. Although ELK is currently one of the fastest reasoners for $\mathcal{EL}$, caching of sub- and superclass relations yields a great performance boost since ELASTIQ needs to access these relationships for the same class several times.

The Definition for $\sim_i^{\max}$ suggests to iterate over all subsets of CN and SC in order to find the maximal similarity. This exponential procedure can be improved by looking at the primitive similarities between elements. Let $d \in \mathcal{I}'_{Q,\mathcal{T}}$ and $e \in \mathcal{I}'_{\mathcal{K}}$. By definition of $\sim_i^{\max}$ we are looking for those subsets of the concept names and successors of e that maximize the similarity. Instead of iterating over all subsets of $\mathrm{CN}(e)$, it is easy to see that if $B \in \mathrm{CN}(e)$ for which there exists some $A \in \mathrm{CN}(d)$ with $A \sim_p B = 1$, we can always keep B in the subset of $\mathrm{CN}(e)$, because it can only increase the similarity. Conversely, if $B' \in \mathrm{CN}(e)$ such that for all $A \in \mathrm{CN}(d)$ we have $A \sim_p B' = 0$, then B' can be left out of the subset of $\mathrm{CN}(e)$ since it cannot increase the maximal similarity value. Analogously, we can remove (s, q) from $\mathrm{SC}(e)$ if for all $(r, p) \in \mathrm{SC}(d)$ we have $r \sim_p s = 0$. This can dramatically reduce the number of subsets that need to be checked. In fact, for the default primitive measure, this means that the best subset of the concept names of e can always be computed in linear time; it is simply $\mathrm{CN}(e) \cap \mathrm{CN}(d)$.

4.3.3 Experimental Evaluation

For the experimental evaluation of ELASTIQ, we use two test cases: one hand-crafted ontology, and one set of real-world ontologies. The first test is done with a small hand-crafted ontology that describes describes bicycles; in this ontology, it is possible to evaluate the quality of the relaxed answers, in order to see if the relaxed instance query approach works as intended. The second test is based on the Gene Ontology [Gen00], and compares fragments of this ontology of a different size. This is useful in order to judge how the size of both the TBox and the ABox influence the performance of ELASTIQ.

Bicycle ontology

This ontology is a small made-up example ontology used to perform some qualitative evaluation of relaxed instance queries. The ontology contains 58 concept names ordered in the hierarchy given in Figure 4.3 and 15 individuals that describe a variety of bikes (5 city bikes, 1 cyclocross bike, and 3 road races, trekking bikes and mountain bikes each). The ontology uses the following 8 role names: hasPart, hasWeight, hasPrice, hasColor, height, numberOfGears, madeFrom, and useCase. Besides the concept hierarchy, the TBox also encodes some additional knowledge, for example that mountain bikes always have treaded tires and allow the use case Offroad.

In order to evaluate ELASTIQ on this ontology, we asked a sequence of 4 relaxed instance queries (note that we abbreviated some of the concept and role names):

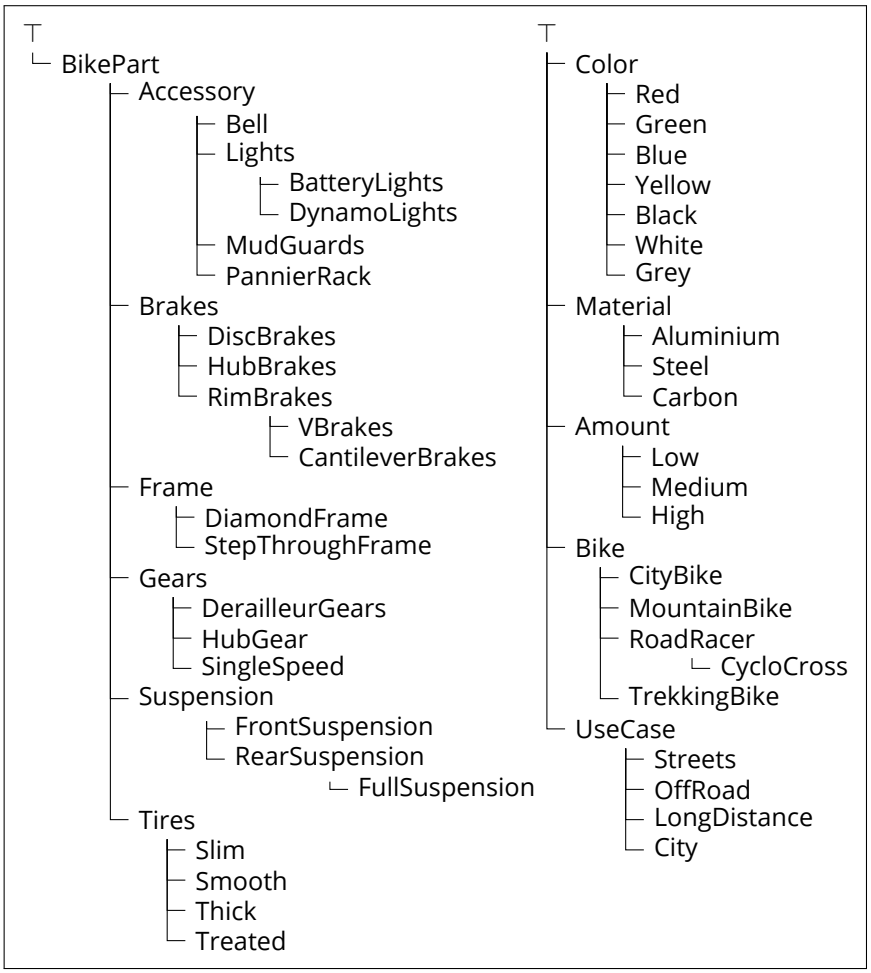

Figure 4.3: Concept hierarchy for the bicycle ontology

1. $Q_1 = \mathsf{Bike} \sqcap \exists \mathsf{price.Low} \sqcap \exists \mathsf{part.}(\mathsf{Fame} \sqcap \exists \mathsf{height.High}) \sqcap \exists \mathsf{part.HubGear}$ asks for any cheap bikes that have a large frame and hub gear instead of derailleur gears.
2. $Q_2 = \mathsf{MountainB} \sqcap \mathsf{RoadRacer} \sqcap \exists \mathsf{weight.Low} \sqcap \exists \mathsf{price.Medium}$ ask for bikes that are both mountain bikes and road racers and have a low weight, but medium price.
3. $Q_3 = \mathsf{CycloCross} \sqcap \exists \mathsf{price.Low} \sqcap \exists \mathsf{part.SingleSpeed}$ ask for a cheap cyclocross bike without gears.
4. $Q_4 = \mathsf{TrekkingB} \sqcap \exists \mathsf{part.DynamoLights} \sqcap \exists \mathsf{part.PannierRack} \sqcap \exists \mathsf{part.HubGear} \sqcap \exists \mathsf{weight.Low}$ asks for light trekking bikes with hub gear, dynamo lights and a pannier rack.

For answering relaxed queries we use a primitive measure that coincides with the default primitive measure for all pair except for $\mathsf{Low} \sim_p \mathsf{Medium} = 0.5$ and $\mathsf{Medium} \sim_p \mathsf{High} = 0.5$. We also use a weighting function that assigns weight 0.1

to the general concept names (all concept names that have $\top$ or BikePart as direct superconcept), and weight 1 to all other concept names. The discounting factor was set at $w = 0.8$.

When answering the queries Q_1 to Q_4 relaxed with a threshold of $t = 0.8$ we get the following answers:

- Q_1 returns 12 answers with a similarity of up to 0.92, the answers include all bikes except the cyclocross bike, one trekking bike and one road racer.
- Q_2 returns 7 answers (all mountain bikes and road racers, as well as the cyclocross bike); the best answer is the cyclocross bike with a similarity of 0.89, since it is closest to being both a road racer and a mountain bike and has a low weight, but high price.
- Q_3 returns only a single answer, since the query is pretty unrealistic. The answer is indeed the cyclocross bike, and has a similarity of 0.91.
- Q_4 returns 5 answers; the three trekking bikes take the first three places, two of the city bikes follow. The other types of bikes mostly have no or very few accessories. The most similar answer is trekking bike #2 with a similarity of 0.96; this bike indeed has all the accessories, has a hub gear, but medium weight.

In a second test we focused on query Q_1: Assume that the customer is unhappy that the answer set includes nearly all bikes, and wants to put more emphasis on the parts that are important, which for him are the price and the frame size. As such, he sets the weights of height and hasPrice to 3. With just this small change, the number of answers immediately drop to 5, and the best answer with a similarity of 0.95 is now shared between and mountain bike and a city bike, both of which have a low price, but a medium sized frame.

During this evaluation, we found that while the answers and the ordering are usually quite reasonable, and the approach to relax instance queries using a similarity measure seems indeed quite useful for this application, there are still a few problems. For example, even after reducing the weights of the more general concept names to 0.1 and with a high discounting factor of 0.8, the similarity values itself are often higher than expected. This means that choosing a threshold t that returns a reasonable number of answers is quite hard and usually involves trying out a lower number first and looking at the similarity values of the returned answers. An alternative to this would be to return the best k answers instead of those above a threshold (top-k query answering). This is currently not implemented in ELASTIQ, but would be easy to add, since ELASTIQ already computes the maximal similarity values of all individuals.

Also, another problem is that it can be quite hard to really focus on features at a role-depth of 2 or deeper. In the example query Q_1, when we wanted to make the height of the frame more important, we could only increase the weight of the role height itself, but not the role hasPart for the frame, since otherwise the gear hub

would have increased in importance as well, which is unintended. As it is now, this query prefers bikes with a low price instead of a large frame, since the price is at role-depth 1, the frame size at role-depth 2. This is an unintended side-effect of the discounting and makes the relaxed query approach dependent on the modeling of the ontology.

Gene Ontology

The second evaluation of ELASTIQ used different fragments of the Gene Ontology that describe *schizosaccharomyces pombe* – a species of yeast that reproduces via medial fission. We used a set of 15 different, increasingly more complex fragments of the Gene Ontology that ranged from 9,157 concept names and 34,875 individuals in the first version to 51,949 concept names and 289,206 individuals for the 15th version. The sizes of canonical models $\mathcal{I}'_{\mathcal{K}}$ ranged from 77,941 to 602,548 elements.

We obtained our test ontologies by the custom dataset generation provided by the Manchester OWL Corpus [MBP13b]. These ontologies are some of the only $\mathcal{EL}$ ontologies with an ABox and a complex TBox that suitable for testing ELASTIQ. However, these Gene Ontology versions are anonymized and therefore any contentual interpretation of the results is virtually impossible. Thus, this evaluation is restricted solely to the performance of ELASTIQ.

We discovered that for each individual e there exists a very fragmented concept assertion in the ABox of the form $\exists \mathsf{is\_a}.C_e(e)$, where the qualification C_e is rarely larger than 3 conjuncts with a role-depth of at most 2.

Our test suite contains 10 randomly generated query concepts with increasingly complex structures. These queries were built over the common signature of versions 1–15 of the Gene Ontology (approximately 1,000 concept names and 4 roles). The smallest query (Query 1) only contained 6 concept and role names and had a role-depth of 2, while the Query 10 had a size of 670 and a role-depth of 5. Due to the plain structure of the concept assertions we wrapped each query concept Q_i in an existential restriction $\exists \mathsf{is\_a}.Q_i$ in order to provoke a more complex computation. For these queries, the sizes of the canonical models $\mathcal{I}'_{Q,\mathcal{T}}$ ranged from 2 to 236 elements.

We evaluated the queries for the default primitive measure and weighting function, and counted the number of relaxed instances for a threshold of $t = 0.333$. The test system had a 1800 MHz dual core processor AMD Turion II Neo and 6 GB of RAM. Figure 4.4 shows the runtime of ELASTIQ for answering all 10 relaxed instance queries w.r.t. each ontology version. ELASTIQ was written in Java and uses OwlAPI 3.5.0.

The high runtimes for ontology versions 11, 12, and especially 13–15 is mainly due to the increase of the size of the canonical model $\mathcal{I}_{\mathcal{K}}$. Query 1 and 2 actually had no relaxed answers in any version of the ontology, i.e., none of the individuals was similar enough. However, most queries returned a lot more relaxed instances for the ontologies 11–15 than for ontologies 1–10. Queries 8 and 9 returned the largest number of relaxed instances, up to over 200,000 for Query 8 evaluated on ontology 15.

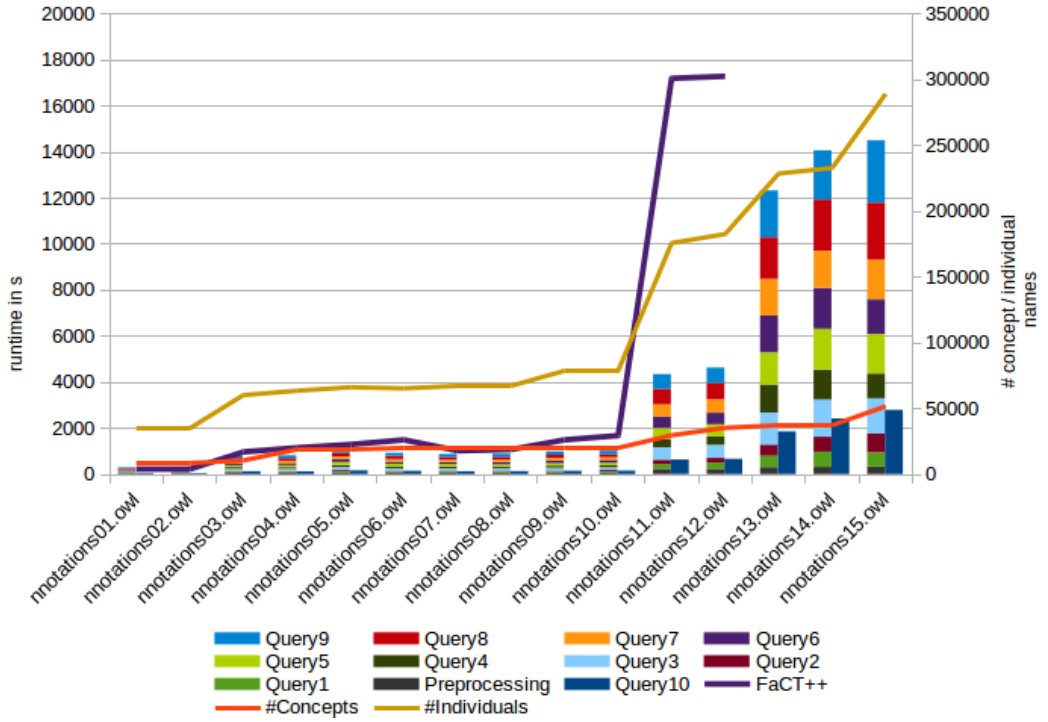

Figure 4.4: ELASTIQ's runtime for answering relaxed instance queries in different versions of the Gene Ontology

When breaking down the times for preprocessing and the query answering further, it shows that the preprocessing time is dominated by the flattening of the ontology, the reasoning done by ELK, and the construction of the canonical model $\mathcal{I}_{\mathcal{K}}$, while the time to construct the canonical models $\mathcal{I}_{Q,\mathcal{T}}$ is negligible. However, the overall query answering time is largely spent on Step 3, i.e., the iterations to compute the maximal similarity.

ELASTIQ performs ABox realization to obtain $\mathcal{I}_{\mathcal{K}}$ and in addition a kind of relaxed ABox realization for the query concepts in the test suite. Now, while it is clear that ELASTIQ is slower than ELK for ABox realization, it showed, surprisingly, that this does not need to be the case for other optimized DL reasoners. We compared ELASTIQ's overall reasoning times with the ABox realization times of the commonly used FaCT++ reasoner [TH06]. Figure 4.5 shows that ELASTIQ mostly performed better than FaCT++, although solving a more complex task[2]. Note that FaCT++ classification resulted in an error for ontologies 13–15. Of course, FaCT++ works for much more expressive DLs than ELASTIQ. However, with computation times of more than a minute for 10 relaxed queries over ABoxes with 1,000 individuals and considering that much larger ontologies are in practical use, this still calls for further improvement depending on the use case.

[2]Since ELASTIQ uses ELK to perform ABox realization as one of its steps, ELK would perform much better than both FaCT++ and ELASTIQ in the above figure.

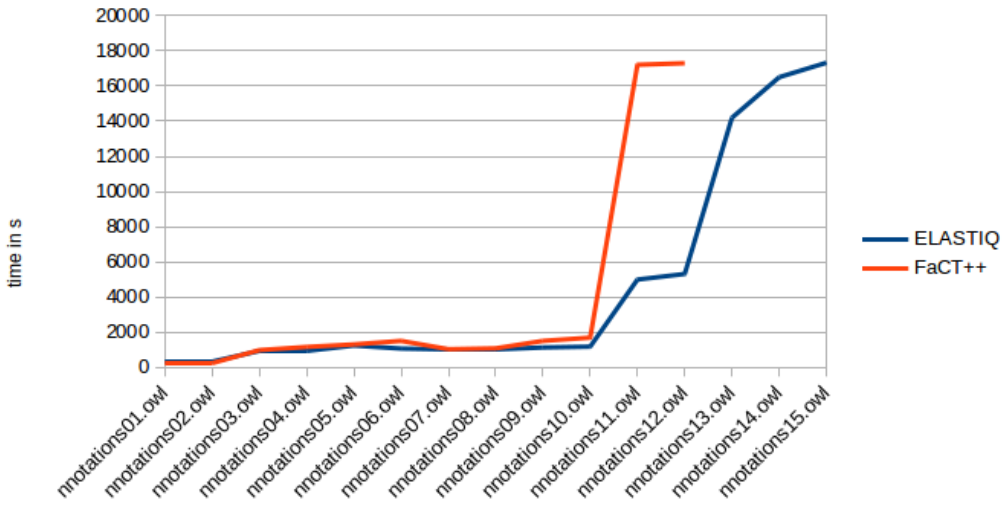

Figure 4.5: Runtime of ELASTIQ compared to FaCT++ ABox Realization times.

4.4 Using Membership Degree Functions to Relax Queries for General $\mathcal{EL}$ TBoxes

In [BBF15], the authors introduce the DL $\tau\mathcal{EL}$, which extends $\mathcal{EL}$ with threshold concepts of the form $C_{\sim t}$ for $t \in [0,1]$ and $\sim \in \{<, >, \leq, \geq\}$. To realize this, they use the notion of graded membership functions:

Definition 63. A *graded membership function* m is a family of functions that contains for every interpretation $\mathcal{I}$ a function $m^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \mathfrak{C}(\mathcal{EL}) \to [0,1]$, satisfying the following conditions (for $C, D \in \mathfrak{C}(\mathcal{EL})$):

- M1: $d \in C^{\mathcal{I}} \Leftrightarrow m^{\mathcal{I}}(d, C) = 1$ for all $d \in \Delta^{\mathcal{I}}$, and
- M2: $C \equiv D \Leftrightarrow$ for all $d \in \Delta^{\mathcal{I}} : m^{\mathcal{I}}(d, C) = m^{\mathcal{I}}(d, D)$. ♢

Using those graded membership functions, the threshold concept $C_{\sim t}$ for $\sim \in \{\leq, \geq, <, >, =\}$is interpreted as $(C_{\sim t})^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid m^{\mathcal{I}}(d, C) \sim t\}$. The authors proceed with introducing a specific membership function $\deg^{\mathcal{I}}(d, C)$, which is defined as a maximal weighted partial homomorphism between the concept tree of the concept C and the pointed interpretation $(\mathcal{I}, d)$.

In the context of relaxed instance query answering, the following properties are notable:

1. deg is very similarly defined to a directed version of $\sim_c$, i.e., where only one direction of cm and sm is evaluated instead of both. Indeed, [BBF15] shows that when converting an instance of the similarity measure *simi* introduced in [LT12] into a graded membership function in a straight-forward way, then the resulting graded membership function m is equivalent to deg.
2. The threshold logic $\tau\mathcal{EL}$ can be used to compute the instances of a threshold

concept $C_{\geq t}$ w.r.t. a classical unfoldable $\mathcal{EL}$ knowledge base $\mathcal{K}$. This is possible in polynomial time [BBF15].

3. Together, the previous points imply that relaxed instances of a query concept Q w.r.t. t, an unfoldable KB $\mathcal{K}$, and the CSM $\sim_i$ can be computed in polynomial time by computing all instances of the threshold concept $Q_{\geq t}$ w.r.t. the membership degree function deg.

Note that, while the definitions of $\sim_c$ and the membership degree function deg seem quite similar, and both can be used to answer relaxed instance queries, there are some important differences. First of all, $\tau\mathcal{EL}$ was not introduced for relaxed instance query answering, but for general reasoning in the full logic with threshold concepts, including algorithms that check subsumption of $\tau\mathcal{EL}$ concepts. A second, important difference concerns the measures $\sim_c$ and deg itself: $\sim_c$ is symmetric, it compares all features of both concepts to find all commonalities and differences. The membership degree function deg is inherently asymmetric though: It only checks if the individual contains the features (concept names and successors) that the threshold concept has, not the other way around. This means that membership degree functions can be used directly to relax queries (i.e., directly compute the membership degree of an individual to the query concept), while for similarity measures one needs the detour of checking the similarity of all concepts that the individual is instance of towards the query concept.

In [BBF15], the threshold logic is only introduced for unfoldable TBoxes. In the following, we will show that the similarity between relaxed instance queries and membership degree functions can be exploited to extend deg to work w.r.t. general TBoxes, and show how to use this to find all instances of a threshold concept of the form $C_{\geq t}$ w.r.t. deg and a general $\mathcal{EL}$ KB. We will not extend other inferences like subsumption and will not look at the case $C_{\leq t}$, since these are not relevant for relaxing instance queries.

4.4.1 Extending the Graded Membership Function deg to General $\mathcal{EL}$ TBoxes

Originally, $deg^{\mathcal{I}}(d, C)$ was defined as the maximal weighted partial tree-to-graph homomorphism between the expanded concept tree C and the pointed interpretation graph $(\mathcal{I}, d)$. This definition generalizes the subsumption relation in $\mathcal{EL}$, which can be defined as the existence of a homomorphism between C and $(\mathcal{I}, d)$. We want to define deg for general $\mathcal{EL}$ TBoxes in a similar way. However, expanding the concept C w.r.t. a general TBox is not possible; instead, we use again the canonical model of C, which will lead to graph-to-graph homomorphisms.

Before introducing the homomorphism, we need to formally define the notion of description graphs. A description graph G is a tuple $G = (V_G, E_G, \ell_G)$ consisting of a set of nodes V_G, a set of edges $E_G \subseteq V_G \times \mathsf{N_R} \times V_G$ labeled with role names, and a labeling function $\ell_G : V_G \to \mathcal{P}(\mathsf{N_C})$ that assigns to each node a set of concept names. DL interpretations can be viewed as description graphs.

Let $G = (V_G, E_G, \ell_G)$ be a description graph with $v \in V_G$. With $\text{Path}(G, v)$ we denote the set of all finite paths in G starting from the node v:

$$\text{Path}(G, v) = \{v_0 r_1 v_1 \dots r_n v_n \mid v_0 = v \wedge (v_{i-1}, r_i, v_i) \in E_G \text{ for } 1 \leq i \leq n\}$$

Then we can define partial graph-to-graph homomorphisms as follows:

Definition 64 (partial graph-to-graph homomorphism). Let $G = (V_G, E_G, \ell_G)$ and $H = (V_H, E_H, \ell_H)$ be two description graphs with a distinguished node $v_0 \in V_G$ such that G is finitely branching. A partial mapping $h : \text{Path}(G, v_0) \nrightarrow V_H$ is a *partial graph-to-graph homomorphism (pggh)* from G to H, if the following conditions are satisfied:

- $\text{dom}(h)$ is prefix-closed, i.e., whenever there is a path $v_0 r_1 v_1 \dots v_{n-1} r_n v_n \in \text{dom}(h)$, then we also have $v_0 r_1 v_1 \dots v_{n-1} \in \text{dom}(h)$;
- for each path $v_0 \dots v_{n-1} r_n v_n \in \text{dom}(h)$ with $n > 0$, we have $(h(v_0 \dots v_{n-1}), r_n, h(v_0 \dots v_{n-1} r_n v_n)) \in E_H$. ♢

Note that although the pggh h maps from paths in G to a node in H, we want this path not to be understood as a path, but instead as the last node in addition to the history of where we came from; this history is only important to distinguish between nodes that we revisit later. This means that h should be understood as a mapping from nodes of G to nodes of H, except that the same node can be mapped to different targets depending on the history.

The second condition implies that all paths in G are mapped to paths in H, i.e., we have that $h(v_0) r_1 h(v_0 r_1 v_1) \dots h(v_0 \dots v_{n-1}) r_n h(v_0 \dots v_{n-1} r_n v_n)$ is a path in H for all $v_0 \dots v_{n-1} r_n v_n \in \text{dom}(h)$.

As in the unfoldable case, it is possible to characterize subsumption between concepts w.r.t. a general $\mathcal{EL}$ TBox by the existence of a complete graph-to-graph homomorphism (as given in Def. 68) between the description graphs of their canonical models. This is basically the same result as for simulations, compare Thm. 11. The partial graph-to-graph homomorphisms generalize this notion. They induce a weighted homomorphism h_w, which measures how close the partial homomorphisms are to being complete homomorphisms.

Definition 65. Let $G = (V_G, E_G, \ell_G)$ and $H = (V_H, E_H, \ell_H)$ be two description graphs with a distinguished node $v_0 \in V_G$ such that G is finitely branching and let $h : \text{Path}(G, v_0) \nrightarrow V_H$ be a pggh from G to H as defined above. We define the *weighted homomorphism induced by* h as a function $h_w : \text{Path}(G, v_0) \to [0, 1]$:

$$h_w(v_0 \dots v_n) = \begin{cases} 1 & \text{if } |\ell_G(v_n)| + k^*(v_n) = 0 \\ \dfrac{|\ell_G(v_n) \cap \ell_H(h(v_0 \dots v_n))| + \sum\limits_{1 \leq i \leq k} (1-w) + w \cdot h_w(v_0 \dots v_n r_i v_i)}{|\ell_G(v_n)| + k^*(v_n)} & \text{otherwise} \end{cases}$$

The elements used to define h_w have the following meaning. For a given $v \in V_G$,

$k^*(v)$ denotes the number of successors of v in G, i.e.,

$$k^*(v) = |\{(r, v') \mid (v, r, v') \in E_G\}|,$$

and for a path $v_0 \dots v_n \in \mathrm{dom}(h)$, we denote with $(r_1, v_1), \dots, (r_k, v_k)$ the extensions of the path $v_0 \dots v_n$ in G such that $v_0 \dots v_n r_i v_i \in \mathrm{dom}(h)$. Additionally, we have a discounting factor $0 < w < 1$. ♢

This definition properly extends the weighted homomorphism for $\mathcal{EL}$ concepts w.r.t. unfoldable TBoxes defined in [BBF15]. In particular, if G is a description tree instead of a more general graph and H is a description graph, then the partial tree-to-graph homomorphisms from [BBF15] and the partial graph-to-graph homomorphism defined here coincide, as do the weighted homomorphisms induced by them (there are notational differences, though).

However, it is not obvious that the well-definedness of weighted homomorphisms given in [BBF15] and its nice properties also transfer. In particular, if G is cyclic or infinite then for any pggh h dom_h may contain infinite sequences of increasing paths. As such, we need to show that the weighted homomorphism h_w is indeed well-defined. This result is very similar to Theorem 29, except now the resulting metric space can be infinite.

Lemma 66. *Let $G = (V_G, E_G, \ell_G)$ and $H = (V_H, E_H, \ell_H)$ be two description graphs with a distinguished node $v_0 \in V_G$ such that G is finitely branching and let further $h : Path(G, v_0) \twoheadrightarrow V_H$ be a pggh from G to H. Then the weighted homomorphism h_w induced by h is well-defined.*

Proof. If $\mathrm{dom}(h)$ is finite, then well-definedness of h_w follows trivially from the definition. Otherwise, let $p = (p_0, p_1, p_2, \dots)$ be a fixed enumeration of all paths $p_i \in \mathrm{dom}(h)$, and let S be the set of all infinite sequences $s = (s_1, s_2, s_3, \dots)$ with values $s_i \in [0, 1]$. With $s(v_0 \dots v_n)$ we denote the value s_i of the sequence with the same index that the path $v_0 \dots v_n$ has in p. We define a mapping $f_h : S \to S$ as follows:

$$f_h((s_i)_{i\geq 0}) = (t_i)_{i\geq 0} \text{ with}$$

$$t_i = \begin{cases} 1 & \text{if } |\ell_G(v_n)| + k^*(v_n) = 0 \\ \dfrac{|\ell_G(v_n) \cap \ell_H(h(p_i))| + \sum\limits_{1\leq j\leq k} (1-w) + w \cdot s(p_i r_j v_j)}{|\ell_G(v_n)| + k^*(v_n)} & \text{otherwise} \end{cases},$$

where v_n is the last node in the path p_i

This mapping basically computes the new membership values for each path by evaluating the right-hand side of the equations h_w for the old membership values. By definition of f_h, the solutions of h_w correspond exactly to the fixed points of f_h.

Thus it is enough to show that f_h has a unique fixed point. This follows from the Banach fixed point theorem if we can show that f_h is a contraction mapping

on the space S of all sequences, since S together with the supremum-norm $\|s\| = \sup_{s_i \in s} s_i$ is a bounded sequence space (l^∞) and thus also a complete metric space.

Let $d(s,t) = \sup_{i \geq 0} |s_i - t_i|$ be the distance between two sequences s and t, and let $s' = f_h(s)$ and $t' = f_h(t)$. Then we have that for each $i \geq 0$, either $|\ell_G(v_n)| + k^*(v_n) = 0$ and thus $s'_i = t'_i = 1$, or

$$\begin{aligned}
|s'_i - t'_i| &= \frac{|\ell_G(v_n) \cap \ell_H(h(p_i))| + \sum\limits_{1 \leq j \leq \mathrm{rk}(p_i)} (1-w) + w \cdot s(p_i r_j v_j)}{|\ell_G(v_n)| + k^*(v_n)} \\
&\quad - \frac{|\ell_G(v_n) \cap \ell_H(h(p_i))| + \sum\limits_{1 \leq j \leq \mathrm{rk}(p_i)} (1-w) + w \cdot t(p_i r_j v_j)}{|\ell_G(v_n)| + k^*(v_n)} \\
&= \frac{\sum\limits_{1 \leq j \leq \mathrm{rk}(p_i)} w \cdot s(p_i r_j v_j) - w \cdot t(p_i r_j v_j)}{|\ell_G(v_n)| + k^*(v_n)} \\
&= w \frac{\sum\limits_{1 \leq j \leq \mathrm{rk}(p_i)} s(p_i r_j v_j) - t(p_i r_j v_j)}{|\ell_G(v_n)| + k^*(v_n)} \\
&\leq w \frac{\sum\limits_{1 \leq j \leq \mathrm{rk}(p_i)} d(s,t)}{|\ell_G(v_n)| + k^*(v_n)} \leq w \cdot d(s,t)
\end{aligned}$$

Thus we have that $d(f_h(s), f_h(t)) \leq w \cdot d(s,t)$ for $w < 1$, and thus f_h is a contraction mapping. The Banach fixed-point Theorem 26 implies that f_h has a unique fixed point and indeed for any sequence $s^0 \in S$, the sequence $(s^0, s^1, s^2, \dots)$ of sequences with $s^{i+1} = f_h(s^i)$ converges to this fixed point. The unique fixed point corresponds exactly to the unique solution of h_w, and thus h_w is well-defined. □

With $\mathcal{H}(G, H, d, e)$ we denote the set of all pgghs h from graph G to graph H with $d \in V_G$ being the root of $\mathrm{dom}(h)$ and $h(d) = e$. Now we can finally define the graded membership function deg for general $\mathcal{EL}$ TBoxes:

Definition 67. Let $\mathcal{T}$ be a general $\mathcal{EL}$ TBox, and $\mathcal{I}$ be a model of $\mathcal{T}$. The *membership degree function deg*$^{\mathcal{I}}$ is defined for all elements $e \in \Delta^{\mathcal{I}}$ and all $\mathcal{EL}$ concepts C as follows:

$$\deg^{\mathcal{I}}(e, C, \mathcal{T}) = \sup_{h \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)} (h_w(d_C)),$$

where $G_{C,\mathcal{T}}$ is the description graph of the canonical model $\mathcal{I}_{C,\mathcal{T}}$ in normal form (see Def. 34), and $G_{\mathcal{I}}$ is the description graph of the interpretation $\mathcal{I}$. ♢

For deg defined in Definition 67 to be a valid graded membership function, we also need to show that it satisfies the properties M1 and M2 given in Definition 63.

4.4.2 Properties of deg

In order to show that deg satisfies properties M1 and M2, we will need to introduce some notations. For some description graph G, a node $v \in V_G$, and $n \in \mathbb{N}$, we

use $\text{Path}_n(G, v)$ to denote the set of all paths in $\text{Path}(G, v)$ of length at most n. We also need to define *complete homomorphisms*, which basically transfer the notion of a simulation to description graphs:

Definition 68. Let G and H be two $\mathcal{EL}$ description graphs with a distinguished node $v_0 \in V_G$ and G finitely branching. A mapping $h : \text{Path}(G, v_0) \to V_H$ is a *complete homomorphism* iff:

- h is pggh and $\text{dom}(h) = \text{Path}(G, v_0)$, and
- for all paths $p = v_0 \dots v_k \in \text{Path}(G, v_0)$: $\ell_G(v_k) \subseteq \ell_H(h(p))$. ♢

Then, we can show that weighted homomorphisms induced by the complete homomorphisms have always value 1.

Lemma 69. *Let G and H be two $\mathcal{EL}$ description graphs with a distinguished node $v_0 \in V_G$ and G finitely branching. Additionally, let $h : \text{Path}(G, v_0) \to V_H$ be a complete homomorphism in the sense of Definition 68. Then, $h_w(v_0) = 1$.*

Proof. This is easy to see from the definition of h_w, since due to $\ell_G(v_k) \subseteq \ell_H(h(p))$ for all $p = v_0 \dots v_k \in \text{Path}(G, v_0)$ we have $|\ell_G(v_k) \cap \ell_H(h(p))| = |\ell_G(v_k)|$, and due to $\text{dom}(h) = \text{Path}(G, v_0)$, we have that if $h_w(pr_j v_j) = 1$ for all extensions $pr_j v_j$ of p in G, then also

$$\sum_{1 \leq i \leq k} (1 - w) + w \cdot h_w(v_0 \dots v_n r_i v_i) = k^*(v_n).$$

Thus, setting $h_w(p) = 1$ for all $p \in \text{Path}(G, v_0)$ gives a valid solution, and since we know that h_w is well-defined, this must be the only solution. Then, in particular, $h_w(v_0) = 1$. □

Conversely, we now show some properties which are consequences of having membership degree value of 1.

Lemma 70. *Let C be an $\mathcal{EL}$ concept description, $\mathcal{T}$ a general $\mathcal{EL}$ TBox, $\mathcal{I}$ a model of $\mathcal{T}$ and $d \in \Delta^{\mathcal{I}}$. If $\deg^{\mathcal{I}}(e, C, \mathcal{T}) = 1$, then for all $n \geq 0$ there exists a pggh h from $\text{Path}(G_{C,\mathcal{T}}, d_C)$ to $G_{\mathcal{I}}$ such that:*

1. *$h(d_C) = e$,*
2. *$\text{Path}_n(G_{C,\mathcal{T}}, d_C) \subseteq \text{dom}(h)$,*
3. *for all paths $\pi = d_C r_1 d_1 \dots r_k d_k \in \text{dom}(h)$ with $k \leq n$: $\ell_G(d_k) \subseteq \ell_{\mathcal{I}}(h(\pi))$.*

Proof. By definition of deg we have:

$$\sup_{h \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)} (h_w(d_C)) = 1$$

Assume that there is $m \in \mathbb{N}$ falsifying the claim. Then, for each $h \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)$ it must be the case that either:

- $\text{Path}_m(G_{C,\mathcal{T}}, d_C) \not\subseteq \text{dom}(h)$, or
- exists a path $\pi = d_C r_1 d_1 \dots r_k d_k \in \text{dom}(h)$ ($k \leq m$) such that:

$$\ell_G(d_k) \not\subseteq \ell_{\mathcal{I}}(h(\pi))$$

In both cases, there will be a constant

$$c = \left(\frac{w}{b \cdot |\mathsf{N_C} \cap (\text{sig}(\mathcal{T}) \cup \text{sig}(\mathcal{T}))|} \right)^m > 0$$

such that $h_w(d_C) \leq 1 - c$, where b is the maximal out-degree of $G_{C,\mathcal{T}}$. This is easy to see by the definition of h_w, where at each node of a path the minimal factor is

$$\frac{w}{b \cdot |\mathsf{N_C} \cap (\text{sig}(\mathcal{T}) \cup \text{sig}(\mathcal{T}))|}.$$

But then $\sup_{h \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)}(h_w(d_C)) \leq 1 - c$. This contradicts $\deg^{\mathcal{I}}(e, C, \mathcal{T}) = 1$. Thus, such a value m cannot exists and our claim holds. □

The following corollary is a direct consequence of the previous lemma.

Corollary 71. *Let C be an $\mathcal{EL}$ concept description, $\mathcal{T}$ a general $\mathcal{EL}$ TBox, $\mathcal{I}$ a model of $\mathcal{T}$ and $d \in \Delta^{\mathcal{I}}$. If $\deg^{\mathcal{I}}(e, C, \mathcal{T}) = 1$, then there exists a sequence of pggh $\{h_0, h_1, \dots\} \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)$ such that:*

- *$\text{Path}_i(G_{C,\mathcal{T}}, d_C) \subseteq \text{dom}(h_i)$,*
- *for all $\pi = d_C r_1 d_1 \dots r_k d_k \in \text{Path}(G_{C,\mathcal{T}}, d_C)$:*
  - *$\ell_G(d_k) \subseteq \ell_{\mathcal{I}}(h_k(\pi))$, and*
  - *$h_j(\pi) = h_k(\pi)$ for all $j > k$.*

Proof. The proof is by contradiction. Suppose that such a sequence do not exists. Let $s = \{h_0, h_1, \dots, h_m\}$ be the largest sequence satisfying such conditions. Lemma 70 yields a pggh $h_{m+1} \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)$ such that:

- $\text{Path}_{m+1}(G_{C,\mathcal{T}}, d_C) \subseteq \text{dom}(h_{m+1})$, and
- $\pi = d_C r_1 d_1 \dots r_k d_{k \leq m+1} \in \text{Path}(G_{C,\mathcal{T}}, d_C)$ implies $\ell_G(d_k) \subseteq \ell_{\mathcal{I}}(h_{m+1}(\pi))$.

Then, it is easy to see that h_{m+1} can be used to define a sequence $\{h_0, h_1, \dots, h_{m+1}\}$ satisfying the same conditions as s. Hence, we have obtained a contradiction against m being the largest possible value. □

We are now ready to show that deg satisfies properties *M1* and *M2*.

Lemma 72. *deg satisfies property M1.*

Proof. Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox, $\mathcal{I}$ be a model of $\mathcal{T}$, $d \in \Delta^{\mathcal{I}}$, and C an $\mathcal{EL}$ concept.

($\Rightarrow$) Assume $d \in C^{\mathcal{I}}$. By the characterization of membership for $\mathcal{EL}$ w.r.t. general TBoxes (see Theorem 13), there is a simulation S between $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}$ such that $(d_C, d) \in S$, i.e., $(\mathcal{I}_{C,\mathcal{T}}, d_C) \lesssim (\mathcal{I}, d)$.

Then, we use S to build a mapping h from $\text{Path}(G_{C,\mathcal{T}}, d_C)$ to $V_{\mathcal{I}}$ such that $h(d_C) = d$ and h is a homomorphism in the sense of Definition 68. The construction is done inductively as follows:

1. $h(d_C) = d$,
2. let $\pi r_k d_k \in \text{Path}(G_{C,\mathcal{T}}, d_C)$ be a path of length k, such that $(d_k, h(\pi r_k d_k)) \in S$. For all $(d_k, r, d^*) \in E_{C,\mathcal{T}}$, we choose e^* such that $(d^*, e^*) \in S$ and $(h(\pi r_k d_k), r, e^*) \in E_{\mathcal{I}}$. Then, $h(\pi r_k d_k r d^*) = e^*$.

Note that if $(d_k, h(\pi r_k d_k)) \in S$, the rest of the construction in the second step is always possible, since S is a simulation. Therefore, as $(d_C, d) \in S$ the whole construction is well-defined. It is then easy to see that h is a complete homomorphism in the sense of Definition 68.

From Lemma 69 we have $h_w(d_C) = 1$. Thus, $\deg^{\mathcal{I}}(d, C, \mathcal{T}) = 1$.

($\Leftarrow$) Assume that $\deg^{\mathcal{I}}(d, C, \mathcal{T}) = 1$. We use the sequence $\{h_0, h_1, \ldots\}$ from Corollary 71 to construct a relation $S \subseteq \Delta^{\mathcal{I}_{C,\mathcal{T}}} \times \Delta^{\mathcal{I}}$ as follows: $(x, y) \in S$ *if, and only if*, there exists a path $\pi = d_C r_1 d_1 \ldots r_k d_k \in \text{Path}(G_{C,\mathcal{T}}, d_C)$ such that $d_k = x$ and $h_k(\pi) = y$.

We now show that S is a simulation between $\mathcal{I}_{C,\mathcal{T}}$ and $\mathcal{I}$. For all $(x, y) \in S$, there is a path $\pi = d_C r_1 d_1 \ldots r_k d_k \in \text{Path}(G_{C,\mathcal{T}}, d_C)$ such that $d_k = x$ and $h_k(\pi) = y$. Hence,

- the properties of $\{h_0, h_1, \ldots\}$ imply that $\ell_G(x) \subseteq \ell_{\mathcal{I}}(y)$.
- let $(x, x') \in r^{\mathcal{I}_C,\mathcal{T}}$. Then, $\pi r x' \in \text{Path}_{k+1}(G_{C,\mathcal{T}}, d_C)$, and from Corollary 71 we obtain $\pi r x' \in \text{dom}(h_{k+1})$. Furthermore, it is also the case that $h_k(\pi) = h_{k+1}(\pi) = y$. Hence, by definition of a pggh there must exits y' such that $h_{k+1}(\pi r x') = f'$ and $(y, r, y') \in E_{\mathcal{I}}$. Finally, by construction of S it follows that $(x', y') \in S$.

We have just shown that S is a simulation. Since $(d_C, d) \in S$, the characterization of membership gives $d \in C^{\mathcal{I}}$. □

Lemma 73. *deg satisfies property M2.*

Proof. The right to left direction follows from Property *M1*. For the other direction we use the characterization of subsumption w.r.t. general TBoxes. Assume $C \equiv_{\mathcal{T}} D$, then we have $(\mathcal{I}_{C,\mathcal{T}}, d_C) \lesssim (\mathcal{I}_{D,\mathcal{T}}, d_D)$ and $(\mathcal{I}_{D,\mathcal{T}}, d_D) \lesssim (\mathcal{I}_{C,\mathcal{T}}, d_C)$. Using the first simulation one can show the following: for any pggh h from $\text{Path}(G_{C,\mathcal{T}}, d_C)$ to $G_{\mathcal{I}}$ with $h(d_C) = d$, there is a pggh g from $\text{Path}(G_{D,\mathcal{T}}, d_D)$ to $G_{\mathcal{I}}$ with $g(d_D) = d$ such that: $h_w(d_C) \leq g_w(d_D)$. Hence, we obtain:

$$\deg^{\mathcal{I}}(d, C, \mathcal{T}) \leq \deg^{\mathcal{I}}(d, D, \mathcal{T})$$

Using the same reasoning, from the second simulation it holds:

$$\deg^{\mathcal{I}}(d, D, \mathcal{T}) \leq \deg^{\mathcal{I}}(d, C, \mathcal{T})$$

Thus, $\deg^{\mathcal{I}}(d, C, \mathcal{T}) = \deg^{\mathcal{I}}(d, D, \mathcal{T})$. □

Therefore, deg is indeed a valid graded membership function for general $\mathcal{EL}$ TBoxes. Finally, we show that the set of all instance of a threshold concept $C_{\geq t}$ w.r.t. some $\mathcal{EL}$ KB $\mathcal{K}$ and the deg can be computed in polynomial time.

4.4.3 Complexity

If the interpretation $\mathcal{I}$ is finite, it is possible to compute $\deg^{\mathcal{I}}(d, C, \mathcal{T})$. In this case, both description graphs $G_{C,\mathcal{T}}$ and $G_{\mathcal{I}}$ are finite. We will introduce an equation system with variables x_{v_1,v_2} for $v_1 \in V_{G_{C,\mathcal{T}}}$ and $v_2 \in V_{G_{\mathcal{I}}}$:

$$x_{v_1,v_2} = \begin{cases} 1 & \text{if } |l_G(v_1)| + k^*(v_1) = 0 \\ \dfrac{|l_G(v_1) \cap l_H(v_2)| + \sum\limits_{(v_1,r,v_3) \in E_{G_{C,\mathcal{T}}}} \max\limits_{(v_2,r,v_4) \in E_{G_{\mathcal{I}}}} (1-w) + w \cdot x_{v_3,v_4}}{|l_G(v_1)| + k^*(v_1)} & \text{otherwise} \end{cases} \tag{4.2}$$

Note that this equation system again uses nearly the same formula as the definition of weighted homomorphisms, except now we consider all possible matches for a successor in $G_{\mathcal{I}}$ and choose the one that gives the maximal membership degree. This equation system has again a unique solution, the proof is analog to the one for Lemma 66. Let $\{x_{v_i,v_j} = v_{v_i,v_j} \mid v_i \in V_{G_{C,\mathcal{T}}}, v_j \in V_{G_{\mathcal{I}}}\}$ be the unique solution of the equation system (4.2). Then we can show that the local maximization in the equation system and the global maximization over all homomorphism given in Definition 67 yield the same result, i.e. $\deg^{\mathcal{I}}(e, C, \mathcal{T}) = v_{d_C,e}$.

Lemma 74. *Let $\mathcal{T}$ be an $\mathcal{EL}$ TBox, C be an $\mathcal{EL}$ concept, $\mathcal{I}$ be a model of $\mathcal{T}$ with finite domain, and $e \in \Delta^{\mathcal{I}}$. Then $\deg^{\mathcal{I}}(e, C, \mathcal{T}) = v_{d_C,e}$, where $v_{d_C,e}$ is the value assigned to the variable $x_{d_C,e}$ in the unique solution of equation system* (4.2)*.*

Proof. To show this, we will construct a pggh between $G_{C,\mathcal{T}}$ and $G_{\mathcal{I}}$. The pggh h is defined as follows:

$$h(d_C) = e,$$

and for all $h(d_C \dots d) = e$ and $(d, r, d') \in E_{G_{C,\mathcal{T}}}$ we have

$$h(d_C \dots d\, r\, d') = \operatorname*{arg\,max}_{(e,r,e') \in E_{G_{\mathcal{I}}}} (1 - w) + w \cdot v_{d',e'}$$

For all paths $d_C \dots d \in \mathrm{dom}_h$ with $h(d_C \dots d) = e$, we have that $h_w(d_C \dots d) = v_{d,e}$, since this assignment satisfies all equations from Definition 65 (and we know that the solution is unique). In particular, this means that $h_w(d_C) = v_{d_C,e}$.

It remains to be shown that for all other pgghs $h' \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)$ the degree $h'_w(d_C)$ can not be larger than $h_w(d_C)$. This is easy to show by contradiction: If there was such a pggh h' with $h'_w(d_C) > v_{d_C,e}$, then the value of $x_{d_C,e}$ in the solution of the equation system (4.2) would be larger than $v_{d_C,e}$ as well.

Together, this implies that $h_w(d_C, e)$ is the maximal homomorphism and thus $\deg^{\mathcal{I}}(e, C, \mathcal{T}) = \sup_{h' \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, e)} h'_w(d_C) = h_w(d_C) = v_{d_C,e}$. □

The equation system (4.2) has only finitely many variables (exactly $|V_{G_{C,\mathcal{T}}}| \cdot |V_{G_{\mathcal{I}}}|$), and thus the membership degree can be computed for finite interpretations $\mathcal{I}$. We can show that it can actually done in polynomial time.

Lemma 75. *Let $\mathcal{T}$ be a TBox, and $\mathcal{I}$ be a model of $\mathcal{T}$ with finite domain. Then $deg^{\mathcal{I}}(d, C, \mathcal{T})$ is computable in time polynomial in the size $|C| + |Delta^{\mathcal{I}}|$.*

Proof. We can reformulate equation system (4.2) as a linear optimization problem:

$$x_{v_1,v_2} \geq \begin{cases} 1 & \text{if } |l_G(v_1)| + k^*(v_1) = 0 \\ \dfrac{|l_G(v_1) \cap l_H(v_2)| + \sum\limits_{(v_1,r,v_3) \in E_{G_{C,\mathcal{T}}}} y_{r,v_3,v_2}}{|l_G(v_1)| + k^*(v_1)} & \text{otherwise} \end{cases}$$

$$y_{r,v_3,v_2} \geq (1 - w) + w \cdot x_{v_3,v_4} \qquad \text{for all } (v_2, r, v_4) \in E_{G_{\mathcal{I}}}$$

Minimizing the objective function $\sum_{v_1 \in V_{G_{C,\mathcal{T}}}, v_2 \in V_{G_{\mathcal{I}}}} x_{v_1,v_2}$, this leads to a unique solution, which corresponds to the solution to equation system (4.2). The argument proceeds analogous to the proof of Lemma 31. The linear optimization problem has polynomial size, and can be solved in PTime, so the Lemma follows. □

In $\mathcal{EL}$, an individual d is an instance of a concept C, if $d^{\mathcal{I}}$ is an element of $C^{\mathcal{I}}$ in all models $\mathcal{I}$ of the KB. In order to define instance membership for threshold concepts, we also need to check that the membership degree is larger than the threshold in all models of the KB. Of course, there may be infinitely many models, so simply computing the membership degrees for each of them is not possible. However, we can show that just checking the canonical model is indeed enough, as for all other models, the membership degree can never be smaller than in the canonical model.

Lemma 76. *Let $\mathcal{I}$ and $\mathcal{J}$ be two interpretations with $(\mathcal{I}, d) \lesssim (\mathcal{J}, e)$, and let C be an $\mathcal{EL}$-concept. Then $deg^{\mathcal{I}}(d, C, \mathcal{T}) \leq deg^{\mathcal{J}}(e, C, \mathcal{T})$.*

Proof. For each pggh $h : \text{Path}(G_{C,\mathcal{T}}, d_C) \to G_{\mathcal{I}}$, we can construct a second pggh $h' : \text{Path}(G_{C,\mathcal{T}}, d_C) \to G_{\mathcal{J}}$ such that $h_w(d_C) \leq h'_w(d_C)$. But then, this implies

$$\sup_{h \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{I}}, d_C, d)} (h_w(d_C)) \leq \sup_{h \in \mathcal{H}(G_{C,\mathcal{T}}, G_{\mathcal{J}}, d_C, e)} (h_w(d_C)),$$

and therefore $\deg^{\mathcal{I}}(d, C, \mathcal{T}) \leq \deg^{\mathcal{J}}(e, C, \mathcal{T})$.

Let $h : \text{Path}(G_{C,\mathcal{T}}, d_C) \to G_{\mathcal{I}}$ be a pggh with $h(d_C) = d$. We inductively construct h' with $\text{dom}(h') = \text{dom}(h)$ such that for all $p \in \text{dom}(h)$ we have $(\mathcal{I}, h(p)) \lesssim (\mathcal{J}, h'(p))$ as follows:

- $h'(d_C) = e$.
- Let $p \in \mathrm{dom}(h)$ with $h(p) = d'$ and $h'(p) = e'$. We know by construction of h' that $(\mathcal{I}, d') \lesssim (\mathcal{J}, e')$. Then for each extension $pr_i d_i \in \mathrm{dom}(h)$ and $h(pr_i d_i) = d'_i$ the simulation implies that there is a $e_i \in \Delta^{\mathcal{J}}$ with $(\mathcal{I}, d'_i) \lesssim (\mathcal{J}, e_i)$; thus we set $h'(pr_i d_i) = e_i$.

We now show that $h_w(d_C) \leq h'_w(d_C)$. To see that, take any path $p = v_0 \dots v_n$ from $\mathrm{dom}(h)$ and compare the equations $h_w(p)$ and $h'_w(p)$. If $h_w(p) = 1$ because $|l_G(v_n)| + k^*(v_n) = 0$, then also $h'_w(p) = 1$. Otherwise, we have that $l_{G_{\mathcal{I}}}(h(p)) \subseteq l_{G_{\mathcal{J}}}(h'(p))$, and thus $|l_{G_{C,\mathcal{T}}}(v_n) \cap l_{G_{\mathcal{I}}}(h(p))| \leq |l_{G_{C,\mathcal{T}}}(v_n) \cap l_{G_{\mathcal{J}}}(h(p))|$; apart from that the equations for $h_w(p)$ and $h'_w(p)$ are exactly the same. This implies that $h_w(p) \leq h'_w(p)$ for all paths $p \in \mathrm{dom}(h)$, in particular $h_w(d_C) \leq h'_w(d_C)$. □

Since $(\mathcal{I}_{\mathcal{K}}, d_a) \lesssim (\mathcal{J}, e)$ for any model $\mathcal{J}$ of $\mathcal{K}$ with $e = a^{\mathcal{J}}$, this implies that $\deg^{I_{\mathcal{K}}}(d_a, C, \mathcal{T}) \leq \deg^{\mathcal{J}}(e, C, \mathcal{T})$. In particular, this also leads to the following result.

Corollary 77. *Let $\mathcal{K}$ be an $\mathcal{EL}$ KB, C be an $\mathcal{EL}$ concept, a an individual occurring in $\mathcal{K}$, and $t \in [0, 1]$. Then $\mathcal{K} \models C_{\geq t}(a)$ iff $\deg^{\mathcal{I}_{\mathcal{K}}}(d_a, C, \mathcal{T}) \geq t$. This is decidable in time polynomial in the size of $\mathcal{K}$ and C.*

This gives a better complexity than relaxed instance query answering w.r.t. the CSM $\sim_c$. The main reason for this is that graded membership functions are already asymmetric and can be applied directly, while $\sim_c$ is symmetric, and in order to compute the relaxed instances one needs to check all subsuming concepts of the individuals to find the maximal similarity value.

Finally, we conjecture that one could easily extend the graded membership function deg with a primitive measure and a weighting function in a similar way this is done for $\sim_c$. This would make the graded membership function even more useful for answering relaxed instances.

In conclusion, graded membership functions give a nice, and less complex, alternative to similarity measures for relaxing instance queries. Since graded membership functions don't need to check all subsuming concepts of each individual, but can be applied directly, their application in relaxed instance queries might be more intuitive. One the other hand, similarity measures have many other uses as well, and seem like a more natural measure in general. A practical evaluation would be useful to compare these to approaches more in-depth.

Chapter 5

Reasoning with Prototypes using Weighted Tree Automata

In this chapter, we present an approach that allows to make prototypical definitions. Prototypical definitions, in essence, allow to define a concept by comparing elements to a prototype or prototypical expression. Only those elements that are close enough to this prototype are still considered to belong to the concept.

In order to be used within a formal knowledge representation language with automated reasoning capabilities, such prototypes need to be equipped with a formal semantics. To obtain such a semantics, we use the ideas underlying Gärdenfors' conceptual spaces [Gär00], where categories are explained in terms of convex regions, which are defined using the distance from a focal point. To obtain a concrete representation language, we need to define what focal points are and how to define the distance of an individual to such a focal point. Instead of employing prototypical individuals or concepts as focal points, we take a more abstract approach based on automata, which is inspired by the automata-approach for reasoning in DLs (see Section 3.2 in [Baa09] for an introduction). Basically, in this approach, a given concept C and a TBox $\mathcal{T}$ are translated into a tree automaton $\mathcal{A}_{C,\mathcal{T}}$ that accepts all the tree-shaped models of $\mathcal{T}$ whose root belongs to C. Testing satisfiability of C w.r.t. $\mathcal{T}$ then boils down to the emptiness test for $\mathcal{A}_{C,\mathcal{T}}$, i.e., checking whether there is a tree accepted by $\mathcal{A}_{C,\mathcal{T}}$.

Instead of using a classical automaton that returns 1 (accepted) or 0 (not accepted) for an input tree, we propose to use a weighted automaton [DKV09]. Intuitively, this automaton receives as input a tree-shaped interpretation and returns as output a non-negative integer, which we interpret as the distance, or dissimilarity, of the individual at the root of the tree to the prototype (focal point) described by the automaton. This approach can be applied to non-tree-shaped models by the usual unraveling operation. In order to integrate such prototypes into a Description Logic, we propose to use thresholds to derive concepts from prototypes. More precisely, the threshold concept $P_{\sim n}(\mathcal{A})$ for $\sim \in \{<, \leq, >, \geq\}$ is interpreted as the set of all elements with a distance $\sim n$ according to the weighted automaton $\mathcal{A}$. The concepts obtained this way can then be used like atomic concepts within a DL.

It might appear to be more intuitive to use concepts or individuals rather than automata to describe prototypes. However, in these alternative settings, one then needs to give formal definitions of the distance between two individuals or between

an individual and a concept, whereas in our approach this comes for free by the definition of the semantics of weighted automata. We show that these alternative settings can actually be seen as instances of our weighted automata approach.

5.1 Prototypical Reasoning

In general, a prototype can be seen as some kind of structure that can be compared to elements of an interpretation, distinguishing elements that are closer (more similar or related) to the prototype from elements that are further away (dissimilar or different). More specifically, one may view a prototype as a function that assigns to each element a distance value from the focal point, where small distances correspond to similar elements, and large distances to dissimilar elements.

Definition 78. A *prototype distance function* (pdf) d is a function that assigns to each element e of an interpretation $\mathcal{I}$ a distance value $d_{\mathcal{I}}(e) \in \mathbb{N}$. The constructor $P_{\sim n}(d)$ for a threshold $n \in \mathbb{N}$ is interpreted in an interpretation $\mathcal{I}$ as the set of all elements $e \in \Delta^{\mathcal{I}}$ such that $d_{\mathcal{I}}(e) \sim n$, for $\sim \in \{<, \leq, >, \geq\}$. If D is a set of pdfs, we use $\mathcal{ALCP}(D)$ to denote the Description Logic $\mathcal{ALC}$ extended with the prototype constructor for pdfs from D. ♢

Notice that, instead of treating the focal point and the distance as separate entities, prototype distances functions combine both notions together, and directly assign a distance value to each element; the focal point itself is implicit in the pdf; if one element has distance 0 w.r.t. some pdf, then this point might be considered a focal point (though not necessarily the unique focal point). Otherwise, the focal point is just an abstract entity and cannot be represented directly.

As explained before, we will use weighted alternating tree automata to define pdfs. These automata can express distance functions between trees (in our case, tree-shaped pointed interpretations) to the non-negative integers $\mathbb{N}$. By unraveling pointed interpretations we can extend this to a function from arbitrary pointed interpretations to $\mathbb{N}$, i.e., a prototype distance function.

5.1.1 Using Weighted Tree Automata for Prototype Distance Functions

Before introducing the weighted automata model used to represent pdfs, we will first define some basic notions and an unweighted automata model. This will make it easier to understand the more difficult weighted case, and will also be required to show decidability.

A *tree domain* is a prefix-closed, non-empty set $D \subseteq \mathbb{N}^*$, i.e., for every $ui \in D$ with $u \in \mathbb{N}^*$ and $i \in \mathbb{N}$ we also have $u \in D$. The elements of D are called *nodes*, the node ε is the *root* of D and, for every $u \in D$, the nodes $ui \in D$ are called *children* of u. A node is called a *leaf* if it has no children. A *path* π in D is a subset $\pi \subseteq D$ such that $\varepsilon \in \pi$ and for every $u \in \pi$, u is either a leaf or there is a unique $i \in \mathbb{N}$

with $ui \in \pi$. Given an alphabet Σ, a Σ-labeled tree is a pair (dom_T, T) consisting of a tree domain dom_T and a labeling function $T : \text{dom}_T \to \Sigma$. Instead of the pair (dom_T, T) we often use only T to denote a labeled tree. With $\text{Tree}(\Sigma)$ we denote the set of all Σ-labeled trees. Note that this definition allows the existence of infinite trees.

The automata type we introduce now is based mainly on the alternating tree automata defined by Wilke [Wil01], which are working on $\mathcal{P}(\Sigma)$-Trees, which are labeled with the power set of some finite alphabet Σ. Given such a Σ and a set of states Q, a transition condition $\text{TC}(\Sigma, Q)$ is one of the following: true; false; σ or $\neg\sigma$ for $\sigma \in \Sigma$; $q_1 \wedge q_2$ or $q_1 \vee q_2$ for $q_1, q_2 \in Q$; or $\Box q$ or $\Diamond q$ for $q \in Q$. These transition conditions allow to accept or reject the current path, check for existence or absence of a symbol σ at the current node, and allow to split the automaton in multiple copies for the current node or all successors, for which all copies or one of them must be accepting, respectively.

Definition 79. An *alternating parity tree automaton (apta)* $\mathcal{A}$ working on $\mathcal{P}(\Sigma)$-trees is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, \Omega)$, where

1. Σ is a finite alphabet;
2. Q is a finite set of states and $q_0 \in Q$ is the initial state;
3. the transition function $\delta : Q \to \text{TC}(\Sigma, Q)$ assigns to each state a transition condition; and
4. $\Omega : Q \to \mathbb{N}$ is the priority function that specifies the parity acceptance condition. ♢

Given a $\mathcal{P}(\Sigma)$-labeled tree T, a *run* is a $(\text{dom}_T \times Q)$-labeled tree R with $\varepsilon \in \text{dom}_R$, $R(\varepsilon) = (\varepsilon, q_0)$, and that respects all the transition conditions, i.e., for all $u \in \text{dom}_R$ with $R(u) = (v, q)$ we have:

- $\delta(q) \neq \mathsf{false}$
- if $\delta(q) = \sigma$, then $\sigma \in T(v)$; and if $\delta(q) = \neg\sigma$, then $\sigma \notin T(v)$;
- if $\delta(q) = q_1 \wedge q_2$, then there exists $i_1, i_2 \in \mathbb{N}$ such that $R(ui_1) = (v, q_1)$ and $R(ui_2) = (v, q_2)$;
- if $\delta(q) = q_1 \vee q_2$, then there exists $i \in \mathbb{N}$ such that $R(ui) = (v, q_1)$ or $R(ui) = (v, q_2)$;
- if $\delta(q) = \Diamond q'$, then there exists $i, j \in \mathbb{N}$ with $R(ui) = (vj, q')$; and
- if $\delta(q) = \Box q'$, then for every $j \in \mathbb{N}$ with $vj \in \text{dom}_T$ there exists $i \in \mathbb{N}$ with $R(ui) = (vj, q')$.

A run is *accepting*, if every infinite path π in R satisfies the *parity acceptance condition* specified by Ω, i.e., the largest priority $\Omega(u)$ occurring infinitely often along the path $u \in \pi$ is even. The language accepted by an apta $\mathcal{A}$, $L(\mathcal{A})$, is the set of all $\mathcal{P}(\Sigma)$-trees T for which there exists an accepting run R of $\mathcal{A}$ on T.

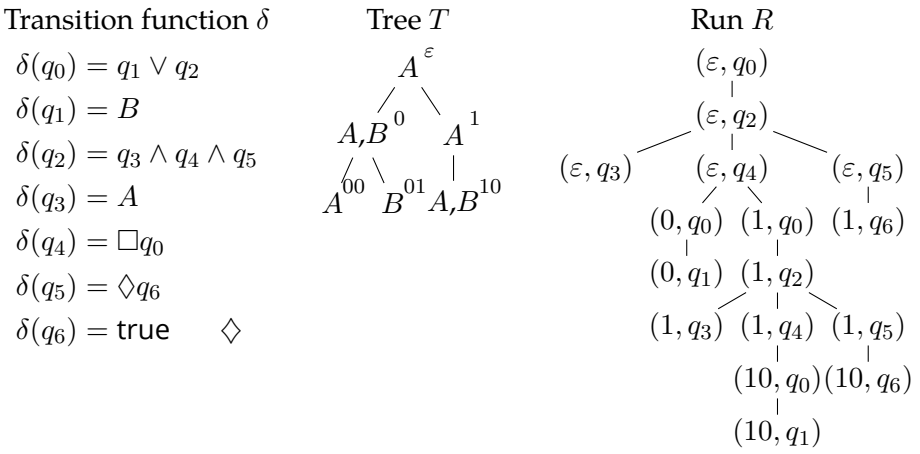

Figure 5.1: Transition function δ, $\mathcal{P}(\Sigma)$-tree T, and accepting run R of $\mathcal{A}_{\text{ex}}$ on T.

Note that apta are bisimulation invariant: If there exist a bisimulation between two trees $T_1 \cong T_2$, then we have $T_1 \in L(\mathcal{A})$ iff $T_2 \in L(\mathcal{A})$ [Wil01]. The emptiness problem for apta, i.e., deciding whether $L(\mathcal{A}) = \emptyset$, is in ExpTime; the complement automaton which accepts the complement language $\text{Tree}(\Sigma) \setminus L(\mathcal{A})$ can be constructed in linear time [Wil01]. Note that, instead of only the transition conditions mentioned above, one could allow for complex transition conditions like $\Box(q_1 \wedge \neg B) \vee q_2$. Automata with complex transition conditions can be transformed into equivalent automata using only simple transition conditions by introducing new states for each subformula of the transition condition [Wil01].

Example 80. Let $\mathcal{A}_{\text{ex}} = (\Sigma, Q, q_0, \delta, \Omega)$ be an apta with alphabet $\Sigma = \{A, B\}$, with states $Q = \{q_0, \ldots, q_6\}$, initial state q_0, transition function δ as given in Figure 5.1, and priority function Ω with $\Omega(q) = 1$ for all $q \in Q$.

This automaton accepts only trees where the root label contains B (state q_1), or it is labeled with A and all of its successors (at least one) are again of this form. Since the parity function prohibits infinite paths in the run (though not in the input tree), $\mathcal{A}_{\text{ex}}$ accepts exactly those trees where all paths start with nodes labeled with A until eventually a node with a label containing B is encountered. Figure 5.1 shows such a tree T and an accepting run R of $\mathcal{A}_{\text{ex}}$ on T.

Now, we can introduce a weighted version of this automata model in order to describe prototype distance functions. The main idea behind the use of weighted automata for pdfs is that the automaton can punish a pointed interpretation by increasing the distance value whenever a feature described by the automaton is not as expected. For example, the automaton can require the current node to be labeled with the concept name Cup, and increase the distance by some number if this is not the case. Using this idea, the most natural interpretation of the transition conditions in the weighted setting is as follows: $q_1 \wedge q_2$ will compute the sum of the distances for q_1 and q_2 (both features should be present), $\vee$ will be interpreted as the minimum (one of the feature should be present), $\Diamond$ will also be interpreted

as the minimum (one of the successors should have the feature, i.e., we choose the best one); and $\Box$ will be interpreted as the maximum (all successors should have the feature; if not, we take the distance of the worst)[1].

A weighted alternating parity tree automaton is nearly the same as in the unweighted case, with the exception that the transition function may also contain non-negative integers. Given an alphabet Σ and a set of states Q, a weighted transition condition $\mathrm{wTC}(\Sigma, Q)$ is one of the following: $n \in \mathbb{N}$; σ or $\neg\sigma$ for $\sigma \in \Sigma$; $q_1 \wedge q_2$ or $q_1 \vee q_2$ for $q_1, q_2 \in Q$; or $\Box q$ or $\Diamond q$ for $q \in Q$.

Definition 81. A *weighted alternating parity tree automaton (wapta)* $\mathcal{A}$ working on $\mathcal{P}(\Sigma)$-trees is a tuple $\mathcal{A} = (\Sigma, Q, q_0, \delta, \Omega)$, where

1. Σ is a finite alphabet;
2. Q is a finite set of states and $q_0 \in Q$ is the initial state;
3. $\delta : Q \to \mathrm{wTC}(\Sigma, Q)$ is the transition function; and
4. $\Omega : Q \to \mathbb{N}$ is the priority function. $\Diamond$

Runs are defined as in the unweighted case, however nodes labeled with a state for which the transition function yields a number do not need to satisfy any additional conditions, they can be leafs in the run. In order to interpret the $\Box$-operator as the maximum, we need to further split runs into their $\Box$-fixations; these $\Box$-fixations basically chooses for a $\Box$-operator a single successor node instead of all of them. Formally, given a run R, a $\Box$-fixation is a tree R' with $\mathrm{dom}_{R'} \subseteq \mathrm{dom}_R$, which can be obtained from R as follows: starting with the root, we keep all the successors for nodes where the transition function does not yield a box; for nodes u labeled with a state q for which the transition function is of the form $\delta(q) = \Box q'$, the $\Box$-fixation R' keeps at most one successor $ui \in \mathrm{dom}_R$. All nodes $u \in \mathrm{dom}_{R'}$ have the same label $R'(u) = R(u)$ as in R.

Then, we can define the behavior of the automaton as a function $\|\mathcal{A}\|$ from $\mathcal{P}(\Sigma))$-trees to $\mathbb{N} \cup \{\infty\}$. The weight of a $\Box$-fixation R' of a run R is defined as

$$\mathrm{weight}_{\mathcal{A}}(R') = \sum_{u \in \mathrm{dom}_{R'}, R'(u)=(d,q), \delta(q,T(u))=n \in \mathbb{N}} n,$$

i.e., the weight of R' is simply the sum of all numbers to which the nodes in R' are mapped. Note that this (possibly infinite) sum is well-defined: If infinitely many values $n > 0$ occur in R', the weight of R' is ∞; otherwise it is the finite sum of all weights in R'. The weight of a run R on T is

$$\mathrm{weight}_{\mathcal{A}}(R) = \sup_{R' \text{ } \Box\text{-fixation of } R} \mathrm{weight}_{\mathcal{A}}(R'),$$

[1] Another reasonable interpretation of $\Box$ would be to add up the distances of all successors instead of taking the maximum. However, it is easy to see that this interpretation would mean that the automaton model is no longer bisimulation-invariant: Simply duplicating a successor with non-zero weight would create a bisimilar interpretation, but would increase the total weight. Basically, this semantics would allow the weighted automata to count. Bisimulation-invariance is helpful later to show that reasoning with prototypes is decidable.

and the behavior of the wapta $\mathcal{A}$ on a tree T is

$$\|\mathcal{A}\|(T) = \min_{R \text{ accepting run on } T} \text{weight}_{\mathcal{A}}(R).$$

Similar to the unweighted case, wapta are equivalence invariant, i.e., if $T_1 \cong T_2$, then we have $\|\mathcal{A}\|(T_1) = \|\mathcal{A}\|(T_2)$. The main reason for this is that $\Diamond$ and $\Box$ are interpreted as idempotent operators, min and max respectively. This property will become important later.

Notice that if one wants to let the automata run on tree-shaped interpretations, we have a slight disparity: trees as introduced above do not have labeled edges, while interpretations do. To overcome this, we push role names into the labels of the children [Hla07, pp. 59–62]. Thus, in the following, the alphabet Σ always consists of all concept and role names of C and $C_{\mathcal{T}}$, i.e., $\Sigma = (\text{sig}(C) \cup \text{sig}(C_{\mathcal{T}})) \cap (\mathsf{N_C} \cup \mathsf{N_R})$.

As said before, such wapta can be used to define prototype distance functions, in order to measure the distance from pointed interpretations to some prototype. In the following section we give two examples of such constructions, which show that wapta are indeed usable for this purpose.

Constructions of prototype automata

In the following we will give a concrete example of how a weighted automaton can be constructed from an $\mathcal{ALC}$-concept.

Example 82 (Constructing a wapta from an $\mathcal{ALC}$ concept). Recall from the beginning of this chapter that a prototypical cup is a small container with handles, which can hold liquids and is made of plastic or porcelain[2]. We can express this as an $\mathcal{ALC}$-concept:

$$\begin{aligned}&\mathsf{Container} \sqcap \mathsf{Small} \sqcap \forall \mathsf{material}.(\mathsf{Glass} \sqcup \mathsf{Porcelain})\\ &\quad \sqcap \exists \mathsf{hasPart}.\mathsf{Handle} \sqcap \forall \mathsf{holds}.\mathsf{Liquid}\end{aligned}$$

This concept can directly be translated into a complex transition condition for an alternating tree automaton:

$$\begin{aligned}&\mathsf{Container} \wedge \mathsf{Small} \wedge \Box(\neg\mathsf{material} \vee (\mathsf{Glass} \vee \mathsf{Porcelain}))\\ &\quad \wedge \Diamond(\mathsf{hasPart} \wedge \mathsf{Handle}) \wedge \Box(\neg\mathsf{holds} \vee \mathsf{Liquid})\end{aligned}$$

Finally, we can add weights in order to punish those features of the transition condition that an element might be missing:

$$\begin{aligned}&(\mathsf{Container} \vee 3) \wedge (\mathsf{Small} \vee 1) \wedge \Box(\neg\mathsf{material} \vee ((\mathsf{Glass} \vee 1) \vee (\mathsf{Porcelain} \vee 1)))\\ &\quad \wedge (\Diamond(\mathsf{hasPart} \wedge \mathsf{Handle}) \vee 1) \wedge \Box(\neg\mathsf{holds} \vee (\mathsf{Liquid} \vee 2))\end{aligned}$$

[2] We previously mentioned a few more properties that prototypical cups should satisfy. In order to keep this example small, we will ignore those additional properties.

The meaning of this weighted transition condition is as follows: If an element is not a container, it will be punished with a weight of 3 since there cannot be a run that uses the option Container at the root. Otherwise, there is such a run, which does not contribute a weight. Accordingly, the absence of the feature small is punished with weight 1. If the cup does not have a successor that is labeled with both hasPart and Handle, then a weight of 1 is added. Finally, if there is a material-successor that is not labeled with Glass or Porcelain, then this is punished with weight 1. If the cup does not have any material-successors, or all of them are glass or porcelain, no weight is added. Similarly for holding only liquids. ♢

Such a construction works for arbitrary $\mathcal{ALC}$ concepts. In essence, after translating the concept into a transition condition, choosing the weights appropriately allows us to punish the absence of different features by different values.

For universal restrictions, the weights of several offending successors are not added up, but rather the supremum is taken. As a consequence, equivalent concepts may not yield equivalent wapta using this approach. For example, $\forall r.(A \sqcap B) \equiv \forall r.A \sqcap \forall r.B$, but the corresponding transition conditions after adding weights may lead to different results. However, one can argue that, when viewed as prototype descriptions, these two concept descriptions do actually encode different intentions. While in the first case we want to make sure that all r-successors are instance of A and B simultaneously (and pick the weight of the worst offender if there is one), in the second case we want to enforce both features separately, and punish for the worst offenders separately.

Besides translating $\mathcal{ALC}$ concepts into wapta, one can also create prototypes from finite pointed interpretations, i.e., prototypical elements. For this, one introduces a state for each element of the interpretation, and as transition condition for each state one simply conjoins all the concept names the element is instance of, negations of all concept names it is not instance of, and a $\Diamond$-transition for each successor in the interpretation, labeled with both the role name and the state of the successor-element. If one also introduces a $\Box$-transition with a disjunction of all possible successor-states and adds positive weights as in the above example, this weighted automaton will only give distance 0 to pointed interpretations that are bisimilar to the prototypical interpretation, and otherwise punish each difference by increasing the distance accordingly.

5.2 Reasoning with Prototypes

In this section, we will show how to reason in $\mathcal{ALCP}$. Note that in DLs with negation, subsumption can be reduced to concept satisfiability. Indeed, we have $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable in $\mathcal{T}$. Therefore, an algorithm that decides concept satisfiability can also be used to decide subsumption.

5.2.1 Deciding Concept Satisfiability using Alternating Parity Tree Automata

Before giving an algorithm to decide concept satisfiability in $\mathcal{ALCP}$, we will show that alternating parity tree automata can be used for decide the satisfiability of $\mathcal{ALC}$ concepts w.r.t. to an $\mathcal{ALC}$ TBox without prototypes. This result is a simple adaptation of the approach in [SV01] to $\mathcal{ALC}$.

It is well-known [BRV01] that $\mathcal{ALC}$ has the tree model property, i.e., every satisfiable $\mathcal{ALC}$-concept C has a tree-shaped model in which the root of the tree is an instance of C. Thus, to decide concept satisfiability, it is enough to consider tree-shaped interpretations. The automaton approach to decide concept satisfiability requires the concept and the TBox to be in negation normal form.

Definition 83 (negation normal form). An $\mathcal{ALC}$-concept C is in negation normal form if negation occurs only directly in front of concept names. A concept C can be transformed in linear time into an equivalent concept in negation normal form [Baa09] by applying the following rules to subconcepts of C until no more rule is applicable:

$$\begin{aligned}
\neg\top &\rightsquigarrow \bot \\
\neg\bot &\rightsquigarrow \top \\
\neg\neg C &\rightsquigarrow C \\
\neg(C \sqcap D) &\rightsquigarrow \neg C \sqcup \neg D \\
\neg(C \sqcup D) &\rightsquigarrow \neg C \sqcap \neg D \\
\neg\exists r.C &\rightsquigarrow \forall r.\neg C \\
\neg\forall r.C &\rightsquigarrow \exists r.\neg C
\end{aligned}$$

We write $\mathrm{nnf}(C)$ to denote the concept C transformed into negation normal form, and say that an TBox is in negation normal form if all concepts occurring in it are in negation normal form. ♢

We can transform a TBox $\mathcal{T}$ into a single, equivalent concept $C_\mathcal{T} = \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} \neg C \sqcup D$; then an interpretation satisfies $\mathcal{T}$ iff it satisfies the GCI $\top \sqsubseteq C_\mathcal{T}$. We now construct an automaton that decides concept satisfiability in $\mathcal{ALC}$. Given a TBox $\mathcal{T}$ and a concept C, the idea underlying this approach is that the constructed automaton will accept exactly the tree models of $\mathcal{T}$ for which the root is an instance of C. The automaton contains a state for each subconcept of $\mathrm{nnf}(C)$ and $\mathrm{nnf}(C_\mathcal{T})$, which are used to simulate the semantics of $\mathcal{ALC}$. Cycles in $\mathcal{T}$ can enforce infinite tree models; infinite paths are always accepting if they satisfy the axioms in $\mathcal{T}$.

Definition 84. Let $\mathcal{T}$ be an $\mathcal{ALC}$-TBox of the form $\{\top \sqsubseteq C_\mathcal{T}\}$ and C an $\mathcal{ALC}$-concept with both C and $C_\mathcal{T}$ in negation normal form. We define the automaton $\mathcal{A}_{C,\mathcal{T}} = (\Sigma, Q, q_0, \delta, \Omega)$ as follows:

- $\Sigma = \mathrm{sig}(C) \cup \mathrm{sig}(C_\mathcal{T})$ and $\Omega(q) = 0$ for all $q \in Q$,

- $Q = \{q_D \mid D \in \mathsf{sub}(C) \cup \mathsf{sub}(C_\mathcal{T})\} \cup \{q_r, q_{\neg r} \mid r \in \mathsf{sig}(C) \cup \mathsf{sig}(C_\mathcal{T})\} \cup \{q_0, q_\mathcal{T}\}$,
- the transition function δ is defined as follows (where $\sigma \in \mathsf{N_C} \cup \mathsf{N_R}$):

$$\begin{aligned} \delta(q_0) &= q_C \wedge q_\mathcal{T} & \delta(q_\mathcal{T}) &= q_{C_\mathcal{T}} \wedge \Box q_\mathcal{T} \\ \delta(q_\sigma) &= \sigma & \delta(q_{\neg\sigma}) &= \neg\sigma \\ \delta(q_{C_1 \sqcap C_2}) &= q_{C_1} \wedge q_{C_2} & \delta(q_{C_1 \sqcup C_2}) &= q_{C_1} \vee q_{C_2} \\ \delta(q_{\exists r.C}) &= \Diamond(q_r \wedge q_C) & \delta(q_{\forall r.C}) &= \Box(q_{\neg r} \vee q_C) \end{aligned} \qquad \Diamond$$

The proof of the following proposition is similar to the one in [Hla07, pp. 59–62]. It relies on the fact that any tree with accepting run can be interpreted as a model of the TBox with the root being an instance of C, and any model of the TBox can be unraveled into a tree for which an accepting run can be inductively constructed.

Proposition 85. *Given an $\mathcal{ALC}$-TBox $\mathcal{T}$ and an $\mathcal{ALC}$-concept C, the concept C is satisfiable w.r.t. $\mathcal{T}$ iff $L(\mathcal{A}_{C,\mathcal{T}}) \neq \emptyset$.*

Proof. We have to show both directions.

"$\Leftarrow$" Let $L(\mathcal{A}_{C,\mathcal{T}}) \neq \emptyset$, i.e., there is a tree T and an accepting run R of $\mathcal{A}_{C,\mathcal{T}}$ on T. We define the interpretation $\mathcal{I}_T$ as follows:

$$\begin{aligned} \Delta^{\mathcal{I}_T} &= \mathrm{dom}_T, \\ A^{\mathcal{I}_T} &= \{u \in \mathrm{dom}_T \mid A \in T(u)\}, \\ r^{\mathcal{I}_T} &= \{(u, ui) \in \mathrm{dom}_T \times \mathrm{dom}_T \mid r \in T(ui)\}. \end{aligned}$$

Claim 86. *For any $u \in \Delta^{\mathcal{I}_T}$ and $E \in sub(C) \cup sub(C_\mathcal{T})$, if (u, q_E) occurs as a label of a node in R then $u \in E^{\mathcal{I}_T}$.*

This claim can be shown by induction on the structure of E. For example, if $E = \exists r.F$, then by definition of $\mathcal{A}_{C,\mathcal{T}}$ we have $\delta(q_E) = \Diamond(q_r \wedge q_F)$. If $R(v) = (u, q_E)$ occurs in R at some node v, then there are $i, j, k, l \in \mathbb{N}$ with $R(vi) = (ul, (q_r \wedge q_F))$, $R(vij) = (ul, q_r)$, and $R(vik) = (ul, q_F)$. By induction hypothesis we then have $ul \in F^{\mathcal{I}_T}$ and $(u, ul) \in r^{\mathcal{I}_T}$, and hence $u \in (\exists r.F)^{\mathcal{I}_T}$. The other cases are similar.

Since $\delta(q_0) = q_C \wedge q_\mathcal{T}$ and $R(\varepsilon) = q_0$, the claim yields $\varepsilon \in C^{\mathcal{I}_T}$. It remains to show that $\mathcal{I}_T$ is a model of $\mathcal{T}$. However, since $(\varepsilon, q_\mathcal{T})$ occurs in R and $\delta(q_\mathcal{T}) = q_{C_\mathcal{T}} \wedge \Box q_\mathcal{T}$, the claim yields that for every node $u \in \Delta^{\mathcal{I}_T}$ we have $u \in C_\mathcal{T}^{\mathcal{I}_T}$ and thus all GCIs are satisfied by $\mathcal{I}_T$.

"$\Rightarrow$" Assume that C is satisfiable w.r.t. $\mathcal{T}$, i.e., there exists a finitely branching interpretation $\mathcal{I}$ and an element $d_0 \in \Delta^\mathcal{I}$ such that $\mathcal{I}$ is a model of $\mathcal{T}$ and

$d_0 \in C^{\mathcal{I}}$. We will inductively construct a Σ-tree T and a mapping $L : T \to \Delta^{\mathcal{I}}$ as follows:

$$\begin{aligned} \varepsilon &\in \mathrm{dom}_T, \\ L(\varepsilon) &= d_0, \\ T(\varepsilon) &= \{A \in \mathsf{N_C} \mid d_0 \in A^{\mathcal{I}}\} \end{aligned}$$

Let $u \in \mathrm{dom}_T$, with $L(u) = d$. Let $S = \{(r, d') \in \mathsf{N_R} \times \Delta^{\mathcal{I}} \mid (d, d') \in r^{\mathcal{I}}\}$ be the set of successors, i.e., elements that are connected to d via a role r, and let $(r_1, d_1), (r_2, d_2), \ldots, (r_n, d_n)$ be an arbitrary enumeration of those successors. For $1 \leq i \leq n$ we define:

$$\begin{aligned} ui &\in \mathrm{dom}_T, \\ L(ui) &= d_i, \\ T(ui) &= \{A \in \mathsf{N_C} \mid d_i \in A^{\mathcal{I}}\} \cup \{r_i\} \end{aligned}$$

For this tree, one can inductively construct a run R starting from $R(\varepsilon) = (\varepsilon, q_0)$, and just following the transition function δ, such that whenever we have a node $v \in R$ with $R(v) = (u, q_E)$, then $L(u) \in E^{\mathcal{I}}$. In fact, it is easy to show that this is an accepting run of $\mathcal{A}_{C,\mathcal{T}}$ on T. Consequently, $T \in L(\mathcal{A}_{C,\mathcal{T}})$ and thus $L(\mathcal{A}_{C,\mathcal{T}}) \neq \emptyset$. □

Since the automaton $\mathcal{A}_{C,\mathcal{T}}$ is polynomial in the size of the TBox $\mathcal{T}$ and the concept C, this approach yields an ExpTime algorithm for concept satisfiability, which is worst-case optimal [Sch91].

To reason in $\mathcal{ALC}$ with prototypes, we now have to achieve two more things: First, for each prototype constructor $P_{\leq n}(\mathcal{A})$, we have to transform the wapta $\mathcal{A}$ into an unweighted automaton that accept exactly those trees T for which $\|\mathcal{A}\|(T) \leq n$. Then we need to combine the alternating tree automaton $\mathcal{A}_{C,\mathcal{T}}$ defined just now with the unweighted automata for the prototypes such that the resulting automaton accepts exactly the tree models of C w.r.t. $\mathcal{T}$. An emptiness test can then be used to decide (un-)satisfiability for $\mathcal{ALCP}$.

5.2.2 Cut-point Automata

Given a weighted alternating parity tree automaton $\mathcal{A}$ and a threshold value $n \in \mathbb{N}$, we want to construct an unweighted apta $\mathcal{A}_{\leq n}$ that accepts exactly the cut-point language, i.e. $L(\mathcal{A}_{\leq n}) = \{T \in \mathrm{Tree}(\mathcal{P}(\Sigma)) \mid \|\mathcal{A}\|(T) \leq n\}$. In this cut-point automaton, each state needs to keep track of both the weight and the current state of the corresponding weighted automaton. However, instead of tracking the weight that has already been accumulated, it needs to track the weight that the automaton is still allowed to spend. The reason for this is that, for trees, each state can have multiple successors, and thus we have to budget the allowed weight for each of the successors so that the sum is not greater than the threshold.

Definition 87. Given a wapta $\mathcal{A} = (\Sigma, Q, q_0, \delta, \Omega)$, the cut-point automaton $\mathcal{A}_{\leq n} = (\Sigma, Q', q_0', \delta', \Omega')$ for the threshold $n \in \mathbb{N}$ is an apta defined as follows:

- $Q' = \{(q, i) \in Q \times \mathbb{N} \mid i \leq n\} \cup \{q_0'\}$,
- $\Omega'((q, i)) = \Omega(q)$, and
- the transition function δ' with

$$\begin{aligned}
\delta'(q_0') &= \bigvee_{0 \leq i \leq n} (q_0, i) \\
\delta'((q, i)) &= \delta(q) \text{ if } \delta(q) = \sigma, \neg\sigma \\
\delta'((q, i)) &= \mathsf{true} \text{ if } \delta(q) = j \leq i \\
\delta'((q, i)) &= \mathsf{false} \text{ if } \delta(q) = j > i \\
\delta'((q, i)) &= \Diamond(q', i) \text{ if } \delta(q) = \Diamond q' \\
\delta'((q, i)) &= \Box(q', i) \text{ if } \delta(q) = \Box q' \\
\delta'((q, i)) &= (q_1, i) \vee (q_2, i) \text{ if } \delta(q) = q_1 \vee q_2 \\
\delta'((q, i)) &= \bigvee_{0 \leq j \leq i} (q_1, j) \wedge (q_2, i - j) \text{ if } \delta(q) = q_1 \wedge q_2.
\end{aligned}$$ ♢

This automaton $\mathcal{A}_{\leq n}$ accepts exactly the cut-point language of $\mathcal{A}$ w.r.t. n.

Proposition 88. *Let $\mathcal{A}$ be a wapta and $\mathcal{A}_{\leq n}$ the cut-point automaton derived from $\mathcal{A}$ using the threshold $n \in \mathbb{N}$. Then $\mathcal{A}_{\leq n}$ accepts the cut-point language, i.e., $L(\mathcal{A}_{\leq n}) = \{T \in \mathit{Tree}(\mathcal{P}(\Sigma)) \mid \|\mathcal{A}\|(T) \leq n\}$.*

Proof. We have to prove both directions. Given a tree $T \in L(\mathcal{A}_{\leq n})$, and an accepting run R of $\mathcal{A}_{\leq n}$ on T, we can construct a run R' of $\mathcal{A}$ on T by removing all weights from the labels of R. By induction on the weight i, we can then show that whenever we have $R(u) = (v, (q, i))$ for some node $u \in \mathrm{dom}_R$, all $\Box$-fixations of R' starting from u will have a weight at most i. This follows from the claim that the sum of the weights of the children of a node v is never larger than the weight of v itself for all $\Box$-fixations. Since the first successor of the root of R is labeled with $R(0) = (\varepsilon, (q_0, n))$, this means that $\mathrm{weight}_{\mathcal{A}}(R') \leq n$.

Similarly, if we have a tree $T \in \mathrm{Tree}(\mathcal{P}(\Sigma))$ with $\|\mathcal{A}\|(T) \leq n$, and a run R of $\mathcal{A}$ on T with $\mathrm{weight}_{\mathcal{A}}(R) \leq n$, we can construct a run R' of $\mathcal{A}_{\leq n}$ on T by setting $R'(u) = (v, (q, i))$ where $R(u) = (v, q)$ and i is the weight assigned by $\mathcal{A}$ to the subtree of R rooted at u, starting in state q. It can then be shown that the run R' obtained this way is an accepting run of $\mathcal{A}_{\leq n}$ on T, i.e., it satisfies all transition conditions and all infinite paths are accepting. □

The cut-point automaton $\mathcal{A}_{\leq n}$ has $O(n \cdot q)$ states, where q is the number of states of the weighted automaton $\mathcal{A}$. Thus, if n is encoded in unary, this construction is polynomial, otherwise it is exponential.

5.2.3 Reasoning in $\mathcal{ALC}$ with Prototypes

We want to combine the cut-point automata constructed from prototype concepts with the automaton from Definition 84 in order to decide the concept satisfiability problem in $\mathcal{ALCP}$(wapta); more specifically, we want to construct an automaton $\mathcal{A}$ that accepts all those (tree-shaped) pointed interpretations that are instances of an $\mathcal{ALCP}$(wapta)-concept w.r.t. an $\mathcal{ALCP}$(wapta)-TBox.

For $\mathcal{ALCP}$(wapta)-concepts, one can again define a normal form. This extends the negation normal form from Definition 83 by requiring that prototype constructors occur only in the form $P_{\leq n}(\mathcal{A})$, possibly negated. For example, one can transform $P_{\geq n}(\mathcal{A})$ for $n \geq 1$ into negation normal form by replacing it with $\neg P_{\leq n-1}(\mathcal{A})$; $P_{\geq 0}(\mathcal{A})$ can be replaced by $\top$. The set of subconcepts now contains such prototype concepts as well.

In case a prototype constructor occurs negated, the complement automaton $\bar{\mathcal{A}}$ for a cut-point automaton $\mathcal{A}$ can be constructed in linear time, by exchanging true and false, $\vee$ and $\wedge$, $\Box$ and $\Diamond$, and σ and $\neg\sigma$ for all $\sigma \in \Sigma$ in all transition conditions, as well as adding one to the priority of all states [Wil01].

Then, we can define the apta $\mathcal{A}_{\mathcal{P},C,\mathcal{T}}$ as follows:

Definition 89. Let $\mathcal{T}$ be an $\mathcal{ALCP}$(wapta)-TBox of the form $\{\top \sqsubseteq C_{\mathcal{T}}\}$ and C an $\mathcal{ALCP}$(wapta)-concept, with both C and $C_{\mathcal{T}}$ in negation normal form, and let $\mathcal{A}_{i,\leq n}$ be the cut-point automaton of the wapta $\mathcal{A}_i$ for each prototype constructor $P_{\leq n}(\mathcal{A}_i)$ occurring in C or $C_{\mathcal{T}}$.

The apta $A_{\mathcal{P},C,\mathcal{T}}$ is the disjoint union of $\mathcal{A}_{C,\mathcal{T}}$ from Definition 84, all automata $\mathcal{A}_{i,\leq n}$ for prototypes $P_{\leq n}(\mathcal{A}_i)$ occurring positively in C or $C_{\mathcal{T}}$, and all automata $\bar{\mathcal{A}}_{i,\leq n}$ for negated prototypes $\neg P_{\leq n}(\mathcal{A}_i)$ occurring in C or $C_{\mathcal{T}}$, such that the transition function of $\mathcal{A}_{C,\mathcal{T}}$ additionally is defined for subconcepts of the form $P_{\leq n}(\mathcal{A}_i)$ and $\neg P_{\leq n}(\mathcal{A}_i)$ as follows:

$$\delta(q_{P_{\leq n}(\mathcal{A}_i)}) = q_i \text{ where } q_i \text{ is the initial state of } \mathcal{A}_{i,\leq n}$$
$$\delta(q_{\neg P_{\leq n}(\mathcal{A}_i)}) = q_i \text{ where } q_i \text{ is the initial state of } \bar{\mathcal{A}}_{i,\leq n} \qquad \Diamond$$

The following theorem is an easy consequence of Proposition 85 and Proposition 88.

Theorem 90. *Given an $\mathcal{ALCP}$(wapta)-TBox $\mathcal{T}$ and an $\mathcal{ALCP}$(wapta)-concept C, the concept C is satisfiable w.r.t. $\mathcal{T}$ iff $L(\mathcal{A}_{\mathcal{P},C,\mathcal{T}}) \neq \emptyset$.*

Because of the size of the cut-point automata and the ExpTime-emptiness test for alternating tree automata, concept satisfiability can thus be deciding in ExpTime if the numbers are given in unary. This is worst-case optimal. If the numbers are given in binary, the complexity of the algorithm increases to 2ExpTime. It is an open problem whether this second exponential blowup can be avoided.

In conclusion, the automaton $\mathcal{A}_{\mathcal{P},C,\mathcal{T}}$ will accept all tree models of an $\mathcal{ALCP}$ TBox $\mathcal{T}$ where the root is an element of the $\mathcal{ALCP}$ concept C. In example 82, we showed

how to construct a wapta $\mathcal{A}_{\text{cup}}$ that measures the distance of objects to prototypical cups. This prototype can be integrated into $\mathcal{T}$ or C via a concept $P_{\leq t}(\mathcal{A}_{\text{cup}})$. For example, the concept

$$C = P_{\leq 2}(\mathcal{A}_{\text{cup}}) \sqcap \exists \mathsf{filledWith}.(\mathsf{GreenTea} \sqcap \exists \mathsf{temperature}.\mathsf{Hot})$$

can be used to describe all objects that are similar to prototypical cups (with a distance of of at most 2) and are currently filled with hot green tea. This concept is satisfiable w.r.t. an $\mathcal{ALCP}$ TBox $\mathcal{T}$ (which may also contain $\mathcal{A}_{\text{cup}}$ or other prototypes), if $\mathcal{A}_{\mathcal{P},C,\mathcal{T}}$ accepts a non-empty language.

So far, we only considered one semantics for the weighted automata used to define prototypes. As mentioned before, other semantics might be interesting as well, in particular the semantics that interprets $\Box$ as a sum instead of the supremum, and thus add the distances of all successors. However, this would mean that the wapta is no longer bisimulation-invariant, and thus its cut-point language cannot be recognized by a (bisimulation-invariant) apta. However, we conjecture that a stronger automaton model like graded alternating parity automata defined in [KSV02] are expressive enough to describe the cut-point language if wapta with this new semantics, although the construction will be much more involved.

Chapter 6

Conclusions

In this chapter we will provide a brief summary of achieved results, and mention points of future work.

6.1 Main Results

The main results of this work can be split in two parts. In the first part, spanning Chapter 3 and Chapter 4, we have been concerned with concept similarity measures and their application to relax instance queries.

In Chapter 3, we introduced a new similarity measure $\sim_c$, that works for general $\mathcal{EL}$ TBoxes and fulfills all formal properties stated in Definition 20. To our knowledge, this is the only CSM of this kind. The measure $\sim_c$ is parameterizable using a primitive measure $\sim_p$, a weighting function g, and a discounting factor w. As such, it can be tweaked to different use cases, and for example encode preferences of which features are more or less important for the overall similarity value. Despite being used for relaxed instance queries, we believe that $\sim_c$ can be very useful in other applications as well.

We extended this similarity measure to $\mathcal{EL}^{++}$, meaning that it can handle concrete domains, role inclusions and nominals. While concrete domains in $\mathcal{EL}^{++}$ must be p-admissible and have therefore very limited expressiveness, even just value assignments can be useful when computing concept similarity values by choosing an appropriate primitive concrete measure $\sim_{\mathcal{D}}$.

Chapter 4 is concerned with a new reasoning service for DLs that allows to relax instance queries by means of concept similarity measures. By choosing appropriate similarity measures and parameters, this allows for domain- and context-dependent relaxations of the query in order to not just get certain instances of the query concept, but also similar alternatives. We have explored two methods for computing relaxed instances in the description logic $\mathcal{EL}$. The first method works for arbitrary CSMs, as long as they are equivalence invariant and role-depth bounded, but only w.r.t. unfoldable $\mathcal{EL}$-TBoxes. Since we have no further knowledge about the CSM employed for relaxing the query concept, we were only able to give a very inefficient approach, which needs to check a non-elementary number of concepts in the worst case. A second approach for answering relaxed instance queries restricts to the $\sim_c$ family of CSMs, but can handle general $\mathcal{EL}$-TBoxes. We were able to show that in this case, relaxed instance queries can be answered in NP.

In Section 4.3 we presented the ELASTIQ system, an implementation of the relaxed instance answering approach for general $\mathcal{EL}$ TBoxes, and evaluated the system using two ontologies, one hand-crafted, and one real-world ontology. This evaluation allowed us to conclude that the definition of relaxed instances is useful, but has its quirks. Additionally, the performance of ELASTIQ was decent, but not great.

Finally, in the last section of Chapter 4, we have discussed the connection of relaxed instance queries to the DL $\tau\mathcal{EL}$ defined in [BBF15]. We showed that graded membership functions, which form the basis of $\tau\mathcal{EL}$, can be defined for general $\mathcal{EL}$ TBoxes using similar methods as for $\sim_c$, and yield an alternative approach towards relaxing instance queries with better computational properties.

In Chapter 5 we introduced an extension to Description Logics that allows to define and reason over prototypes. In particular, we introduced the prototype constructor $P_{\sim n}(d)$ that is interpreted as the set of all elements of the interpretations that have a distance $\sim n$ according to a prototype distance functions d. We gave examples of pdfs encoded as weighted alternating parity tree automata, and showed that reasoning in $\mathcal{ALCP}$(wapta) w.r.t. general TBoxes is EXPTIME-complete, if the numbers are coded in unary.

6.2 Future Work

There are many directions in which future investigations would be interesting. For once, we have no lower bounds on the complexity of relaxed instance query answering, neither in case of unfoldable, nor general TBoxes, and thus don't know if the derived complexities are worst-case optimal or not. On the other hand, these approaches could always be extended to look at more expressive DLs, especially more expressive Horn-DLs that also induce finite canonical models, or to work with other kinds of similarity measures.

Similarly, it would be interesting to extend the query language, for example to relax conjunctive queries instead of instance queries. However, the similarity measure itself is only defined for (essentially tree-shaped and rooted) concepts and not for arbitrary query graphs. Therefore, an extension to conjunctive queries would also require to extent the notion of similarity measures to queries. On the other hand, we conjecture that the approach based on graded membership functions can quite easily be extended to relax conjunctive queries. A practical evaluation of the approach based on graded membership functions would be very interesting as well, in particular in comparison with ELASTIQ.

As for prototypical definitions, the semantics of weighted alternating tree automata currently only allows weights to be combined using the the operations $+$, min, and max. One can easily imagine other useful semantics, e.g. in order to allow for discounting of weights with increasing depth; however, those semantics would usually not admit regular cut-points anymore. The same is true if one wants to interpret the $\Box$-operator by a sum instead of max, which would add the distances of all successors instead of choosing the worst one, which makes the wapta no

more bisimulation-invariant. Investigating more expressive automata models like graded automata [KSV02] might be one way to solve this problem.

Two very useful extensions to prototypical definitions on the DL side would certainly be nominals and quantified number restrictions. With those, it would be possible to also reason over individual objects given in an ABox, and allow prototypes to count. For example, this would allow to specify that a prototypical cup has exactly one handle, and increase the distance for each additional handle. We believe the approach can be extended by using fully enriched automata as proposed in [BLM+06], which extend apta, amongst other things, with graded modalities and nominals.

Bibliography

[AP94] Agnar Aamodt and Enric Plaza. "Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches". In: *AI Communications* 7.1 (1994), pages 39–59 (cited on page 4).

[APS14] Tahani Alsubait, Bijan Parsia, and Uli Sattler. "Measuring Similarity in Ontologies: A New Family of Measures". In: *Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2014)*. Edited by Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen. Volume 8876. Lecture Notes in Computer Science. Springer, 2014, pages 13–25. DOI: 10.1007/978-3-319-13704-9_2 (cited on pages 23, 43).

[ALP04] Sihem Amer-Yahia, Laks V. S. Lakshmanan, and Shashank Pandit. "FleXPath: Flexible Structure and Full-text Querying for XML". In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. SIGMOD '04. Paris, France: ACM, 2004, pages 83–94. ISBN: 1-58113-859-8. DOI: 10.1145/1007568.1007581 (cited on pages 6, 55).

[BBL05] F. Baader, S. Brandt, and C. Lutz. "Pushing the $\mathcal{EL}$ Envelope". In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*. Edinburgh, UK: Morgan-Kaufmann Publishers, 2005 (cited on pages 2, 16, 17, 20, 21).

[Baa03] Franz Baader. "Terminological Cycles in a Description Logic with Existential Restrictions". In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. IJCAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc., 2003, pages 325–330 (cited on page 2).

[Baa09] Franz Baader. "Description Logics". In: *Reasoning Web: Semantic Technologies for Information Systems, 5th International Summer School*. Volume 5689. Lecture Notes in Computer Science. Springer, 2009, pages 1–39 (cited on pages 85, 92).

[BBL08] Franz Baader, Sebastian Brandt, and Carsten Lutz. "Pushing the $\mathcal{EL}$ Envelope Further". In: *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*. Edited by Kendall Clark and Peter F. Patel-Schneider. 2008 (cited on page 20).

[BBF15] Franz Baader, Gerhard Brewka, and Oliver Fernández Gil. "Adding Threshold Concepts to the Description Logic $\mathcal{EL}$". In: *Proceedings of the 10th International Symposium on Frontiers of Combining Systems (FroCoS'15)*. Edited by Carsten Lutz and Silvio Ranise. Volume 9322. Lec-

tures Notes in Artificial Intelligence. Wroclaw, Poland: Springer, 2015, pages 33–48 (cited on pages 7, 56, 73, 74, 76, 100).

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0-521-78176-0 (cited on pages 1, 11, 14).

[BH91] Franz Baader and Philipp Hanschke. "A Scheme for Integrating Concrete Domains into Concept Languages". In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, pages 452–457. ISBN: 1-55860-160-0 (cited on page 21).

[BK98] Franz Baader and Ralf Küsters. "Computing the least common subsumer and the most specific concept in the presence of cyclic ALN-concept descriptions". In: *Advances in Artificial Intelligence: 22nd Annual German Conference on Artificial Intelligence*. Edited by Otthein Herzog and Andreas Günter. Springer Berlin Heidelberg, 1998, pages 129–140. DOI: 10.1007/BFb0095434 (cited on page 18).

[Ban22] Stefan Banach. "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales". In: *Fundamenta Mathematicae* 3.1 (1922), pages 133–181 (cited on pages 31, 66).

[BRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. New York, NY, USA: Cambridge University Press, 2001. ISBN: 0-521-80200-8 (cited on page 92).

[BLM+06] Piero A. Bonatti, Carsten Lutz, Aniello Murano, and Moshe Y. Vardi. "The complexity of enriched μ-calculi". In: *Proc. ICALP 2006*. Volume 4052. LNCS. Springer, 2006, pages 540–551. DOI: 10.1007/11787006_46 (cited on page 101).

[BWH05] A. Borgida, T. Walsh, and H. Hirsh. "Towards Measuring Similarity in Description Logics". In: *Proceedings of the 18th International Workshop on Description Logics (DL-2005)*. Volume 147. CEUR Workshop Proceedings. 2005 (cited on pages 5, 43).

[Bre91] G. Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge: Cambridge University Press, 1991 (cited on page 7).

[CDH+06] Surajit Chaudhuri, Gautam Das, Vagelis Hristidis, and Gerhard Weikum. "Probabilistic Information Retrieval Approach for Ranking of Database Query Results". In: *ACM Transactions on Database Systems* 31.3 (2006), pages 1134–1168. DOI: 10.1145/1166074.1166085 (cited on pages 6, 55).

[dFE05] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. "A Semantic Similarity Measure for Expressive Description Logics". In: *Convegno Italiano di Logica Computazionale* (2005) (cited on pages 5, 23, 43).

[dSF08] Claudia d'Amato, Steffen Staab, and Nicola Fanizzi. "On the Influence of Description Logics Ontologies on Conceptual Similarity". In: *Proceedings of Knowledge Engineering: Practice and Patterns, 16th International Conference (EKAW 2008)*. Edited by Aldo Gangemi and Jérôme Euzenat. Volume 5268. Lecture Notes in Computer Science. Springer, 2008, pages 48–63. ISBN: 978-3-540-87695-3 (cited on page 43).

[Dis08] Felix Distel. *Model-based Most Specific Concepts in Description Logics with Value Restrictions*. Technical report 08-04. See http://lat.inf.tu-dresden.de/research/reports.html. Dresden, Germany: Institute for theoretical computer science, TU Dresden, 2008 (cited on page 19).

[DAB14] Felix Distel, Jamal Atif, and Isabelle Bloch. "Concept Dissimilarity with Triangle Inequality". In: *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*. Short Paper. Vienna, Austria: AAAI Press, 2014 (cited on pages 23, 43).

[DSW+09] Peter Dolog, Heiner Stuckenschmidt, Holger Wache, and Jörg Diederich. "Relaxing RDF Queries Based on User and Domain Preferences". In: *Journal of Intelligent Information Systems* 33.3 (2009), pages 239–260. DOI: 10.1007/s10844-008-0070-7 (cited on pages 6, 55).

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 2009. ISBN: 978-3-64201-491-8 (cited on page 85).

[EPT13a] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Computing Role-depth Bounded Generalizations in the Description Logic $\mathcal{ELOR}$". In: *Proceedings of the 36th German Conference on Artificial Intelligence (KI 2013)*. Volume 8077. Lecture Notes in Artificial Intelligence. Koblenz, Germany: Springer-Verlag, 2013, pages 49–60 (cited on page 19).

[EPT13b] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Towards Instance Query Answering for Concepts Relaxed by Similarity Measures". In: *Workshop on Weighted Logics for AI (in conjunction with IJCAI'13)*. Beijing, China, 2013 (cited on page 57).

[EPT14] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Answering Instance Queries Relaxed by Concept Similarity". In: *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*. Vienna, Austria: AAAI Press, 2014, pages 248–257 (cited on page 57).

[EPT15] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. "Similarity-based Relaxed Instance Queries". In: *Journal of Applied Logic* 13.4, Part 1 (2015). Special Issue for the Workshop on Weighted Logics for AI

2013, pages 480–508. DOI: http://dx.doi.org/10.1016/j.jal.2015.01.002 (cited on page 57).

[Gär00] Peter Gärdenfors. *Conceptual spaces - the geometry of thought*. MIT Press, 2000. ISBN: 978-0-262-07199-4 (cited on pages 7, 85).

[Gen00] The Gene Ontology Consortium. "Gene Ontology: Tool for the Unification of Biology". In: *Nature Genetics* 25 (2000), pages 25–29 (cited on pages 3, 4, 68).

[GM97] Dedre Gentner and Arthur B. Markman. "Structure mapping in analogy and similarity". In: *AMERICAN PSYCHOLOGIST* 52 (1997), pages 45–56 (cited on page 4).

[HRJ+13] Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. "Semantic Measures for the Comparison of Units of Language, Concepts or Entities from Text and Knowledge Base Analysis". In: *Computing Research Repository* abs/1310.1285 (2013). URL: http://arxiv.org/abs/1310.1285 (cited on pages 1, 3).

[Hla07] Jan Hladik. "To and Fro Between Tableaus and Automata for Description Logics". Dissertation. TU Dresden, Germany, 2007. URL: http://www.qucosa.de/fileadmin/data/qucosa/documents/846/1201792812059-1908.pdf (cited on pages 90, 93).

[HPH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank Van Harmelen. "From SHIQ and RDF to OWL: The Making of a Web Ontology Language". In: *Journal of Web Semantics* 1 (2003), page 2003 (cited on page 2).

[HLZ08] Hai Huang, Chengfei Liu, and Xiaofang Zhou. "Computing Relaxed Answers on RDF Databases". In: *Proceedings of the 9th International Conference on Web Information Systems Engineering*. WISE '08. Auckland, New Zealand: Springer-Verlag, 2008, pages 163–175. ISBN: 978-3-540-85480-7. DOI: 10.1007/978-3-540-85481-4_14 (cited on pages 6, 55).

[HPW08] Carlos A. Hurtado, Alexandra Poulovassilis, and Peter T. Wood. "Journal on Data Semantics X". In: edited by Stefano Spaccapietra. Berlin, Heidelberg: Springer-Verlag, 2008. Chapter Query Relaxation in RDF, pages 31–61. ISBN: 978-3-540-77687-1 (cited on pages 6, 55).

[Jac01] Paul Jaccard. "Étude comparative de la distribution florale dans une portion des Alpes et des Jura". In: *Bulletin del la Société Vaudoise des Sciences Naturelles* 37 (1901), pages 547–579 (cited on page 4).

[Jan06] Krzysztof Janowicz. "Sim-DL: Towards a Semantic Similarity Measurement Theory for the Description Logic $\mathcal{ALCNR}$ in Geographic Information Retrieval". English. In: *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*. Edited by Robert Meersman, Zahir Tari, and Pilar Herrero. Volume 4278. Lecture Notes in Computer

Science. Springer Berlin Heidelberg, 2006, pages 1681–1692. ISBN: 978-3-540-48273-4. DOI: 10.1007/11915072_74. URL: http://dx.doi.org/10.1007/11915072_74 (cited on pages 5, 23, 43).

[JW09] Krzysztof Janowicz and Marc Wilkes. "SIM-DLA: A Novel Semantic Similarity Measure for Description Logics Reducing Inter-concept to Inter-instance Similarity". In: *Proceedings of 8th Extended Semantic Web Conference (ESWC'09)*. Volume 5554. Lecture Notes in Computer Science. Springer-Verlag, 2009, pages 353–367 (cited on page 43).

[Kar84] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Combinatorica* 4.4 (1984), pages 373–395. DOI: 10.1007/BF02579150 (cited on page 36).

[KKS14] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. "The Incredible ELK: From Polynomial Procedures to Efficient Reasoning with $\mathcal{EL}$ Ontologies". In: *Journal of Automated Reasoning* 53.1 (2014), pages 1–61. DOI: 10.1007/s10817-013-9296-3 (cited on page 66).

[KSV02] Orna Kupferman, Ulrike Sattler, and Moshe Y. Vardi. "The complexity of the graded μ-calculus". In: *Proc. of the 18th Int. Conf. on Automated Deduction*. Volume 2392. Lecture Notes in Computer Science. Springer, 2002, pages 423–437. DOI: 10.1007/3-540-45620-1_34 (cited on pages 97, 101).

[KR99] Natasha Kurtonina and Maarten de Rijke. "Expressiveness of concept expressions in first-order description logics". In: *Artificial Intelligence* 107.2 (1999), pages 303–333. DOI: http://dx.doi.org/10.1016/S0004-3702(98)00109-X (cited on page 16).

[KM01] R. Küsters and R. Molitor. "Approximating Most Specifc Concepts in Description Logics with Existential Restrictions". In: *Proceedings of the 24th German Annual Conference on Artificial Intelligence (KI-01)*. Edited by F. Baader, G. Brewka, and T. Eiter. Volume 2174. Lecture Notes In Artificial Intelligence. Springer, 2001, pages 33–47 (cited on pages 18, 19).

[Lab73] William Labov. "The boundaries of words and their meanings". In: *New ways of analyzing variation in English*. Edited by Charles-James N Bailey and Roger W Shuy. Georgetown University Press, 1973 (cited on page 6).

[Lee02] Dongwon Lee. "Query Relaxation for XML Model". PhD thesis. University of California, Los Angeles, 2002. URL: http://pike.psu.edu/publications/dongwon-dissertation.pdf (cited on pages 6, 55).

[LT12] Karsten Lehmann and Anni-Yasmin Turhan. "A Framework for Semantic-based Similarity Measures for $\mathcal{ELH}$-Concepts". In: *Proceedings of the 13th European Conference on Logics in A.I. (JELIA 2012)*. Edited by Luis Fariñas del Cerro, Andreas Herzig, and Jérôme Mengin. Lecture Notes

In Artificial Intelligence. Springer-Verlag, 2012, pages 307–319 (cited on pages 5, 23–25, 27, 42, 43, 56, 73).

[Lin98] Dekang Lin. "An Information-Theoretic Definition of Similarity". In: *Proceedings of the Fifteenth International Conference on Machine Learning*. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pages 296–304 (cited on pages 3–5, 23).

[LWZ03] C. Lutz, F. Wolter, and M. Zakharyaschev. "Reasoning about concepts and similarity". In: *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*. Edited by D. Calvanese, G. De Giacomo, and E. Franconi. CEUR-WS 81. 2003 (cited on pages 6, 55).

[LBF+06] Carsten Lutz, Franz Baader, Enrico Franconi, Domenico Lembo, Ralf Möller, Riccardo Rosati, Ulrike Sattler, Boontawee Suntisrivaraporn, and Sergio Tessaris. "Reasoning Support for Ontology Design". In: *In Proceedings of the second international workshop OWL: Experiences and Directions*. Edited by Bernardo Cuenca Grau, Pascal Hitzler, Connor Shankey, and Evan Wallace. Volume 216. CEUR Workshop Proceedings. CEUR-WS.org, 2006 (cited on page 2).

[LW10] Carsten Lutz and Frank Wolter. "Deciding Inseparability and Conservative Extensions in the Description Logic $\mathcal{EL}$". In: *Journal of Symbolic Computation* 45.2 (2010), pages 194–228. DOI: 10.1016/j.jsc.2008.10.007 (cited on pages 17, 18, 45, 48).

[MBP13a] Nicolas Matentzoglu, Samantha Bail, and Bijan Parsia. "A Corpus of OWL DL Ontologies." In: *Proceedings of the 26th International Workshop on Description Logics (DL-2013)*. Volume 1014. CEUR Workshop Proceedings. CEUR-WS.org, 2013, pages 829–841 (cited on page 2).

[MBP13b] Nicolas Matentzoglu, Samantha Bail, and Bijan Parsia. "A Snapshot of the OWL Web". In: *Proceedings of the 12th International Semantic Web Conference (ISWC 2013)*. 2013, pages 331–346 (cited on page 71).

[MET11] Julian Mendez, Andreas Ecke, and Anni-Yasmin Turhan. "Implementing completion-based inferences for the $\mathcal{EL}$-family". In: *Proceedings of the International Description Logics workshop*. Edited by Riccardo Rosati, Sebastian Rudolph, and Michael Zakharyaschev. Volume 745. CEUR, 2011 (cited on page 19).

[OWL09] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. Available at http://www.w3.org/TR/owl2-overview/. W3C Recommendation, 27 October 2009 (cited on page 64).

[PSS+08] Jeff Z. Pan, Giorgos B. Stamou, Giorgos Stoilos, Stuart Taylor, and Edward Thomas. "Scalable querying services over fuzzy ontologies". In: *Proceedings of the 17th International Conference on World Wide Web (WWW'08)*. ACM, 2008, pages 575–584 (cited on pages 6, 55).

[PT11] R. Peñaloza and A.-Y. Turhan. "A Practical Approach for Computing Generalization Inferences in $\mathcal{EL}$". In: *Proceedings of the 8th European Semantic Web Conference (ESWC'11)*. Edited by Marko Grobelnik and Elena Simperl. Lecture Notes in Computer Science. Springer, 2011, pages 410–423 (cited on page 19).

[PTT14] Rafael Peñaloza, Veronika Thost, and Anni-Yasmin Turhan. *Conjunctive Query Answering in Rough $\mathcal{EL}$*. LTCS-Report 14-04. See `http://lat.inf.tu-dresden.de/research/reports.html`. Dresden, Germany: Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden, 2014 (cited on pages 6, 55).

[RMB+89] R. Rada, H. Mili, E. Bicknell, and M. Blettner. "Development and application of a metric on semantic nets". In: *IEEE Transactions on Systems, Man and Cybernetics*. 1989, pages 17–30 (cited on pages 5, 23).

[RL78] E. Rosch and B. B. Lloyd. *Cognition and Categorization*. L. Erlbaum Associates, 1978. ISBN: 9780835734042 (cited on page 6).

[SV01] Ulrike Sattler and Moshe Y. Vardi. "The hybrid μ-calculus". In: *Proc. IJCAR 2001*. Volume 2083. LNCS. Springer, 2001, pages 76–91. DOI: 10.1007/3-540-45744-5_7 (cited on page 92).

[Sch91] Klaus Schild. "A correspondence theory for terminological logics: Preliminary report". In: *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*. Edited by John Mylopoulos and Raymond Reiter. Morgan Kaufmann, 1991, pages 466–471 (cited on page 94).

[SS91] Manfred Schmidt-Schauß and Gert Smolka. "Attributive Concept Descriptions with Complements". In: *Artificial Intelligence* 48.1 (1991), pages 1–26. DOI: 10.1016/0004-3702(91)90078-X (cited on page 2).

[SSP+05] J. L. Sevilla, V. Segura, A. Podhorski, E. Guruceaga, J. M. Mato, L. A. Martinez-Cruz, F. J. Corrales, and A. Rubio. "Correlation between gene expression and GO semantic similarity". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2.4 (2005), pages 330–338. DOI: 10.1109/TCBB.2005.50 (cited on page 4).

[She62] Roger N Shepard. "The analysis of proximities: Multidimensional scaling with an unknown distance function. I." In: *Psychometrika* 27.2 (1962), pages 125–140 (cited on page 4).

[She87] Roger N Shepard. "Toward a universal law of generalization for psychological science". In: *Science* 237.4820 (1987), pages 1317–1323. DOI: 10.1126/science.3629243 (cited on page 4).

[SE13] P. Shvaiko and J. Euzenat. "Ontology Matching: State of the Art and Future Challenges". In: *IEEE Transactions on Knowledge and Data Engineering* 25.1 (2013), pages 158–176. DOI: 10.1109/TKDE.2011.253 (cited on page 5).

[Str06a] Umberto Straccia. "Answering vague queries in fuzzy DL-Lite". In: *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-06)*. 2006, pages 2238–2245 (cited on pages 6, 55).

[Str06b] Umberto Straccia. "Towards Top-k Query Answering in Description Logics: The Case of DL-Lite". In: *Proceedings of the 10th European Conference on Logics in A.I. (JELIA 2006)*. Volume 4160. Lecture Notes in Computer Science. Springer, 2006, pages 439–451 (cited on pages 6, 55).

[Sun13] Boontawee Suntisrivaraporn. "A Similarity Measure for the Description Logic $\mathcal{EL}$ with Unfoldable Terminologies". In: *5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*. 2013, pages 408–413. DOI: 10.1109/INCoS.2013.77 (cited on pages 5, 23, 25, 43).

[SBS+07] Boontawee Suntisrivaraporn, Franz Baader, Stefan Schulz, and Kent Spackman. "Replacing SEP-Triplets in SNOMED CT using Tractable Description Logic Operators". In: *Proceedings of the 11th Conference on Artificial Intelligence in Medicine (AIME'07)*. Edited by Jim Hunter, Riccardo Bellazzi, and Ameen Abu-Hanna. Lecture Notes in Computer Science. Springer-Verlag, 2007 (cited on page 3).

[TH06] Dmitry Tsarkov and Ian Horrocks. "FaCT++ Description Logic Reasoner: System Description". In: *Proceedings of the Third International Joint Conference on Automated Reasoning*. IJCAR'06. Seattle, WA: Springer-Verlag, 2006, pages 292–297. ISBN: 978-3-540-37187-8. DOI: 10.1007/11814771_26. URL: http://dx.doi.org/10.1007/11814771_26 (cited on page 72).

[Tve77] A. Tversky. "Features of similarity". In: *Psychological Review* 84 (1977), pages 327–352 (cited on pages 3, 4, 24).

[Wil01] Thomas Wilke. "Alternating tree automata, parity games, and modal μ-calculus". In: *Bull. Belg. Math. Soc.* 8 (2001), pages 359–391 (cited on pages 87, 88, 96).

[ZGB+07] Xuan Zhou, Julien Gaugaz, Wolf-Tilo Balke, and Wolfgang Nejdl. "Query Relaxation Using Malleable Schemas". In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. SIGMOD '07. Beijing, China: ACM, 2007, pages 545–556. ISBN: 978-1-59593-686-8. DOI: 10.1145/1247480.1247541 (cited on pages 6, 55).